openSUSE Leap 15.7

セキュリティ強化ガイド

セキュリティ強化ガイド
openSUSE Leap 15.7

このガイドでは、システムセキュリティに関する基本的な考え方と AppArmor や SELinux 、監査システム等のように、製品に含まれる各種のセキュリティソフトウエアの 使用方法を説明しています。このほか、システムをより強固にするためのシステム管理者向けガイドも含まれています。

発行日: 2026/11/05

SUSE LLC
1800 South Novell Place
Provo, UT 84606
USA
https://documentation.suse.com

Copyright © 2006– 2026 SUSE LLC and contributors. All rights reserved.

訳: SUSE LLC および貢献者が全権利を留保しています。

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled 「GNU Free Documentation License」.

訳: この文書を、フリーソフトウェア財団発行の GNU フリー文書利用許諾契約書バージョン 1.2 または (希望すれば) 1.3 が定める条件の下で複製、頒布、あるいは改変することを許可します。ただし、この著作権とライセンス表記については変更不可部分とします。この利用許諾契約書の複製物は、「GNU フリー文書利用許諾契約書」という章に含まれています。

For SUSE trademarks, see https://www.suse.com/company/legal/ . All third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

訳: SUSE 社の商標については、https://www.suse.com/company/legal/ をご覧ください。その他の商標は各所有者の所有物です。商標シンボル (®, ™ など) は、それぞれ SUSE 社およびその関連会社の商標であることを示しています。また、アスタリスク (*) は第三者の商標を示しています。

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither SUSE LLC, its affiliates, the authors nor the translators shall be held liable for possible errors or the consequences thereof.

訳: この書籍内にある全ての情報は、細部に至るまで最大限の注意を払って制作されていますが、完全に正確であることを保証するものではありません。SUSE LLC やその関連会社, 著者, 翻訳者のいずれも、本書籍内の誤りとそこから生じる結果について、一切の保証はいたしません。



注記

なお、本文書は原文 (英語) の翻訳文書であり、公式な文書ではありません。あらかじめご了承ください。

目次

前書き xx
1 利用可能なドキュメンテーション xx
2 ドキュメンテーションの改善 xx
3 文書規約 xxi
1 セキュリティと機密保持 1
1.1 概要 1
1.2 パスワード 2
1.3 バックアップ 2
1.4 システムの完全性保護 3
1.5 ファイルへのアクセス権 4
1.6 ネットワーク 5
1.7 ソフトウエアの脆弱性 5
1.8 マルウエア 6
1.9 主なセキュリティヒント 7
1.10 セキュリティ問題の報告 8
I 認証 9
2 PAM を利用した認証 10
2.1 PAM とは? 10
2.2 PAM 設定ファイルの構造 11
2.3 sshd の PAM 設定 14
2.4 PAM モジュールの設定 16
pam_env.conf 16 • pam_mount.conf.xml 17 • limits.conf 18

2.5 pam-config を利用した PAM の設定 18
2.6 PAM の手動設定 19
2.7 ローカルログインのための U2F キーの設定 19
アカウントと U2F キーの紐付け 20 • PAM 設定の更新 21
2.8 さらなる情報 22
3 NIS の使用 23
3.1 NIS サーバの設定 23
NIS マスターサーバの設定 23 • NIS スレーブサーバの設定 28
3.2 NIS クライアントの設定 29
4 YaST を利用した認証クライアントの設定 31
4.1 YaST を利用した認証クライアントの設定 31
4.2 SSSD 31
状態の確認 32 • キャッシュ 32
5 389 Directory Server を利用した LDAP サービス 33
5.1 LDAP のディレクトリツリー構造 33
5.2 389 Directory Server の Docker コンテナの作成と管理 36
5.3 389 Directory Server のインストール 37
389 Directory Server のインスタンスの構築 37 • 独自の設定ファイルを利用した 389 Directory Server インスタンスの作成 39 • テンプレートからの 389 Directory Server インスタンスの作成 41 • 389 Directory Server の起動と停止 42 • ローカル管理用の管理者認証情報の設定 43
5.4 ファイアウオールの設定 44
5.5 389 Directory Server でのバックアップと復元 45
LDAP サーバ設定のバックアップ 45 • LDAP データベースに対するオフラインバックアップの採取と復元 46 • LDAP データベースに対するオンラインバックアップの採取と復元 47

5.6 LDAP でのユーザとグループの管理 47
既存の LDAP ユーザとグループの表示 47 • ユーザの作成とパスワードの管理 48 • グループの作成と管理 49 • ユーザ／グループの削除、およびグループからのユーザ脱退 49

5.7 プラグインの管理 50
389 Directory Server でのサポート対象外プラグイン 51

5.8 LDAP 認証管理のための SSSD の使用 51
サポート対象外のパスワードハッシュと認証方式 54

5.9 OpenLDAP から 389 Directory Server への移行 55
OpenLDAP からのテスト移行 55 • saslauthd を使用する OpenLDAP サーバのテスト移行 59 • 移行の計画 63

5.10 TLS 証明書と鍵の取り込み 64

5.11 レプリケーションの設定 66
非同期書き込み 66 • 構成設計 67 • レプリケーションの設計例 67 • 用語 70 • レプリケーションの設定 71 • 監視とヘルスチェック 74 • バックアップの作成 74 • レプリケーションの一時停止と再開 75 • Changelog max-age 75 • レプリカの削除 76 • 389 Directory Server でのレプリケーションの制限 76

5.12 Microsoft Active Directory との同期 77
同期トポロジの計画 77 • Active Directory 側の事前要件 78 • 389 Directory Server 側の事前要件 78 • Active Directory から 389 Directory Server への同意の作成 78

5.13 さらなる情報 79

6 Kerberos を利用したネットワーク認証 81

6.1 考え方の概要 81

6.2 Kerberos で使用する用語 81

6.3 Kerberos の動作概要 83
最初の通信 83 • サービスの要求 84 • 相互認証 85 • チケット発行許諾 - 全てのサーバへの通信 85

6.4 Kerberos のユーザ側からの見た目 86

6.5 Kerberos のインストールと管理 87
Kerberos ネットワークトポロジ 87 • Kerberos レルムの選択 88 • KDC ハードウエアの設定 89 • 時刻同期の設定 90 • KDC の設定 91 • Kerberos クライアントの設定 94 • リモートからの Kerberos 管理の設定 97 • Kerberos サービスプリンシパルの作成 99 • Kerberos に対する PAM サポートの有効化 100 • Kerberos 認証用の SSH の設定 101 • LDAP と Kerberos の使用 102

6.6 Kerberos と NFS 105
グループのメンバー 106 • 性能とスケーラビリティ 107 • マスター KDC／複数ドメイン／信頼関係 108

6.7 さらなる情報 108

7 Himmelblau を利用した Microsoft Entra ID による認証 109

7.1 Himmelblau のインストール 109

7.2 Himmelblau 認証の設定 109

7.3 設定の確認 110

7.4 サインインできるユーザの制限 111

7.5 Intune ポリシー強制の有効化 111

8 Active Directory サポート 112

8.1 Linux と Active Directory 環境の統合 112

8.2 Linux Active Directory サポートに対する背景情報 113
ドメインへの参加 115 • ドメインログインとユーザのホームディレクトリ 116 • オフラインサービスとポリシーサポート 117

8.3 Active Directory 向けの Linux クライアントの設定 118
Active Directory に接続する際に使用する YaST モジュールの選択 119 • [ユーザのログイン管理] を利用した Active Directory への参加 120 • [Windows ドメインメンバーシップ] を利用した Active Directory ドメインへの参加 125 • Active Directory の接続状態の確認 127

8.4 Active Directory ドメインへのログイン 127
GDM 127 • コンソールログイン 128

8.5 パスワードの変更 128

8.6 Active Directory での証明書自動登録 129
サーバ側での証明書自動登録設定 129 • クライアント側での証明書自動登録の有効化 130

9 FreeRADIUS サーバの構築 132

9.1 openSUSE Leap でのインストールとテスト 132

II ローカルセキュリティ 135

10 物理的なセキュリティ 136

10.1 システムの施錠 136

10.2 BIOS のロックダウン 137

10.3 ブートローダでのセキュリティ 138

10.4 機密データを含む Linux サーバの廃棄 138
scrub: ディスクの上書きユーティリティ 139

10.5 リムーバブルメディアへのアクセス制限 141

10.6 USBGuard による USB デバイスの認可とシステムの保護 142
USBGuard のインストール 143 • USBGuard の使用方法 144 • さらなる情報 145

11 ソフトウエア管理 146

11.1 不要なソフトウエアパッケージ (RPM) の削除 146

11.2 Linux システムへの修正の適用 148
YaST オンライン更新 148 • 自動オンライン更新 148

12 ファイルの管理 150

12.1 ディスクのパーティション 150

12.2 特定のシステムファイルに対するパーミッションの変更 151

12.3 ホームディレクトリに対する 755 から 700 へのパーミッション変更 152

12.4 既定の umask 153
既定の umask 値の調整 154

12.5 SUID/SGID フラグの付いたファイル 155

12.6 全てのユーザが書き込むことのできるファイル 156

12.7 孤立したファイル／所有者のいないファイル 157

13 パーティションやファイルの暗号化 158

13.1 YaST を利用した暗号化ファイルシステムの設定 158
インストール時における暗号化パーティションの作成 159 • 動作中のシステムにおける暗号化パーティションの作成 160 • リムーバブルメディアの内容の暗号化 161

13.2 GPG を利用したファイルの暗号化 161

13.3 Rage によるファイルの暗号化 162
その他の情報 164

14 cryptctl を利用したアプリケーション向けのストレージ暗号化 165

14.1 `cryptctl` サーバの設定 166

14.2 `cryptctl` クライアントの設定 168

14.3 LUKS ボリューム向け /etc/fstab の設定 171

14.4 サーバ側のコマンドでパーティションの解読状態を確認する方法 171

14.5 手作業での暗号化パーティションの解読 172

14.6 メンテナンス時の停止手順 173

14.7 cryptctl-server での HA 環境の構築 173

14.8 さらなる情報 177

15 ユーザ管理 178

15.1 様々なアカウントのチェック 178
ロック (施錠) されていないアカウント 178 • 使用されていないアカウント 179

15.2 パスワードの有効期限の設定 179

15.3 複雑なパスワードの強制 181

15.4 PAM を利用したパスワードとログインの管理 181
パスワードの強度 182 • 過去に使用したことのあるパスワードの制限 183 • ログイン失敗時のユーザアカウントのロックアウト 184
15.5 root ログインの制限 185
ローカルのテキストコンソールへのログイン制限 186 • グラフィカルセッションへのログインの制限 187 • SSH ログインの制限 188
15.6 sudo を利用できるユーザの制限 188
15.7 対話型シェルセッションに対する無操作タイムアウトの設定 190
15.8 不用意なサービス拒否攻撃の防止 193
システムリソースの制限例 193
15.9 ログインバナーの表示 195
15.10 各種のアカウント関連ユーティリティ 196
16 cron と at の制限 198
16.1 cron デーモンの制限 198
16.2 at スケジューラの制限 199
17 Spectre/Meltdown チェッカー 201
17.1 spectre-meltdown-checker の使用 201
17.2 さらなる情報 203
18 YaST を利用したセキュリティの設定 204
18.1 [セキュリティの概要] 204
18.2 [事前定義済みのセキュリティ設定] 205
18.3 [パスワード設定] 206
18.4 [起動設定] 207
18.5 [ログイン設定] 207
18.6 [ユーザ追加] 207
18.7 [その他の設定] 207

19 Polkit 認可フレームワーク 209
19.1 考え方の概要 209
認証エージェント 209 • Polkit の設定 210 • Polkit のユーティリティ 210
19.2 認可の種類 211
暗黙の認可 211 • SUSE 既定の権限 212
19.3 権限の問い合わせ 212
19.4 Polkit の設定変更 213
Polkit ポリシーファイルの上書き 213 • JavaScript 認可ルールの追加 214 • SUSE 既定の権限設定の変更 215
19.5 SUSE 既定の権限設定の復元 216
20 Linux でのアクセス制御リスト 217
20.1 従来型のファイルアクセス権 217
setuid ビット 218 • setgid ビット 218 • sticky ビット 219
20.2 ACL の利点 219
20.3 定義 220
20.4 ACL の処理 220
ACL エントリとファイルモード許可ビットの関係 222 • ACL の設定されたディレクトリ 222 • 既定の ACL が設定されたディレクトリ 225 • ACL チェックアルゴリズム 228
20.5 アプリケーション内での ACL サポート 228
20.6 さらなる情報 228
21 AIDE を利用した侵入検知 229
21.1 AIDE を使用する理由 229
21.2 AIDE データベースの設定 229
21.3 ローカルの AIDE チェック 232
21.4 システムから独立したチェック 234
21.5 さらなる情報 235

III ネットワークセキュリティ 236

22 X Window System と X 認証 237

23 OpenSSH によるネットワーク操作の機密保持 238

23.1 OpenSSH の概要 238

23.2 サーバのセキュリティ強化 242

23.3 パスワード認証 244

23.4 ユーザ鍵とホスト鍵の管理 244
ユーザ SSH 鍵の生成 245 • SSH サーバのホスト鍵作成 247

23.5 ホスト鍵の切り替え 248

23.6 公開鍵認証 250

23.7 パスフレーズ無しの公開鍵認証 251

23.8 OpenSSH での証明書認証 252
新しい証明機関の構築 253 • サーバ証明書の作成 255 • ユーザ向けの CA 設定 256 • ユーザ証明書の作成 257 • ホスト鍵の失効処理 257

23.9 gnome-keyring を利用した公開鍵ログインの自動化 258

23.10 ssh-agent を利用した公開鍵ログインの自動化 258
`ssh-agent` の X セッション内での利用 259

23.11 SSH 機密鍵のパスフレーズ変更 259

23.12 鍵の指紋 (fingerprint) の取得 260

23.13 ネットワーク上の離れたホストでの X11 アプリケーションの起動 260

23.14 エージェント転送 261

23.15 `scp` - 機密を確保したファイルのコピー 262

23.16 `sftp` - 機密を保持したファイルの転送 263
`sftp` の使用 263 • ファイルアップロードのパーミッション設定 263

23.17 ポート転送 (SSH トンネリング) 264

23.18 さらなる情報 265

23.19 fail2ban による SSH ブルートフォース攻撃の抑止 266
fail2ban とは？ 266 • SSH ブルートフォース攻撃を抑止するための fail2ban の使用 266

24 マスカレードとファイアウオール 269

24.1 iptables によるパケットフィルタリング 269

24.2 基本的なマスカレード処理 272

24.3 基本的なファイアウオール処理 273

24.4 firewalld 274
コマンドラインからのファイアウオール設定 275 • 動的にポートを割り当てるタイプのサービスへのアクセス 280

24.5 SuSEfirewall2 からの移行 286

24.6 さらなる情報 288

25 VPN サーバの設定 289

25.1 考え方の概要 289
用語 289 • VPN のシナリオ 290

25.2 シンプルな構成の構築例 293
VPN サーバの設定 294 • VPN クライアントの設定 295 • シンプルな構成のテスト 296

25.3 証明機関を利用した VPN サーバの構築 296
証明機関の作成 297 • VPN サーバの設定 297 • VPN クライアントの設定 300

25.4 YaST を利用した VPN サーバおよびクライアントの設定 301

25.5 さらなる情報 302

26 X Window System で動作する PKI マネージャ XCA による管理 303

26.1 XCA のインストール 303

26.2 新しい PKI の作成 303
新しいルート証明機関の作成 304 • 新しいホスト証明書の作成 305 • 証明書の失効化 306

27 `sysctl` 変数によるネットワークセキュリティの改善 307

IV APPARMOR による権限の制限 310

28 AppArmor の紹介 311
28.1 AppArmor のコンポーネント 311
28.2 AppArmor プロファイリングに関する背景となる情報 312

29 入門 313
29.1 AppArmor のインストール 313
29.2 AppArmor の有効化／無効化 314
29.3 プロファイルを作成するアプリケーションの選択 315
29.4 プロファイルの作成と修正 316
29.5 プロファイルの更新 317

30 プログラムに対する予防接種 318
30.1 AppArmor フレームワークの紹介 319
30.2 予防接種を行うプログラムの決定 321
30.3 cron ジョブへの予防接種 322
30.4 ネットワークアプリケーションへの予防接種 322
Web アプリケーションへの予防接種 324 • ネットワークエージェントに対する予防接種 326

31 プロファイルのコンポーネントと文法 328
31.1 AppArmor プロファイル内の各部の意味 329
31.2 プロファイルの種類 331
標準プロファイル 332 • 未接続プロファイル 332 • ローカルプロファイル 333 • ハット 333 • ルール変更 333
31.3 include ステートメント 334
抽象 336 • プログラムチャンク 336 • チューナブル 337
31.4 ケーパビリティ項目 (POSIX.1e) 337

31.5 ネットワークアクセス制御 337

31.6 プロファイル名／フラグ／パス／グロブ 338
プロファイルフラグ 340 • プロファイル内での変数の使用 340 • パターンマッチ 341 • ネームスペース 342 • プロファイル名と接続仕様 342 • 別名ルール 343

31.7 ファイルアクセス許可のアクセスモード 344
読み込みモード (r) 344 • 書き込みモード (w) 344 • 追記モード (a) 345 • ファイル施錠 (ロック) モード (k) 345 • リンクモード (l) 345 • リンク対 345 • 任意指定の `allow` および `file` ルール 346 • 所有者条件ルール 347 • 拒否ルール 348

31.8 マウントルール 348

31.9 ピボットルートルール 350

31.10 PTrace ルール 350

31.11 シグナルルール 351

31.12 実行モード 352
専用プロファイル実行モード (px) 352 • 専用ローカルプロファイル実行モード (cx) 353 • 無制限実行モード (ux) 353 • 安全ではない実行モード 353 • 継承実行モード (ix) 354 • 実行可能なマッピングの許可 (m) 354 • 名前付きプロファイル遷移 354 • プロファイル遷移に対するフォールバックモード 355 • 実行モード内での変数設定 356 • `safe` キーワードおよび `unsafe` キーワード 357

31.13 リソース制限制御 357

31.14 監査ルール 359

32 AppArmor のプロファイルリポジトリ 360

33 YaST を利用したプロファイルの構築と管理 361

33.1 プロファイルの手動追加 361

33.2 プロファイルの編集 362
項目の追加 363 • 項目の編集 366 • 項目の削除 366

33.3 プロファイルの削除 367

33.4 AppArmor の管理 367
AppArmor の状態変更 368 • 個別のプロファイルに対するモード変更 368

34 コマンドラインからのプロファイル構築 370

34.1 AppArmor の状態確認 370

34.2 AppArmor プロファイルの構築 371

34.3 AppArmor プロファイルの追加と作成 372

34.4 AppArmor プロファイルの編集 372

34.5 未知の AppArmor プロファイルの読み込み解除 373

34.6 AppArmor プロファイルの削除 373

34.7 2 種類のプロファイル作成方法 373
単独プロファイル作成 374 • 一括プロファイル作成 375 • プロファイル作成ツールの概要 376

34.8 主なファイル名とディレクトリ 397

35 チェンジハット機能による Web アプリケーションのプロファイル作成 399

35.1 mod_apparmor を利用するための Apache の設定 400
仮想ホストのディレクティブ 401 • Location ディレクティブと Directory ディレクティブ 401

35.2 チェンジハット対応のアプリケーションの管理 402
AppArmor のコマンドラインツールによる方法 402 • YaST によるハットと項目の追加方法 408

36 pam_apparmor によるユーザの制限 410

37 プロファイルを作成したアプリケーションの管理 411

37.1 アクセス拒否イベントへの対応 411

37.2 セキュリティプロファイルの管理 411
セキュリティプロファイルのバックアップ 411 • セキュリティプロファイルの変更 412 • お使いの環境に対する新規ソフトウエアの導入 412

38 サポート 413
38.1 AppArmor のオンライン更新 413
38.2 マニュアルページの使用 413
38.3 さらなる情報 415
38.4 トラブルシューティング 415
おかしなアプリケーション動作に対する対応 415 • プロファイルが期待通りに動作しない場合の対応 415 • Apache での問題解決 419 • 使用するプロファイルの一覧から特定のプロファイルを除外する方法 419 • システムにインストールされていないアプリケーションのプロファイル管理方法 419 • AppArmor の文法エラーの検出および修正方法 420
38.5 AppArmor のバグ報告 421
39 AppArmor 用語集 422
V SELINUX 425
40 SELinux の設定 426
40.1 なぜ SELinux を使用するのか 426
サポート状態 427 • SELinux のコンポーネント 427
40.2 SELinux ポリシーの概要 428
40.3 SELinux パッケージのインストール 429
40.4 SELinux ポリシーのインストール 429
40.5 SELinux での許容 (permissive) モードへの移行 430
40.6 SELinux での強制 (enforcing) モードへの移行 431
40.7 SELinux の設定 432
40.8 SELinux の管理 433
セキュリティコンテキストの表示 433 • SELinux モードの選択 435 • SELinux のコンテキストタイプの変更 436 • ファイルコンテキストの適用 438 • SELinux ポリシーの設定 439 • SELinux モジュールの使用 440
40.9 トラブルシューティング 441

VI LINUX 監査フレームワーク 445

41 Linux 監査システムの概要 446

41.1 Linux 監査システムのコンポーネント紹介 448

41.2 監査デーモンの設定 450

41.3 `auditctl` による監査システムの制御 451

41.4 監査システムに対するパラメータ指定 454

41.5 監査ログの仕組みとレポートの生成 457
監査ログの仕組み 457 • 独自の監査レポートの生成 463

41.6 `ausearch` による監査デーモンログへの問い合わせ 470

41.7 `autrace` によるプロセスの解析 473

41.8 監査データの可視化 474

41.9 監査イベント通知の中継 476

42 Linux 監査フレームワークの設定 479

42.1 監査対象のコンポーネントの決定 480

42.2 監査デーモンの設定 480

42.3 システムコールに対する監査の有効化 482

42.4 監査ルールの設定 483

42.5 監査レポートの設定 485

42.6 ログの可視化の設定 489

43 監査ルールセットの紹介 491

43.1 基本的な監査設定パラメータの追加 492

43.2 監査ログファイルと設定ファイルに対する監視の追加 492

43.3 ファイルシステムオブジェクトの監視 493

43.4 セキュリティ設定ファイルとデータベースの監視 495

43.5 その他のシステムコールに対する監視 497

43.6 システムコールのパラメータでのフィルタリング 497
43.7 キーを利用した監査イベントレコードの管理 500
44 その他の情報源 502
A GNU ライセンス 504

前書き

改訂履歴

2023-03-16

1 利用可能なドキュメンテーション

オンラインドキュメンテーション

ドキュメンテーションは https://doc.opensuse.org で公開されています。ここから直接読むこともできますし、様々な形式でダウンロードを行うこともできます。



注記: 最新の更新について

ドキュメンテーションの最新版は通常、英語版が先に作成されます。

SUSE ナレッジベース

何らかの問題に直面した場合は、https://www.suse.com/support/kb/ からアクセスできる技術情報文書 (TID) をご確認ください。ここからお客様からの問い合わせで発生した、様々な SUSE 社の知識情報が検索できます。

お使いのシステム内

オフライン環境での利用を想定して、お使いのシステム内の `/usr/share/doc/release-notes` ディレクトリにはリリースノートが保存されているほか、各パッケージに対するドキュメンテーションが `/usr/share/doc` 内に存在しています。
また、多くのコマンドに対して、マニュアルページ も用意されています。マニュアルページは `man` コマンドで表示することができます。このコマンドの後ろに参照したいコマンド名を指定してください。なお、`man` コマンドがインストールされていない場合は、`sudo zypper install man` を実行してインストールしてください。

2 ドキュメンテーションの改善

このドキュメンテーションに対するご意見だけでなく、改善や追記などの貢献をいただければ幸いです。それぞれ下記のチャンネル経由でお送りいただくことができます:

バグ報告

ドキュメンテーション内に誤記などを見つけた場合は、https://bugzilla.opensuse.org/ で問題を報告してください。

なお、この文書の HTML 版の各章には、問題を報告するための［Report a bug］というアイコンが用意されていますので、こちらをお使いのうえ報告をお願いいたします。これにより、Bugzilla で対象の製品や分類、そしてリンク先などをそれぞれ自動で設定するようになっています。あとは問題点の説明を記述するだけです。
なお、報告には Bugzilla のアカウントが必要となるほか、英語でのやり取りが必要となりますので、あらかじめご了承ください。

貢献

このドキュメンテーションに対して追記もしくは更新すべき具体的な内容をお持ちの場合は、この文書の HTML 版の各章に用意されている［EDIT SOURCE］のリンクをお使いください。これにより、GitHub 内にある 英語版原文の ソースコードを表示し編集することができますので、作業が終わり次第 pull request を送信してください。
なお、GitHub のアカウントが必要となります。



注記: ［Edit Source］は英語版のみに提供される件について

［EDIT SOURCE］のリンクは各文書の英語版原文を編集する目的でのみご利用いただけます。その他の言語については［Report a bug］でお知らせください。

なお、本文書で使用しているドキュメンテーション環境の情報については、リポジトリ内の README をお読みください。

電子メール

本製品のドキュメンテーションに対するフィードバックは、`doc-team@suse.com` でも受け付けております。なお、文書のタイトルと製品のバージョン、およびドキュメンテーションの発行日付をそれぞれご記入ください。また、問題点の報告や記述の追加に関するご提案は、それぞれ概要と対応するセクション番号、およびページ (もしくは URL) をご記入ください。

ヘルプ

openSUSE Leap に対するさらなる支援をご希望の場合は、https://ja.opensuse.org/Portal:Support をご覧ください。

3 文書規約

この文書内では、下記のような記述ルールを使用しています:

- `/etc/passwd` : ディレクトリ名やファイル名を示しています
- `PLACEHOLDER` : `PLACEHOLDER` の箇所は、実際の値に置き換えるべきものであることを示しています

- `PATH` : 環境変数であることを示しています
- `ls` , `--help` : コマンドやオプション、パラメータであることを示しています
- `user` : ユーザ名やグループ名であることを示しています
- **`パッケージ名`** : ソフトウエアのパッケージ名を示しています
- `Alt` , `Alt`–`F1` : キー入力や組み合わせキー入力を示しています; キーはキーボードに書かれているとおりに大文字で示されます
- [ファイル] , [ファイル] › [名前を付けて保存] : メニュー項目やボタンなどを示しています
- 第1章「章のタイトル」: 本ガイド内の他の箇所への参照を示しています。
- 下記は `root` ユーザの権限で実行しなければならないコマンドを示しています。一般ユーザから実行する場合は、これらのコマンドの前に `sudo` を付けることで、`root` で実行できるようになります:

```
# コマンド
> sudo コマンド
```

- 下記は一般ユーザで実行できるコマンドを示しています:

```
> コマンド
```

- また、行末にバックスラッシュ文字 (`\`) を付けることで、コマンドを複数行に分けて記述している場合もあります。バックスラッシュ文字は、シェルに対して、これ以降にもコマンドが続くことを示す文字になります:

```
> echo a b \
c d
```

- このほか、行頭にプロンプトが書かれたコマンド行に続いて、そのコマンドを実行した場合の出力例を示す場合もあります:

```
> コマンド
出力
```

- 各種の情報について



警告: 警告

実際に実施したりする前に、注意しておかなければならない、きわめて重要な情報を記述しています。セキュリティ面の問題のほか、データを失ってしまう可能性への告知、ハードウエアの損傷の可能性や物理的な障害が発生する可能性を示しています。



重要: 重要な情報

実際に実施する前に注意すべき点を説明しています。



注記: 一般的な情報

一般的な補足情報を示しています。たとえばソフトウエアバージョン間での違いなどを説明しています。



ヒント: その他のヒント

ガイドラインや実践的なアドバイスなど、ヒントとなる情報を示しています。

- 簡潔な補足情報



一般的な補足情報を示しています。たとえばソフトウエアバージョン間での違いなどを説明しています。



ガイドラインや実践的なアドバイスなど、ヒントとなる情報を示しています。

1 セキュリティと機密保持

改訂履歴

2024-06-26

本章では、コンピュータセキュリティにおける基本的な考え方を説明しています。また、現実的に発生しうる脅威のほか、脅威からコンピュータを保護するための緩和策なども説明しています。このほか、さらに詳しい情報を得るための他の章やガイド、Web サイトに対する参照も示しています。

1.1 概要

Linux の主な特長のうちの 1 つとして、1 台のコンピュータ内で複数のユーザを同時に扱うことができ (マルチユーザ対応) 、かつ同時に複数の処理を行うことができる (マルチタスク対応) ことがあげられます。また、ユーザにとってはローカルに保存されているデータも、ネットワーク内に保存されているデータも、作業を行うにあたっては違いが無いという点もあります。

このようにマルチユーザ対応の仕組みであることから、それぞれのユーザが保存する情報は別々に管理されなければならず、かつセキュリティとプライバシーの両方を保証しなければならないことになります。それと同等に重要な項目として、ハードディスクなどのデータメディアを紛失してしまったりした場合や、損傷を受けてしまったような場合でも、できる限りデータを保護し、正当な利用者からは読み込め、それ以外の利用者からは読み込めないような仕組みも必要となります。

本章では、主に機密保持とプライバシーに主眼を置いて説明しています。ところが、このようなセキュリティの考え方は日を追うごとに進化するものであるばかりか、場合によっては現実的ではなくなってしまうこともあります。また、バックアップの存在も重要となります。バックアップを採取せずに進めてしまうと、データが改ざんされた場合やハードウエア障害などで壊れてしまった場合、その状態からのデータの復元は難しくなります。

セキュリティを維持するにあたっては、多層防御 の考え方を取り入れておくことをお勧めします。それぞれの脅威に対する緩和策は、お使いのシステムやデータを完璧に保護するものではなく、多数の階層でそれぞれ保護を実施することで、攻撃をより困難にする、という考え方です。このような多層防御の考え方では、下記のような仕組みを使用します:

- パスワードのハッシュ化 (例: PBKDF2, bcrypt, scrypt) とソルト化
- データの暗号化 (AES など)
- ログ記録、監視、侵入検知
- ファイアウオール

- アンチウイルススキャナ
- 緊急時の対応手順の策定と文書化
- バックアップ
- 物理セキュリティ
- 監査、セキュリティスキャン、侵入テスト

openSUSE Leap では、上述のリストの要件に適合する様々なソフトウエアを提供しています。下記の章では、システムを保護するための出発点となる情報を提供しています。

1.2 パスワード

Linux システムでは、ハッシュ化されたパスワードのみが保存されています。ハッシュ化とは一方向関数とも呼ばれ、元のデータに戻すのは非常に難しいデジタル指紋を生成する関数を意味します。

ハッシュ化されたパスワードは、一般ユーザでは読み込めない `/etc/shadow` というファイル内に保存されます。これは、非常に高性能なコンピュータを使用すれば元のデータを探り当てることができてしまうため、一般ユーザにはアクセスできないようにしておく必要があるためです。

また、National Institute of Standards and Technology (NIST) ではパスワードに対するガイドラインを発行しています。詳しくは https://pages.nist.gov/800-63-3/sp800-63b.html#sec5 (OpenID Foundation Japan による日本語訳: https://openid-foundation-japan.github.io/800-63-3/sp800-63b.ja.html) をお読みください。

パスワードポリシーの設定方法について、詳しくは 18.3項「[パスワード設定]」をお読みください。Linux における認証方法についての詳細は、パートI「認証」をお読みください。

1.3 バックアップ

システム内に不正に侵入されて破壊されてしまったような場合、バックアップを利用して以前の状態に戻すことができます。それ以外にも、バグや障害などでシステムが壊れてしまった場合にも、以前の状態と比較する目的でバックアップを使用することができます。本番環境のシステムの場合は、災害対策の目的で複数のオフサイトバックアップを用意しておくのが重要です。これはたとえば、テープなどの記録媒体にストレージのコピーを採取しておいて保管しておいたりなどの方法が考えられます。

また、法律上の理由により、企業や組織ではバックアップに多数の情報を記録しておき、それを長期に保管しなければならない場合もあります。お使いの環境内に古い文書の保管ルールが存在する場合は、そのポリシー内に Linux バックアップに関する考慮を加える必要があるかもしれません。

このほか、サーバの物理的なセキュリティも重要な要素ですし、バックアップを採取する場合には、暗号化を行う必要もあります。これらはいずれも個別に配慮して実施すべき内容になります。バックアップメディアを輸送中などに紛失してしまった場合、何も暗号化を行っていないと、データに不正アクセスされる危険性があります。もちろんバックアップシステムそのものへの不正侵入対策も必要です。いずれの場合にしても、バックアップを暗号化しておくことは、ある程度の機密保護になります。もちろん緊急事態の際には、正当な管理者が暗号を解読できなければならないことに注意する必要はあります。また、暗号鍵そのものも盗まれる可能性がありますから、暗号鍵そのものの置き換えも考慮しておく必要があります。

システムに対して不正侵入が発生した場合や、それが疑われる場合は、何よりもバックアップの整合性が確保されているかどうかを確認しておくことが重要です。システムへの不正侵入が長期間にわたって判明しなかったような場合、バックアップそれ自身も不正に書き換えられている可能性があり、不正な設定ファイルやプログラムが混入している可能性があるためです。いずれの場合にしても、十分な期間のバックアップを保持しておくことで、不用意な変更が発生していないかどうかを確認することができるようになります。

既知のセキュリティ違反が見つからなかった場合であっても、バックアップ内の設定ファイルの差分を定期的に確認しておくことで、設定ミスを含め様々なセキュリティ問題を検出できることになります。このようなアプローチは、ファイルや環境が頻繁に変更されるようなものでない限り、最適な方法です。

1.4 システムの完全性保護

コンピュータに対して物理的にアクセスができてしまうと、ファームウエアや起動処理に改ざんを加えることで、認可されたユーザがコンピュータを起動したタイミングで、不正アクセスができるようになってしまいます。もちろん全てのコンピュータを檻の中に閉じこめておくことはできませんが、コンピュータを保護する際に最初に考慮すべきことは、何よりもコンピュータをサーバルーム内などに施錠しておくことです。

また、古い環境を廃棄する際にも注意を払わなければなりませんし、ブートローダの安全性を高める手法やリムーバブルメディアの制限なども物理的なセキュリティ向上に役立ちます。物理的な機密保持について、詳しくは 第10章「物理的なセキュリティ」をお読みください。

このほか、下記のような点についても考慮する必要があります:

- リムーバブルデバイスから起動できないようにシステムを設定すること。
- UEFI パスワード, Secure Boot, GRUB2 パスワードをそれぞれ設定して、起動処理の安全性を高めること。
- Linux システムはブートローダと呼ばれるシステムを利用して起動しますが、ブートローダには Linux カーネルに対してオプションを指定する機能が用意されています。そのため、ブートローダに対してもパスワードを設定して、起動時に不正なパラメータを指定されたりしないようにする必

要があります。これはシステムのセキュリティを保護する際の重要な設定で、ブートローダはカーネルに対して影響を及ぼすことができることから、`root` の権限で動作するカーネルと同様に保護されなければならないためです。
ブートローダへのパスワード設定について、詳しくは『リファレンス』、第12章「ブートローダ GRUB 2」、12.2.6項「起動パスワードの設定」をお読みください。

- ハードディスクの暗号化を実施すること。詳しくは 第13章「パーティションやファイルの暗号化」をお読みください。
- `cryptctl` コマンドを利用して接続されているストレージの暗号化を行うこと。詳しくは 第14章「cryptctl を利用したアプリケーション向けのストレージ暗号化」をお読みください。
- AIDE を利用して、システムの設定変更を検知できるようにすること。詳しくは 第21章「AIDE を利用した侵入検知」をお読みください。

1.5 ファイルへのアクセス権

Linux では 全てのものをファイルとして扱う アプローチのもと作られている仕組みであるため、ほとんどの資源に対してアクセスを制御することのできるファイルアクセス権の設定は、セキュリティでは重要な要素となります。ファイルへのアクセス権は通常のファイルやディレクトリに対する設定だけでなく、ハードウエアデバイスに対するアクセスも設定することができます。既定では、ほとんどのハードウエアデバイスへのアクセスは `root` にのみ許可されていますが、シリアルポートなど一部のハードウエアに対しては、一般ユーザからアクセスできるものもあります。

一般的には、行うべき作業に対する最小限の権限のみを利用すべきです。たとえば電子メールの読み書きの作業程度であれば、`root` の権限は不要です。もしも電子メールプログラム内にバグが存在し、プログラムを実行しているのと同じ権限で不正なスクリプトを実行されてしまうようなものがあった場合、`root` で電子メールを読み書きしていると、`root` でプログラムが実行されてしまうことになるためです。一般ユーザで実行していれば、システムに対する被害は最小限に抑えられます。

詳しくは 20.1項「従来型のファイルアクセス権」と 20.2項「ACL の利点」をお読みください。

AppArmor と SELinux は、アプリケーションとユーザの両方に対して制限を課すことができる仕組みです。詳しくは パートIV「AppArmor による権限の制限」 および パートV「SELinux」をお読みください。

インストールされているオペレーティングシステム以外から、ハードディスクに対してアクセスできる可能性がある場合、たとえばライブ CD/DVD などを使用するシステムや、ハードウエアの取り外しができるシステムの場合は、データの暗号化を行うとよいでしょう。openSUSE Leap では、データが含まれているディスクに対しても、オペレーティングシステムが含まれているディスクに対しても、暗号化を行うことができます。詳しくは 第13章「パーティションやファイルの暗号化」をお読みください。

1.6 ネットワーク

ネットワークサービスの機密保護は重要です。OSI モデル で規定されている各レイヤに対して、できる限りの保護を実施しておくことをお勧めします。

全ての通信には認証を求め、最新の技術を利用してトランスポート層もしくはアプリケーション層で暗号化しておくことをお勧めします。また、必要に応じて Virtual Private Network (VPN; 仮想プライベートネットワーク) を構築して、物理ネットワークに対する追加保護を実施してもかまいません。

openSUSE Leap には、ネットワークを保護するための様々な仕組みが用意されています:

- X509 形式の証明書を作成するには、`openssl` を使用します。X509 形式の証明書は数多くのサービスで使用され、暗号化と証明の両方の目的で使われます。必要であれば、独自の 証明機関 (CA) を設立して、ネットワーク内の証明インフラを構築することもできます。詳しくは `man openssl` をお読みください。
- 少なくともネットワーク通信の一部はインターネット側に公開されます。そのため、ファイアウオールで意図しないポートを閉じたり、不要なサービスをアンインストールもしくは無効化したりするなどして、攻撃の足がかりとなりうる箇所を減らすことも行います。こちらについて詳しくは、第24章「マスカレードとファイアウオール」をお読みください。
- OpenVPN を使用することで、信頼のできない物理ネットワークを介して安全な通信手段を構築することもできます。詳しくは 第25章「VPN サーバの設定」をお読みください。
- ネットワークサービスに対しては、強固な認証を設定してください。詳しくは パートI「認証」をお読みください。

1.7 ソフトウエアの脆弱性

ソフトウエアの脆弱性は、認可されていないアクセス手段を取得したり、システムの動作を不安定にさせたりすることのできる、悪用可能なソフトウエアの問題を意味します。脆弱性は特に、HTTP サーバなどのリモートサービスに影響があるような場合、特に重要となります。コンピュータシステムは複雑なものであるため、常に脆弱性が存在しうるものであると考える必要があります。

このような脆弱性が既知のものになると、ソフトウエアの開発者がソフトウエアを修正して、穴を塞がなければならなくなります。その結果公開される更新は、システム管理者が影響のあるシステムに対して、素早く安全に適用しなければなりません。

脆弱性はアメリカ政府が管理している National Vulnerability Database (NVD; 国家脆弱性情報データベース) や JPCERT/CC と情報処理推進機構 (IPA) が管理している Japan Vulnerability Notes (JVN) で情報公開されます。これらの団体が提供する情報を購読することで、新しく発見された脆弱性に関する詳細を知ることができます。ただし、場合によってはソフトウエアの更新が脆弱

性情報の公開に間に合わないこともあります。また、脆弱性には Common Vulnerabilities and Exposures (CVE) 番号や Common Vulnerability Scoring System (CVSS) スコアが割り当てられます。CVSS のスコアは脆弱性の緊急度を表す値です。

SUSE では、セキュリティ勧告のフィードを提供しています。詳しくは https://www.suse.com/ja-jp/support/update/ をご覧ください。また、CVE 番号によるセキュリティ更新の一覧も提供しています。こちらについては https://www.suse.com/support/security/ をご覧ください。

一般的には、管理者は自分自身が管理するシステムに対して、脆弱性への備えを行っておくべきです。これには、システムに属する全てのコンピュータをできる限り強固にしておくことも含まれます。また、致命的な脆弱性が見つかった場合、即時に更新をインストールするために、必要な手順をあらかじめ策定しておくこともお勧めしています。

発生しうる攻撃からの被害を最小限に抑えるため、ファイルのアクセス権は厳格に設定しておくことをお勧めします。詳しくは 20.1項「従来型のファイルアクセス権」をお読みください。

その他の便利なリンク:

- http://lists.opensuse.org/opensuse-security-announce/: openSUSE セキュリティ告知のメーリングリスト (英語)
- https://nvd.nist.gov/: 国家脆弱性情報データベース (英語)
- https://cve.mitre.org/: MITRE's CVE データベース (英語)
- https://jvn.jp/: Japan Vulnerability Notes (日本国内向けの脆弱性情報)
- https://www.first.org/cvss/: CVSS のスコアシステムについての情報

1.8 マルウエア

マルウエア とはコンピュータの正常な動作を妨げたり、データを盗み出したりすることを目的にしたソフトウエアのことを意味します。これにはウイルスのほか、ワームやランサムウエア、ルートキットなどが含まれます。マルウエアによっては、ソフトウエアの脆弱性を利用するものもあります。ただしこれらのマルウエアは、不明な提供元から公開されたソフトウエアをユーザが実行してしまうことによって組み込まれてしまうことがほとんどです。openSUSE Leap ではダウンロードリポジトリを通じて、様々なプログラム (パッケージ) を公開しています。SUSE が提供する全てのパッケージには電子署名が付与されていますので、openSUSE Leap のパッケージマネージャでインストールを行う際、その署名を確認することによって、不正なソフトウエアが混入しないようにしています。

たとえば `rpm --checksig RPM_ファイル` のように実行すると、チェックサムとパッケージの署名をチェックすることができます。また、openSUSE Leap のメディアには証明書 (署名鍵) が付属していますし、世界中の鍵サーバにも登録が行われています。

このほか、ClamAV アンチウイルスソフトウエアを使用することで、お使いのシステム内にマルウエアが存在しているかどうかを調べることもできます。ClamAV はメールサーバや HTTP プロキシなど、いくつかのサービスに組み込むこともできます。これにより、ユーザにマルウエアが届くよりも前に、マルウエアを防ぐことができます。

また、ユーザに与える権限をできる限り少なくすることによっても、不用意なコード実行のリスクを減らすことができます。

1.9 主なセキュリティヒント

下記のヒントは、これまでの章で説明したことの概要となっています:

- 最新のセキュリティ問題に対して、常に情報を得ておきましょう。また、セキュリティ通知で推奨される更新パッケージについては、できる限り素早くインストールしておきましょう。
- できる限り `root` 権限の使用は避けましょう。また、ファイルのアクセス権限もできるだけ厳格に設定しましょう。
- ネットワーク上で通信を行う際は、必ず暗号化を行いましょう。
- 間違いなく必要となるネットワークサービス以外は、無効化しておきましょう。
- 定期的なセキュリティ監査を受けましょう。たとえば開いているネットワークポートの調査などを行いましょう。
- `AIDE` (Advanced Intrusion Detection Environment) を利用して、お使いのシステムの完全性を監視しましょう。
- サーとパーティ製のソフトウエアをインストールする場合は、特に注意して作業を行いましょう。
- 全てのバックアップは定期的に採取しましょう。
- logwatch などでログファイルは定常的に監視しましょう。
- 明示的に通さなければならないものを除いて、それ以外のポートはファイアウオールで閉じておきましょう。
- セキュリティの仕組みは多階層で実施し、1 つが突破されても別の仕組みで防げるようにしましょう。
- できる限り暗号化を使用しましょう。たとえばモバイルコンピュータであれば、ハードディスクの暗号化を行いましょう。

1.10 セキュリティ問題の報告

何らかのセキュリティ問題を発見したら、まずはソフトウエア更新が提供されていないかどうかをご確認ください。その問題に対して何も更新が提供されていない場合は、`security@suse.de` 宛に電子メールにてご連絡をお願いいたします。その際には、問題の詳細な説明と、対象となるパッケージのバージョン情報も添付してください。また、電子メールは GPG で暗号化しておくことをお勧めします。

SUSE 社の GPG 鍵の最新版は、https://www.suse.com/support/security/contact/ で公開しています。

I 認証

2 PAM を利用した認証 10
3 NIS の使用 23
4 YaST を利用した認証クライアントの設定 31
5 389 Directory Server を利用した LDAP サービス 33
6 Kerberos を利用したネットワーク認証 81
7 Himmelblau を利用した Microsoft Entra ID による認証 109
8 Active Directory サポート 112
9 FreeRADIUS サーバの構築 132

2 PAM を利用した認証

改訂履歴

2024-03-14

Linux では認証処理に際して PAM (Pluggable Authentication Modules; プラグインを使用することのできる認証モジュール) を利用し、ユーザとアプリケーションとの間の仲介を行っています。PAM のモジュールはシステム全体に対して提供されているため、様々なアプリケーションから利用できるようになっています。本章では、このようなモジュール型の認証機構の動作と、設定方法について説明しています。

2.1 PAM とは?

システム管理者やプログラマは、システムの特定の箇所や、アプリケーション内の特定の機能に対してアクセスを制限したいと考えることがあります。PAM の無い状態では、それぞれのアプリケーションは LDAP や Samba, Kerberos など、様々な認証機構に対応しなければならなくなります。このような開発には時間を要するばかりか、開発によってエラーを引き起こす可能性すらあります。このような欠点を防ぐ唯一の方法は、認証の仕組みとアプリケーションを分離するという方法で、集中管理されたモジュール内で、認証の仕組みを代行させるやり方です。このような仕組みであれば、新しい認証方式が開発された場合でも、対応する PAM モジュール を作成すれば、アプリケーション側ですぐに利用できるようになります。

PAM は下記のような考え方から成り立っています:

- PAM モジュール: 特定の認証機構に対応する共有ライブラリです。
- モジュールスタック: 複数の PAM モジュールから構成されるものです。
- PAM 対応の サービス: モジュールスタック、もしくは PAM モジュールを利用して、認証の仕組みを利用するものです。通常、サービスは `login` や `su` のように、対応するアプリケーションに対する分かりやすい名前を指定します。サービス名 `other` は、既定のルールで使用される予約語です。
- モジュールのパラメータ: 特定の PAM モジュールに対して、その処理に影響を与えるためのものです。
- 結果: それぞれの PAM モジュールの実行結果を意味します。正の値が返された場合、次の PAM モジュールを実行します。負の値は設定によって異なる動作となり、「何も影響を与えず、そのまま次のモジュールに進む」から「即時に終了する」まで、様々な処理を行うことができます。

2.2 PAM 設定ファイルの構造

PAM は下記の 2 種類の方式で設定することができます:

ファイルベースの設定 (`/etc/pam.conf`)
: それぞれのサービスに対する設定を `/etc/pam.conf` に書き込む方式です。この方式は、メンテナンス性と利便性が損なわれることから、openSUSE Leap では使用していません。

ディレクトリベースの設定 (`/etc/pam.d/`)
: PAM の仕組みを利用する各サービス (もしくはプログラム) に対する設定を、`/etc/pam.d/` ディレクトリ内のそれぞれの設定ファイルに保存する方式です。たとえば `sshd` サービスに対する設定は、`/etc/pam.d/sshd` ファイル内に書き込むことになります。

`/etc/pam.d/` ディレクトリ内のファイルでは、認証に使用する PAM モジュールを設定します。ファイル内の各行には最大で 4 つの値を指定します:

```
種類　制御
  モジュールパス　モジュールパラメータ
```

それぞれの意味は下記のとおりです:

`種類`
: PAM モジュールの種類を指定します。PAM モジュールはスタックとして処理されます。それぞれの種類はそれぞれ別々の目的で使用します。たとえば一方のモジュールではパスワードをチェックし、他方のモジュールではシステムにアクセスする際のアクセス元をチェックし、さらに別のモジュールではユーザ固有の設定を読み込む、などです。PAM では、下記 4 つのモジュールの種類に対応しています:

  `auth`
  : 従来であればパスワードの問い合わせなど、ユーザの正当性を確認するための種類です。現在では IC カードによる認証や生体認証 (指紋認証、虹彩認証など) も使用することができます。

  `account`
  : そのサービスを使用できる権限があるかどうかを確認するための種類です。たとえば有効期限の切れたユーザ名でログインできないようにする、などの目的で使用します。

  `password`
  : 認証トークン (通常はパスワード) の変更を有効化するための種類です。

`session`

ユーザセッションを管理したり設定したりするための種類です。ログインに際しては認証前と認証後の両方で開始され、ユーザ固有の環境 (メールアカウント、ホームディレクトリ、システム制限など) を設定するために使用します。

制御

PAM モジュールの動作を表す箇所です。それぞれのモジュールには、下記のような制御フラグを設定することができます:

`required`

このフラグを設定したモジュールは、認証を成功させるにあたって必ず成功しなければならないものであることを表します。`required` フラグを設定したモジュールが失敗すると、同じフラグを設定した他のモジュールは、ユーザが認証失敗メッセージを受け取る前に処理が行われます。

`requisite`

このフラグを設定したモジュールは、`required` フラグと同様に処理が成功しなければならないものであることを表します。ただし、このフラグを設定したモジュールが失敗すると、ユーザに対して即時にフィードバックが送られ、その他のモジュールは処理されなくなります。成功した場合は、`required` フラグを設定した場合と同様に、その他のモジュールは引き続き処理されます。`requisite` フラグは、認証を成功させるために必要な条件をチェックするための基本フィルタとして動作します。

`sufficient`

このフラグを設定したモジュールの処理が成功すると、それより前に存在する `required` フラグ付きのモジュールの処理が失敗しない限り、サービスは成功を示すメッセージを即時に受け取り、その他のモジュールが処理されなくなります。なお、`sufficient` フラグが設定されたモジュールの処理が失敗しても、直接的な結果には至らず、残りのモジュールは通常通り処理されます。

`optional`

このフラグを設定したモジュールの処理は、成功した場合も失敗した場合も、直接的な影響を与えません。このフラグは、メッセージ (たとえばメールが到着しているなど) を表示し、それ以外の処理を行わないモジュールを使用する場合に有用です。

`include`

このフラグを指定した場合、この位置に指定した追加設定を取り込むことを表します。

モジュールパス

PAM モジュールに対する完全なファイル名を指定します。モジュールが既定のディレクトリ (32 ビット版の場合は `/lib/security`, 64 ビット版の場合は `/lib64/security`) 内に存在している場合、ディレクトリを指定する必要はありません。

モジュールパラメータ

PAM モジュールの動作に影響を与えるための、スペース区切りのパラメータリストです。たとえば `debug` (デバッグの有効化) や `nullok` (空のパスワードを許可する) などがあります。

これらに加えて、`/etc/security` ディレクトリ内には PAM モジュール向けのグローバルな設定ファイルが用意されています。このディレクトリ内には、モジュールに対する動作設定が書かれています (`pam_env.conf` や `time.conf` など) 。PAM モジュールを使用する各アプリケーションは、PAM の機能を実際に呼び出したあと、様々な設定ファイルに含まれる情報を処理して、結果を返します。

PAM モジュールの作成やメンテナンスを簡略化する目的で、`auth`, `account`, `password`, `session` の各種類に対応する一般的な既定の設定ファイルが用意されています。これらは、各アプリケーション側の PAM 設定から読み込まれるようになっているため、これらの設定ファイルが更新されると、それぞれの PAM 設定ファイルを個別に書き換えることなく、各アプリケーションに適用できるようになっています。

このような PAM のグローバル設定ファイルは、`pam-config` ツールを利用して管理します。このツールは設定ファイルに新しいモジュールを自動的に追加したり、既存の設定ファイルを変更したり、既存の設定ファイルからモジュール (もしくはオプション) を削除したりすることができます。このような仕組みにより、PAM の設定ファイルを手作業で管理する手間が最小化できているばかりか、場合によっては全く不要にもなっています。



注記: 64 ビットアプリケーションと 32 ビットアプリケーションの混在する環境について

64 ビットのオペレーティングシステムをお使いの場合、32 ビットアプリケーション向けのランタイム環境をそろえることもできます。この場合、32 ビット版の PAM モジュールを追加でインストールする必要があります。

2.3 sshd の PAM 設定

sshd の PAM 設定ファイルを例として取り上げてみます:

例 2.1: SSHD の PAM 設定 (/etc/pam.d/sshd)

```
#%PAM-1.0 ❶


auth     requisite      pam_nologin.so                          ❷
auth     include        common-auth                             ❸
account  requisite      pam_nologin.so                          ❷
account  include        common-account                          ❸
password include        common-password                         ❸
session  required       pam_loginuid.so                         ❹
session  include        common-session                          ❸
session  optional       pam_lastlog.so   silent noupdate showfailed ❺
```

❶ 設定ファイルのバージョンを表す文字列です。慣習として書かれているだけの項目ですが、将来のバージョンでも使用できるよう、必ず記述しておくことをお勧めします。

❷ `/etc/nologin` ファイルが存在するかどうかをチェックするモジュールの指定です。存在した場合、`root` 以外のユーザがログインできなくなります。

❸ 4 種類のモジュールタイプに対して、それぞれ設定ファイルを参照しています (`common-auth` , `common-account` , `common-password` , `common-session`) 。これらの 4 種類のファイルには、それぞれのモジュールの種類に対応する既定の設定が含まれています。

❹ 認証を行ったプロセスに対して、ログイン UID プロセス属性を設定します。

❺ ユーザの最終ログイン情報を表示します。

それぞれの PAM モジュールに対する設定ファイルに個別に追加していく代わりに、他の設定ファイルを取り込むように指定することで、管理者側が既定値を変更するだけで、その設定を取り込み元のモジュールに反映させることができるようになります。従来は PAM に対して変更があったり、新しいアプリケーションをインストールしたりするたびに、全ての設定ファイルを手作業で調整しなければなりませんでした。現在はこのような仕組みになっているため、設定の変更を一箇所で行うことができるようになっていて、各サービスの PAM 設定にもそのまま反映できるようになっています。

最初の `include` では `common-auth` という設定ファイルを取り込むように指定していますが、ここでは `auth` の種類に対して、3 種類のモジュール (`pam_env.so` , `pam_gnome_keyring.so` , `pam_unix.so` を呼び出しています。詳しくは 例2.2「auth セクションに対する既定の設定 (common-auth)」) をご覧ください。

例 2.2: auth セクションに対する既定の設定 (common-auth)

```
auth  required  pam_env.so                    ❶
auth  optional  pam_gnome_keyring.so         ❷
auth  required  pam_unix.so  try_first_pass  ❸
```

❶ pam_env.so は /etc/security/pam_env.conf ファイルを読み込んで、そのファイルに指定されているとおりに環境変数を設定します。これは、pam_env ではログインが行われた場所に関する情報を保持していることから、DISPLAY 環境変数を適切な値に設定するために使用されています。

❷ pam_gnome_keyring.so では、GNOME 鍵リングに対してユーザのログイン名とパスワードをチェックします。

❸ pam_unix は、ユーザのログイン名とパスワードを、/etc/passwd および /etc/shadow でチェックします。

auth 内の全てのモジュールは、sshd 側でログインが成功したかどうかについて、結果を得る前に処理が行われます。また、制御フラグ required が設定された全てのモジュールは、sshd が肯定応答を受け取るよりも前に成功していなければなりません。これらのうちのいずれかのモジュールが成功以外の状態になると、残りのモジュールはそのまま処理されるものの、最終的には sshd が否定応答を受け取ることになります。

auth の種類が設定された全てのモジュールが成功すると、その他のステートメントが処理されるようになります。この場合は 例2.3「account セクションに対する既定の設定 (common-account)」で示されている内容になります。この common-account には 1 つのモジュール (pam_unix) のみが書かれています。この pam_unix は対象のユーザが存在しているかどうかを返すもので、sshd が成功を表すメッセージを受け取ると、さらに続くモジュールスタック (この場合は 例2.4「password セクションに対する既定の設定 (common-password)」に示されている password) を実行することになります。

例 2.3: account セクションに対する既定の設定 (common-account)

```
account  required  pam_unix.so  try_first_pass
```

例 2.4: password セクションに対する既定の設定 (common-password)

```
password  requisite  pam_cracklib.so
password  optional   pam_gnome_keyring.so  use_authtok
password  required   pam_unix.so  use_authtok nullok shadow try_first_pass
```

sshd の PAM 設定では、common-password のファイル名で指定されている password モジュール向けの既定の設定のみが include で設定されています。ここには、アプリケーションが認証トークンの変更を要求した場合、必ず成功しておかなければならないもの (制御フラグ requisite および required) が書かれています。

パスワードなどの認証トークンを変更する際には、セキュリティチェックを必要とします。セキュリティチェックは `pam_cracklib` モジュールで行います。`pam_unix` モジュールはその後に動作するモジュールで、`pam_cracklib` から古いパスワードと新しいパスワードを受け取ります。これにより、パスワードの変更に際しても、ユーザ側で再度認証を行う必要が無いようにしています。また、このような仕組みにより、`pam_cracklib` 側のチェックは回避できないようにもなっています。`account` や `auth` の種類でパスワードの有効期限を確認するよう設定する場合は、`password` モジュールも使用する必要があります。

例 2.5: `session` セクションに対する既定の設定 (`common-session`)

```
session  required  pam_limits.so
session  required  pam_unix.so  try_first_pass
session  optional  pam_umask.so
session  optional  pam_systemd.so
session  optional  pam_gnome_keyring.so auto_start only_if=gdm,gdm-password,lxdm,lightdm
session  optional  pam_env.so
```

最後のステップとして、`session` の種類が指定されたモジュール (`common-session` ファイル内に書かれています) を呼び出して、対象のユーザの設定に従ってセッションを設定します。`pam_limits` モジュールは `/etc/security/limits.conf` ファイルを読み込むモジュールで、特定のシステム資源の使用に対して制限を設定することができるものです。ここでは再度 `pam_unix` モジュールを使用しています。`pam_umask` モジュールは、ファイル作成時のマスクを設定するためのモジュールです。このモジュールには `optional` フラグが設定されているため、このモジュールの処理が失敗しても、セッションモジュールスタックの処理には影響を与えないようになっています。なお、`session` の種類が設定されたモジュールは、ユーザがログアウトする際にも再度呼び出されます。

2.4 PAM モジュールの設定

PAM モジュールによっては、設定ファイルが別途用意されているものがあります。設定ファイルは `/etc/security` 内に配置します。本章では、sshd の例に示されているモジュール (`pam_env.conf` および `limits.conf`) について、概要を説明しています。

2.4.1 pam_env.conf

`pam_env.conf` は、`pam_env` モジュールが呼び出された際に、対象のユーザに対して標準化された環境を設定するために使用するファイルです。このファイルでは、下記のような書式で環境変数を設定します:

```
変数名  [DEFAULT=値]  [OVERRIDE=値]
```

変数名

設定したい環境変数の名前を指定します。

[DEFAULT=<値>]

管理者側で設定したい既定値を指定します。

[OVERRIDE=<値>]

pam_env に対して問い合わせ、必要であれば既定値を上書きして設定する値を指定します。

pam_env の一般的な用途としては、DISPLAY 環境変数があげられます。これはリモートからログインが行われることで、常に変化する可能性があるものです。詳しくは 例2.6「pam_env.conf」 をご覧ください。

例 2.6: PAM_ENV.CONF

```
REMOTEHOST  DEFAULT=localhost          OVERRIDE=@{PAM_RHOST}
DISPLAY     DEFAULT=${REMOTEHOST}:0.0  OVERRIDE=${DISPLAY}
```

最初の行では環境変数 REMOTEHOST の値に対して、pam_env 側で判断が付かなかった場合、localhost を設定するよう指定しています。2 行目の DISPLAY 環境変数は、REMOTEHOST の値をもとに設定を行う意味です。詳しくは /etc/security/pam_env.conf 内のコメント (英語) をお読みください。

2.4.2 pam_mount.conf.xml

pam_mount は、ログインの処理中にユーザのホームディレクトリをマウントし、ログアウト時にマウントを解除するためのモジュールです。一般的には、中央のファイルサーバ内にユーザのホームディレクトリが存在しているような環境で使用するものです。この方式を使用することで、/home 全体を共有してマウントする必要がなくなり、ログインしているユーザのホームディレクトリのみをマウントして使用できるようになります。

pam_mount をインストールすると、/etc/security 内に pam_mount.conf.xml に対するテンプレートファイルが用意されるようになります。要素の説明について、詳しくは man 5 pam_mount.conf で表示されるマニュアルページをお読みください。

なお、YaST を利用することで、この機能に対する基本的な設定を行うことができます。[ネットワークサービス] › [Windows ドメインメンバーシップ] › [熟練者向け設定] を選択して、ファイルサーバを追加してください。

2.4.3 limits.conf

システムに対する制限は、`pam_limits` モジュールが読み込む `limits.conf` ファイルで設定することができます。制限はユーザ単位やグループ単位で設定することができます。このファイルでは、超過を許さないハード制限のほか、一時的であれば超過を許すソフト制限も設定することができます。文法とオプションについて、詳しくは `/etc/security/limits.conf` ファイル内のコメント (英語) をお読みください。

2.5 pam-config を利用した PAM の設定

`pam-config` ツールは、グローバルな PAM 設定ファイル (`/etc/pam.d/common-*`) やいくつかのアプリケーション設定を行う際の支援ツールとなるものです。対応するモジュールの一覧は、`pam-config --list-modules` を実行して得ることができます。また、`pam-config` コマンドは、お使いの PAM 設定ファイルを管理するためにも使用することができます。PAM 設定への新しいモジュールの追加や、既存のモジュールの削除、モジュールに対する設定の変更などを行うことができます。グローバルな PAM 設定ファイルを変更する場合は、個別のアプリケーションに対して PAM の設定を手作業で調整する必要はありません。

`pam-config` の使用例としては、下記のようなものがあります:

1. **新しい Unix 形式の PAM 設定の自動生成:** pam-config に対して、あとから追加や変更ができるよう、最もシンプルな設定を作成するように指示します。具体的には `pam-config --create` というコマンドを実行することで、シンプルな Unix 認証設定を作成することができます。pam-config で管理されていない設定が既に存在していた場合、設定は上書きされてしまいますが、バックアップコピーが `*.pam-config-backup` という形式で保存されます。

2. **新しい認証方式の追加:** 新しい認証方式 (たとえば SSSD) をお使いの PAM モジュールスタックに追加するには、`pam-config --add --sss` のようにコマンドを実行します。このコマンドを実行すると、全ての `common-*-pc` PAM 設定ファイルの該当箇所に対して SSSD が追加されます。

3. **テスト目的でのデバッグの有効化:** 新しい認証の仕組みが期待通りに動作しているかどうかを確認するには、全ての PAM 関連操作に対して、デバッグ機能を有効化する必要があります。たとえば `pam-config --add --sss-debug` のように実行すると、SSSD 関連の PAM 操作に対して、デバッグが有効化されます。デバッグ出力は、`systemd` のジャーナル (詳しくは『リファレンス』、第11章「`journalctl` : systemd ジャーナルへの問い合わせコマンド」をお読みください) 経由で読むことができます。

4. **設定の問い合わせ:** 新しい PAM の設定を適用する際には、必要な全てのオプションを指定しているかどうかを確認しておいてください。`pam-config --query --` モジュール名 コマンドを実行すると、使用されている種類の一覧と、モジュールに対して指定されているオプションを表示することができます。

5. **デバッグオプションの削除:** 設定に問題がないことを確認し終えたら、最後にデバッグオプションを削除します。たとえば `pam-config --delete --sss-debug` のように実行すると、たとえば `pam_ssh.so` に対して設定していたデバッグを無効化することができます。他のモジュールの場合も、同じようなコマンドで無効化することができます。

`pam-config` コマンドに対する詳しい使用方法と利用可能なオプションについて、詳しくは `pam-config(8)` のマニュアルページをお読みください。

2.6 PAM の手動設定

PAM の設定ファイルを手作業で作成もしくは管理したい場合は、これらのファイルに対する `pam-config` を無効化する必要があります。

`pam-config --create` コマンドを利用して独自の PAM 設定ファイルを手作業で作成する場合、`common-*` から `common-*-pc` ファイルに対するシンボリックリンクが作成されます。`pam-config` では `common-*-pc` の設定ファイルのみを修正しますので、これらのシンボリックリンクを削除することで、pam-config を効率的に無効化することができます。

警告: 設定内に pam_systemd.so を含める必要性について

独自の PAM 設定を作成する場合は、`pam_systemd.so` モジュールを `session optional` で指定するようにしてください。`pam_systemd.so` を含めるように指定しないと、`systemd` のタスク制限で問題が発生するようになります。詳しくは `pam_systemd.so` のマニュアルページをお読みください。

2.7 ローカルログインのための U2F キーの設定

ローカルログイン時にさらにセキュリティを高める方法として、`pam-u2f` フレームワークと YubiKey 等の U2F 機能を持つセキュリティキーを利用した 2 要素認証を設定することができます。

お使いの システムに対して U2F を設定するには、まず システムで鍵とアカウントの紐付けを行う必要があります。紐付け後は鍵を使用するように設定するだけです。手順はそれぞれ後続の章で説明しています。

2.7.1 アカウントと U2F キーの紐付け

U2F キーとアカウントの紐付けを行うには、下記の手順を行います:

1. まずは通常通りシステムにログインします。
2. U2F キーを接続します。
3. U2F キーの設定のためのディレクトリを作成します:

```
> sudo mkdir -p ~/.config/Yubico
```

4. `pamu2fcfg` コマンドを実行して、出力結果を設定として出力します:

```
> sudo pamu2fcfg > ~/.config/Yubico/u2f_keys
```

5. 接続したデバイスに搭載された LED が点灯したら、接続確認のためにデバイスの金属接点に触ってください。

なお、予備の U2F デバイスを用意して設定しておくことをお勧めします。下記のようなコマンドを実行します:

1. 下記を実行します:

```
> sudo pamu2fcfg -n >> ~/.config/Yubico/u2f_keys
```

2. 接続したデバイスに搭載された LED が点灯したら、接続確認のためにデバイスの金属接点に触ってください。

出力されたファイルは、既定の場所から `sudo` での許可が必要なディレクトリに移動させておくことで、さらにセキュリティを高めることができます。たとえば `/etc` ディレクトリ以下に移動したい場合は、下記のような手順を実施します:

1. まずは `/etc` 以下にディレクトリを作成します:

```
> sudo mkdir /etc/Yubico
```

2. あとは作成したファイルを移動するだけです:

```
> sudo mv ~/.config/Yubico/u2f_keys /etc/Yubico/u2f_keys
```

注記: u2f_keys ファイルを非標準の場所に配置する場合について

出力されたファイルを既定のディレクトリ（ `$HOME/.config/Yubico/u2f_keys` ）以外に移動させる場合、2.7.2項「PAM 設定の更新」に示された手順に従って、`/etc/pam.d/login` ファイル内にそのディレクトリを設定しておく必要があります。

2.7.2 PAM 設定の更新

U2F キーの設定が終わったら、あとはシステムの PAM 設定を調整するだけです。

1. まずは `/etc/pam.d/login` ファイルを開きます。
2. 下記のようにして `auth required pam_u2f.so` という行を追加します:

```
#%PAM-1.0
auth      include    common-auth
auth      required   pam_u2f.so
account   include    common-account
password  include    common-password
session   optional   pam_keyinit.so revoke
session   include    common-session
#session  optional   pam_xauth.so
```

3. なお、`$HOME/.config/Yubico/u2f_keys` 以外のディレクトリに `u2f_keys` ファイルを配置している場合は、`/etc/pam.d/login` ファイル内で、`authfile` オプションを指定する必要があります:

```
#%PAM-1.0
auth     requisite pam_nologin.so
auth     include   common-auth
auth  required pam_u2f.so authfile=<U2F_キーのディレクトリ>
...
```

ここで、`<U2F_キーのディレクトリ>` には、`u2f_keys` ファイルのあるディレクトリを絶対パスで指定します。

2.8 さらなる情報

`pam-doc` パッケージをインストールすると、`/usr/share/doc/packages/pam` ディレクトリ内に下記のような追加のドキュメンテーションが提供されるようになります:

README
: このディレクトリの直下には `modules` サブディレクトリがあり、ここには利用可能な PAM モジュールに関する説明 (英語) の書かれた README ファイルがあります。

The Linux-PAM System Administrators' Guide (Linux-PAM システム管理者ガイド)
: この文書には、システム管理者が PAM について知っておくべき内容を説明しています。設定ファイルの文法から、PAM のセキュリティ要素までの範囲を説明しています。

The Linux-PAM Module Writers' Manual (Linux-PAM モジュール開発者マニュアル)
: この文書には、開発者の観点からの内容を説明しています。具体的には、標準に準拠した PAM モジュールの作成方法などを説明しています。

The Linux-PAM Application Developers' Guide (Linux-PAM アプリケーション開発者マニュアル)
: この文書には、PAM ライブラリを使用したいアプリケーション開発者に対して、必要な情報が含まれています。

PAM マニュアルページ
: PAM の一般的な説明と、全てのコンポーネントの機能をよりよく説明するためのモジュール向けマニュアルページが用意されています。

3 NIS の使用

改訂履歴

2024-05-10

ネットワーク内に複数の Unix システムが存在していて、共通のリソースを使用するように設計している場合、それらの Unix システムでは同じユーザおよびグループの識別子を使用する必要が生じます。また、ネットワークはユーザにとって透過的な存在であるため、使用している Unix システムによって環境が異なるべきではありません。このような要件に対応するため、NIS と NFS のサービスを併用する環境をお勧めします。

NIS (Network Information Service) はデータベースのようなサービスであり、ネットワークを介して `/etc/passwd`，`/etc/shadow`，`/etc/group` の内容を提供することができます。NIS はそれ以外の目的 (たとえば `/etc/hosts` や `/etc/services` などのファイルで設定する項目の提供など) にも使用することができますが、本章では説明を行っていません。また、NIS は YP (「Yellow Pages」、いわゆる電話帳) としても知られています。

3.1 NIS サーバの設定

NIS の情報をネットワーク経由で配布するには、全てのクライアントに対してサービスを提供するサーバを 1 台だけ (マスター) を用意するか、もしくはマスターに対して情報を要求して、その応答を対応するクライアントに返却する NIS スレーブサーバを追加で用意する環境を構成します。

- ネットワーク内に 1 台の NIS サーバ (マスターサーバ) を構築する場合は、3.1.1項「NIS マスターサーバの設定」をお読みください。
- NIS マスターサーバがスレーブサーバに対してデータを提供する必要がある場合は、まず 3.1.1項「NIS マスターサーバの設定」の手順に従ってマスターサーバを構築したあと、サブネット内のスレーブサーバを 3.1.2項「NIS スレーブサーバの設定」の手順に従って構築してください。

3.1.1 NIS マスターサーバの設定

YaST で NIS サーバの機能を管理したい場合は、root で `zypper in yast2-nis-server` コマンドを実行して、`yast2-nis-server` パッケージをインストールしてください。お使いのネットワーク内に NIS マスターサーバを配置するには、下記の手順を実施します:

1. [YaST] › [ネットワークサービス] › [NIS サーバ] を選択して起動します。

2. お使いのネットワーク内に NIS サーバを 1 台だけ配置したい場合、もしくは後から設定する NIS スレーブサーバ向けにマスターサーバとして動作させたい場合は、[NIS マスタサーバを作成する] を選択します。選択に従って YaST は必要なパッケージをインストールします。



ヒント: 既に NIS サーバソフトウエアがインストールされている場合について

NIS サーバソフトウエアがお使いのマシン内にインストール済みである場合、[NISマスタサーバを作成する] を選択して NIS マスターサーバを作成します。



図 3.1: NIS サーバのセットアップ

3. ここでは NIS の基本的な設定オプションを指定します:

   a. NIS ドメイン名を入力します。

   b. このホストを NIS クライアントとしても動作させる (NIS サーバを利用してユーザのログインとデータへのアクセスを行う) 場合は、[このホストはNISクライアントでもあります] を選択します。

   c. NIS サーバを異なるサブネット内の NIS スレーブサーバに対するマスターサーバとして動作させる場合は、[アクティブなNISスレーブサーバが存在する] を選択します。[アクティブなNISスレーブサーバが存在する] を選択している場合にのみ、[高速マップ配布] が有用になります。これにより、スレーブに対するマップ転送が高速化されます。

d. `yppasswd` コマンドで、ネットワーク内のユーザ (ローカルユーザと NIS サーバで管理されているユーザの両方) に対して、自分自身のパスワード変更を許可したい場合は、[パスワードの変更を許可する] を選択します。また、これを選択した場合は、追加で [GECOSフィールドの変更を許可する] と [ログインシェルの変更を許可する] を選択できるようになります。「GECOS」とは `ypchfn` コマンドで変更できる項目で、名前やアドレスなどの情報を設定することができるものです。また、「シェル」は既定のシェル (Bash や Tcsh など) の意味で、`ypchsh` コマンドで変更できるようにするのかを設定します。ただし、設定するシェルはあらかじめ `/etc/shells` 内で指定されていなければなりません。

e. YaST 側で NIS サーバにあわせてファイアウオールの設定を調整したい場合は、[ファイアウオールでポートを開く] を選択します。

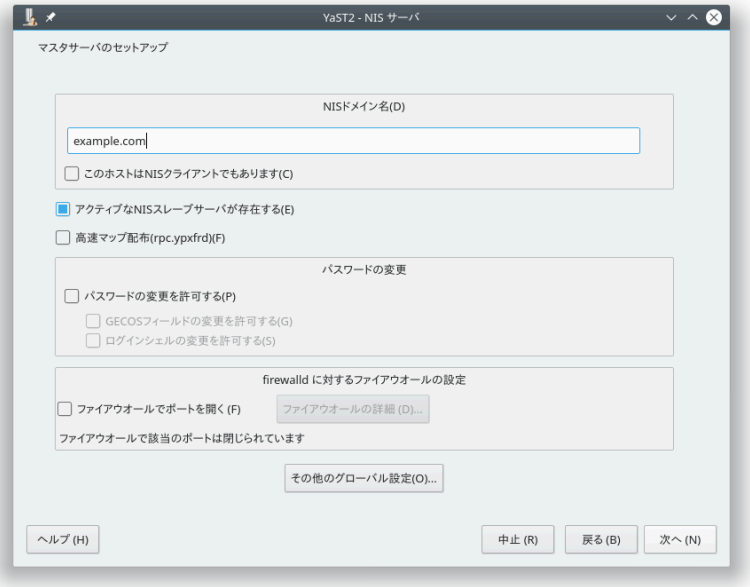


図 3.2: マスターサーバのセットアップ

f. あとは [次へ] を押して次のダイアログに進むか、もしくは [その他のグローバル設定] を押して追加設定を行います。
[その他のグローバル設定] には、NIS サーバの情報源となるディレクトリ (既定では `/etc`) の変更のほか、パスワードのマージ (合成) などの処理を設定することができます。マージの設定は、`/etc/passwd`, `/etc/shadow`, `/etc/group` にあるシステムの認証ファイルからユーザデータベースを作成する場合、選択しておくべき項目です。また、NIS 側で提供する最小のユーザ ID およびグループ ID を指定することもできます。最後に [OK] を押すと、設定を適用して元の画面に戻ることができます。

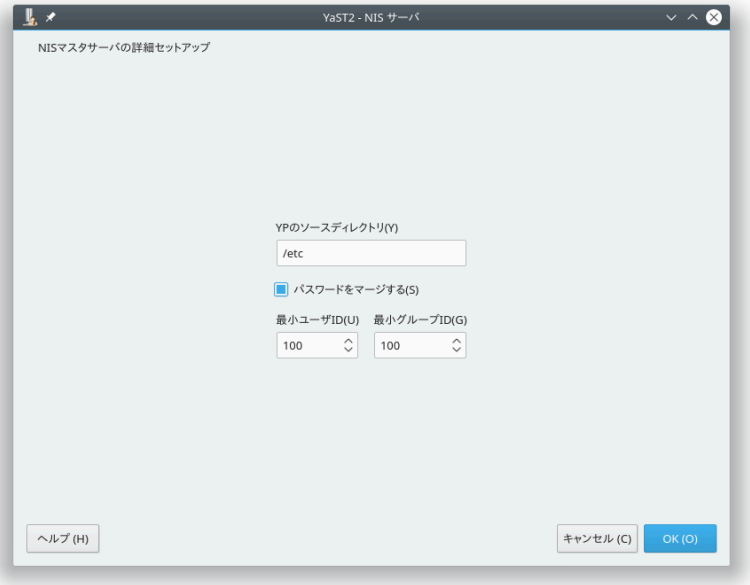


図 3.3: NIS サーバに対するディレクトリと同期するファイルの変更

4. 以前の画面で［アクティブなNISスレーブサーバが存在する］を選択していた場合は、ここでスレーブとして使用するホスト名を入力して［次へ］を押します。スレーブサーバが存在しない場合は、ここでの設定はそのまま飛ばしてかまいません。

5. 続いてデータベースの設定ダイアログが表示されます。NIS サーバのマップのセットアップ では、NIS サーバからクライアントに対して、データベースの一部のみを転送する設定です。通常は特に変更する必要はありません。そのまま［次へ］を押して進みます。

6. ここではどのマップを利用できるようにするのかを選択します。［次へ］を押すと進むことができます。

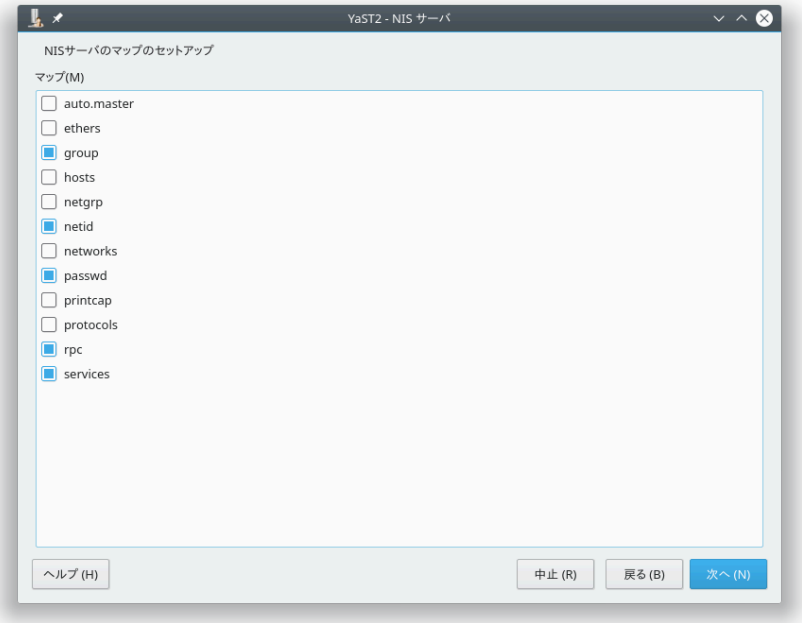


図 3.4: NIS サーバのマップのセットアップ

7. 次に、どのホストから NIS サーバに対して問い合わせを送信できるのかを設定します。ここではそれぞれのボタンを押すことで、ホストの追加や編集、削除を行うことができます。ここでは NIS サーバに対してリクエストを送信することのできる発信元ネットワークを指定します。通常は内部ネットワークを指定します。全てのネットワークに対してアクセスを許可するには、下記のような設定を行います:

```
255.0.0.0       127.0.0.0
0.0.0.0         0.0.0.0
```

最初の行は NIS サーバ自身からの接続を受け入れる設定、2 行目は全てのホストからの問い合わせを受け入れる設定です。

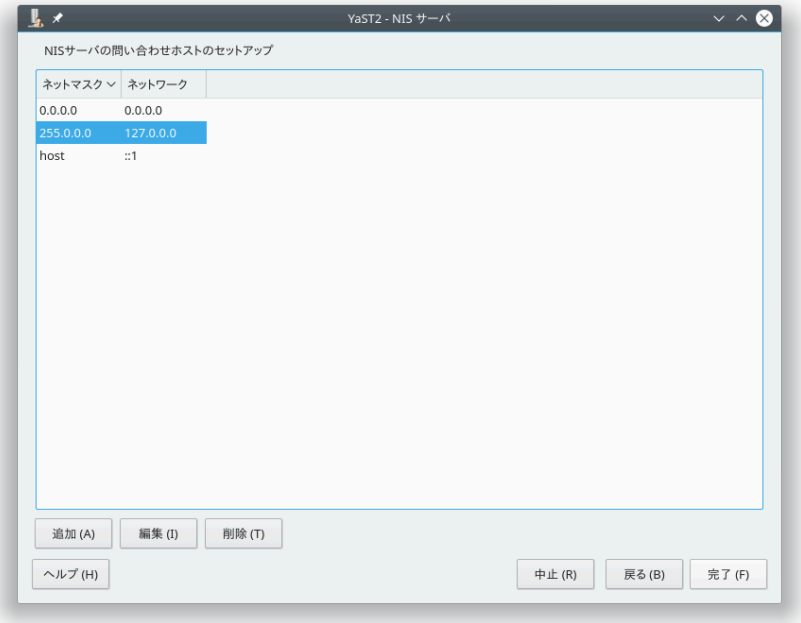


図 3.5: NIS サーバに対するリクエスト許可の設定

8. [完了] を押すと設定を保存して、セットアップを終了することができます。

3.1.2 NIS スレーブサーバの設定

お使いのネットワーク環境内に追加の NIS スレーブサーバ を設定するには、下記の手順で行います:

1. [YaST] › [ネットワークサービス] › [NIS サーバ] を選択して起動します。
2. [NISスレーブサーバを作成する] を選択して [次へ] を押します。

   ヒント

   NIS サーバソフトウエアがお使いのマシン内にインストール済みである場合、[NISスレーブサーバを作成する] を選択して NIS スレーブサーバを作成します。

3. NIS スレーブサーバの基本的な設定を行います:
   a. NIS のドメインを入力します。
   b. マスターサーバのホスト名もしくは IP アドレスを入力します。

c. そのサーバに対してユーザがログインできるようにする場合は、［このホストはNISクライアントでもあります］を選択します。

d. ［ファイアウオールでポートを開く］を選択して、ファイアウオールの設定を調整します。

e. ［次へ］を押します。

4. 次に、どのホストから NIS サーバに対して問い合わせを送信できるのかを設定します。ここではそれぞれのボタンを押すことで、ホストの追加や編集、削除を行うことができます。ここでは NIS サーバに対してリクエストを送信することのできる発信元ネットワークを指定します。通常は内部ネットワークを指定します。全てのネットワークに対してアクセスを許可するには、下記のような設定を行います:

```
255.0.0.0       127.0.0.0
0.0.0.0         0.0.0.0
```

最初の行は NIS サーバ自身からの接続を受け入れる設定、2 行目は全てのホストからの問い合わせを受け入れる設定です。

5. ［完了］を押すと設定を保存して、セットアップを終了することができます。

3.2 NIS クライアントの設定

NIS をワークステーションで使用したい場合は、下記のようにして行います:

1. ［YaST］›［ネットワークサービス］›［NIS クライアント］を選択して起動します。

2. ［NIS を使用する］を選択します。

3. NIS のドメイン名を入力します。この値は通常、管理者が指定するドメイン名か、DHCP で取得できる固定の IP アドレスになります。DHCP について、詳しくは『リファレンス』、第20章「DHCP」をお読みください。

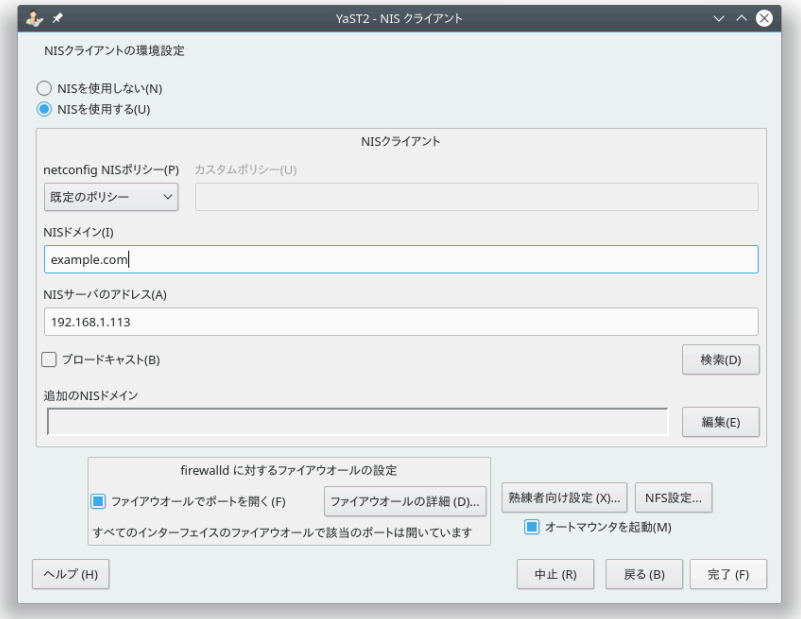


図 3.6: NIS サーバのドメインとアドレスの設定

4. 使用する NIS サーバをスペース区切りで指定します。NIS サーバのアドレスが分からない場合は、［検索］を押すことで、YaST 側でドメイン内の NIS サーバを検索することができます。なお、ローカルネットワークの規模にもよりますが、場合によっては時間のかかる処理になることもあります。［ブロードキャスト］を選択すると、指定したサーバに対する問い合わせが失敗した際、ローカルネットワークに対して NIS サーバを問い合わせるようになります。

5. お使いの環境によっては、automounter (オートマウンター) を使用する必要があることもあります。このオプションを選択すると、必要に応じて追加のソフトウエアをインストールします。

6. お使いのクライアントに対して、他のホストからの問い合わせに答えないように設定したい場合は、［熟練者向け設定］に移動して、［リモートホストに応答する］のチェックを外してください。また、［ブロークンサーバ］にチェックを入れると、非特権ポートを介した通信でサーバからの応答を受け取るようになります。詳しくは `man ypbind` をお読みください。

7. 最後に［完了］を押すと、設定を保存して YaST のコントロールセンターに戻ることができます。これで NIS の設定が完了します。

4 YaST を利用した認証クライアントの設定

改訂履歴

2023-11-30

Kerberos は認証時に使用するプロトコルですが、LDAP は認可と識別を行うためのプロトコルです。両方を組み合わせて使用することもできます。LDAP に関する詳細は 第5章「389 Directory Server を利用した LDAP サービス」を、Kerberos に関する詳細は 第6章「Kerberos を利用したネットワーク認証」をそれぞれお読みください。

4.1 YaST を利用した認証クライアントの設定

YaST では、下記のようなモジュールを利用して、クライアントに対する認証を設定することができます:

- [ユーザのログイン管理]: 識別情報提供サービス (通常は LDAP) とユーザ認証サービス (通常は Kerberos) の 両方を併用して設定を行います。この設定機能は SSSD をベースにした仕組みで、Active Directory ドメインへの参加を行う場合には最適な選択肢となります。
  このモジュールについては 8.3.2項「[ユーザのログイン管理] を利用した Active Directory への参加」で説明しています。
- [Windows ドメインメンバーシップ]: Kerberos と LDAP をそれぞれ利用して Active Directory ドメインへの参加を行います。この設定機能は `winbind` をベースにした仕組みで、NTLM での認証が必要な Active Directory ドメインに参加する必要がある場合や、フォレストを 跨いだ認証が必要な場合に最適な選択肢となります。
  このモジュールについては 8.3.3項「[Windows ドメインメンバーシップ] を利用した Active Directory ドメインへの参加」で説明しています。

4.2 SSSD

YaST では、2 種類のモジュールで SSSD を使用しています。それぞれ [ユーザログイン管理] と [LDAP と Kerberos の認証] です。

SSSD は System Security Services Daemon の略であり、ネットワーク上離れた場所に存在するディレクトリサービスにアクセスして、ユーザデータの提供や様々な認証方式 (LDAP, Kerberos, Active Directory (AD) など) の提供を受けることができるデーモンです。このデーモンには NSS (Name Service Switch) と PAM (Pluggable Authentication Module) の各インターフェイスも用意されています。

SSSD はユーザデータをローカルにキャッシュ (一時記憶) してデータを提供しています。そのため、実際のディレクトリサービスに対して、一時的にアクセスができない状況に陥った場合も、SSSD はサービスを続けることができます。

4.2.1 状態の確認

YaST の認証モジュールで設定を行ったあとは、下記のように実行することで、SSSD が動作しているかどうかを確認することができます:

```
# systemctl status sssd
sssd.service - System Security Services Daemon
   Loaded: loaded (/usr/lib/systemd/system/sssd.service; enabled)
   Active: active (running) since Thu 2015-10-23 11:03:43 CEST; 5s ago
   [...]
```

4.2.2 キャッシュ

認証用のバックエンドが利用できなくなってしまっている状態でもログインができるようにするため、SSSD はキャッシュの有効期限が切れてもキャッシュを使用するようになっています。これはバックエンドが再度利用できるようになるまで動作します。

キャッシュの内容を消去したい場合は、`sss_cache -E` (`sss_cache` コマンドは `sssd-tools` パッケージに含まれています) を実行します。

SSSD キャッシュを削除したい場合は、下記を実行します:

```
> sudo systemctl stop sssd
> sudo rm -f /var/lib/sss/db/*
> sudo systemctl start sssd
```

5 389 Directory Server を利用した LDAP サービス

改訂履歴

2024-06-26

軽量ディレクトリアクセスプロトコル (Lightweight Directory Access Protocol (LDAP)) は情報ディレクトリへのアクセスや管理に使用するプロトコルです。LDAP はユーザやグループのほか、システム設定の管理やアドレスの管理などにも使用することができます。openSUSE Leap 15.7 では、389 Directory Server が提供する LDAP サービスを使用することができます。これは従来の OpenLDAP の代替として提供されているものです。

実際には、中央のサーバのディレクトリ内にデータを保管しておき、明確に定義されたプロトコルで全てのクライアントに配布します。データは構造化されているため、様々なアプリケーションがそれらのデータにアクセスできるようになります。中央のサーバ (リポジトリとも呼びます) を用意することで管理の手間を省き、LDAP のようなオープンで標準化されたプロトコルを使用することで、そこで管理されている情報にアクセスするプログラムをできる限り多く用意することができます。

なお、本章での「ディレクトリ」とは、読み込みや検索の速度や効率性の点で最適化されたデータベースを意味します。ディレクトリ内に保存されたデータは長期にわたって保存され、あまり変更が発生しないものとされます。通常のデータベースシステムであれば、短い時間内にデータの書き込みが多数発生する前提で最適化が行われますが、LDAP サービスは読み込みに対して大きく最適化されたデータベースとなります。

5.1 LDAP のディレクトリツリー構造

本章では、LDAP ディレクトリツリーの基本的な配置と、LDAP に関する基本的な用語を説明しています。既に LDAP に関する知識をお持ちの場合は、5.3.1項「389 Directory Server のインスタンスの構築」からお読みください。

LDAP ディレクトリはツリー型の構造になっています。ディレクトリ内の全ての項目 (オブジェクトと呼びます) は、この階層構造内での位置が決められています。この階層構造を ディレクトリ情報ツリー (Directory Information Tree (DIT)) と呼びます。また、特定の項目に対して、それを正確に識別するためのパス情報を、識別名 (Distinguished Name (DN)) と呼びます。またツリー内にあるオブジェクトは 相対識別名 (Relative Distinguished Name (RDN)) でお互いを区別します。識別名は、その項目に対する全ての 相対識別名 から構成されています。

LDAP ディレクトリツリーの構造を図に表すと、図5.1「LDAP ディレクトリの構造」のようになります。

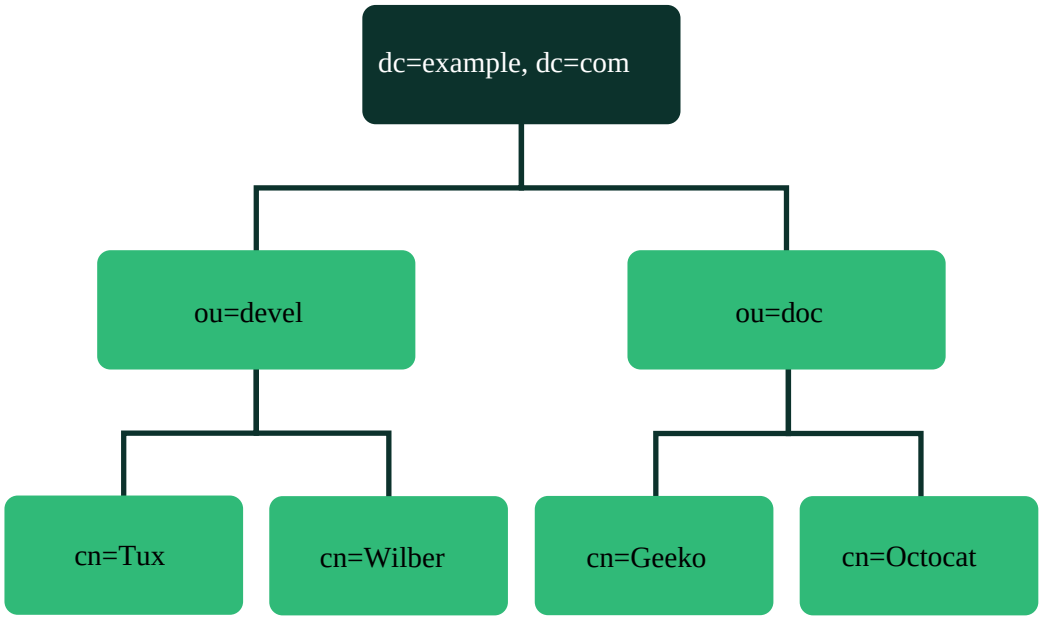


図 5.1: LDAP ディレクトリの構造

上記は架空のディレクトリ情報ツリーを表しています。そのうち、3 階層分の項目のみを図示しています。図の中では、各項目を 1 つの箱として表しています。たとえば架空の従業員である `Geeko Linux` に対する完全な 識別名 (DN) は、`cn=Geeko Linux,ou=doc,dc=example,dc=com` となります。これは相対識別名である `cn=Geeko Linux` に対して、階層構造上の親である `ou=doc,dc=example,dc=com` を追加することによります。

ディレクトリ情報ツリー (DIT) 内に保管できるオブジェクトの種類は、スキーマ と呼ばれるものに従って設定されます。オブジェクトの種類は オブジェクトクラス と呼ばれ、それに対応するスキーマで、そのオブジェクトが設定しなければならない値や、設定できる値が示されます。そのためスキーマには、LDAP サーバ内で使用される全てのオブジェクトクラスや属性の定義を含んでいることになります。一方の属性は、構造化されたデータ型です。それらの書式や順序などの情報も、スキーマ内に記述されています。LDAP サーバには幅広い環境で動作することができるよう、中枢 (コア) スキーマセットが用意されています。必要であれば、独自のスキーマをアップロードすることもできます。

表5.1「よく使用されるオブジェクトクラスと属性」には、`00core.ldif` や `06inetorgperson.ldif` で規定されているオブジェクトクラスの一部と、それらで必須とされている属性、そして属性値の書式を紹介しています。389 Directory Server パッケージをインストールしている環境であれば、これらは `/usr/share/dirsrv/schema` 内に存在しています。

表 5.1: よく使用されるオブジェクトクラスと属性

オブジェクトクラス	意味	値の例	必須の属性
`domain`	ドメインの名前パート	example	displayName

オブジェクトクラス	意味	値の例	必須の属性
organizationalUnit	組織単位	documentation	ou
nsPerson	イントラネットもしくはインターネットの個人関連データ	Tux Linux	cn

また 例5.1「CN=schema からの抜粋」では、スキーマ定義の例を示しています。

例 5.1: CN=SCHEMA からの抜粋

```
attributetype (1.2.840.113556.1.2.102 NAME 'memberOf' ❶
       DESC 'Group that the entry belongs to' ❷
       SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 ❸
       X-ORIGIN 'Netscape Delegated Administrator') ❹

objectclass (2.16.840.1.113730.3.2.333 NAME 'nsPerson' ❺
       DESC 'A representation of a person in a directory server' ❻
       SUP top STRUCTURAL ❼
       MUST ( displayName $ cn ) ❽
       MAY ( userPassword $ seeAlso $ description $ legalName $ mail \
             $ preferredLanguage ) ❾
       X-ORIGIN '389 Directory Server Project'
  ...
```

❶ 属性の名前とユニークな OID (オブジェクト識別子 (Object Identifier)) (数字) 、そして属性の略称を表しています。

❷ DESC では属性に対する簡潔な説明を記します。定義の元となった RFC などがあれば、それも記述されます。

❸ 属性内に保持できるデータの型。この場合、大文字と小文字を区別しない文字列であることを示しています。

❹ スキーマ要素のソース (たとえばプロジェクト名) 。

❺ オブジェクトクラス nsPerson の定義の始まりです。オブジェクトクラスの定義は属性の定義と同じで、OID とオブジェクトクラスの名前をそれぞれ記します。

❻ そのオブジェクトクラスに対する簡潔な説明です。

❼ SUP top と書くことで、このオブジェクトクラスの上位に存在するオブジェクトクラスが存在していないことを表しています。

❽ MUST 以下には、nsPerson のオブジェクトで設定されなければならない全ての属性を記します。

❾ MAY 以下には、そのオブジェクトクラスで設定することのできる全ての属性を記します。

5.2 389 Directory Server の Docker コンテナの作成と管理



注記

本章は 任意 で参照すべき項目であり、389 Directory Server を Docker コンテナ下で動作させたい場合にのみ読むべきものです。通常のサーバ内への 389 Directory Server のインストールや管理については、本章以外の章をお読みください。

389 Directory Server インスタンスを Docker のコンテナとして作成して管理したい場合は、下記の手順をお読みください:

389 Directory Server イメージの取得

コンテナレジストリから 389 Directory Server のイメージを取得するには、下記のようなコマンドを実行します:

```
>  docker pull registry.suse.com/suse/389-ds:latest
```

新しいボリュームの作成

コンテナに対して新しいボリュームを作成するには、下記のようなコマンドを実行します:

```
>  docker volume create ボリューム名
```

コンテナの作成 (基本設定も併せて実施)

基本的な設定を行いながらコンテナを作成したい場合は、下記のようなコマンドを実行します:

```
>  docker create \
 -u ユーザ名 \
 -e SUFFIX_NAME="dc=example,dc=com" \
 -e DS_DM_PASSWORD=パスワード \
 -m 1024M \
 -p 3389:3389 -p 3636:3636 \
 -v ボリューム名:/data \
 --name インスタンス名 \
 registry.suse.com/suse/389-ds:latest
```

389 Directory Server の Docker コンテナの起動

Docker コンテナを起動するには、下記のようなコマンドを実行します:

```
>  docker start インスタンス名
```

動作中の 389 Directory Server コンテナ内でのコマンドの実行

ここでは、コンテナ内でのプライマリプロセス (`PID 1`) が動作していることを前提にしています。389 Directory Server コンテナ内でコマンドを実行するには、下記のような書式で記述して実行します:

```
> sudo docker exec -u ユーザ名 -i -t インスタンス名 コマンド
```



注記: コマンド は実行可能なものでなければならない件について

複数のコマンドを連続して実行したり、コマンド内で引用符を利用したりしたい場合は、コンテナに添付されている `sh` シェルを介して実行する方法があります:

```
> sudo docker exec -u ユーザ名 -i -t インスタンス名 sh -c "コマンド"
```

389 Directory Server のコンテナの停止

動作中の Docker コンテナを停止するには、下記のようなコマンドを実行します:

```
>  docker stop インスタンス名
```

389 Directory Server の Docker コンテナの削除

Docker コンテナを削除するには、下記のようなコマンドを実行します:

```
>  docker rm インスタンス名
```

5.3 389 Directory Server のインストール

下記のコマンドで 389 Directory Server をインストールすることができます:

```
> sudo zypper install 389-ds
```

インストールを行った後は、5.3.1項「389 Directory Server のインスタンスの構築」に示されている内容に従って、 サーバの設定を行ってください。

5.3.1 389 Directory Server のインスタンスの構築

`dscreate` コマンドを使用することで、389 Directory Server のインスタンスを作成したりきれいに削除したりすることができます。

インスタンスの作成方法には 2 つの種類があります。1 つめは設定ファイルを使用する方法、もう 1 つは自動生成されたテンプレート (雛型) ファイルを使用する方法です。なお、自動生成されたテンプレートファイルを使用して本番環境を作成する場合は、内容をよく読んで注意深く設定してください。

インスタンスを作成したら、あとは管理者用のアカウントを作成して、必要なユーザとグループを管理してサービスを動作させてください。

389 Directory Server は主に 3 つのコマンドで制御します:

`dsctl`
: ローカルのインスタンス管理に使用するコマンドで、`root` の権限が必要となります。なお、このコマンドを実行するには、ディレクトリサーバインスタンスの動作しているサーバ内にいなければなりません。起動や停止、データベースのバックアップなどを行うことができます。

`dsconf`
: サーバの管理や設定を行うための主要なツールです。外部インターフェイスを介してインスタンスの設定を管理することができます。このコマンドを使用することで、インスタンスをリモートから設定変更することができます。

`dsidm`
: 識別子 (ユーザ／グループ／パスワードなど) を管理するために使用します。なお、アクセス制御機能で許可を得ておく必要があります。一般ユーザの場合、自分自身のパスワードのリセットや詳細情報の変更などを行うことができます。

下記の手順を実施することで、いくつかのサンプル用エントリを含むテスト目的や開発目的のインスタンスを構築することができます。

1. 独自の設定ファイルを利用した 389 Directory Server インスタンスの作成
2. テンプレートからの 389 Directory Server インスタンスの作成
3. ローカル管理用の管理者認証情報の設定
4. LDAP でのユーザとグループの管理
5. LDAP 認証管理のための SSSD の使用
6. プラグインの管理
7. TLS 証明書と鍵の取り込み

5.3.2 独自の設定ファイルを利用した 389 Directory Server インスタンスの作成

シンプルな設定ファイルを使用することで、新しい 389 Directory Server インスタンスを作成することができます。このファイルは INF と呼ばれる形式でなければなりませんが、ファイル名に関しては任意の名前を使用することができます。

既定のインスタンス名は `localhost` です。インスタンス名は作成後に変更することができませんので、混乱を避けるとともに、動作を良く理解できるよう、既定値ではなく独自の名前を指定しておくことをお勧めします。なお、下記の例では `LDAP1` というインスタンス名で、サフィックス (LDAP での既定のドメイン名) が dc= `LDAP1` ,dc= `COM` である場合の例を示しています。

例 5.2 には新しい 389 Directory Server インスタンスを作成する際の設定ファイルの例が示されています。この内容を修正せずそのまま利用してもかまいません。

1. 下記の内容をホームディレクトリ内の `LDAP1.inf` というファイルに保存します:

例 5.2: 389 DIRECTORY SERVER を構築するための最小限のインスタンス設定ファイル

```
# LDAP1.inf

[general]
config_version = 2 ❶

[slapd]
root_password = パスワード❷
self_sign_cert = True ❸
instance_name = LDAP1

[backend-userroot]
sample_entries = yes ❹
suffix = dc=LDAP1,dc=COM
```

❶ この行は必ず指定しておいてください。ここでは INF ファイルのバージョンが 2 であることを示しています。

❷ `cn=Directory Manager` という LDAP ユーザに対するパスワードを、`root_password` で指定しています。このユーザは、ディレクトリサービスに接続 (バインド) するために使用するもので、複雑で強力なパスワードでなければなりません。

❸ 自己署名型のサーバ証明書を `/etc/dirsrv/slapd-LDAP1` に作成することを宣言しています。

❹ サンプルユーザとサンプルグループをそれぞれ新しいインスタンス内に作成するよう指示しています。

2. 例 5.2 から 389 Directory Server のインスタンスを作成するには、下記を実行します:

```
> sudo dscreate -v from-file LDAP1.inf | \
tee LDAP1-OUTPUT.txt
```

上記のコマンドを実行するとインスタンスが作成され、作成の際に出力された様々な情報が `LDAP1-OUTPUT.txt` に書き込まれます。出力されたテキストファイルには様々な情報が含まれています。このファイルを作成したくない場合は、上記のコマンドラインのうち、`| tee LDAP1-OUTPUT.txt` の部分を削除してください。

3. `dscreate` コマンドの実行が失敗した場合には、メッセージ内にその理由が示されますので、その原因を解決してからいったんインスタンスを削除 (詳しくは ステップ 5 をお読みください) してから、インスタンスを作成し直してください。

4. インスタンスの作成が成功すると、`Completed installation for LDAP1` というメッセージが出力されます。あとは作成したインスタンスの状態を確認します:

```
> sudo dsctl LDAP1 status
Instance "LDAP1" is running
```

5. 下記のコマンドはインスタンスをきれいに削除するためのコマンドです。前者のコマンドでは削除が正しくできるかどうかを確認 (ドライラン) し、後者のコマンドで実際の削除を行っています。後者のコマンドでは、`--do-it` オプションを忘れずに指定してください:

```
> sudo dsctl LDAP1 remove
Not removing: if you are sure, add --do-it

> sudo dsctlLDAP1 remove --do-it
```

このコマンドを実行することで、部分的にインストールが完了していたり、壊れてしまったりしたインスタンスを削除することもできます。インスタンスの作成も削除も、必要に応じて何度でも実行することができます。

インスタンス名を忘れてしまった場合は、`dsctl` コマンドを実行して全てのインスタンスを表示させてください:

```
> sudo dsctl -l
slapd-LDAP1
```

5.3.3 テンプレートからの 389 Directory Server インスタンスの作成

`dscreate` コマンドを使用することで、新しい 389 Directory Server インスタンスを作成するためのテンプレート (雛型) を自動生成することができます。このコマンドでは、そのまま使用できる形でテンプレートが作成されますので、あとは必要に応じて様々な箇所を修正するだけでインスタンスを作成することができます。ただし、そのまま使用するのはテスト用途に留めるものとし、本番環境の場合は要件に合わせて適切に変更してください。なお、既定値はテンプレートファイル内のコメントとして書かれています (英語) 。設定を変更する場合は既定値のコメント文字を外して、必要な値を記入していってください。いずれのオプションに対しても詳しく説明が書かれています。

下記のコマンドを入力して実行すると、設定例を標準出力に出力することができます:

```
> sudo dscreate create-template
```

上記のコマンドを実行すると標準出力にテンプレートが出力されますが、このままでは使用できません。下記のようにして任意のファイル名を指定して、そのファイル内にテンプレートを出力してください:

```
> sudo dscreate create-template TEMPLATE.txt
```

下記は出力されたテンプレートの抜粋です:

```
# full_machine_name (str)
# Description: Sets the fully qualified hostname (FQDN) of this system. When
# installing this instance with GSSAPI authentication behind a load balancer, set
# this parameter to the FQDN of the load balancer and, additionally, set
# "strict_host_checking" to "false".
# Default value: ldapserver1.test.net
;full_machine_name = ldapserver1.test.net

# selinux (bool)
# Description: Enables SELinux detection and integration during the installation
# of this instance. If set to "True", dscreate auto-detects whether SELinux is
# enabled. Set this parameter only to "False" in a development environment.
# Default value: True
;selinux = True
```

システムの完全修飾ドメイン名 (テンプレート内の `full_machine_name`) など、既存の環境からの既定値を自動的に設定している様子がわかるかと思います。あとはこのファイルを何も変更せずそのまま利用して、新しいインスタンスを作成してみます:

```
> sudo dscreate from-file TEMPLATE.txt
```

これにより、`localhost` という名前の新しいインスタンスが作成され、作成が完了すると自動的に開始されます:

```
> sudo dsctl localhost status
```

```
Instance "localhost" is running
```

既定値のままインスタンスを作成しても問題なく動作しますが、いくつかの設定値を変更しておくことで、より本番に近い環境を作成することができます。

インスタンス名は作成後に変更することができませんので、混乱を避けるとともに、動作を良く理解できるよう、既定値ではなく独自の名前を指定しておくことをお勧めします。インスタンス名を変更するには、`;instance_name = localhost` の行のコメント文字 (`;`) を外して、`localhost` を任意の名前に変更してください。ここでは `LDAP1` という名前を使用しています。

また、もう 1 つ変更すべき箇所として、ユーザとグループのサンプルを作成する機能があります。必要であれば `;sample_entries = no` の箇所のコメント文字を外して、`no` を `yes` にしてください。サンプル作成を指定すると、`demo_user` (ユーザ) と `demo_group` (グループ) がそれぞれ作成されます。

このほか `;root_password` の行のコメント文字 (;) を外して、設定したいパスワードを入力してもかまいません。

また、テンプレート内には既定のサフィックス (LDAP での既定のドメイン名) を指定していませんので、下記のようにして `suffix` 行で指定を行うことができます:

```
suffix = dc=LDAP1,dc=COM
```

作成したインスタンスをきれいに削除してやり直すには、`dsctl` で下記のようなコマンドを入力して実行します:

```
> sudo dsctl LDAP1 remove --do-it
```

5.3.4 389 Directory Server の起動と停止

389 Directory Server のインスタンスを管理するには、`systemd` を使用します。まずはサービスの状態を表示するには、下記のように入力して実行します (下記はインスタンス名が `LDAP1` である場合の例です):

```
> systemctl status --no-pager --full dirsrv@LDAP1.service
   • dirsrv@LDAP1.service - 389 Directory Server LDAP1.
     Loaded: loaded (/usr/lib/systemd/system/dirsrv@.service; enabled; vendor preset:
 disabled)
     Active: active (running) since Thu 2021-03-11 08:55:28 PST; 2h 7min ago
    Process: 4451 ExecStartPre=/usr/lib/dirsrv/ds_systemd_ask_password_acl
       /etc/dirsrv/slapd-LDAP1/dse.ldif (code=exited, status=0/SUCCESS)
   Main PID: 4456 (ns-slapd)
     Status: "slapd started: Ready to process requests"
      Tasks: 26
     CGroup: /system.slice/system-dirsrv.slice/dirsrv@LDAP1.service
             └─4456 /usr/sbin/ns-slapd -D /etc/dirsrv/slapd-LDAP1 -i /run/dirsrv/slapd-
LDAP1.pid
```

LDAP サーバの起動／停止／再起動を行うには、それぞれ下記のように入力して実行します:

```
> sudo systemctl start dirsrv@LDAP1.service
> sudo systemctl stop dirsrv@LDAP1.service
> sudo systemctl restart dirsrv@LDAP1.service
```

`systemctl` の使用方法について、詳しくは『リファレンス』、第10章「systemd デーモン」をお読みください。

なお、`dsctl` コマンドでもサービスの起動と停止を行うことができます:

```
> sudo dsctl LDAP1 status
> sudo dsctl LDAP1 stop
> sudo dsctl LDAP1 restart
> sudo dsctl LDAP1 start
```

5.3.5 ローカル管理用の管理者認証情報の設定

389 Directory Server のローカル管理を行う目的で、`/root` ディレクトリ内に `.dsrc` 設定ファイルを作成することができます。これにより、管理者 (root) や sudo 経由で管理権限を取得したユーザが、わざわざ 389 Directory Server の管理者情報を入力したりすることなく管理できるようになります。例 5.3 には、サーバをローカル管理する場合の例が示されています。ここではインスタンス名が `LDAP1` 、ベース DN が `com` である場合を示しています。

`/root/.dsrc` ファイルを作成したら、あとは新しいユーザを作成していきます (詳しくは 5.6項「LDAP でのユーザとグループの管理」をお読みください) 。

例 5.3: ローカル管理用の `.dsrc` ファイル

```
# LDAP1 インスタンスを管理するための /root/.dsrc ファイルの例

[LDAP1] ❶

uri = ldapi://%%2fvar%%2frun%%2fslapd-LDAP1.socket ❷
basedn = dc=LDAP1,dc=COM
binddn = cn=Directory Manager
```

❶ ここには管理対象のインスタンス名を指定します。

❷ `ldapi` は接続しようとしているユーザの UID や GID を検出することができるプロトコルで、UID/GID が `0/0` もしくは `dirsrv:dirsrv` であった場合、ディレクトリサーバに対して管理者 (つまり `cn=Directory Manager`) でログインできるようになります。
なお URI の指定では、スラッシュを `%%2f` に置き換える必要があります。この例では、`/var/run/slapd-LDAP1.socket` というパスを指定しています。

重要: sudoers.ldap の次世代新機能について

`sudo` 1.9.9 以前のバージョンでは、sudoers.ldap のうち `sudoUser` , `sudoRunAsUser` , `sudoRunAsGroup` の各属性で否定表現が正しく動作しませんでした。たとえば下記のようになっていました:

```
 # joe 以外の全員にマッチさせるつもりが、
# 全員を対象外としてしまう例
sudoUser: !joe

# 同様に joe 以外の全員にマッチさせるつもりが、
# joe を含む全員を対象としてしまう例
sudoUser: ALL
sudoUser: !joe
```

`sudo` のバージョン 1.9.9 およびそれ以降では、`sudoUser` における否定表現が動作するようになっています。詳しくは `man 5 sudoers.ldap` をお読みください。

5.4 ファイアウオールの設定

389 Directory Server の既定の TCP ポートは 389 と 636 です。TCP:389 は暗号化を行わない接続および STARTTLS を使用する接続向けに、TCP:636 は TLS での暗号化を行う接続向けに使用します。

`firewalld` は openSUSE Leap における既定のファイアウオールマネージャです。下記のコマンドを実行すると、`ldap` および `ldaps` の各サービスを有効化することができます:

```
> sudo firewall-cmd --add-service=ldap --zone=internal
> sudo firewall-cmd --add-service=ldaps --zone=internal
> sudo firewall-cmd --runtime-to-permanent
```

なお、zone 以下の値はお使いのサーバに合わせて設定してください。なお、TLS での接続の暗号化に関する詳細は 5.10項「TLS 証明書と鍵の取り込み」を、`firewalld` に関する詳細は 24.3項「基本的なファイアウオール処理」をそれぞれお読みください。

5.5 389 Directory Server でのバックアップと復元

389 Directory Server ではオフラインとオンラインの両方のバックアップ形態に対応しています。`dsctl` コマンドではオフラインによるデータベースバックアップを、`dsconf` コマンドではオンラインによるデータベースバックアップを、それぞれ採取することができます。また、LDAP サーバの設定ディレクトリをバックアップしておくことで、重大な障害が発生してサーバが復旧不可能になってしまったような場合に対応することもできます。

5.5.1 LDAP サーバ設定のバックアップ

LDAP のサーバ設定は `/etc/dirsrv/slapd-インスタンス名` のディレクトリ内に存在しています。このディレクトリ内には証明書や鍵のほか、`dse.ldif` ファイルが含まれています。これらは `tar` コマンドを利用して圧縮しながらバックアップすることができます:

```
> sudo tar caf \
config_slapd-インスタンス名-$(date +%Y-%m-%d_%H-%M-%S).tar.gz \
/etc/dirsrv/slapd-インスタンス名/
```



compact注記: tar で出力される無害なメッセージについて

なお、上記のような `tar` コマンドを実行すると、`tar: メンバ名から先頭の `/' を取り除きます` のようなメッセージが表示されますが、無視してかまいません。

設定を復元したい場合は、上記で採取したバックアップを同じディレクトリ内に展開します:

1. まずは既存の設定を上書きしてしまわないよう、既存の設定を移動します:

   ```
   > sudo old /etc/dirsrv/slapd-インスタンス名/
   ```

2. 採取したバックアップから展開します:

   ```
   > sudo tar -xvzf \
   config_slapd-インスタンス名-日付.tar.gz
   ```

3. 展開したディレクトリとファイルを `/etc/dirsrv/slapd-インスタンス名` にコピーします:

   ```
   > sudo cp -r etc/dirsrv/slapd-インスタンス名 \
   /etc/dirsrv/slapd-インスタンス名
   ```

5.5.2 LDAP データベースに対するオフラインバックアップの採取と復元

`dsctl` コマンドを利用することで、オフラインバックアップを採取することができます。まずはサーバを停止します:

```
> sudo dsctl インスタンス名 stop
Instance "インスタンス名" has been stopped
```

下記の例のとおりに実行すると、`/var/lib/dirsrv/slapd-LDAP1/bak/インスタンス名-日付` のようなディレクトリ内にバックアップが作成されます。

```
> sudo dsctl インスタンス名 db2bak
db2bak successful
```

たとえばテスト用に作成した ldap1 インスタンスの場合、下記のようなディレクトリになります:

```
/var/lib/dirsrv/slapd-ldap1/bak/ldap1-2021_10_25_13_03_17
```

バックアップから復元したい場合は、下記のようにしてバックアップを含むディレクトリを指定します:

```
> sudo dsctl インスタンス名 bak2db \
/var/lib/dirsrv/slapd-インスタンス名/bak/インスタンス名-日付/
bak2db successful
```

あとはサーバを起動します:

```
> sudo dsctl インスタンス名 start
Instance "インスタンス名" has been started
```

なお、LDIF 形式でのバックアップを採取することもできます:

```
> sudo dsctl インスタンス名 db2ldif --replication userRoot
ldiffile: /var/lib/dirsrv/slapd-インスタンス名/ldif/インスタンス名-userRoot-日付.ldif
db2ldif successful
```

LDIF 形式のバックアップから復元したい場合も同様に、ファイル名を指定して復元し、サーバを起動するだけです:

```
> sudo dsctl ldif2db userRoot \
/var/lib/dirsrv/slapd-インスタンス名/ldif/インスタンス名-userRoot-日付.ldif
> sudo dsctl インスタンス名 start
```

5.5.3 LDAP データベースに対するオンラインバックアップの採取と復元

`dsconf` コマンドを利用することで、LDAP データベースのオンラインバックアップを採取することができます:

```
> sudo dsconf インスタンス名 backup create
The backup create task has finished successfully
```

上記を実行すると、`/var/lib/dirsrv/slapd-インスタンス名/bak/インスタンス名-日付` のようなディレクトリ内にバックアップが作成されます。

復元は下記のようにして行います:

```
> sudo dsconf インスタンス名 backup restore \
/var/lib/dirsrv/slapd-インスタンス名/bak/インスタンス名-日付
```

5.6 LDAP でのユーザとグループの管理

ユーザやグループの作成や削除、管理を行うには、`dsidm` コマンドを使用します。

5.6.1 既存の LDAP ユーザとグループの表示

下記では、既存のユーザとグループを一覧表示する例を示しています。なお、インスタンス名は `LDAP1` であるものとします。実際に実行する際には、お使いのインスタンス名に置き換えて実行してください。

```
> sudo dsidm LDAP1 user list
> sudo dsidm LDAP1 group list
```

特定のユーザに対する全ての情報を表示させるには、下記のように入力して実行します:

```
> sudo dsidm LDAP1 user get ユーザ名
```

特定のグループに対する全ての情報を表示させるには、下記のように入力して実行します:

```
> sudo dsidm LDAP1 group get グループ名
```

グループのメンバー一覧を表示するには、下記のように入力して実行します:

```
> sudo dsidm LDAP1 group members グループ名
```

5.6.2 ユーザの作成とパスワードの管理

下記の例では 1 人のユーザ `wilber` を作成しています。このとき、サーバインスタンスは `LDAP1` 、サフィックスは `dc=LDAP1,dc=COM` であるものとします。

手順 5.1: LDAP ユーザの作成

下記の例では、389 DS インスタンス内にユーザ Wilber Fox を作成しています:

1. ```
   > sudo dsidm LDAP1 user create --uid wilber \
     --cn wilber --displayName 'Wilber Fox' --uidNumber 1001 --gidNumber 101 \
     --homeDirectory /home/wilber
   ```

2. ユーザの `distinguished name` (DN) 名 (ディレクトリオブジェクトに対する完全修飾名、つまり唯一性の保証された名前) を参照するには、下記のように入力して実行します:

   ```
   > sudo dsidm LDAP1 user get wilber
   dn: uid=wilber,ou=people,dc=LDAP1,dc=COM
   [...]
   ```

   ユーザのパスワード変更などの処理を行うには、`distinguished name` (DN) を指定する必要があります。

3. `wilber` ユーザに対するパスワードを作成するには、下記のように入力して実行します:

   a. ```
      > sudo dsidm LDAP1 account reset_password \
        uid=wilber,ou=people,dc=LDAP1,dc=COM
      ```

   b. あとは `wilber` に設定するパスワードを 2 回入力します。
      パスワードの設定が完了すると、下記のようなメッセージが表示されるはずです:

      ```
      reset password for uid=wilber,ou=people,dc=LDAP1,dc=COM
      ```

      設定済みのパスワードを変更したい場合も、同じコマンドで行うことができます。

4. 認証が正しく動作することを確認します:

   ```
   > ldapwhoami -D uid=wilber,ou=people,dc=LDAP1,dc=COM -W
   Enter LDAP Password: PASSWORD
   dn: uid=wilber,ou=people,dc=LDAP1,dc=COM
   ```

5.6.3 グループの作成と管理

ユーザを作成したあとは、グループを作成してユーザを割り当てます。下記の手順では `server_admins` という名前のグループを作成し、そのグループに `wilber` ユーザを所属させます。このとき、インスタンス名は `LDAP1` 、サフィックスは `dc=LDAP1,dc=COM` であるものとします。

手順 5.2: LDAP グループの作成とユーザの割り当て

1. まずはグループを作成します:

   ```
   > sudo dsidm LDAP1 group create
   ```

   すると、グループ名の入力を求められます。たとえば `SERVER_ADMINS` というグループを作成するには、下記のように入力します:

   ```
   Enter value for cn : SERVER_ADMINS
   ```

2. あとは作成したグループに `wilber` を追加します (ユーザは 手順5.1「LDAP ユーザの作成」で作成しています):

   ```
   > sudo dsidm LDAP1 group add_member SERVER_ADMINS \
   uid=wilber,ou=people,dc=LDAP1,dc=COM
   added member: uid=wilber,ou=people,dc=LDAP1,dc=COM
   ```

5.6.4 ユーザ／グループの削除、およびグループからのユーザ脱退

ユーザの削除やグループからのユーザ脱退、そしてグループそのものの削除についても、`dsidm` コマンドで行います。下記の例では、server_admins グループから wilber ユーザを脱退させています:

```
> sudo dsidm LDAP1 group remove_member SERVER_ADMINS \
uid=wilber,ou=people,dc=LDAP1,dc=COM
```

次にユーザを削除します:

```
> sudo dsidm LDAP1 user delete \
uid=wilber,ou=people,dc=LDAP1,dc=COM
```

さらにグループを削除します:

```
> sudo dsidm LDAP1 group delete SERVER_ADMINS
```

5.7 プラグインの管理

下記のように入力して実行することで、有効化されているプラグインと無効化されているプラグインの両方を一覧表示することができます。ここでは 389 Directory Server のインスタンス名ではなく、サーバのホスト名を指定することに注意してください。下記はホスト名が `LDAPSERVER1` である場合の例です:

```
> sudo dsconf -D "cn=Directory Manager" ldap://LDAPSERVER1 plugin list
Enter password for cn=Directory Manager on ldap://LDAPSERVER1: PASSWORD

7-bit check
Account Policy Plugin
Account Usability Plugin
ACL Plugin
ACL preoperation
[...]
```

下記のコマンドは、5.8項「LDAP 認証管理のための SSSD の使用」で利用している `MemberOf` プラグインを有効化するためのコマンドです。`MemberOf` はユーザの検索を簡略化するための仕組みで、コマンドを 1 つ実行するだけで、ユーザが所属するグループの一覧を出力することができます。`MemberOf` プラグインがないと、グループへの所属情報を検索するのに複数回のコマンド実行が必要になってしまいます。

```
> sudo dsconf -D "cn=Directory Manager" ldap://LDAPSERVER1 plugin memberof enable
```

なお、コマンド内で使用するプラグイン名は、小文字で指定しなければならないことに注意してください。そのため、一覧で表示される名前とは異なる指定になります。プラグイン名の指定が誤っていると、下記のようなエラーメッセージが表示されます:

```
dsconf instance plugin: error: invalid choice: 'MemberOf' (choose from
'memberof', 'automember', 'referential-integrity', 'root-dn', 'usn',
'account-policy', 'attr-uniq', 'dna', 'linked-attr', 'managed-entries',
'pass-through-auth', 'retro-changelog', 'posix-winsync', 'contentsync', 'list',
'show', 'set')
```

プラグインを有効化した場合は、サーバの再起動が必要です:

```
> sudo systemctl restart dirsrv@LDAPSERVER1.service
```

次にプラグインの設定を行います。下記の例では、`MemberOf` プラグインを全ての項目に対して有効化します。なお、ここではサーバのホスト名ではなく、インスタンス名を指定してください:

```
> sudo dsconf LDAP1 plugin memberOf set --scope dc=example,dc=com
Successfully changed the cn=MemberOf Plugin,cn=plugins,cn=config
```

`MemberOf` プラグインを有効化して設定を行うと、全ての新しいユーザやグループは自動的に `MemberOf` のターゲットとして設定されます。ただし、それまでに作成されたユーザやグループはそうではありません。そのため、手作業でそれらを設定する必要があります:

```
> sudo dsidm LDAP1 user modify suzanne add:objectclass:nsmemberof
Successfully modified uid=suzanne,ou=people,dc=ldap1,dc=com
```

これで suzanne 本人の情報とメンバーの情報を、単一のコマンドで出力することができるようになります:

```
> sudo dsidm LDAP1 user get suzanne
dn: uid=suzanne,ou=people,dc=ldap1,dc=com
cn: suzanne
displayName: Suzanne Geeko
gidNumber: 102
homeDirectory: /home/suzanne
memberOf: cn=SERVER_ADMINS,ou=groups,dc=ldap1,dc=com
```

多数のユーザを修正する場合はそれなりの負荷がかかります。下記の例では、1 つの `fixup` コマンドで既存の全ユーザに対して `MemberOf` を設定しています:

```
> sudo dsconf LDAP1 plugin memberof fixup -f '(objectClass=*)' dc=LDAP1,dc=COM
```

5.7.1 389 Directory Server でのサポート対象外プラグイン

下記のプラグインは、389 Directory Server においてサポートされておりません:

- Distributed Numeric Assignment (分散型数値配置; DNA) プラグイン
- Managed Entries Plug-in (管理項目プラグイン; MEP)
- Posix Winsync プラグイン

5.8 LDAP 認証管理のための SSSD の使用

システムセキュリティサービスデーモン (System Security Services Daemon (SSSD)) は、リモートのユーザに対する認証や識別、アクセス制御などを行うためのシステムです。本章では、389 Directory Server と SSSD を利用して、ユーザ認証やユーザ識別を行うための手順を説明しています。

SSSD は LDAP サーバとクライアントとの間を仲介します。LDAP のほか、Active Directory や Kerberos などのバックエンドにも対応しています。サービス側としては SSH, PAM, NSS, sudo などに対応しています。SSSD は ID や資格情報をキャッシュする仕組みが存在することから、性能面だけ

でなく柔軟性も兼ね備えています。キャッシュ機能は 389 Directory Server サーバの負荷を減らすことにつながるばかりか、バックエンドが接続不可能な状態になっても、認証機能を継続して提供することができるようになります。

なお、ネームサービスキャッシュデーモン (Name Services Caching Daemon (nscd)) がネットワーク内で動作している場合は、無効化するか削除しておいてください。nscd は passwd, group, hosts, service, netgroup などの名前解決要求のみをキャッシュする仕組みであることから、SSSD と競合してしまうためです。

LDAP サーバをプロバイダ (提供元) として使用する場合、SSSD のインスタンスはプロバイダに対するクライアントとして動作します。389 DS サーバ内で SSSD を動作させてもかまいませんが、異なるマシンで動作させることで、389 DS サーバが接続不可能な状況に陥った場合でもサービスを継続させることができます。下記の手順では、SSSD クライアントのインストールと設定について説明しています。なお、下記では 389 DS のインスタンス名が `LDAP1` であるものとして記しています:

1. まずは `sssd` と `sssd-ldap` の各パッケージをインストールします:

```
> sudo zypper in sssd sssd-ldap
```

2. まずは `/etc/sssd/sssd.conf` ファイルをバックアップしておきます:

```
> sudo old /etc/sssd/sssd.conf
```

3. 次に新しい SSSD の設定テンプレートを作成します。出力ファイル名には `sssd.conf` と `ldap.conf` のいずれかの名前を指定します。なお `display` を指定すると、標準出力に出力を行います。下記の例では、`/etc/sssd/sssd.conf` ファイルにクライアント設定を出力しています:

```
> sudo cd /etc/sssd
> sudo dsidm LDAP1 client_config sssd.conf
```

4. 出力された内容を確認して、環境に合わせた変更を行います。下記は `/etc/sssd/sssd.conf` ファイルの設定例 (全体) です。



重要: MemberOf について

この LDAP のアクセスフィルタは、`MemberOf` プラグインを使用しています。詳しくは 5.7項「プラグインの管理」をお読みください。

```
[sssd]
services = nss, pam, ssh, sudo
config_file_version = 2
domains = default
```

```
[nss]
homedir_substring = /home

[domain/default]
# 巨大なグループを扱う場合（たとえばメンバーが 50 人を超えるような場合）は、
# True を指定してください
ignore_group_members = False
debug_level=3
cache_credentials = True
id_provider = ldap
auth_provider = ldap
access_provider = ldap
chpass_provider = ldap

ldap_schema = rfc2307bis
ldap_search_base = dc=example,dc=com
# ldaps を使用しておくことを強くお勧めします
ldap_uri = ldaps://ldap.example.com
ldap_tls_reqcert = demand
ldap_tls_cacert = /etc/openldap/ldap.crt
ldap_access_filter = (|(memberof=cn=<login group>,ou=Groups,dc=example,dc=com))
enumerate = false

ldap_user_member_of = memberof
ldap_user_gecos = cn
ldap_user_uuid = nsUniqueId
ldap_group_uuid = nsUniqueId
ldap_account_expire_policy = rhds
ldap_access_order = filter, expire
# 下記の行を /etc/ssh/sshd_config に追加してください:
#  AuthorizedKeysCommand /usr/bin/sss_ssh_authorizedkeys
#  AuthorizedKeysCommandUser nobody
ldap_user_ssh_public_key = nsSshPublicKey
```

5. ファイルの所有者を root にして、root のみが読み書きできるように設定します:

```
> sudo chown root:sssd /etc/sssd/sssd.conf
> sudo chmod 640 /etc/sssd/sssd.conf
```

6. SSSD サーバ内の `/etc/nsswitch.conf` 設定ファイルを編集し、下記のような行が含まれるようにします:

```
passwd: compat sss
group:  compat sss
shadow: compat sss
```

7. SSSD サーバ内の PAM 設定である `common-account-pc` , `common-auth-pc` , `common-password-pc` , `common-session-pc` をそれぞれ編集します。openSUSE Leap では `pam-config` というコマンドを使用することで、これら全てのファイルを一括変更することができます:

```
> sudo pam-config -a --sss
```

8. あとは修正後の設定を確認します:

```
> sudo pam-config -q --sss
auth:
account:
password:
session:
```

9. さらに、389 Directory Server サーバ内にある `/etc/dirsrv/slapd-LDAP1/ca.crt` ファイルを SSSD サーバ内の `/etc/openldap/certs` にコピーして、ハッシュを再作成します:

```
> sudo c_rehash /etc/openldap/certs
```

10. SSSD を有効化して開始します:

```
> sudo systemctl enable --now sssd
```

systemctl での sssd.service の制御方法については、第4章「YaST を利用した認証クライアントの設定」をお読みください。

5.8.1 サポート対象外のパスワードハッシュと認証方式

`nsslapd-rootpwstoragescheme` や `passwordStorageScheme` の設定値として、もしくはアカウントポリシーオブジェクト内での `passwordStorageScheme` の値としては、下記の値はサポート対象外となっております:

- SHA
- SSHA
- SHA256
- SSHA256
- SHA384
- SSHA384
- SHA512

- SSHA512
- NS-MTA-MD5
- clear
- MD5
- SMD5



注記

なお、これらの値を含むデータベースからのインポートに関しては、`nsslapd-enable-upgrade-hash` が `on` (デフォルト値: `on`) になっていればサポート対象内となります。

5.9 OpenLDAP から 389 Directory Server への移行

OpenLDAP は 廃止予定となり、389 Directory Server に置き換えられています。SUSE ではこの移行作業を支援するため、`389-ds` パッケージ内に `openldap_to_ds` というユーティリティを提供しています。

`openldap_to_ds` は移行作業をできる限り自動的に実施できるよう設計されています。ですが、それぞれの LDAP 環境は異なるものであることから、全ての環境に適合するツールを作成するのは不可能です。そのため、必要に応じて手作業による介入が必要となるほか、本番環境の移行にあたっては、あらかじめ移行作業のテストを行う必要もあります。



重要: `help` ページの参照について

移行ツールである `openldap_to_ds` を使用する前に、あらかじめ `openldap_to_ds --help` コマンドで表示される内容をよくお読みになることを強くお勧めします。ここには移行ツールに用意されている機能の一覧のほか、制限事項に関する説明も書かれているため、誤った仮定に基づく誤解を防ぐことができるようになっています。

5.9.1 OpenLDAP からのテスト移行

OpenLDAP と 389 Directory Server との間にはそれなりの差異が存在することから、移行を行うにあたっても事前のテストと調整を繰り返し実施する必要があります。そのため、手早く移行テストを実施して、必要な追加手順を素早く洗い出すことが肝心です。

あらかじめ用意しておくべきもの:

- 389 Directory Server インスタンスの動作するマシン
- OpenLDAP `slapd` の設定ファイル、もしくは ldif 形式による動的な設定ディレクトリ
- OpenLDAP データベースの ldif 形式によるフルバックアップ

slapd の設定が動的な ldif 形式ではない場合、まずは `slaptest` を利用して動的なコピーを作成します。`/root/slapd.d/` のような `slapd.d` ディレクトリを作成して、下記のコマンドを実行します:

```
> sudo slaptest -f /etc/openldap/slapd.conf -F /root/slapd.d
```

上記を実行すると、下記のようにいくつかのファイルが作成されるはずです:

```
> sudo ls /root/slapd.d/*

/root/slapd.d/cn=config.ldif

/root/slapd.d/cn=config:
cn=module{0}.ldif  cn=schema.ldif                 olcDatabase={0}config.ldif
cn=schema          olcDatabase={-1}frontend.ldif  olcDatabase={1}mdb.ldif
```

サフィックスごとに 1 つの ldif ファイルが作成されます。以降の例ではサフィックスが dc= `LDAP1` ,dc= `COM` であるものとします。また、`/etc/openldap/slapd.conf` 形式を使用している場合は、下記のようなコマンドを実行することで、ldif のバックアップファイルを作成することができます:

```
> sudo slapcat -f /etc/openldap/slapd.conf -b dc=LDAP1,dc=COM \
-l /root/LDAP1-COM.ldif
```

あとは `openldap_to_ds` ユーティリティを利用して設定とファイルを分析させ、移行プランを表示させます (この時点では何も変更は行われません):

```
> sudo openldap_to_ds LDAP1\
/root/slapd.d /root/LDAP1-COM.ldif.ldif
```

前述のとおり上記では何も変更されませんが、下記のような出力が現れるはずです:

```
Examining OpenLDAP Configuration ...
Completed OpenLDAP Configuration Parsing.
Examining Ldifs ...
Completed Ldif Metadata Parsing.
The following migration steps will be performed:
 * Schema Skip Unsupported Attribute -> otherMailbox (0.9.2342.19200300.100.1.22)
 * Schema Skip Unsupported Attribute -> dSAQuality (0.9.2342.19200300.100.1.49)
 * Schema Skip Unsupported Attribute -> singleLevelQuality (0.9.2342.19200300.100.1.50)
 * Schema Skip Unsupported Attribute -> subtreeMinimumQuality (0.9.2342.19200300.100.1.51)
 * Schema Skip Unsupported Attribute -> subtreeMaximumQuality (0.9.2342.19200300.100.1.52)
 * Schema Create Attribute -> suseDefaultBase (SUSE.YaST.ModuleConfig.Attr:2)
```

```
 * Schema Create Attribute -> suseNextUniqueId (SUSE.YaST.ModuleConfig.Attr:3)
[...]
 * Schema Create ObjectClass -> suseDhcpConfiguration (SUSE.YaST.ModuleConfig.OC:10)
 * Schema Create ObjectClass -> suseMailConfiguration (SUSE.YaST.ModuleConfig.OC:11)
 * Database Reindex -> dc=example,dc=com
 * Database Import Ldif -> dc=example,dc=com from example.ldif -
excluding entry attributes = [{'structuralobjectclass', 'entrycsn'}]
No actions taken. To apply migration plan, use '--confirm'
```

あとは下記のように実行することで、実際の移行を行うことができます。変更しない場合とは出力が異なることに注意してください:

```
  > sudo openldap_to_ds LDAP1 /root/slapd.d /root/LDAP1-COM.ldif --confirm
Starting Migration ... This may take some time ...
migration: 1 / 40 complete ...
migration: 2 / 40 complete ...
migration: 3 / 40 complete ...
[...]
Index task index_all_05252021_120216 completed successfully
post: 39 / 40 complete ...
post: 40 / 40 complete ...
# Migration complete!
----------------------
You should now review your instance configuration and data:
 * [ ] - Create/Migrate Database Access Controls (ACI)
 * [ ] - Enable and Verify TLS (LDAPS) Operation
 * [ ] - Schedule Automatic Backups
 * [ ] - Verify Accounts Can Bind Correctly
 * [ ] - Review Schema Inconistent ObjectClass -> pilotOrganization
 (0.9.2342.19200300.100.4.20)
 * [ ] - Review Database Imported Content is Correct -> dc=ldap1,dc=com
```

移行が完了すると `openldap_to_ds` は、実施しておかなければならない移行後のチェック作業の一覧を表示します。この作業はいずれも移行を最適に行うために必要な手順ですので、本番環境でも同じようにしておくことをお勧めします。あとはテストクライアントとアプリケーションを、移行後の 389 Directory Server インスタンスに向けてテストを行ってください。



重要: ロールバックプランの策定

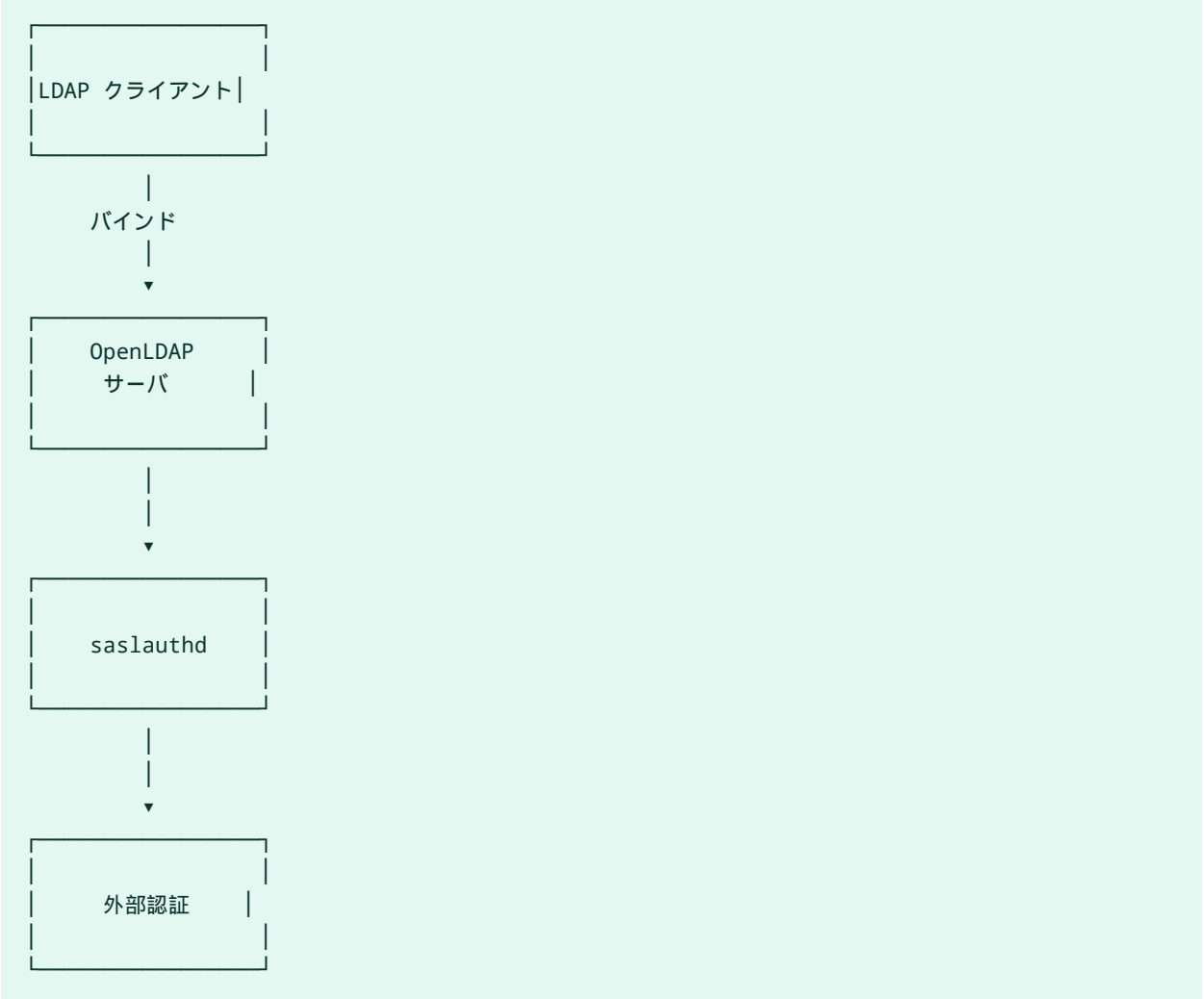
移行時、もしくは移行後に何らかの問題に直面することを想定して、ロールバックプラン (手戻りのための手順) と移行の成功判断基準を策定しておくことが重要です。具体的には、動作させなければならないクライアントやアプリケーションは何か、後から移行しても問題のないクライアントやアプリケーションは何か、それらでテストしておくべきことは何か、どのようにして手戻りの被害を最小限にするのか、巻き込むべきチームはどこなのかを事前に決めておく必要があります。

用途や範囲などが環境によって様々に異なることから、画一的な移行プラン／手戻りプランを策定することは難しいモノです。まずは移行処理をおおまかにテストしてみて、そこから少しずつ問題点を洗い出していくのがよいでしょう。また、下記のような一般的な手法も役に立つはずです:

- ホスト名や DNS の TTL を、一般的な 48 時間から 5 分に短縮しておくことで、手戻りを素早く実施できるようになります。
- 全てのデータ同期や受信データ処理を一時的に停止してください。これにより、移行時に OpenLDAP 環境が変化しなくなります。
- 移行前に 389 Directory Server のホストをよく確認しておくことも重要です。
- テスト移行時に発生した細かい問題などを詳しく記録しておくことも重要です。

5.9.2 saslauthd を使用する OpenLDAP サーバのテスト移行

OpenLDAP 環境の場合、パススルー型のユーザ認証を行うのに `saslauthd` を使用することがあります。この場合、認証処理では下記のようなコンポーネントが関わってきます:



設定が正しいことを確認したり、そこから先のトラブルシューティングを実施したりする場合は、下記のような情報が必要となります:

- OpenLDAP は、ユーザの `userPassword` 属性が `userPassword:{SASL}ユーザ名@レルム` の形式になっていることを検出すると、パススルー認証を使用するユーザであると判断します。なお、パススルー認証を使用するには、OpenLDAP サーバの構築時に `--enable-spasswd` オプションを指定しておく必要があります。
- OpenLDAP では、`saslauthd` との接続に関する設定を `/usr/lib/sasl2/slapd.conf` から取得します。

- `saslauthd` は、`/etc/sysconfig/saslauthd` 内に設定されたコマンドラインパラメータを利用して、関連するモジュールを検出します。
- `saslauthd` のバックエンドモジュールは、`man saslauthd` に書かれているとおり、独自の設定ファイルを使用します。

OpenLDAP でのパススルー認証に関する詳細については、公式文書である OpenLDAP Admin Guide (https://openldap.org/doc/admin24/security.html#Pass-Through%20authentication) ↗ (英語) をお読みください。

5.9.2.1 SASL パススルー認証を使用している環境での OpenLDAP から 389 Directory Server への移行

OpenLDAP で SASL によるパススルー認証を使用している場合、389 Directory Server への移行は下記の手順が最適です:

1. まずは移行前に、OpenLDAP サーバ内で `testsaslauthd` を実行し、問題が発生しないことを確認しておきます。

   ```
   > sudo testsaslauthd -u ユーザ名@レルム -p パスワード
   ```

   `saslauthd` は、レルムの指定から使用すべき認証バックエンドを判断し、ユーザ名は識別子として認証を行います。

2. 389 Directory Server から `saslauthd` に接続できるようにするため、`pam_saslauthd` パッケージをインストールします。

   ```
   > sudo zypper install -y pam_saslauthd
   ```

3. `openldap_to_ds` コマンドを実行して、OpenLDAP から 389 Directory Server への移行処理を行います。移行処理に関する詳細は、5.9.1項「OpenLDAP からのテスト移行」をお読みください。



注記

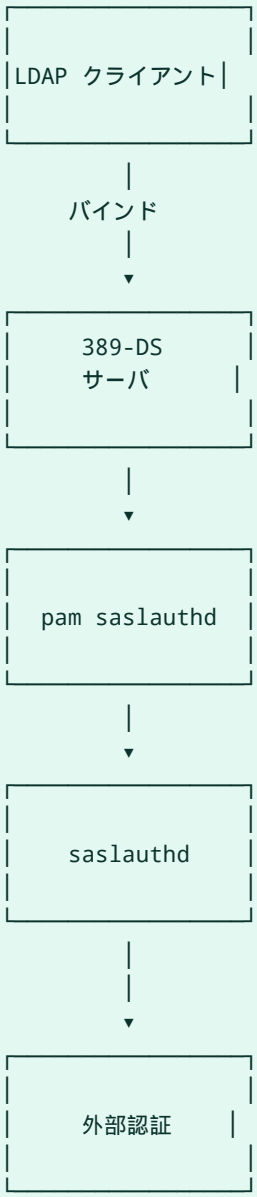
なお、`openldap_to_ds` を実行すると、ユーザの `userPassword` 属性が `userPassword: {SASL}ユーザ名@レルム` 形式になっていることを検出すると、この属性は削除され、`nsSaslauthId` 属性に `nsSaslauthId: ユーザ名@レルム` が設定されます。これに加えて、この属性に対応するよう、`objectClass: nsSaslauthAccount` が自動的に追加されます。

4. 移行が終わったあとは、下記のようなコマンドを実行して、PAM のパススルー認証が正しく設定されているかどうかを確認します:

```
> sudo dsconf インスタンス名 plugin pass-through-auth show
```

```
> sudo dsconf インスタンス名 plugin pam-pass-through-auth list
```

移行が全て終わると、パススルー認証は下記のような構成になります:



5.9.2.2 saslauthd パススルー認証のトラブルシューティング

OpenLDAP から 389 Directory Server への移行前後で発生する `saslauthd` パススルー認証に関するトラブルシューティングについては、それぞれ下記を確認するとよいでしょう:

`testsaslauthd` が `ユーザ名@レルム` の形式で動作すること。

`testsaslauthd` での確認手順については、5.9.2項「saslauthd を使用する OpenLDAP サーバのテスト移行」をお読みください。
正しく動作しない場合は、下記を試してみてください:

- まずは `/etc/sysconfig/saslauthd` ファイル内で、`saslauthd` バックエンドモジュールが正しく設定されていることを確認してください。`saslauthd` のバックエンドモジュールに関する詳細と設定方法については、`man saslauthd` をお読みください。
- `/etc/sysconfig/saslauthd` ファイルに `SASLAUTHD_PARAMS="-d"` を設定することで、デバッグ出力を有効化することもできます。
- なお `saslauthd` のログは、`journalctl` で表示することができます。

PAM `saslauthd` が正しく動作すること。

PAM `saslauthd` が正しく動作するかどうかを確認したい場合は、https://github.com/kanidm/pam_tester にある `pam_tester` ツールをお使いください。



注記

なお、`pam_tester` ツールは公式にサポートされているツールではありません。

PAM パススルー認証プラグインが有効化されていること。

PAM パススルー認証プラグインの状態を確認するには、下記のようなコマンドを入力して実行します:

```
> sudo dsconf インスタンス名 plugin pam-pass-through-auth status
```

プラグインを有効化するには、下記のようなコマンドを入力して実行します:

```
> sudo dsconf インスタンス名 plugin pam-pass-through-auth enable
```

PAM パススルー認証プラグインが有効化されていること。

サーバインスタンスに対して PAM パススルー認証プラグインの設定がされているかどうかを確認するには、下記のようなコマンドを入力して実行します:

```
> sudo dsconf インスタンス名 plugin pass-through-auth show
```

サーバインスタンスのユーザに対して出力されたログ。

ログファイルは `/var/lib/サーバユーザ名/インスタンス名/error` にあります。

5.9.3 移行の計画

OpenLDAP は「様々な部品から構成されるソフトウエアパッケージ」であり、様々なカスタマイズを実施できることから、「画一的な手順」で移行ができるものではありません。まずは OpenLDAP の環境と設定、そして使用範囲をよく調査してください。調査の対象としては、下記のようなものがあります (下記だけであるとも限りません):

- レプリケーションのトポロジ
- 高可用性と負荷分散の設定
- 外部のデータフロー (データ管理, 人事管理, Active Directory など)
- 設定済みのオーバーレイ (389 Directory Server のプラグイン)
- クライアント側の設定とサーバ側に求められる機能
- スキーマのカスタマイズ内容
- TLS の設定

最終的に 389 Directory Server をどのように配置するのかについても、よく計画しておく必要があります。これはオーバーレイをプラグインで置き換えること以外の、サーバの配置やインストールに関する内容です。現在の環境をよく調査し、389 Directory Server の配置計画を立てたら、あとは移行プランの策定になります。OpenLDAP の環境と並行して 389 Directory Server の環境を動作させて、お互いに切り替えることができるようにしても良いでしょう。

OpenLDAP から 389 Directory Server への移行は一方通行です。相互運用の意味でもこれらの間には十分な差異がありますし、逆方向 (389 Directory Server から OpenLDAP へ) の移行は提供されていませんので、よく注意しておく必要があります。下記には、様々な類似性と差異の一覧を示しています。

機能	OpenLDAP	389 Directory Server	互換性
双方向レプリケーション	SyncREPL	389 DS 固有のシステム	いいえ
MemberOf	オーバーレイ	プラグイン	はい (シンプルな設定にのみ対応)

機能	OpenLDAP	389 Directory Server	互換性
外部認証	プロキシ	-	いいえ
Active Directory との同期	-	Winsync プラグイン	いいえ
内蔵スキーマ	OLDAP スキーマ	389 スキーマ	はい (移行ツールで対応)
独自スキーマ	OLDAP スキーマ	389 スキーマ	はい (移行ツールで対応)
データベースの取り込み	LDIF	LDIF	はい (移行ツールで対応)
パスワードのハッシュ化	各種	各種	はい (Argon2 を除く全ての形式に対応)
OpenLDAP と 389 DS のレプリケーション	-	-	いいえ (389 DS との同期は不可能)
時間ベースのワンタイムパスワード (TOTP)	TOTP オーバーレイ	-	いいえ (現時点でサポートされていません)
entryUUID	OpenLDAP の一部として提供	プラグイン	はい

5.10 TLS 証明書と鍵の取り込み

389 Directory Server の証明書や鍵を管理するには、`certutil`, `openssl`, `pk12util` のようなコマンドラインツールを使用します。

新しい 389 DS インスタンスを作成すると、`dscreate` は独自の証明機関を作成し、自己署名型のサーバ証明書を発行します。こちらはテスト用途にのみ使用されるべきもので、証明書のファイルは `/etc/dirsrv/slapd-インスタンス名/` 内に配置されます。

本番環境を構築する場合は、Let's Encrypt, CAcert.org, SSL.com など、第三者が発行する証明書を使用するようにしてください。それぞれサーバ証明書、クライアント証明書、ルート証明書が必要になります。

重要

また、Mozilla NSS (Network Security Services) ツールキットでは、証明書ストア内の証明書に対してニックネーム (名前) を設定しますが、サーバ用の証明書に対しては Server-Cert という名前を指定してください。

1. インスタンスから自己署名用の証明機関と証明書を削除するには、下記のコマンドを実行します:

```
> sudo dsctl インスタンス名 tls remove-cert Self-Signed-CA
> sudo dsctl インスタンス名 tls remove-cert Server-Cert
```

ここで、`インスタンス名` にはディレクトリサーバのインスタンス名を指定します。前章での手順では LDAP1 という名前を設定しているものです。

2. まずは証明書への署名を実施した証明機関の証明書をインポートします。

```
> sudo sudo dsctl インスタンス名 tls import-ca
  /path/to/CA/in/PEM/format/CA.pem  CA_の名前
```

ここで、**インスタンス名** にはディレクトリサーバのインスタンス名を、`/path/to/CA/in/PEM/format/CA.pem` には証明機関の証明書 (PEM 形式) のファイルに対するフルパスを指定します。また `CA_の名前` には、証明機関の名前を指定します。

3. あとはサーバ証明書とそれに対応する鍵をインポートします。

```
> sudo dsctl インスタンス名 tls import-server-key-cert
  /path/to/SERVER.pem /path/to/SERVER.key
```

ここで、**インスタンス名** にはディレクトリサーバのインスタンス名を、`/path/to/SERVER.pem` にはサーバ証明書 (PEM 形式) のファイルに対するフルパスを指定します。また `/path/to/SERVER.key` には、サーバ証明書に対応する機密鍵 (PEM 形式) のファイルに対するフルパスを指定します。

4. あとはインスタンスを再起動すると、新しい証明書を使用するようになります。

```
> sudo systemctl restart dirsrv@インスタンス名.service
```

ここで、**インスタンス名** にはディレクトリサーバのインスタンス名を指定します。

5.11 レプリケーションの設定

389 Directory Server では複数のサーバ間でデータベースをレプリケーション (複製) することができます。レプリケーションの構成にもよりますが、下記のような利点を提供することになります:

- 性能と応答時間の高速化
- 耐障害性と冗長性の確保
- 負荷分散
- 高可用性

データベースは複製可能なディレクトリの最小単位でもあります。データベース全体を複製することはできますが、データベース内のサブツリーのみを複製することはできません。また、1 つのデータベースは必ず 1 つのサフィックスに結びつけられるものであり、2 つまたはそれ以上のデータベースにまたがったサフィックスを複製することはできません。

また、複製すべきデータを他に送信するサーバをサプライヤ (supplier) と呼びます。逆に、サプライヤからのデータを受け取るサーバをコンシューマ (consumer) と呼びます。レプリケーションは常にサプライヤ側から行われ、1 つのサプライヤからは複数のコンシューマに送信することができます。また一般に、マルチサプライヤ (multi-supplier) レプリケーションの場合を除いて、サプライヤは読み書き可能なレプリカとして、コンシューマは読み込み専用のサーバとして提供されます。マルチサプライヤレプリケーションの場合、サプライヤとコンシューマを兼ねるサーバとなります。

5.11.1 非同期書き込み

389 DS は他のデータベースとは異なるレプリケーション管理を行います。レプリケーションは非同期でありながらも、結果整合性を持つ仕組みになっています。つまり、下記のような動作になります:

- サーバへの書き込みや変更は、即時に受け入れられます。
- 特定のサーバ内に書き込まれたあと、他のサーバにレプリケーションされて見えるようになるまでの間に、時間的な遅れが発生します。
- いずれかのサーバ内での書き込みに矛盾が発生した場合、特定のポイントまでロールバック (巻き戻される) 可能性があります。
- レプリケーションの遅延により、特定の時点で各サーバ内に存在するデータが異なる可能性があります。

一般的に LDAP は、「書き込みの少ない」データベースであるため、これらの特徴を考慮しても、サーバが一貫性のない状態にはならず、常に既知の安定状態に居続けることができます。また、この安定状態からの変更は少しずつしか行われないため、日々の運用ではこのような欠点に気付くようなことはありません。

5.11.2 構成設計

レプリケーションの構成 (トポロジ) を設計するにあたっては、下記のような要素を決めておく必要があります。

- レプリケーションに対する要件: 高可用性、物理的な配置場所、読み込み性能やそれらの組み合わせなど。
- 構成内に配置するレプリカ (ノード、サーバ) 数。
- データの流れとその方向。構成内に流入するデータと流出するデータ。
- クライアント側からリクエストを送信する際の、ノード選択方式 (クライアント側で LDAP URI を複数個指定して分散させるか、SRV レコードで分散させるか、ロードバランサを用いるか等) 。

これらの要素はいずれも構成そのものに影響します (詳しい構成例については 5.11.3項「レプリケーションの設計例」をお読みください) 。

5.11.3 レプリケーションの設計例

下記の章では、2 ノードから 6 ノードまでの 389 Directory Server ノードを使用する、いくつかのレプリケーション構成例を示しています。なお、レプリケーションを構成する際の最大サプライヤ数は 20 ノードまでです。実際の運用経験からすると、レプリケーション効率を高めるための最適なノード数は最大でも 8 ノードまでと考えられています。

5.11.3.1 2 つのレプリカ

例 5.4: 2 つのサプライヤレプリカ



例5.4「2 つのサプライヤレプリカ」には S1 と S2 という 2 つのレプリカが示されています。これらは双方向レプリケーションとして構成する場合の例で、これらのノードはいずれもサプライヤとコンシューマの両方を兼ね備えた存在となります。S1 と S2 は別々のデータセンター内であっても、同じデータセン

ター内であってもかまいません。LDAP URI や負荷分散装置、DNS SRV レコードのいずれかでクライアント側からのアクセスを分散させることができます。また、これは高可用性を提供する際の最もシンプルな構成でもあります。ただし、一方のサーバで障害が発生してオフラインになった場合、他方のサーバでは全てのクライアントからの処理を受け付ける必要があることに注意してください。また、2 ノードのレプリケーションでは一方のノードがオフラインになった場合、残った 1 つのノードのみで全ての処理を行わなければならないことから、水平分散としては不適切です。

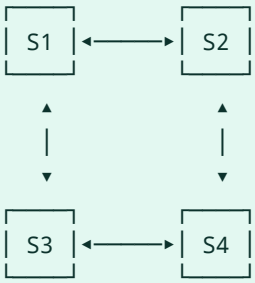


注記: 既定の構成

2 ノードの構成は最も簡単に管理できることから、既定の構成と考えられます。この構成を初期構成として、必要に応じて構成を広げていってもかまいません。

5.11.3.2 4 つのサプライヤレプリカ

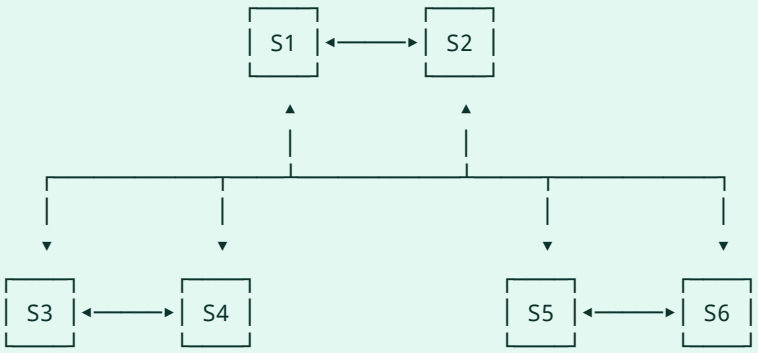
例 5.5: 4 つのサプライヤレプリカ



例5.5「4 つのサプライヤレプリカ」は 4 つのサプライヤレプリカを構成する場合の例です。これらは相互にデータを同期する構成で、4 つのデータセンターに分散させても、1 つのデータセンターごとに 2 つのサーバを配置してもかまいません。4 つのデータセンターに分散させた場合、各ノードは 100% のクライアント負荷を受け付ける処理能力が必要となります。2 つのデータセンターに 2 つのサーバを配置した場合は、各ノードは 50% のクライアント負荷を受け付ければ十分になります。

5.11.3.3 6 つのレプリカ

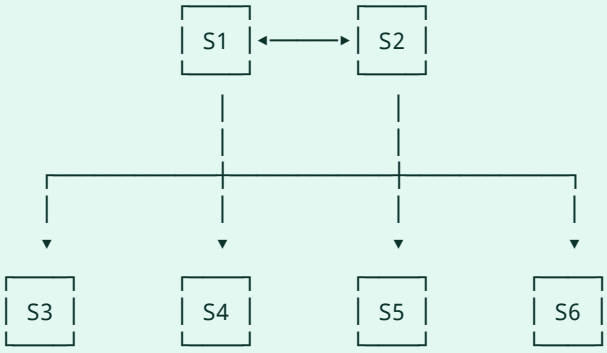
例 5.6: 6 つのレプリカ



例5.6「6 つのレプリカ」は各サーバ対を別々の場所に配置した場合の構成例です。S1 と S2 はサプライヤ、S3, S4, S5, S6 は S1 と S2 のコンシューマとして構成します。また、各サーバ対はお互いにレプリケーションしているものとし、S3, S4, S5, S6 でも書き込み要求は受け付けるものとしますが、ほとんどのレプリケーションは S1, S2 で行われます、このような構成は、高可用性と分散処理の両方を、地理的な離れた場所から提供する際の例となります。

5.11.3.4 読み込み専用コンシューマのある 6 つのレプリカ

例 5.7: 読み込み専用コンシューマのある 6 つのレプリカ



例5.7「読み込み専用コンシューマのある 6 つのレプリカ」では、S1 と S2 の両方がサプライヤとなり、残りの 4 つのサーバは読み込み専用のコンシューマとして構成する場合の例となります。全ての書き込み要求は S1, S2 でのみ行われ、残りの 4 つのサーバに複製していく構成です。読み込み専用のコンシューマには、データ開示を最小限に留めるため、データベースの一部や項目の一部のみを保存させるようにすることもできます。このように部分的な複製のみを持つ構成は DMZ のようなネットワークを持つ場合に最適で、この場合は DMZ 内に配置したコンシューマへの書き込みはできなくなります。

5.11.4 用語

ここまでに説明してきた 389 DS の構成例やこの後の説明では、様々な用語を使用しています。下記ではそれら用語に対する説明を示しています。

レプリカ
: データベースを持つ 389 DS のインスタンスを意味しています。

読み書き可能なレプリカ
: 読み込みと書き込みの両方の要求を受け付けることができる、データベースの完全コピーを持つレプリカを意味しています。

読み込み専用レプリカ
: 読み込み要求のみを受け付けることができる、データベースの完全コピーを持つレプリカを意味しています。

部分的な複製のみを持つレプリカ
: データベースのうちの一部のみを複製し、読み込み要求のみを受け付けることができるレプリカを意味しています。

サプライヤ
: 他のレプリカにデータベースの内容を提供することができるレプリカを意味しています。

コンシューマ
: 他のレプリカからデータを受け取って、自身のデータベースに書き込むレプリカを意味しています。

レプリケーション同意
: 他のレプリカとの関係性 (サプライヤ／コンシューマ) に対するサーバ設定を意味しています。

構成
: レプリケーション同意を介したレプリカ同士の接続構成を意味しています。

レプリカ ID
: レプリケーション構成内で重複しない 389 Directory Server のインスタンス識別子を意味しています。

レプリケーションマネージャ
: ディレクトリ内でレプリケーション権限を持つアカウントを意味しています。

5.11.5 レプリケーションの設定

最小構成例として、まずは 2 ノードの双方向レプリケーションと 1 ノードの読み込み専用サーバを構成してみます。下記の例では、読み書き可能なレプリカのサーバ名は RW1, RW2 とし、読み込み専用のレプリカのサーバ名は RO1 とします (もちろん必要に応じて自由な名前を設定してかまいません) 。

この構成内にあるサーバには、全て同じサフィックスを設定する必要があります。また、初期状態では RW1 にのみデータが存在するものとします。

5.11.5.1 2 ノードレプリケーションの設定

下記のコマンドは 2 ノード構成 (例5.4「2 つのサプライヤレプリカ」) の場合のコマンド例で、ホスト名を RW1, RW2 にしています (実際には独自の名前でかまいません) 。



警告: 強力なレプリケーションマネージャパスワードの作成について

レプリケーションマネージャのアカウントはディレクトリマネージャのアカウントと同等の権限を持つことから、セキュリティを確保するために強力なパスワードを設定しておくことを強くお勧めします。

各サーバで異なるレプリケーションマネージャのパスワードを設定する場合は、それぞれのサーバが使用するアカウントをよく覚えておいてください。たとえば RW1 側のレプリケーション同意を設定する場合、パスワードとして設定すべきなのは RW2 側のレプリケーションマネージャのパスワードとなります。

まずは RW1 側を設定します:

```
> sudo dsconf インスタンス名 replication create-manager
> sudo dsconf インスタンス名 replication enable \
--suffix dc=example,dc=com \
--role supplier --replica-id 1 --bind-dn "cn=replication manager,cn=config"
```

RW2 側の設定は下記のようになります:

```
> sudo dsconf インスタンス名 replication create-manager
> sudo dsconf インスタンス名 replication enable \
--suffix dc=example,dc=com \
--role supplier --replica-id 2 --bind-dn "cn=replication manager,cn=config"
```

上記のコマンドを実行することで、RW1, RW2 の両方にレプリケーション用のデータを設定することができます。なお、RW1 と RW2 では `replica-id` の値が異なっていることに注意してください。また、このコマンドを実行すると、レプリケーションマネージャのアカウントを作成することになります。このアカウントは、2 つのノードが互いに認証する際に使用するアカウントとなります。

これで RW1 と RW2 の間でレプリケーションを行う準備ができました。あとは RW1 から RW2 に対して、データを発信するための最初の合意を作成します。

```
> sudo dsconf インスタンス名 repl-agmt create \
--suffix dc=example,dc=com \
--host=RW2 --port=636 --conn-protocol LDAPS --bind-dn "cn=replication manager,cn=config" \
--bind-passwd パスワード --bind-method SIMPLE RW1_to_RW2
```

上記のコマンドだけではデータベースの同期が始まりません。同期を始めるには、初期化または再初期化と呼ばれる完全同期が必要となります。初期化または再初期化を行うと、RW2 側の全てのデータがリセットされ、RW1 と全く同じデータになります。これを行うには、下記のように入力して実行します:

```
> sudo dsconf インスタンス名 repl-agmt init \
--suffix dc=example,dc=com RW1_to_RW2
```

あとは RW1 側で下記のようなコマンドを入力して実行し、状況を確認します:

```
> sudo dsconf インスタンス名 repl-agmt init-status \
--suffix dc=example,dc=com RW1_to_RW2
```

同期が完了すると、`Agreement successfully initialized` というメッセージが表示されるはずです。それ以外のエラーメッセージが表示された場合は、エラーログを調べてみてください。エラーが表示されなければ、RW1 と RW2 は全く同じデータになっているはずです。

あとはデータを双方向に複製するため、RW2 から RW1 に対するレプリケーション合意も設定します:

```
> sudo dsconf インスタンス名 repl-agmt create \
--suffix dc=example,dc=com \
--host=RW1 --port=636 --conn-protocol LDAPS \
--bind-dn "cn=replication manager,cn=config" --bind-passwd パスワード \
--bind-method SIMPLE RW2_to_RW1
```

これで RW1 と RW2 の双方向レプリケーションが完成し、一方での変更が他方にも反映されるようになっています。いずれかのサーバで下記のようなコマンドを入力して実行し、状態を確認してください:

```
> sudo dsconf インスタンス名 repl-agmt status \
--suffix dc=example,dc=com \
--bind-dn "cn=replication manager,cn=config" \
--bind-passwd パスワード RW2_to_RW1
```

5.11.5.2 読み込み専用ノードの設定

読み込み専用ノードを作成するには、まずレプリケーションマネージャのアカウントとメタデータを作成するところから始めます。ここでのサーバ名は RO3 とします:



警告: 強力なレプリケーションマネージャパスワードの作成について

レプリケーションマネージャのアカウントはディレクトリマネージャのアカウントと同等の権限を持つことから、セキュリティを確保するために強力なパスワードを設定しておくことを強くお勧めします。

各サーバで異なるレプリケーションマネージャのパスワードを設定する場合は、それぞれのサーバが使用するアカウントをよく覚えておいてください。たとえば RW1 側のレプリケーション同意を設定する場合、パスワードとして設定すべきなのは RW2 側のレプリケーションマネージャのパスワードとなります。

```
> sudo dsconf インスタンス名 replication create-manager
> sudo  dsconf インスタンス名 \
replication enable --suffix dc=EXAMPLE,dc=COM \
--role consumer --bind-dn "cn=replication manager,cn=config"
```

なお、読み込み専用のレプリカに対しては、レプリカ ID の指定は不要であるほか、役割の指定を `consumer` にします。これにより、特殊な読み込み専用のレプリカ ID が割り当てられます。読み込み専用のレプリカを作成したら、あとは RW1, RW2 のそれぞれに対してレプリケーション同意を追加します。下記は RW1 側での実行例です:

```
> sudo dsconf インスタンス名 \
repl-agmt create --suffix dc=EXAMPLE,dc=COM \
--host=RO3 --port=636 --conn-protocol LDAPS \
--bind-dn "cn=replication manager,cn=config" --bind-passwd パスワード
--bind-method SIMPLE RW1_to_RO3
```

下記の例は、RW2 と RO3 との間でのレプリケーション同意を RW2 側で設定する場合の例です:

```
> sudo dsconf インスタンス名 repl-agmt create \
--suffix dc=EXAMPLE,dc=COM \
--host=RO3 --port=636 --conn-protocol LDAPS \
--bind-dn "cn=replication manager,cn=config" --bind-passwd パスワード \
--bind-method SIMPLE RW2_to_RO3
```

これらの作業が完了したら、あとは RW1, RW2 から RO3 へデータベースの初期化を行います。下記の例では、RW2 から RO3 に初期化を行っています:

```
> sudo dsconf インスタンス名 repl-agmt init
--suffix dc=EXAMPLE,dc=COM RW2_to_RO3
```

5.11.6 監視とヘルスチェック

`dsconf` コマンドには監視オプションが用意されています。このコマンドを実行することで、レプリカ内で直接状態を確認することができるほか、他のホストからでも実行することができます。下記の例は RW1 内での実行例で、2 つのレプリカに対する状態の確認と、自分自身の確認をそれぞれ行っています:

```
> sudo dsconf -D "cn=Directory Manager" ldap://RW2 replication monitor
> sudo dsconf -D "cn=Directory Manager" ldap://RO3 replication monitor
> sudo dsconf -D "cn=Directory Manager" ldap://RW1 replication monitor
```

また、`dsctl` コマンドには `healthcheck` というオプションもあります。下記ではローカルの 389 DS インスタンスに対して、レプリケーションの状態確認を行っています:

```
> sudo dsctl インスタンス名 healthcheck --check replication
```

`-v` オプションを追加すると、healthcheck で行っていることの詳細を表示することができます:

```
> sudo dsctl -v インスタンス名 healthcheck --check replication
```

オプション無しで `dsctl インスタンス名 healthcheck` を実行すると、一般的な状態確認のみを行います。

下記のようなコマンドを実行することで、healthcheck を実行した際に行われる処理の内容を調べることもできます:

```
> sudo dsctl インスタンス名 healthcheck --list-checks
config:hr_timestamp
config:passwordscheme
backends:userroot:cl_trimming
backends:userroot:mappingtree
backends:userroot:search
backends:userroot:virt_attrs
encryption:check_tls_version
fschecks:file_perms
[...]
```

個別のチェックのみを実行したい場合は、下記のように入力して実行します:

```
> sudo dsctl インスタンス名 healthcheck \
--check monitor-disk-space:disk_space tls:certificate_expiration
```

5.11.7 バックアップの作成

レプリケーションを有効化した場合は、389 Directory Server のバックアップについてもいくつかの調整を行う必要があります (一般的なバックアップ方法については 5.5項「389 Directory Server でのバックアップと復元」をお読みください) 。具体的には、バックアップ時に `db2ldif` コマンドを実行し

ている場合、`--replication` フラグを指定して、レプリケーションのメタデータを最初にバックアップするようにしてください。また、バックアップは構成内の全てのサーバに対して行うものとし、バックアップからの復元に際しては、構成内の 1 ノードでのみ実施して、残りのノードは再初期化を行ってデータを複製してください。

5.11.8 レプリケーションの一時停止と再開

メンテナンスやその他の運用上の理由で、レプリケーションを一時的に停止することができます。なお、一時停止は変更履歴の保持日数で示された期間 (詳しくは 5.11.9項「Changelog max-age」をお読みください) まで続けることができます。

レプリケーションを一時停止するには、`repl-agmt` コマンドを使用します。RW2 で一時停止する場合は、下記のように入力して実行します:

```
> sudo dsconf インスタンス名 repl-agmt disable \
--suffix dc=EXAMPLE,dc=COM RW2_to_RW1
```

再開する場合は、下記のように入力して実行します:

```
> sudo dsconf インスタンス名 repl-agmt enable \
--suffix dc=EXAMPLE,dc=COM RW2_to_RW1
```

5.11.9 Changelog max-age

レプリカを何らかの理由でしばらくオフラインにする場合には、最大で changelog `max-age` で指定した時間までが許容されます。`max-age` オプションは過去の変更履歴を持つ長さを時間で指定するための値で、ここで指定したよりも過去の履歴は自動的に削除されます。

しばらくオフラインになっていたレプリカが復帰すると、通常のレプリケーションと同様に、他のレプリカとの間で同期が行われます。`max-age` で指定したよりも長い時間オフラインであった場合は、一貫性を失うことから、他のノードからの変更を拒否したり、他のノードへの変更送信を拒否したりするようになります。下記の例では、`max-age` を 7 日間に設定しています:

```
> sudo dsconf インスタンス名 \
replication set-changelog --max-age 7d \
--suffix dc=EXAMPLE,dc=COM
```

5.11.10 レプリカの削除

レプリカを削除するには、まず他のノードに対して変更点の送信を停止する必要があります。あとは削除するノードで受け付けていたレプリケーション合意を削除します。下記の例では、RW2 を削除する場合の例を示しています。まずは RW1 での発信側レプリケーション合意を削除します:

```
> sudo dsconf インスタンス名 repl-agmt delete \
--suffix dc=EXAMPLE,dc=COM RW1_to_RW2
```

あとは削除対象のレプリカ (この例では RW2) で、全ての送信側レプリケーション合意を削除します:

```
> sudo dsconf インスタンス名 repl-agmt delete \
--suffix dc=EXAMPLE,dc=COM RW2_to_RW1
> sudo dsconf インスタンス名 repl-agmt delete \
--suffix dc=EXAMPLE,dc=COM RW2_to_RO3
```

あとは RW2 インスタンスを停止します:

```
> sudo systemctl stop dirsrv@INSTANCE_NAME.service
```

最後に構成内に書かれているレプリカ ID を削除するため、`cleanallruv` コマンドを実行します。下記は RW1 で実行した場合の例です:

```
> sudo dsconf インスタンス名 repl-tasks cleanallruv \
--suffix dc=EXAMPLE,dc=COM --replica-id 2
> sudo dsconf インスタンス名 repl-tasks list-cleanruv-tasks
```

5.11.11 389 Directory Server でのレプリケーションの制限

389 Directory Server でレプリケーションを使用する場合、下記のサポート制限があります:

- 読み書き可能なノードは最大 8 個まで
- レプリケーションハブは最大 20 個まで
- 読み込み専用のサーバは最大 100 個まで
- 読み書き可能なメンバーとしての Winsync Active Directory コンシューマは最大 1 個まで

5.12 Microsoft Active Directory との同期

389 Directory Server では、Microsoft 社の Active Directory からユーザやグループの項目を取得する機能が提供されています。この機能を使用することで、通常は必要とされるドメインへの参加を行うことなく、389 DS を利用して Linux クライアントが識別情報を取得できるようになります。これにより、389 DS が Active Directory と相互に運用できるようになり、用途をさらに広げることができるようになっています。

5.12.1 同期トポロジの計画

同期の形態にもよりますが、最小構成では 1 台の 389 Directory Server と 1 台の Active Directory サーバで同期を構築することができます。ただし、Active Directory は完全なドメインコントローラでなければならず、読み込み専用のドメインコントローラ (RODC) であってはなりません。なお、389 DS はドメイン内の単一フォレストの内容のみを複製するため、グローバルカタログは不要です。

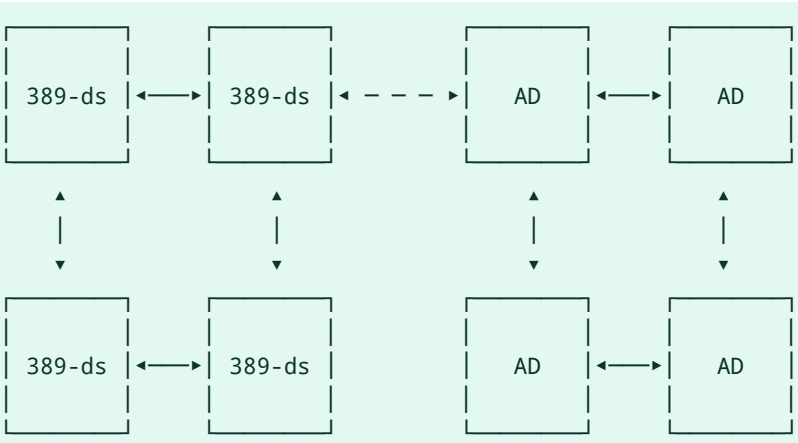
まずはデータの流通方向を選択します。AD から 389 DS だけでなく、389 DS から AD や、その両方を指定することもできます。



注記: パスワード同期について

389 DS と Active Directory との間では、パスワードの同期を行うことができません。将来的には Active Directory から 389 DS に対してパスワードを同期できるようになる予定です。

同期の構成を図に表すと、下記のようになります。389 Directory Server 内や Active Directory 内の構成が異なる場合もありますが、389 DS と Active Directory は 1 つの接続のみで賄われる点が最も重要です。また、これによって発生しうる、389 DS と AD の災害対策やバックアップ計画の策定も非常に重要です。これらを計画しておくことで、単一のレプリケーション接続であっても、正しく復元できるように構成できるからです。



5.12.2 Active Directory 側の事前要件

`Replicating Directory Changes` (ディレクトリ変更のレプリケート) の権限を持つセキュリティグループが必要となります。たとえば `Directory Server Sync` という名前のグループを作成します。具体的には、`Microsoft Metadirectory Services ADMA サービス アカウントの 'ディレクトリ変更のレプリケート' アクセス許可を付与する方法` (https://docs.microsoft.com/ja-JP/troubleshoot/windows-server/windows-security/grant-replicating-directory-changes-permission-adma-service) をお読みのうえ、手順に従って設定してください。



警告: 強力なセキュリティの必要性について

このグループ内のメンバーを設定する際は、ドメイン管理者と同等の注意を払うようにしてください。このグループのメンバーは Active Directory 環境内の機密情報を読み込むことができるため、ユーザに対しては乱数から生成された強力なパスワードを設定するとともに、グループ内のメンバー追加や削除が不適切に行われないよう注意してください。

また、このグループに所属するサービスアカウントも作成する必要があります。

また、389 DS と AD との通信を暗号化 (LDAPS) するため、Active Directory 環境には証明書の設定もしなければなりません。なお、Generic Security Services API/Kerberos (GSSAPI/KRB) は使用できません。

5.12.3 389 Directory Server 側の事前要件

389 Directory Server 側ではバックエンドデータベースを設定し、組織単位 (OU) の各項目を同期できるように設定する必要があります。

また、389 Directory Server 側ではレプリカ ID を設定して、読み込みと書き込みの両方ができるように設定する必要があります (レプリケーションの設定について、詳しくは 5.11項「レプリケーションの設定」をお読みください) 。

5.12.4 Active Directory から 389 Directory Server への同意の作成

下記は 389 Directory Server 側で実行すべきコマンドで、Active Directory から 389 Directory Server への同期合意を作成しています:

```
> sudo dsconf インスタンス名 repl-winsync-agmt create --suffix dc=example,dc=com \
  --host AD-ホスト名 --port 636 --conn-protocol LDAPS \
  --bind-dn "cn=サービスアカウント名,cn=USERS,dc=AD,dc=EXAMPLE,dc=COM" \
```

```
  --bind-passwd "パスワード" --win-subtree "cn=USERS,dc=AD,dc=EXAMPLE,dc=COM" \
  --ds-subtree ou=AD,dc=EXAMPLE,dc=COM --one-way-sync fromWindows \
  --sync-users=on --sync-groups=on --move-action delete \
  --win-domain AD-ドメイン adsync_agreement
```

合意を作成したあとは、初回の同期を実行します:

```
> sudo dsconf インスタンス名 repl-winsync-agmt init --suffix dc=example,dc=com
 adsync_agreement
```

下記のコマンドを使用することで、同期の状況を確認することができます:

```
> sudo dsconf インスタンス名 repl-winsync-agmt init-status --suffix dc=example,dc=com
 adsync_agreement
```



注記: 項目によっては同期されない問題について

初期の同期が成功していても、場合によっては同期されない項目が現れる場合があります。詳しくは `/var/log/dirsrv/slapd-インスタンス名/errors` 内にある 389 DS のログファイルをご覧ください。

下記のコマンドを入力して実行し、合意の状態を確認します:

```
> sudo dsconf インスタンス名 repl-winsync-agmt status --suffix dc=example,dc=com
 adsync_agreement
```

Active Directory と 389 Directory Server のトポロジ内でメンテナンスを実行する場合は、下記のようなコマンドを入力して実行し、合意を一時的に停止します:

```
> sudo dsconf インスタンス名 repl-winsync-agmt disable --suffix dc=example,dc=com
 adsync_agreement
```

一時停止した合意は、下記のように入力して実行することで再開できます:

```
> sudo dsconf インスタンス名 repl-winsync-agmt enable --suffix dc=example,dc=com
 adsync_agreement
```

5.13 さらなる情報

389 Directory Server に関する詳細については、下記をお読みください (いずれも本書記述時点では英語のみの提供です):

- 提供元のドキュメンテーションは https://www.port389.org/docs/389ds/documentation.html にあります。
- man dsconf

- man dsctl
- man dsidm
- man dscreate

6 Kerberos を利用したネットワーク認証

改訂履歴

2025-05-06

Kerberos は暗号化の機能も提供するネットワーク認証プロトコルです。本章では Kerberos の設定方法のほか、LDAP や NFS のようなサービスとの統合方法について説明しています。

6.1 考え方の概要

通常のオープンなネットワークの場合、ワークステーションがユーザを認証する際、パスワード以外の方法をとることができません。一般的なインストール環境では、ネットワーク内に存在する各サービスにアクセスするごとに、パスワードの入力が求められます。Kerberos を利用すると、ユーザの認証を一度行うだけで、他の連携サービスにも自動的に接続できるようになります。なお、ネットワークの機密を保持するため、下記の要件に適合していなければなりません:

- 各サービスを利用するユーザが自身の識別情報をきちんと管理し、第三者によって識別情報を利用されてしまうことがないようにすること。
- それぞれのネットワークサービスの提供側 (サーバ) でも、識別情報を保持すること。サーバ側でも識別情報を持っておかないと、攻撃者は容易にサーバになりすますことができてしまいますので、これによってサーバに送信されるはずの認証情報を窃取することができるようになってしまいます。このような考え方は、クライアントがサーバに対して認証を行うだけでなく、その逆の認証も行うことになることから、相互認証 と呼んでいます。

Kerberos では、強力に暗号化された認証の仕組みによって、このような要件を満たすようになっています。なお、本章での Kerberos の説明は、基本的な考え方にとどまります。より詳しい技術的な説明をご希望の場合は、Kerberos のドキュメンテーションをお読みください。

6.2 Kerberos で使用する用語

Kerberos の世界では、下記のような用語を使用します。

クレデンシャル (信任状)
: ユーザやクライアントは、利用したいサービスに対して、その利用を許可する旨を表す資格情報を提示する必要があります。Kerberos では、チケットとオーセンティケータと呼ばれる 2 種類のクレデンシャルが用意されています。

チケット (切符)

チケットとはサーバごとに発行されるクレデンシャルで、クライアントが特定のサービスを利用する際、その提供者 (サーバ) に対して提示するものです。チケットにはサーバの名前のほか、クライアントの名前とクライアントのインターネットアドレス、タイムスタンプと有効期限、ランダムなセッション鍵が書かれています。このデータの全ては、サーバ側の鍵で暗号化されます。

オーセンティケータ

オーセンティケータはチケットと組み合わせて使用するもので、チケットを提示しているクライアントが正当なクライアントであることを示すためのものです。オーセンティケータはクライアント名と IP アドレス、そしてクライアント側の時刻から構成されていて、クライアントと相手側のサーバのみが知りうるセッション鍵で全て暗号化されています。なお、チケットとは異なり、オーセンティケータは一度しか使用することができません。オーセンティケータはクライアント自身が構築します。

プリンシパル

Kerberos でのプリンシパルとは、チケットを割り当てた先を表すユニークな実体 (ユーザやサービス) を表します。プリンシパルは、下記のような形式で表されます:

```
プライマリ/インスタンス名@レルム
```

- **プライマリ:** プリンシパルの最初の部分です。プリンシパルがユーザを表すものである場合、この項目はユーザ名になります。
- **インスタンス名** (オプション) : プライマリ を特徴付ける、追加の情報を表します。この文字列と プライマリ との間は、`/` で区切ります。
  たとえば `tux@example.org` と `tux/admin@example.org` は、同じ Kerberos システム内に同時に存在しうるプリンシパルで、それぞれは別々のものとして扱われます。
- **レルム:** Kerberos の領域 (レルム) を表します。通常、レルムはドメイン名を大文字で表したものになります。

相互認証

Kerberos では、クライアントとサーバが相互に認証を行います。認証の際、通信の機密を守る目的で、セッション鍵を共有して使用します。

セッション鍵

セッション鍵とは、Kerberos が生成した一時的な機密鍵を意味します。セッション鍵はクライアント側が保持していて、クライアントとサービスを提供するサーバとの間で、チケットのやり取りを行う際に暗号化をするために使用します。

リプレイ

ネットワーク上を流れるほぼ全てのメッセージは、盗聴や盗難、偽装を行うことができてしまいます。Kerberos の世界では、攻撃者が特定のサービスに対するチケットやオーセンティケータを盗むことができてしまうことになりますので、このままでは非常に危険です。たとえそれがサーバ側に届いた後であったとしても、再送信 (リプレイ) を行うことで、特定のユーザになりすますことができてしまいます。しかしながら、Kerberos ではいくつかの仕組みを利用して、この問題を回避しています。

サーバ／サービス

サービス とは、何らかの処理を表す用語です。このサービスを提供するものを サーバ と呼んでいます。

6.3 Kerberos の動作概要

Kerberos はしばしば、第三者の信頼による認証サービスとも呼ばれます。これは、全てのクライアントが他のクライアントの識別情報を判断する際、Kerberos での判断結果を信頼して使用することから、そのように呼ばれています。Kerberos では、全てのユーザに対する情報と、それに対応する機密鍵を保持するデータベースとなっています。

Kerberos を正しく動作させるためには、認証とチケット発行許諾サーバを専用のマシンで動作させるようにしてください。また、このマシンに対しては、物理的にもネットワーク経由でも、管理者のみがアクセスできるようにしてください。また、このマシンで動作させるサービスはごく限られたもののみとし、`sshd` なども停止しておくことを強くお勧めします。

6.3.1 最初の通信

Kerberos に対する最初の通信は、一般的なネットワークシステムにおけるログイン処理に似通っています。まずはユーザ名を入力します。この情報とチケット発行許諾サービスの名前が認証サーバ (Kerberos) に送信されます。認証サーバ側でユーザ名が既知のものであった場合は、クライアントとチケット発行許諾サーバとの間で使用される、乱数を利用したセッション鍵を発行します。あとは認証サーバがチケット発行許諾サーバに対するチケットを発行します。このチケットには下記のような情報が含まれていて、これら全てが認証サーバとチケット発行許諾サーバのみが知りうるセッション鍵で暗号化されます:

- クライアントの名前とチケット発行許諾サーバの名前
- 現在時刻
- チケットの有効期限

- クライアントの IP アドレス
- 新しく生成されたセッション鍵

このチケットは、セッション鍵と共に暗号化された状態のまま、クライアント側に返却されます。ただし、この時点ではクライアントが使用する機密鍵を使用して暗号化を行います。機密鍵はユーザのパスワードを元にして作られるものであるため、Kerberos とクライアントのみが知りうる情報となります。クライアント側では応答を受け取ったあと、利用者に対してパスワードを尋ねるようになります。クライアント内では、このパスワードを機密鍵に変換して、認証サーバから届いたデータを解読する処理を行います。必要な情報が「開梱できた」あとは、パスワードを自身のメモリ内から消去します。チケット内には有効期間が示されていることから、この有効期間が切れるまでの間に新しいチケットの発行を要求することで、クライアント側は認証を必要な期間だけ使い続けられることになります。

6.3.2 サービスの要求

ネットワーク内のサーバが提供するサービスを利用するため、クライアント側のアプリケーションは、サーバに対して識別情報を提示する必要があります。そのため、アプリケーションはオーセンティケータを生成します。オーセンティケータには下記のような情報が含まれています:

- クライアント側のプリンシパル
- クライアントの IP アドレス
- 現在時刻
- チェックサム (クライアント側で選択)

これら全ての情報は、クライアント側で既に受信しているセッション鍵を利用して、暗号化が行われます。サーバ向けのオーセンティケータとチケットが揃ったら、あとはそれらをサーバに送信します。サーバはオーセンティケータを解読するためにセッション鍵のコピーを使用し、サービスを要求するクライアントに対して必要な情報を取得したあと、チケットに書かれている内容と比較します。これにより、サーバ側ではチケットとオーセンティケータが同じクライアントから送信されたものであるかを確認します。

サーバ側では特別なセキュリティを実装しない限り、この処理はリプレイ攻撃に対する標的となり得てしまいます。たとえば誰かが以前に送信したオーセンティケータとチケットを保存しておいて、後からそれを送信して認証をかいくぐる方法が考えられます。このような不正を防止する目的で、サーバ側では以前に受信したものと同じタイムスタンプのチケットは受け付けないようになっているほか、受信した時刻とタイムスタンプがあまりにも違いすぎる場合も、要求を受け付けないようになっています。

6.3.3 相互認証

Kerberos の認証は両方向に使用することができます。つまり、認証はクライアント側にだけ求められるものではありません。サーバ側でも、サービスを要求するクライアントに対して、自分自身を認証する必要があります。そのため、サーバ側ではオーセンティケータ自身を送信します。クライアント側のオーセンティケータのチェックサムに 1 を加えたあと、サーバとクライアントとの間で共有されているセッション鍵で暗号化を行います。クライアント側ではこの応答をもとに、サーバの正当性を確認して後続の処理を行うかどうかを判別します。

6.3.4 チケット発行許諾 - 全てのサーバへの通信

チケットは特定のサーバに対して使用するために設計されています。そのため、異なるサービスに接続する際には、新しいチケットの発行を受ける必要があります。Kerberos では、それぞれのサーバに対するチケットを取得する仕組みが用意されています。このサービスを「チケット発行許諾サービス」と呼びます。チケット発行許諾サービスは、それ自身が (ここまでに説明しているのと同じ) サービスであり、ここまでの説明と同じプロトコルでアクセスを行います。アプリケーション側で新しいチケットが必要になると、チケット発行許諾サーバにアクセスを行います。このリクエストには下記のような情報が含まれています:

- 要求者のプリンシパル
- チケット発行許諾チケット
- オーセンティケータ

他のサーバと同様に、チケット発行許諾サーバでもチケット発行許諾チケットと、オーセンティケータの内容を確認します。それらの情報がいずれも正当なものであった場合、チケット発行許諾サーバは元々のクライアントと新しく接続するサーバとの間で使用するセッション鍵を構築します。新しく接続するサーバに対するチケットには、下記のような情報が含まれています:

- クライアント側のプリンシパル
- サーバのプリンシパル
- 現在時刻
- クライアントの IP アドレス
- 新しく生成されたセッション鍵

新しいチケットにも有効期間が設定されています。この有効期間はチケット発行許諾チケットと同じか、もしくはサービスごとの既定値のうち、いずれか早いほうが設定されます。クライアントは、チケット発行許諾サービスからこのチケットとセッション鍵を受け取ります。ただし、この時点では元々のチケット発行

許諾チケットのセッション鍵で暗号化されています。そのため、クライアント側では利用者に対してパスワードを求めることなく、応答を解読できることになります。これにより、Kerberos はユーザの手を煩わせることなくチケットを取得できることになります。

6.4 Kerberos のユーザ側からの見た目

ユーザ側からの見た目では、Kerberos とのやり取りは端末のログイン時にのみ発生します。ログイン処理ではチケット発行許諾チケットの取得を行います。ログアウト時にはユーザ側の Kerberos チケットが自動的に破棄され、第三者が偽装することができないようにしています。

それぞれのチケットには有効期間が設定されているため、チケット発行許諾チケットに書かれた有効期間 (一般的には 10 時間程度) よりもユーザが長くログインしたままになることがあり得ます。しかしながら、ユーザ側では `kinit` を実行するだけで、新しいチケット発行許諾チケットを受け取ることができるようになっています。ここではパスワードを再入力することで、アクセス中の様々なサービスに対して個別に認証を行わずに済むようになっています。Kerberos で暗黙のうちに取得できるチケットの一覧について、詳しくは `klist` を実行してください。

下記には Kerberos 認証を使用するアプリケーションの一覧を示します。これらのアプリケーションは `krb5-apps-clients` パッケージをインストールしたあと、`/usr/lib/mit/bin` もしくは `/usr/lib/mit/sbin` 内にインストールされるものです。これらのソフトウエアには、一般的な Unix および Linux オペレーティングシステムでの全ての機能に加えて、Kerberos による透過的な認証機構の機能が追加されています:

- `telnet` , `telnetd`
- `rlogin`
- `rsh` , `rcp` , `rshd`
- `ftp` , `ftpd`
- `ksu`

これらのアプリケーションを使用する場合、Kerberos 側で既に識別情報を確認済みであることから、さらにパスワードを入力する必要を無くすことができます。なお、Kerberos に対応するよう `ssh` をコンパイルしてある場合、一方の端末で取得したチケットを他方の端末に転送することもできます。たとえば他のマシンにログインするために `ssh` を使用すると、`ssh` が暗号化されたチケットの内容を必要に応じて調整する処理を行います。これは、チケットには端末固有の情報 (IP アドレスなど) が書かれているためです。XDM や GDM でも Kerberos へのサポートが提供されています。Kerberos に対応するネットワークアプリケーションについて、詳しくは https://web.mit.edu/kerberos にある Kerberos V5 UNIX User's Guide (英語) をお読みください。

6.5 Kerberos のインストールと管理

Kerberos の環境は複数のコンポーネントから構成されています。鍵配送センター (KDC) は、Kerberos 関連のデータの全てを保持する中央データベースです。全てのクライアントからネットワーク経由で適切に認証を行うことができるようにするため、KDC のデータに依存して動作するようになっています。KDC とクライアントでは、お使いの環境に合わせて下記のような設定を行う必要があります:

一般的な準備

ネットワークの設定を確認し、6.5.1項「Kerberos ネットワークトポロジ」で示されている最低要件を満たしていることをご確認ください。また、6.5.2項「Kerberos レルムの選択」で説明している内容をもとに、Kerberos を設定する際のレルムについても適切な値をご用意ください。KDC として動作させるマシンについては、6.5.3項「KDC ハードウエアの設定」の説明に従って注意深く設定を行ってください。さらに、ネットワーク内で正しく時刻が設定されるよう、6.5.4項「時刻同期の設定」の手順に従って設定を行ってください。

基本的な設定

6.5.5項「KDC の設定」と 6.5.6項「Kerberos クライアントの設定」の説明に従って、KDC とクライアントを設定してください。Kerberos サービスに対する遠隔からの管理を有効化し、KDC マシンへの物理的なアクセスを不要にしたい場合は、6.5.7項「リモートからの Kerberos 管理の設定」をお読みください。また、6.5.8項「Kerberos サービスプリンシパルの作成」をお読みになって、レルム内に各サービスに対応するプリンシパルを作成してください。

Kerberos 認証の有効化

ネットワーク内にある様々なサービスが Kerberos に対応しています。Kerberos のパスワードチェックを PAM 経由でアプリケーションに追加したい場合は、6.5.9項「Kerberos に対する PAM サポートの有効化」の手順に従って設定を行ってください。また、SSH や LDAP と Kerberos を設定したい場合は、6.5.10項「Kerberos 認証用の SSH の設定」と 6.5.11項「LDAP と Kerberos の使用」の説明に従って設定を行ってください。

6.5.1 Kerberos ネットワークトポロジ

Kerberos の環境を正しく完全に動作させるには、下記の要件を満たさなければなりません:

- ネットワーク内に DNS サーバを用意して、クライアントとサーバの両方が名前で解決できるようにする必要があります。DNS の設定について、詳しくは『リファレンス』、第19章「ドメインネームシステム」をお読みください。
- ネットワーク内にタイムサーバを用意してください。Kerberos ではチケット内にタイムスタンプを書き込んで有効期限を管理するため、時刻を正しく設定しておくことが重要です。NTP の設定について、詳しくは『リファレンス』、第18章「NTP を利用した時刻同期」をお読みください。

- 鍵配送センター (KDC) は Kerberos の構造の中心に位置するものです。ここには Kerberos のデータベースが保存されます。このマシンに対しては最も厳しいセキュリティポリシーを適用してください。このマシンに対する攻撃が成功してしまうと、Kerberos の構造全体に被害を及ぼしてしまいます。

- クライアント側のマシンを設定して、Kerberos の認証を使用するようにしてください。

下記の図に、Kerberos のインフラストラクチャを構築するのに必要な、最小限のコンポーネントのみを配置した構成例を示します。実際の構成はネットワークの規模やトポロジに応じて決定してください。

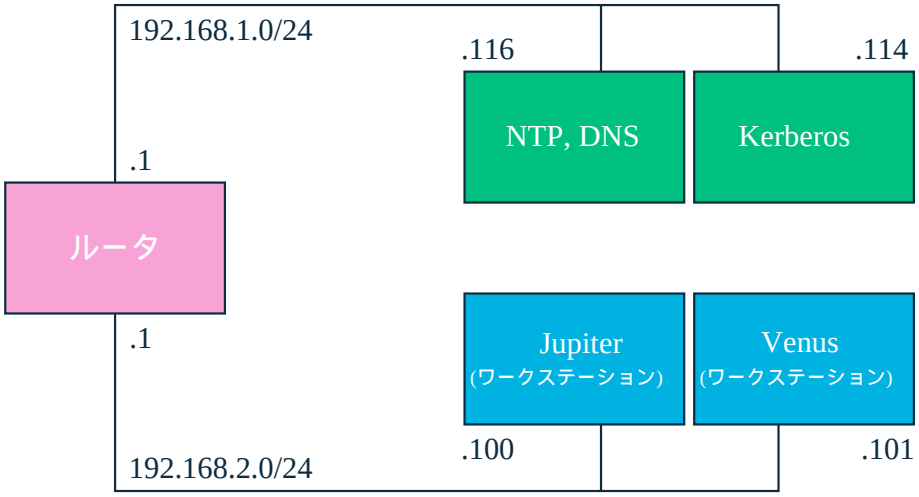


図 6.1: KERBEROS ネットワークトポロジ



ヒント: サブネットルーティングの設定について

図6.1「Kerberos ネットワークトポロジ」に示しているような構成を作成するには、2 つのサブネット (192.168.1.0/24 および 192.168.2.0/24) の間を結ぶルータで、ルーティング (経路制御) を設定する必要があります。YaST でルーティングを設定する場合は、**『リファレンス』、第13章「ネットワークの基礎」、13.4.1.5項「ルーティングの設定」**をお読みください。

6.5.2 Kerberos レルムの選択

Kerberos の世界では認証の管理単位をレルムと呼び、たとえば `EXAMPLE.COM` や単純に `ACCOUNTING` のように、名前を付けて区別します。Kerberos は大文字と小文字を区別して扱う仕組みであることから、`example.com` と `EXAMPLE.COM` は別々のレルムとして扱われます。いずれかお好きなほうをお使いください。ただし、一般的には大文字のレルムを使用します。

また、レルムには DNS ドメイン名 (もしくは `ACCOUNTING.EXAMPLE.COM` のようなサブドメイン) を使用するのが一般的です。このような構成にすることで、KDC やその他の Kerberos サービスを DNS ドメイン名と一括で管理できるようになります。また、レルム名を DNS ドメイン名のサブドメインにしておくと、将来の追加時に便利になることでしょう。

ただし、DNS のドメイン名とは異なり、Kerberos は階層構造にはなっていません。そのため、たとえば `EXAMPLE.COM` 以下に「下位のレルム」として `DEVELOPMENT` と `ACCOUNTING` を作成しても、`EXAMPLE.COM` のプリンシパルを引き継いで使用することはできません。このような構成では、3 つの独立したレルムとして構成をすることになります。ただし、レルム間には相互認証を設定することができますので、一方のレルムに属するユーザがアクセスできるサーバが存在した場合、他方のレルムに属するユーザを許可するような設定を行うことができます。

なお、構成や説明をわかりやすくする目的から、下記の記述では組織内全体を表すレルム `EXAMPLE.COM` を 1 つだけ使用するものとして説明を行います。

6.5.3 KDC ハードウエアの設定

Kerberos を使用するにあたって最初にすべきことは、鍵配送センター (KDC) として動作させるマシンの準備になります。このマシンには Kerberos のユーザデータベース (パスワードなどのユーザに関連する全ての情報) を保存します。

KDC はセキュリティインフラストラクチャで最も重要な構成部品です。そのセキュリティを突破されてしまうと、Kerberos が保持し保護している全てのアカウント情報が漏洩することになってしまいます。また、漏洩した情報をもとに Kerberos 内の任意のユーザになりすますこともできてしまいます。そのため、このマシンに対するセキュリティは、最も強固な設定にしてください:

1. サーバマシン自身を物理的に安全な場所に配置してください。ごく限られた人々のみが入室することのできる、施錠されたサーバルームなどが望ましいです。
2. KDC 以外のネットワークアプリケーションを動作させてはなりません。サーバ機能もクライアント機能も KDC とは決して共存させないでください。具体的には、NFS によるファイルシステムの取り込みや公開、DHCP によるネットワーク設定の取得などを行ってはなりません。
3. まずは最小限のシステムインストールを実施したあと、インストール済みのパッケージの一覧を確認して、不要なパッケージを削除してください。たとえば `inetd` , `portmap` , CUPS などは削除しておくべきですし、X 関連のパッケージも削除しておいてください。また、SSH サーバについても、潜在的なセキュリティリスクとなりますので、削除しておいてください。
4. X サーバは潜在的なセキュリティリスクとなるものであるため、このマシンではグラフィカルログインを動作させないでください。Kerberos では、専用の管理インターフェイスを提供しています。

5. `/etc/nsswitch.conf` ファイルを編集して、ユーザとグループを検索する際にローカルファイルのみを使用するように設定してください。具体的には、`passwd` と `group` の行を下記のように設定します:

```
passwd:         files
group:          files
```

`/etc` ディレクトリ内にある `passwd`，`group`，`shadow` の各ファイルを編集して、`+` 文字で始まる行 (これらは NIS で使用していた項目です) を削除してください。

6. `/etc/shadow` ファイルを編集して、`root` 以外のアカウントを全て無効化するほか、ハッシュ化されたパスワードが書かれている箇所についても `*` や `!` に置き換えてください。

6.5.4 時刻同期の設定

Kerberos を正しく動作させるには、組織内の全システムの時刻誤差が適切な範囲内に収まるよう、時刻を同期させる必要があります。これは他者が送信した認証情報を窃取したあと、再送信して権限を不正に得てしまうリプレイ攻撃を防ぐために重要となります。Kerberos では、このような攻撃を防ぐための様々な仕組みを取り入れていますが、その中の 1 つに、チケット内にタイムスタンプを書き込む、というものがあります。サーバ側では、不正な再送信を防ぐため、現在時刻とは異なるタイムスタンプの書かれたチケットを拒否するようになっています。

もちろんネットワーク上を流れるにあたって、ある程度の時間は必要となることから、タイムスタンプの比較に際しても一定の猶予時間が設けられています。しかしながら、コンピュータの時計は、誤差という観点では非常に不正確なものです。PC の時計が 1 週間のうちに 30 分程度進んだり遅れたりするようなことも、よくある話です。このような理由から、ネットワーク内の全てのコンピュータは、特定の中央の時刻情報源に対して同期を行う必要があります。

時刻同期を行うのに最も簡単な方法は、特定の 1 台のマシンに NTP タイムサーバをインストールして、残りのクライアントはその NTP タイムサーバに対して同期するように設定する方法です。同期する側のコンピュータについては、NTP デーモンである `chronyd` をインストールして動作させることで、同期を行うことができます。なお、KDC 自身も中央の時刻情報源と同期する必要があります。ただし、KDC 自身で NTP デーモンを実行するのはリスクとなりうるため、cron ジョブで `chronyd -q` を実行するよう設定するのが適切でしょう。NTP クライアントの設定方法について、詳しくは『リファレンス』、第18章「NTP を利用した時刻同期」、18.1項「YaST を利用した NTP クライアントの設定」をお読みください。

時刻同期を行うためのもう 1 つの方法としては、専用の NTP サーバを用意するか、もしくは KDC に直接電波時計や GPS 時計などのハードウエア時計を接続して時刻を管理する方法もあります。

なお、Kerberos がタイムスタンプを比較する際の最大許容範囲を設定することもできます。6.5.6.3項「時計の許容誤差の設定」の手順に従って、`krb5.conf` ファイル内の clock skew の値を変更してください。

6.5.5 KDC の設定

本章では、KDC の初期設定とインストール、そして管理者のプリンシパルを作成するまでの手順を説明しています。具体的には、下記のような流れで作業を行います:

1. **RPM のインストール:** まずは KDC として動作させるマシンに、`krb5`，`krb5-server`，`krb5-client` のソフトウエアパッケージをインストールします。

2. **設定ファイルの調整:** 必要に応じて、`/etc/krb5.conf` および `/var/lib/kerberos/krb5kdc/kdc.conf` の設定ファイルを編集します。これらのファイルには、KDC 内の全ての情報が含まれています。詳しくは 6.5.5.1項「サーバの設定」をお読みください。

3. **Kerberos データベースの作成:** Kerberos は全てのプリンシパルに対する情報と、認証時に使用する機密鍵を保持しています。詳しくは 6.5.5.2項「データベースの設定」をお読みください。

4. **ACL ファイルの調整: 管理者の追加:** KDC 内の Kerberos データベースは、ネットワーク上離れた場所から管理することもできます。データベースに対して不正なアクセスがされないよう、Kerberos ではアクセス制御リストを使用します。離れた場所から管理する場合は、データベースを管理することのできる管理者プリンシパルを明示的に指定しなければなりません。Kerberos の ACL ファイルは、`/var/lib/kerberos/krb5kdc/kadm5.acl` にあります。詳しくは 6.5.7項「リモートからの Kerberos 管理の設定」をお読みください。

5. **Kerberos データベースの調整: 管理者の追加:** Kerberos を動作させて管理できるようにするには、1 人以上の管理者プリンシパルを設定する必要があります。このプリンシパルは KDC を起動する前に追加しておかなければなりません。詳しくは 6.5.5.3項「プリンシパルの作成」をお読みください。

6. **Kerberos デーモンの起動:** KDC ソフトウエアをインストールして設定が完了したら、Kerberos デーモンを起動してレルムに対するサービス提供を開始します。詳しくは 6.5.5.4項「KDC の起動」をお読みください。

7. **プリンシパルの作成:** あなた自身を表すプリンシパルを作成します。詳しくは 6.5.5.3項「プリンシパルの作成」をお読みください。

6.5.5.1 サーバの設定

Kerberos サーバの設定は非常に柔軟な仕組みであり、お使いのネットワーク環境や DNS, DHCP のほか、レルムなどのさまざまな構成によって異なる設定になります。また、既定のレルムとドメイン-レルム間のマッピングを設定しなければなりません。なお、下記の設定例は必要最小限に絞った内容を示しているもので、そのままお使いいただけるものではありません。Kerberos の設定について、詳しくは https://web.mit.edu/kerberos/krb5-latest/doc/admin/conf_files/index.html (英語) をお読みください。

例 6.1: KDC の設定例 (/etc/krb5.conf)

```
[libdefaults]
 dns_canonicalize_hostname = false
 rdns = false
 default_realm = example.com
 ticket_lifetime = 24h
 renew_lifetime = 7d

[realms]
  example.com = {
  kdc = kdc.example.com.:88
  admin_server = kdc.example.com
  default_domain = example.com
 }

 [logging]
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log
 default = SYSLOG:NOTICE:DAEMON

[domain_realm]
 .example.com = example.com
 example.com = example.com
```

6.5.5.2 データベースの設定

まずは Kerberos がプリンシパルに対する全ての情報を保存するための、データベースを準備します。また、特にテープメディアにバックアップを採取した場合など、不用意に漏洩してしまうような事態を避けるため、データベースのマスターキーを設定しておくこともお勧めします。マスターキーはパスフレーズから生成するもので、stash と呼ばれるファイル内に補完されます。この仕組みにより、KDC の起動時に毎回パスフレーズを入力しなくても済むようになっています。なお、書籍内の適当なページを開いて目に付いた文章など、適切なパスフレーズを設定するようにしてください。

また、Kerberos データベース (`/var/lib/kerberos/krb5kdc/principal`) をバックアップする際には、stash ファイル (たとえば `/var/lib/kerberos/krb5kdc/.k5.EXAMPLE.COM` など) は除外するように設定してください。stash ファイルまでバックアップしてしまうと、そこから容易にデータベースを解読できてしまうからです。そのため、クラッシュ時に備えてパスフレーズを保管する際には、安全で機密を保持できる別の場所に配置するようにしてください。

stash ファイルとデータベースを作成するには、下記のように実行します:

```
> sudo /usr/lib/mit/sbin/kdb5_util create -r EXAMPLE.COM -s
```

下記のように表示されるはずです:

```
Initializing database '/var/lib/kerberos/krb5kdc/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@EXAMPLE.COM'
You are prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:  ❶
Re-enter KDC database master key to verify:  ❷
```

❶ マスターキー (パスフレーズ) を入力します。

❷ パスフレーズを再度入力します。

構築されたデータベースを確認するには、下記のコマンドを実行します:

```
> kadmin.local

kadmin> listprincs
```

データベース内には既にいくつかのプリンシパルが存在していますが、これらは Kerberos 内部用のものです:

```
K/M@EXAMPLE.COM
kadmin/admin@EXAMPLE.COM
kadmin/changepw@EXAMPLE.COM
krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

6.5.5.3 プリンシパルの作成

次に Kerberos のプリンシパルを 2 つ作成します。1 つは通常時に利用するための個人用プリンシパル、もう 1 つは Kerberos の管理作業用に使用するプリンシパルです。たとえばログイン名が `suzanne` であれば、下記のように実行します:

```
> kadmin.local

kadmin> ank suzanne
```

下記のように表示されるはずです:

```
suzanne@EXAMPLE.COM's Password: ❶
Verifying password: ❷
```

❶ `>suzanne` のパスワードを入力します。

❶ `suzanne` のパスワードを再度入力します。

次に、`kadmin` のプロンプトで続けて `ank suzanne/admin` と入力し、`suzanne/admin` というもう 1 つのプリンシパルを作成します。`admin` という接尾辞を付けて、管理者の 役割 で使用するものであることを指定します。ここで指定した 役割 は、後ほどの手順で Kerberos データベースの管理者を指定する際に使用します。なお、ユーザに対してはそれぞれ別々の目的で設定した複数の役割を指定することができます。役割を指定したアカウントは、通常のものとは別のアカウントであるものとして扱われます。

6.5.5.4 KDC の起動

KDC デーモンと kadmin デーモンを起動します。デーモンを手作業で起動したい場合は、下記のように入力して実行してください:

```
> sudo systemctl start krb5kdc
sudo systemctl start kadmind
```

なお、マシンの起動時に (`krb5kdc`) と kadmind (`kadmind`) を自動的に開始するようにも設定してください。下記のように実行します:

```
> sudo systemctl enable krb5kdc kadmind
```

上記の方法以外にも、YaST の［サービスマネージャ］を使用してもかまいません。

6.5.6 Kerberos クライアントの設定

対応するインフラストラクチャ (DNS, NTP) を用意し、KDC を正しく設定して開始することができたら、あとはクライアント側のマシンの設定になります。Kerberos のクライアントを設定するには、下記で説明している 2 つの手作業のうち、いずれかを実施してください。

Kerberos を設定するにあたっては、2 種類のアプローチをとることができます。1 つは `/etc/krb5.conf` ファイルを利用したクライアントごとの手動の設定方法、もう 1 つは DNS を利用した動的な設定方法です。DNS を利用する場合、Kerberos のアプリケーションは DNS レコードを利用して KDC サービスがどこにあるのかを判断します。クライアントごとの設定を行う場合は、`krb5.conf` 内に KDC サーバのホスト名を設定します (この場合、KDC を別のホストに移動させるには、設定ファイルの更新を行うか、その他の方法で再設定を行う必要があります) 。

DNS ベースの設定のほうがより柔軟な構成を取ることができるほか、クライアントごとの設定作業も大幅に削減することができます。ですが、レルム名と DNS ドメイン名が一致しているか、もしくはレルム名が DNS ドメインのサブドメインでなければならなくなります。それ以外にも、ネームサーバを攻撃してダウンさせてしまったり、DNS レコードを偽装させたりするなどの方法で、Kerberos のインフラストラクチャを混乱させることができてしまいます。とはいえ、最大限に攻撃が成功したとしても、サービスの提供を妨害するまでのことしかできませんし、クライアントごとの設定でも、`krb5.conf` 内に IP アドレスではなくホスト名を記述してしまうと、DNS への攻撃による妨害は発生しうることになります。

6.5.6.1 手動での設定

Kerberos の設定方法のうちの 1 つとして、`/etc/krb5.conf` ファイルを編集する方法があります。このファイルは既定でインストールされるファイルで、いくつかの設定例が書かれています。設定を行う際には、これら全てを削除してから実施してください。また、`krb5.conf` ファイルは複数のセクション (節) から構成されていますが、セクションは `[this]` のように、大括弧で括られて表されます。

Kerberos クライアントを設定するには、下記のセクションを `krb5.conf` ファイルに追加します (なお、`kdc.example.com` の箇所には、お使いの KDC のホスト名を入れてください):

```
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
        EXAMPLE.COM = {
                kdc = kdc.example.com
                admin_server = kdc.example.com
        }
```

上記の例の `default_realm` の行は、Kerberos での既定のレルムを指定しています。複数のレルムを使用する場合は、`[realms]` セクション内でそれぞれのレルムを設定してください。

また、このファイルでは、ホスト名とレルムをどのようにして対応づけるのかを設定することもできます。たとえばリモートのホストに接続する際、Kerberos ライブラリ側ではそのホストがどのレルムに属しているのかを知る必要が発生するためです。この設定は `[domain_realms]` セクション内で設定しなければなりません:

```
[domain_realm]
.example.com = EXAMPLE.COM
www.example.org = EXAMPLE.COM
```

上記のように設定することで、`example.com` の DNS ドメイン名のホストは `EXAMPLE.COM` の Kerberos レルムに属していることになります。これに加えて、`www.example.org` というドメイン名のホストについても、`EXAMPLE.COM` レルムのメンバーであることを宣言しています。

6.5.6.2 DNS ベースの設定

DNS ベースの Kerberos 設定では、DNS の SRV レコードを多用します。詳しくは https://datatracker.ietf.org/doc/html/rfc2052 にある (RFC2052) A DNS RR for specifying the location of services (サービスの場所を指定するための DNS 資源レコード) (英語) をお読みください。

Kerberos に関連する SRV レコードは、`_サービス._プロトコル.レルム` の書式になっていなければなりません (`レルム` は Kerberos のレルムです) 。また、DNS 内のドメイン名は大文字と小文字を区別しない仕組みであるため、区別する Kerberos ではこの方式がうまく動作しない場合があります。なお `_サービス` はサービス名 (KDC やパスワードサービスなどで別々の名前を使用します) 、`_プロトコル` は `_udp` もしくは `_tcp` のいずれかになります (ただし、全てのサービスが両方のプロトコルに対応しているとは限りません) 。

SRV リソースのデータ部分には、優先順位と重み、ポート番号とホスト名をそれぞれ設定します。優先順位は接続を試すべき順序 (低い値ほど優先度が高いことを表します) を、重みは同じ優先順位内での負荷分散用の値を、それぞれ表します。特に必要がなければ、いずれの値とも 0 に設定してかまいません。

MIT Kerberos では、サービスを参照する際に下記の名前を使用します:

_kerberos
: KDC デーモン (認証サーバおよびチケット発行許諾サーバ) の場所を定義します。たとえば下記のように設定します:

```
_kerberos._udp.EXAMPLE.COM.  IN  SRV    0 0 88 kdc.example.com.
_kerberos._tcp.EXAMPLE.COM.  IN  SRV    0 0 88 kdc.example.com.
```

_kerberos-adm
: リモート管理サービスの場所を定義します。たとえば下記のように設定します:

```
_kerberos-adm._tcp.EXAMPLE.COM. IN  SRV    0 0 749 kdc.example.com.
```

なお、kadmind は UDP に対応していないため、`_udp` のレコードは存在すべきではありません。

手動での設定と同様に、`example.com` ドメイン以下に存在していないホストに対しても、`EXAMPLE.COM` レルムの配下として設定する仕組みが用意されています。このような設定を行いたい場合は、TXT レコードで `_kerberos.ホスト名` のようなレコードを作成してください。たとえば下記のようになります:

```
_kerberos.www.example.org.  IN TXT "EXAMPLE.COM"
```

6.5.6.3 時計の許容誤差の設定

クロックスキュー とは、ホスト側のシステム時計と厳密に一致していないタイムスタンプを持つチケットに対して、どの範囲までを許容するのかを指定する項目です。通常、クロックスキューは 300 秒 (5 分) 程度に設定しておきます。この設定の場合、サーバの時計と比較して最大 5 分までの差異までを許す意味になります。

NTP を利用して全てのホストの時刻同期を行う環境であれば、この値を 1 分程度にまで縮めることができます。この場合、`/etc/krb5.conf` 内でのクロックスキューの設定は下記のようになります:

```
[libdefaults]
        clockskew = 60
```

6.5.7 リモートからの Kerberos 管理の設定

KDC のコンソールに直接アクセスすることなく、リモートから Kerberos のデータベースにプリンシパルを追加または削除できるようにするには、Kerberos の管理サーバに対して、どのプリンシパルからの編集を許可するのかを `/var/lib/kerberos/krb5kdc/kadm5.acl` ファイルで設定します。このファイルは ACL (アクセス制御リスト) ファイルと呼ばれ、様々な操作に対して細かく権限を設定することができます。詳しくは `man 8 kadmind` で表示されるマニュアルページをお読みください。

ここでは、あなた自身に対してデータベースの管理者権限を割り当てます。具体的には、下記の内容をファイルに追加してください:

```
suzanne/admin              *
```

なお、`suzanne` の箇所はあなた自身を表すユーザ名に置き換えてください。設定内容を反映させるには、`kadmind` を再起動する必要があります。

これで kadmin ツールを利用して Kerberos の管理作業を行うことができるようになります。まずは管理者ロールに対するチケットを取得して、kadmin サーバの接続時にそのチケットを使用します:

```
> kadmin -p suzanne/admin
Authenticating as principal suzanne/admin@EXAMPLE.COM with password.
Password for suzanne/admin@EXAMPLE.COM:
kadmin:  getprivs
current privileges: GET ADD MODIFY DELETE
kadmin:
```

`getprivs` コマンドを使用すると、取得済みの権限を確認することができます。なお、上記に示されているもので全ての権限となります。

例として、`suzanne` のプリンシパルを編集します:

```
> kadmin -p suzanne/admin
```

```
Authenticating as principal suzanne/admin@EXAMPLE.COM with password.
Password for suzanne/admin@EXAMPLE.COM:

kadmin:  getprinc suzanne
Principal: suzanne@EXAMPLE.COM
Expiration date: [never]
Last password change: Wed Jan 12 17:28:46 CET 2005
Password expiration date: [none]
Maximum ticket life: 0 days 10:00:00
Maximum renewable life: 7 days 00:00:00
Last modified: Wed Jan 12 17:47:17 CET 2005 (admin/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 2
Key: vno 1, Triple DES cbc mode with HMAC/sha1, no salt
Key: vno 1, DES cbc mode with CRC-32, no salt
Attributes:
Policy: [none]

kadmin:  modify_principal -maxlife "8 hours" suzanne
Principal "suzanne@EXAMPLE.COM" modified.
kadmin:  getprinc suzanne
Principal: suzanne@EXAMPLE.COM
Expiration date: [never]
Last password change: Wed Jan 12 17:28:46 CET 2005
Password expiration date: [none]
Maximum ticket life: 0 days 08:00:00
Maximum renewable life: 7 days 00:00:00
Last modified: Wed Jan 12 17:59:49 CET 2005 (suzanne/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 2
Key: vno 1, Triple DES cbc mode with HMAC/sha1, no salt
Key: vno 1, DES cbc mode with CRC-32, no salt
Attributes:
Policy: [none]
kadmin:
```

上記の変更では、チケットの最大有効期限を 8 時間に変更しています。`kadmin` コマンドに関する詳細や、利用可能なオプションの一覧については、`krb5-doc` パッケージ、もしくは `man 8 kadmin` で表示されるマニュアルページをお読みください。

6.5.8 Kerberos サービスプリンシパルの作成

ここまではユーザ認証の部分のみを説明してきましたが、Kerberos に対応するサービスでは、サービス側もクライアントに対して認証を行う必要があります。そのため、レルム内で提供されるサービスに対しては、サービスそのものを表す特殊なサービスプリンシパルを Kerberos データベース内に作成しなければなりません。たとえば ldap.example.com が LDAP サービスを提供するような場合、`ldap/ldap.example.com@EXAMPLE.COM` という名前のサービスプリンシパルを作成する必要があります。

サービスプリンシパルは `サービス名/ホスト名@レルム` の形式で表します。ここで、`ホスト名` にはホストの完全修飾名を指定します。

利用可能なサービス名は下記のとおりです:

サービス記述子	サービス
`host`	Telnet, RSH, SSH
`nfs`	NFSv4 (Kerberos に対応しているもの)
`HTTP`	HTTP (Kerberos 認証の機能があるもの)
`imap`	IMAP
`pop`	POP3
`ldap`	LDAP

サービスプリンシパルはユーザプリンシパルと似ていますが、明らかな違いが存在しています。それは、サービスプリンシパルが鍵で保護されているのに対して、ユーザプリンシパルはパスワードで保護されているという点です。ユーザが KDC からチケット発行許諾チケットを取得した場合、Kerberos 側でチケットを解読する際にパスワード入力が必要となります。この仕組みをそのままサービスプリンシパルにも適用してしまうと、たとえばシステム管理者が SSH デーモン向けのチケットを取得する目的で、8 時間おきにパスワードを入力しなければならなくなってしまうため、現実的ではなくなってしまいます。

その代わり、サービスプリンシパルの場合は初期チケットの解読に鍵を使用するようになっています。鍵は管理者が KDC から一度だけ取り出したあと、keytab と呼ばれるファイルに保存しておくものです。これにより SSH デーモンは鍵を読み込んで、必要なタイミングで新しいチケットを自動的に取得するようになります。既定の keytab ファイルは `/etc/krb5.keytab` にあります。

たとえば `jupiter.example.com` に対するホストサービスプリンシパルを作成したい場合は、kadmin セッション内で下記のようにコマンドを入力します:

```
> kadmin -p suzanne/admin
```

```
Authenticating as principal suzanne/admin@EXAMPLE.COM with password.
Password for suzanne/admin@EXAMPLE.COM:
kadmin:  addprinc -randkey host/jupiter.example.com
WARNING: no policy specified for host/jupiter.example.com@EXAMPLE.COM;
defaulting to no policy
Principal "host/jupiter.example.com@EXAMPLE.COM" created.
```

新しいプリンシパルに対してパスワードを設定する代わりに、`-randkey` フラグを指定して、`kadmin` でランダムな鍵を生成するように指定します。これは、このプリンシパルに対してユーザ操作を不要にするために必要なフラグとなります。これがマシンに対するサーバアカウントとなります。

最後に、鍵を抽出してローカルの keytab ファイル `/etc/krb5.keytab` に保管します。このファイルはスーパーユーザが所有するファイルであるため、下記の kadmin シェルを実行する際には、`root` でなければなりません:

```
kadmin:  ktadd host/jupiter.example.com
Entry for principal host/jupiter.example.com with kvno 3, encryption type Triple
DES cbc mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5.keytab.
Entry for principal host/jupiter.example.com with kvno 3, encryption type DES
cbc mode with CRC-32 added to keytab WRFILE:/etc/krb5.keytab.
kadmin:
```

全ての作業が完了したら、kinit で取得した管理チケットを破棄するため、`kdestroy` を実行します。

6.5.9 Kerberos に対する PAM サポートの有効化



警告: 不完全な設定を行ってしまうとユーザが締め出されてしまう問題について

Kerberos を不完全に設定してしまうと、root ユーザを含め全てのユーザが締め出されてしまうことがあります。このような問題が起こらないようにするには、既存の PAM 設定内に `pam_krb5` を追加した あと 、`pam_krb5` モジュールに `ignore_unknown_principals` ディレクティブを追加してください。具体的には、下記を実行します:

```
> sudo pam-config --add --krb5-ignore_unknown_principals
```

これにより、`pam_krb5` モジュールがアカウントフェーズの失敗を引き起こすエラーを無視するようになります。

openSUSE® Leap には `pam_krb5` と呼ばれる PAM モジュールが用意されています。このモジュールは Kerberos のログインとパスワード更新に対応しており、コンソールのログインや `su` 、もしくは GDM のようなグラフィカルログイン時に使用することができます。つまり、ユーザ側でパスワードを

入力することで、ユーザに成り代わって初期の Kerberos チケットを取得するような、全ての用途に使用することができるモジュールです。PAM で Kerberos に対応するように設定するには、下記のコマンドを実行します:

```
> sudo pam-config --add --krb5
```

上記のコマンドを実行すると、既存の PAM 設定ファイル内に `pam_krb5` を追加し、適切な順序で呼び出されるように設定を調整します。`pam_krb5` に対してより詳しい設定を行いたい場合は、`/etc/krb5.conf` ファイルを編集したあと、PAM に既定のアプリケーションを追加してください。詳しくは `man 5 pam_krb5` で表示されるマニュアルページをお読みください。

`pam_krb5` モジュールは、Kerberos のチケットをユーザ認証処理の一部として受け付けるような、ネットワークサービス向けには設計されていません。このような用途については、後続の章をお読みください。

6.5.10 Kerberos 認証用の SSH の設定

OpenSSH ではプロトコルバージョン 1 および 2 の両方で Kerberos 認証に対応しています。バージョン 1 では Kerberos のチケットを送信するための特殊なプロトコルメッセージが用意されています。バージョン 2 では Kerberos を直接使用するようなことは行っておらず、その代わりに GSSAPI (General Security Services API; 汎用セキュリティサービス API) を使用するようになっています。これは Kerberos 限定のプログラミングインターフェイスではなく、SPKM のような公開鍵認証システムなどでも使用することのできる、下位の認証システムの特異性を隠蔽するように設計されたインターフェイスです。ただし、現時点では GSSAPI ライブラリは Kerberos のみに対応しています。

sshd で Kerberos 認証を使用するには、`/etc/ssh/sshd_config` ファイルを編集して下記のオプションを追加します:

```
# 下記はプロトコルバージョン 1 用
#
# KerberosAuthentication yes
# KerberosTicketCleanup yes

# 下記はプロトコルバージョン 2 用（こちらをお使いください）
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
```

あとは `sudo systemctl restart sshd` を実行して、SSH デーモンを再起動してください。

プロトコルバージョン 2 で Kerberos 認証を使用するには、クライアント側でも同様に有効化を行う必要があります。クライアント側ではシステム全体の設定ファイル `/etc/ssh/ssh_config` で設定することができるほか、ユーザ単位でも `~/.ssh/config` ファイルを編集することで設定を行うことができます。いずれのファイルの場合も、`GSSAPIAuthentication yes` のオプションを追加することになります。

これで Kerberos 認証を使用して接続できるようになります。`klist` を実行すると、有効なチケットの一覧を確認することができます。あとは SSH サーバに接続するだけです。SSH プロトコルバージョン 1 を強制的に使用したい場合は、コマンドラインで `-1` オプションを指定してください。



ヒント: さらなる情報

`/usr/share/doc/packages/openssh/README.kerberos` ファイルには、OpenSSH と Kerberos の関係について、より詳しい説明が書かれています。



ヒント: プロトコルバージョン 2 向けの追加ディレクティブについて

`GSSAPIKeyExchange` の仕組み (RFC 4462) に対応しています。このディレクティブは、どのようにしてホスト鍵を交換するのかを指定します。詳しくは sshd_config のマニュアルページ (`man sshd_config`) をお読みください。

6.5.11 LDAP と Kerberos の使用

Kerberos が認証機能を提供するのに対して、LDAP では認証と識別の両方を提供します。両方のサービスを同時に動作させることもできます。

接続の機密を保持する目的で、389 Directory Server ではさまざまなデータの暗号化方式に対応しています。SSL/TLS 接続や StartTLS 接続、そして SASL による認証にそれぞれ対応しています。Simple Authentication and Security Layer (SASL) は認証用に設計されたネットワークプロトコルです。openSUSE Leap での SASL 実装は `cyrus-sasl` と呼ばれます。それに対して Kerberos 認証は、`cyrus-sasl-gssapi` パッケージが提供する GSS-API (General Security Services API) で行われます。GSS-API を使用することで、389 Directory Server は Kerberos のチケットを利用して、セッションの認証とデータの暗号化を使用することができます。

SASL フレームワークを使用することで、サーバへの認証方式に異なる仕組みを使用することができます。たとえば Kerberos の場合、認証は常に相互的に動作します。つまり、389 Directory Server サーバに対して認証を行っているだけでなく、389 Directory Server 自身もあなたに対して認証を行っていることになります。これは特に、攻撃者が不正に作成した偽サーバにアクセスしたりせず、正しい LDAP サーバに接続していることを確認するために必要な仕組みです。

Kerberos から 389 Directory Server に接続するようにするには、`ldap/ldap.example.com` というプリンシパルを作成して、それを keytab 内に追加する必要があります。389 Directory Server が認証時に使用する認可情報は、keytab によって他のサーバに送信されます。389 Directory Server では keytab を `KRB5_KTNAME` という環境変数を介して割り当てます。

値を設定するには下記の手順を実施します:

1. ```
   > sudo systemctl edit dirsrv@インスタンス名
   ```

   389 Directory Server の既定のインスタンス名を使用している場合は、`インスタンス名` の箇所を `localhost` にしてください。

2. あとは下記を追加します:

   ```
   [Service]
     Environment=KRB5_KTNAME=/etc/dirsrv/slapd-インスタンス名/krb5.keytab
   ```

3. keytab ファイルは 389 Directory Server の動作するアカウント (例: `dirserv`) が読み込めるようにする必要があります:

   ```
   > sudo chown dirsrv:dirsrv /etc/dirsrv/slapd-インスタンス名/krb5.keytab
   > sudo chmod 600 /etc/dirsrv/slapd-インスタンス名/krb5.keytab
   ```

6.5.11.1 LDAP を利用した Kerberos 認証の使用

チケット発行許諾チケットを取得およびキャッシュしたい場合は、6.5.5.3項「プリンシパルの作成」で作成したプリンシパルをお使いください:

```
> kinit suzanne@EXAMPLE.COM
```

GSSAPI 認証が動作していることを確認するには、下記の手順を実施します:

```
> ldapwhoami -Y GSSAPI -H ldap://ldapkdc.example.com
dn: uid=testuser,ou=People,dc=example,dc=com
```

GSSAPI は `ccache` を利用して 389 Directory Server サーバに対してユーザの認証を実施します。このとき、パスワードの入力は不要です。

6.5.11.2 SASL 識別子マッピングの設定

SASL バインド要求を処理する際、389 Directory Server は SASL の認証 ID (ディレクトリサーバとの間で認証時に使用するもの) とサーバ内に保存されている LDAP エントリの紐付け (マッピング) を行います。Kerberos を使用している場合、SASL のユーザ ID は `ユーザ_ID@レルム` (例: `tux`@example.com) のような書式になります。ID はディレクトリサーバ内のユーザの DN 、つまり `uid=tux,ou=people,dc=example,dc=com` などに変換されなければなりません。389 Directory
```

Server では、よくある構成向けの既定のマッピングが提供されていますが、必要であれば独自のマップを作成することもできます。手順6.1「マップの管理」では、マップの表示方法や削除方法、独自のマップを作成する方法について説明しています。

手順 6.1: マップの管理

1. 既存の SASL マップを一覧表示するには、下記のように入力して実行します:

```
> dsconf インスタンス名 sasl list
Kerberos uid mapping
rfc 2829 dn syntax
rfc 2829u syntax
uid mapping
```

2. マップを表示するには、下記のように入力して実行します:

```
> sudo dsconf インスタンス名 sasl get "Kerberos uid mapping"
dn: cn=Kerberos uid mapping,cn=mapping,cn=sasl,cn=config
cn: Kerberos uid mapping
nsSaslMapBaseDNTemplate: dc=\2,dc=\3
nsSaslMapFilterTemplate: (uid=\1)
nsSaslMapRegexString: \(.*\)@\(.*\)\.\(.*\)
objectClass: top
objectClass: nsSaslMapping
```

3. 既定のマップでは、お使いの環境の dc が 2 つのコンポーネントから構成されている場合にのみ動作します。(お使いの環境に合わないなどの理由で) マップを削除するには、下記のように入力して実行します:

```
> sudo dsconf インスタンス名 sasl delete "Kerberos uid mapping"
Deleting SaslMapping cn=Kerberos uid mapping,cn=mapping,cn=sasl,cn=config :
Successfully deleted cn=Kerberos uid mapping,cn=mapping,cn=sasl,cn=config
```

4. 新しいマップを作成するには、下記のように入力して実行します:

```
> sudo dsconf localhost sasl create --cn=bhgssapi --nsSaslMapRegexString "\
(.*\)@EXAMPLE.NET.DE" --nsSaslMapBaseDNTemplate="dc=example,dc=net,dc=de" --
nsSaslMapFilterTemplate="(uid=\1)"
> sudo Enter value for nsSaslMapPriority :
Successfully created bhgssapi
```

5. 新しく作成したマップを表示するには、下記のように入力して実行します:

```
> sudo dsconf localhost sasl get "bhgssapi"
dn: cn=bhgssapi,cn=mapping,cn=sasl,cn=config
cn: bhgssapi
nsSaslMapBaseDNTemplate: dc=example,dc=net,dc=de
nsSaslMapFilterTemplate: (uid=\1)
```

```
nsSaslMapPriority: 100
nsSaslMapRegexString: \(.*\)@EXAMPLE.NET.DE
objectClass: top
objectClass: nsSaslMapping
```

これを利用することで、特定のレルムのユーザのみをチェックして、異なる dc ベースに再マップすることができるようになります。既にご存じのとおり、新しいマップには 3 つの `dc` コンポーネントが存在していますので、既定のマップはこのレルム（ `EXAMPLE.NET.DE` ）では動作せず、`EXAMPLE.NET` のようなレルムでのみ動作します。

6.6 Kerberos と NFS

ほとんどの NFS サーバでは、`sec=sys` として知られている「ネットワークベースの信頼」と、`sec=krb5` ，`sec=krb5i` ，`sec=krb5p` として知られている 3 種類の Kerberos ベースのセキュリティを任意に組み合わせてファイルシステムを公開することができます。`sec` オプションはクライアント側のマウントオプションとして指定します。NFS サービスによっては、いったん `sec=sys` を利用して設定および接続を行い、その後 Kerberos を適用するものもあります。この場合、サーバ側では `sec=sys` といずれかの Kerberos レベルの両方に対応するよう設定したあと、全てのクライアントを移行させてから `sec=sys` サポートを無効化してください。これにより、セキュリティを確実にすることができます。なお、Kerberos への遷移は手順通り行われる限り、透過的に動作します。しかしながら、NFS 側には微妙な動作の違いが存在することもありますので、Kerberos を使用した場合にはうまくいかないこともあります。そのため、NFS サーバを Kerberos で使用するにあたっては、詳細をよくご確認のうえ設定を行う必要があります。詳しくは 6.6.1項「グループのメンバー」をお読みください。

3 種類の Kerberos レベルは、それぞれ異なるセキュリティレベルを構成します。セキュリティレベルを高めれば高めるほど、メッセージの暗号化や解読に際してプロセッサの処理能力が必要になります。NFS 向けに Kerberos を開放するにあたっては、双方で適切なバランスを保つよう選択してください。

`krb5` は認証機能のみを提供します。サーバはリクエストを送信した相手を知っていますし、クライアントは応答を送ったサーバを知っています。ただし、リクエストや応答の内容に対する追加のセキュリティ機能は提供されないため、様々な方法でサーバやクライアントになりすますことができてしまいます。認証済みのユーザが読み込んだり変更したりできなかったファイルに対して、攻撃者が読み込んだり変更したりすることはできませんが、それ以外のほぼ全てのことができてしまいます。

`krb5i` は全てのメッセージに対してチェック機能を追加します。`krb5i` では、攻撃者がリクエストや応答を書き換えたりすることができなくなりますが、交換されているデータは全て盗聴できてしまいます。そのため、読み込んでいるファイルの内容などは容易に盗聴できてしまいます。

`krb5p` はプロトコルにプライバシー機能を追加します。信頼性のある認証機能とチェック機能に加えて、全てのメッセージが暗号化されます。そのため、攻撃者はサーバとクライアントとの間でメッセージが交換されていることを知ることはできますが、メッセージ内に直接含まれている情報は抽出できなくなります。なお、メッセージのタイミングから、どのような状況にあるのかは推測できてしまいますが、こちらは Kerberos では対応のできない範疇外のものとなります。

6.6.1 グループのメンバー

`sec=sys` と Kerberos のセキュリティレベルとの間で、動作が大きく異なるのは、グループのメンバー関係に関する箇所です。Unix や Linux では、ファイルシステムへのアクセスはプロセスから届くものであり、そのプロセスには特定のユーザとグループが結びつけられています。また、ユーザは複数のグループに対するメンバーとなることができます。このような仕組みから、ファイルへのアクセス権は所有者とグループをベースにして行われます。

`sec=sys` を指定した場合、ユーザ ID とグループ ID 、そして最大 16 種類までの補助グループが、各リクエスト内に書き込まれてサーバ側に送られます。

ユーザが 16 種類以上のグループに対するメンバーである場合、それより多くのグループ情報は失われます。そのため、通常はユーザからアクセスできていたはずのファイルに対して、NFS 経由にするとアクセスができなくなってしまうことがあります。このような理由から、NFS を使用する場合は、最大でも 16 種類までの補助グループとなるよう、うまく構成して回避を行ってください。

ユーザが `newgrp` コマンドや set-group-id のプログラムを実行すると、いずれのコマンドとも所属するグループのリストを変更することができますので、これによって NFS へのアクセス結果が即時に変わることがあります。

なお、Kerberos ではグループに関する情報は全く送信されません。ユーザが (Kerberos の「プリンシパル」で) 認証されると、ユーザ ID とそのプリンシパルに対するグループリストを元にして、検索を行うだけです。つまり、ユーザが 16 種類以上のグループに対するメンバーであった場合、ファイルへのアクセス許可判断は、それら全てのグループメンバー情報をもとに行われることになります。しかしながら、ユーザがクライアント側でグループ ID を変更しても、サーバ側にはその変更が送信されないため、アクセス権の判断には影響しません。

多くの場合、多数のグループにアクセスできるようになったことで、大きな利益がもたらされますが、グループが変更できなくてもあまり実害はありません。ただし、Kerberos を使用しているサイトの管理者はその情報を認識し、実際に問題を引き起こしたりすることがないようご注意ください。

6.6.2 性能とスケーラビリティ

セキュリティ確保の目的で Kerberos を使用すると、メッセージの暗号化や解読に際して、CPU の負荷がかかるようになります。どれだけの CPU 性能が必要かや、通常時と比べてどれだけ負荷が高くなるのかは、ハードウエアや用途によって大きく異なります。もしもサーバやクライアント側の CPU 性能が限界に達しているような場合は、`sec=sys` を Kerberos に切り替えることで、顕著な性能劣化が現れることになります。CPU の性能に余裕がある場合であれば、この移行作業で性能が変わることはありません。Kerberos を使用することによる性能変化は、実際のハードウエアと環境でテストしてお確かめください。

負荷を軽減するための唯一の選択は、保護品質の設定です。`sec=krb5p` よりも `sec=krb5` のほうがずっと負荷の軽い処理になりますが、上述のとおり強力な機密保持は行われなくなります。それ以外にも、Kerberos が利用する暗号化方式を変更する手もあります。これによって CPU 要件をより低減させることができます。とはいえ、変更することによる影響をよくよくご検討のうえ、変更を行ってください。

NFS で Kerberos を使用するように設定する際の性能問題という観点では、KDC (鍵配送センター) として知られている Kerberos の認証サーバの可用性についてもご検討ください。

NFS を使用すると、他のサービスに対して Kerberos を使用すると負荷が増えるのと同様に、ある程度の負荷上昇が発生します。あるユーザ (Kerberos のプリンシパル) がサービスに対してセッションを確立した場合、たとえば特定の NFS サーバが公開するファイルにアクセスした場合、クライアントは KDC との間でやり取りを実施する必要があります。やり取りの結果セッション鍵を取得できれば、サーバとクライアントはしばらく直接やり取りできるようになります。これは Kerberos の設定である `ticket_lifetime` の設定に依存して決まります。

Kerberos の KDC サーバの配置を設計するのに重要な項目は、可用性とピーク使用量です。

DNS や LDAP などの名前検索サービスなどの他のコアサービスと KDC サーバとの間は適度に「近く」なるように設計し、各クライアントに対する適度な可用性を提供するようにしてください。Kerberos ではデータベースの伝搬のために柔軟なモデルを採用しているため、複数の KDC サーバを用意することができます。これにより、キャンパス単位やビル単位、キャビネット単位でも、必要なサーバをわかりやすく配置することができます。それぞれのクライアントが近くにある Kerberos サーバを検出できるようにするための適切な方式として、それぞれのビルなどに DNS を水平展開して、それぞれで異なる応答をするように設定するやり方があります。このような方式を採用できない場合は、`/etc/krb5.conf` ファイルをビルごとに別々の値にするなどして、適度に分散させるように設計してください。

Kerberos KDC へのアクセスはあまり頻繁に発生しないことから、負荷として考慮すべきことはピークタイムとなります。たとえば 9:00 から 9:05 の間に多くの人々がログインするような場合、夜間に比べるとその時間帯は非常に多くの毎分リクエストを受け付けることになります。Kerberos サーバの負荷は LDAP サーバの負荷よりも高くなりやすいのですが、桁違いというほどではありません。一般的には、LDAP のレプリカと同じやり方で Kerberos のレプリカを配置したあと、さらにサーバを増設する必要があるかどうかを性能監視で判断するのが良いでしょう。

6.6.3 マスター KDC／複数ドメイン／信頼関係

Kerberos KDC のサービスは、パスワードの変更や新しいユーザの作成などの処理を扱う必要があることから、容易には分散させることができません。これらの処理は、単一のマスター KDC で行わなければなりません。

これらに対する更新処理はそれほど多く発生しないため、負荷に対する要件もあまり厳しくありません。ですが、可用性という観点は重視する必要があります。可用性を考慮しておかないと、新しいユーザを作成したりパスワードを変更したりする際に時間がかかる場合があるほか、拠点によっては一時的にマスター KDC にアクセスできなくなってしまうこともあります。

組織が地理的に分散しているような場合で、各拠点に対する管理作業を拠点内で実施するような場合、それぞれ別々の Kerberos ドメインを作成して、それぞれの管理センターを 1 台ずつ構成するのがよいでしょう。それぞれのドメイン (拠点) には、それぞれのマスター KDC が存在することになります。一方のドメイン内のユーザが他のドメイン内のリソースにアクセスする場合でも、ドメイン間の信頼関係を設定することで解決することができます。

複数ドメインを扱う際の最も簡単な方法は、グローバルドメイン (例: EXAMPLE.COM) とローカルドメイン (例: ASIA.EXAMPLE.COM, EUROPE.EXAMPLE.COM など) をそれぞれ構築する方法です。グローバルドメインはそれぞれのローカルドメインを信頼するように設定し、それぞれのローカルドメインはグローバルドメインを信頼するように設定すれば、任意のドメイン間で完全な信頼を実現できますので、どのドメインに属するプリンシパルであっても、任意のサービスを利用できるようになります。なお、リソースに対して適切なアクセス許可を設定する作業については、ユーザ名の参照方法や NFS ファイルサーバの機能によって異なりますので、ここでは説明していません。

6.7 さらなる情報

MIT Kerberos の公式サイトは https://web.mit.edu/kerberos にあります。ここには Kerberos のインストールから利用、管理ガイドに至るまで、様々な Kerberos の情報に対するリンクが提供されています。

Kerberos—ネットワーク認証システム (Brian Tung (原著), 桑村 潤 (翻訳)) (ISBN 978-4894711464) には、幅広い情報が含まれています。

7 Himmelblau を利用した Microsoft Entra ID による認証

改訂履歴

2026-06-23

Himmelblau は Microsoft Entra ID 認証と Linux 認証を統合するための仕組みです。本章では openSUSE Leap で必要なシステム設定について説明しています。具体的には必要なパッケージのインストールや PAM モジュールの有効化、そして NSS 参照の設定と Entra ID ドメインの設定、そして Himmelblau サービスの起動までを説明しています。

7.1 Himmelblau のインストール

まずは Himmelblau デーモンと PAM モジュール、そして NSS モジュールをそれぞれインストールします:

```
> sudo zypper install himmelblau pam-himmelblau libnss_himmelblau2
```

SSH を介して Entra ID のユーザにログインを許可したい場合は、SSH の設定パッケージもインストールしておきます:

```
> sudo zypper install himmelblau-sshd-config
```

7.2 Himmelblau 認証の設定

下記の章では、Himmelblau を介して Entra ID ユーザを解決し、認証を行うための設定方法を説明しています。

1. まずは `/etc/himmelblau/himmelblau.conf` ファイルを編集し、`[global]` セクション内にプライマリ Entra ID ドメインを設定します:

   ```
   [global]
   domain = example.com
   ```

2. 次に PAM 認証に Himmelblau を追加します:

   ```
   > sudo pam-config --add --himmelblau
   ```

3. `/etc/nsswitch.conf` ファイルを編集して、`passwd` , `group`, `shadow` の各データベースに `himmelblau` を追加します。なお、既存のデータベース設定項目は削除しないでください。たとえば下記のようになります:

```
passwd: files systemd himmelblau
group:  files systemd himmelblau
shadow: files systemd himmelblau
```

4. Himmelblau サービスを有効化して起動します:

```
> sudo systemctl enable --now himmelblaud himmelblaud-tasks
```

5. `himmelblau-sshd-config` パッケージをインストールした場合は、SSH デーモンも再起動しておきます:

```
> sudo systemctl restart sshd.service
```

システムの設定が完了すると、Entra ID のユーザがサインインできるようになります。SSH サポートをインストールしている場合は、SSH 経由でもログインできるようになります。なお、初めてユーザがサインインに成功すると、デバイスが Entra ID に登録され、テナントのポリシーで設定された認証要素を入力するように求める場合があります。

7.3 設定の確認

まずはデーモンが動作していることを確認します:

```
> sudo systemctl status himmelblaud himmelblaud-tasks
```

次に Himmelblau デーモンが動作していることを確認します:

```
> sudo aad-tool status
```

さらに NSS で Entra ID のユーザが解決できていることを確認します:

```
> getent passwd USER
> id USER
```

何らかの問題が発生している場合は、システムジャーナルを調査すると良いでしょう:

```
> sudo journalctl -u himmelblaud -u himmelblaud-tasks
```

7.4 サインインできるユーザの制限

`pam_allow_groups` を設定していない場合、設定した Entra ID 内の全てのユーザが認証できるようになります。アクセスを制限したい場合は、許可したいユーザのプリンシパル名や Entra ID グループオブジェクト ID を、`/etc/himmelblau/himmelblau.conf` 内に設定してください:

```
[global]
pam_allow_groups = user@example.com,00000000-1111-2222-3333-444444444444
```

設定の変更後は、Himmelblau の両方のサービスをそれぞれ再起動してください:

```
> sudo systemctl restart himmelblaud himmelblaud-tasks
```

7.5 Intune ポリシー強制の有効化

Himmelblau に対して Intune の準拠ポリシーを適用したい場合は、`/etc/himmelblau/himmelblau.conf` 内でポリシー処理を有効化します:

```
[global]
apply_policy = true
```

設定の変更後は、Himmelblau の両方のサービスをそれぞれ再起動してください:

```
> sudo systemctl restart himmelblaud himmelblaud-tasks
```

8 Active Directory サポート

改訂履歴

2025-03-05

Active Directory* (AD) は LDAP, Kerberos などのサービスを組み合わせたディレクトリサービスです。Microsoft* Windows* で使用されているサービスで、リソースやサービス、ユーザの管理などを行っています。Microsoft Windows ネットワーク内では、Active Directory はこれらのオブジェクトに対する情報提供のほか、オブジェクトへのアクセス制限、ポリシーの強制などを行っています。openSUSE® Leap では、Active Directory ドメインへの参加のほか、Linux マシンを Windows 環境に統合する機能を提供しています。

8.1 Linux と Active Directory 環境の統合

Active Directory クライアントとして設定し、既存の Active Directory ドメインに参加した Linux マシンでは、単純に openSUSE Leap をクライアントとして設定した場合と比較して、下記のような利点が得られます:

SMB を利用した共有ファイルや共有ディレクトリの参照
: GNOME Files (以前は Nautilus と呼ばれていたソフトウエア) を利用することで、SMB 経由で共有されたリソースにアクセスすることができます。

SMB を利用したファイルやディレクトリの共有
: GNOME Files では、Windows に対してディレクトリやファイルを共有することができます。

Windows サーバ内のユーザデータのアクセスと変更
: GNOME Files を利用することで、Windows のユーザデータにアクセスすることができます。Windows サーバ内にファイルやディレクトリを作成したり削除したり、編集したりすることができます。なお、データにアクセスするにあたってパスワードを再度入力する必要はありません。

オフライン認証について
: ネットワークの接続が切れている (オフライン) の場合であっても、何らかの理由で Active Directory サーバに接続ができない状況であっても、ユーザは Linux マシンにログインしてローカルのデータにアクセスすることができるようになります。

Windows のパスワード変更
: Linux 内で Active Directory サポートを設定することで、Active Directory 内に保存されているパスワードポリシーを取り込むことができるようになります。ディスプレイマネージャやコンソールでも、パスワードを入力して変更することができるようになります。もちろん Linux の `passwd` コマンドを使用しても、Windows のパスワードを設定することができます。

Kerberos 対応のアプリケーションでのシングルサインオン

多くのデスクトップアプリケーションが Kerberos に対応しています。つまり、Web サーバやプロキシ、グループウエアアプリケーションなどで、わざわざパスワードを再入力することなく、認証を再利用できるようになります。



注記: Windows Server * 2016 およびそれ以降における Unix 属性の管理について

Windows Server 2016 もしくはそれ以降のバージョンでは、IDMU/NIS サーバ の役割と、Active Directory ユーザとコンピュータ MMC スナップイン向けの Unix 属性 プラグインが削除されています。

しかしながら、Unix 属性は Active Directory ユーザとコンピュータ MMC スナップイン内にある［高度なオプション］を利用することで、手作業で管理することができます。詳しい情報については、https://blogs.technet.microsoft.com/activedirectoryua/2016/02/09/identity-management-for-unix-idmu-is-deprecated-in-windows-server/ (英語) をお読みください。

それ以外にも、手順8.1「［ユーザのログイン管理］を利用した Active Directory ドメインへの参加」に示されている手順を使用して、クライアント側で属性を設定することもできます (詳しくは ステップ 6.c などをお読みください) 。

下記の章では、これまでの章で説明してきた機能について、その背景となる情報を説明しています。Active Directory を利用したファイルやプリンタの共有については、『GNOME ユーザガイド』をお読みください。

8.2 Linux Active Directory サポートに対する背景情報

Linux クライアントを既存の Windows Active Directory ドメインと統合させるには、多くのシステムコンポーネントは相互作用させて動作させる必要があります。下記の章では、Active Directory サーバとクライアントとの間で使用される、主要な技術についての説明を示しています。

ディレクトリサービスとの間で通信を行うにあたって、クライアントとサーバとの間では下記 2 種類のプロトコルを使用します:

LDAP
: LDAP はディレクトリ情報を管理するために最適化されているプロトコルです。Active Directory 環境での Windows ドメインコントローラは、LDAP プロトコルを利用してクライアントとの間でディレクトリ情報をやり取りします。LDAP について、詳しくは 第5章「389 Directory Server を利用した LDAP サービス」をお読みください。

Kerberos
: Kerberos はサードパーティ型の信頼性のある認証サービスです。いったん Kerberos による認証が成功すれば、どのクライアントでもその認証を信頼するように設計することができますので、これによって Kerberos によるシングルサインオン (SSO) を構築することができます。Windows 側にも Kerberos の実装が用意されていますので、Linux クライアントでも Kerberos SSO を使用することができます。Linux における Kerberos について、詳しくは 第6章「Kerberos を利用したネットワーク認証」をお読みください。

Kerberos 認証を設定するにあたって、どの YaST モジュールを使用するのかによって、アカウント情報や認証データを処理するコンポーネントが変わってきます:

SSSD をベースにしたソリューション

- このソリューションでは、`sssd` デーモンを中枢として使用します。このデーモンが Active Directory サーバとの通信を処理することになります。
- 名前解決を行う場合は、`sssd_nss` を使用します。
- ユーザ認証の際には、PAM 向けの `pam_sss` を使用します。また、Active Directory ユーザがログインした際に Linux クライアント側でホームディレクトリを作成する処理については、`pam_mkhomedir` が実施します。
  PAM に関する詳細については、第2章「PAM を利用した認証」をお読みください。

Winbind (Samba) をベースにしたソリューション

- このソリューションでは、`winbindd` デーモンを中枢として使用します。このデーモンが Active Directory サーバとの通信を処理することになります。
- 名前解決を行う場合は、`nss_winbind` を使用します。
- ユーザ認証の際には、PAM 向けの `pam_winbind` を使用します。また、Active Directory ユーザがログインした際に Linux クライアント側でホームディレクトリを作成する処理については、`pam_mkhomedir` が実施します。
  PAM に関する詳細については、第2章「PAM を利用した認証」をお読みください。

図8.1「Winbind ベースの Active Directory 認証のスキーマ」には、Winbind ベースの Active Directory 認証を使用する際の、主要なコンポーネントを示しています。

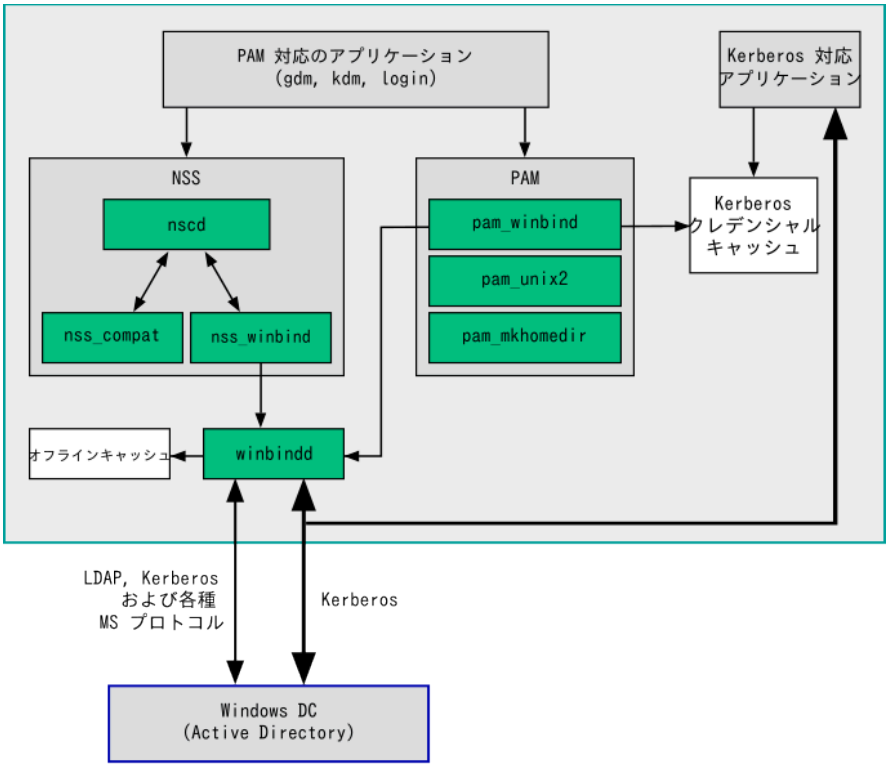


図 8.1: WINBIND ベースの ACTIVE DIRECTORY 認証のスキーマ

ログインルーチンや GNOME ディスプレイマネージャなど、PAM に対応しているアプリケーションであれば、PAM や NSS のレイヤと互いにやり取りを行うことができますので、Windows サーバとの間でも認証を行うことができます。また、ファイルマネージャや Web ブラウザ、電子メールクライアントなど、Kerberos 認証に対応したアプリケーションの場合は、Kerberos のクレデンシャルキャッシュを使用することができますので、ユーザが持つ Kerberos チケットにアクセスすることができます。これにより、SSO 環境を構成することができます。

8.2.1 ドメインへの参加

ドメインへの参加の際、サーバとクライアントの間は緊密な連携を確立します。クライアント側では、Windows ドメインコントローラが提供する既存の LDAP と Kerberos SSO 環境に対して、参加処理を行う必要があります。参加に必要な全ての処理は、YaST のドメインメンバーシップモジュールが実施します。ドメインメンバーシップモジュールは、インストール時にもインストール後にも実行することができます:

1. LDAP と KDC (鍵配送センター) の両方のサービスを提供するドメインコントローラの場所を設定します。

2. クライアントを参加させるためのマシンアカウントを、ディレクトリサービス内に作成します。

3. 最初のチケット発行許諾チケット (TGT) をクライアント向けに取得し、ローカルの Kerberos クレデンシャルキャッシュ内に保存します。クライアント側では、この TGT を利用することで、LDAP の問い合わせを行うディレクトリサーバとの通信など、様々なサービスに対するチケットを取得します。

4. NSS と PAM の設定が調整され、ドメインコントローラとの間でクライアントが認証できるように設定を行います。

クライアントの起動時、winbind デーモンを開始して、マシンアカウントに対応する Kerberos の初期チケットを取得します。winbindd は、有効なチケットを常に確保できるよう、必要に応じて自動更新の処理も行います。また、現在のアカウントポリシーを追跡する目的で、winbindd はドメインコントローラに対して定期的に問い合わせを行います。

8.2.2 ドメインログインとユーザのホームディレクトリ

GNOME のログインマネージャ (GDM) は、Active Directory のドメインログインに対応できるように拡張されています。マシンが直接参加しているドメインのユーザのほか、信頼関係を設定したドメインのユーザがログインすることができます。

ユーザの認証は 8.2項「Linux Active Directory サポートに対する背景情報」で説明しているとおり、いくつかの PAM モジュールを介して行われています。何らかのエラーが発生すると、PAM が提供するエラーコードを読みやすい形式に変換したあと、対応する形式 (GDM, コンソール, SSH など) でエラーメッセージを表示します:

`Password has expired` (パスワードの期限が切れています)
: パスワードの有効期限が切れているため、変更を行う必要があることを示しています。システム側では新しく設定するパスワードを尋ねるほか、新しいパスワードが企業内のパスワードポリシーに合致していない (短すぎる, 単純すぎる, 既に履歴内に存在するなど) 場合に、メッセージを表示する処理も行います。パスワードの変更が失敗した場合は、理由が表示されて再度パスワードの変更を求めます。

`Account disabled` (アカウントが無効化されています)
: アカウントが無効化されているため、ログインできないことを示しています。この場合、システム管理者に連絡を取る必要があります。

`Account locked out` (アカウントがロックアウト (施錠) されています)
: アカウントが施錠されているため、ログインできないことを示しています。この場合、システム管理者に連絡を取る必要があります。

`Password has to be changed` (パスワードを変更する必要があります)

ユーザはログインを行うことができるものの、パスワードを近い将来に変更しなければならないことを示しています。この警告メッセージは、パスワードの有効期限が切れる 3 日前から表示されます。パスワードの有効期限が切れると、ユーザはログインできなくなります。

`Invalid workstation` (不正なワークステーションです)

ユーザが特定のワークステーションからのログインに制限されていて、その中に openSUSE Leap のマシンが含まれていない場合、このメッセージが表示されます。このメッセージが表示された場合、ユーザはログインできなくなります。

`Invalid logon hours` (不正なログイン時刻です)

ユーザが特定の時間帯にのみログインできるように制限されていて、その時間外にログイン使用とした場合、このメッセージが表示されます。このメッセージが表示された場合、ユーザはログインできなくなります。

`Account expired` (アカウントの有効期限が切れています)

管理者は特定のユーザアカウントに対して有効期限を設定することができます。有効期限の超過後に該当のユーザがログインしようとした場合、このメッセージが表示されます。このメッセージが表示された場合、ユーザはログインできなくなります。

認証が成功した場合、クライアントは Active Directory の Kerberos サーバからチケット発行許諾チケット (TGT) を取得し、ユーザのクレデンシャルキャッシュ内に保存します。TGT は裏で自動更新されますので、ユーザ側での更新操作は不要となります。

openSUSE Leap には、Active Directory ユーザに対してローカルのホームディレクトリを提供する機能が用意されています。8.3項「Active Directory 向けの Linux クライアントの設定」の手順に従って YaST で設定を行うと、Linux クライアントで Windows/Active Directory のユーザが初めてログインした際に、ユーザのホームディレクトリは自動的に作成されます。作成されるホームディレクトリは通常のものと同じで、Active Directory のドメインコントローラとは独立して提供される形になります。

Linux 側でローカルのホームディレクトリを使用していて、かつオフライン認証を設定していると、たとえ Active Directory サーバとの接続が切れた場合であっても、このマシン内にあるユーザデータにアクセスができるようになります。

8.2.3 オフラインサービスとポリシーサポート

企業環境では、ネットワークを切り替えたりネットワークを切断したりしても、通常通りにログインすることのできる「ローミング」機能が求められます。ネットワークが接続されていないマシンに対してユーザがログインできるようにする目的で、winbind デーモンに対して様々なキャッシュ機能が用意されています。winbind デーモンはオフライン状態でもパスワードポリシーを守るように動作しますので、ログイン

の失敗回数を記録することができるなど、Active Directory 内で設定されているポリシーに従って動作を行います。オフラインサポートは既定では無効化されていますので、YaST のドメインメンバーシップモジュールで明示的に有効化しなければなりません。

ドメインコントローラと通信ができなくなった場合でも、ユーザは接続が確立していた際に正しい Kerberos チケットを取得しているため、Windows クライアントと同様に、ドメインコントローラ以外のネットワークリソースにアクセスすることができます。ドメインコントローラへのアクセスができない場合は、パスワードの変更については行うことができません。また、特定の Active Directory サーバとの通信が切れている場合は、ユーザはそのサーバにあるデータにもアクセスできなくなります。特定のクライアントが完全にネットワークから切り離されてしまった場合でも、後からネットワークに接続し直すと、ユーザがデスクトップをロック (施錠) してアンロック (解錠) したタイミング (例: スクリーンセーバ) で、openSUSE Leap は新しい Kerberos チケットを取得することができます。

8.3 Active Directory 向けの Linux クライアントの設定

クライアントが Active Directory ドメインに参加できるようになるには、ネットワークの設定に対していくつかの調整を行って、クライアントとサーバの間でのやり取りを問題なく行うことができるようにする必要があります。

DNS
: クライアント側では、DNS のリクエストを Active Directory の DNS サーバに転送することのできる DNS サーバを設定する必要があります。もちろん Active Directory の DNS サーバ自身を DNS サーバとして設定してもかまいません。

NTP
: Kerberos 認証を正しく動作させるには、クライアント側でも正しい時刻を保持していなければなりません。この目的を達成するため、中央の NTP タイムサーバを使用するよう設定しておくことをお勧めします (Active Directory ドメインコントローラ内で NTP サーバを動作させてもかまいません) 。Linux ホストとドメインコントローラとの時刻が一定以上ずれていると、Kerberos の認証は常に失敗するようになりますので、より弱い NTLM (NT LAN Manager) を利用してログインするようになってしまいます。Active Directory を時刻同期に使用する方法について、詳しくは 手順8.2「[Windows ドメインメンバーシップ] を利用した Active Directory ドメインへの参加」をお読みください。

ファイアウオール
: ネットワーク内を探索できるようにするためには、ファイアウオールを完全に無効化するか、探索に使用するインターフェイスを内部ゾーンとして設定してください。

クライアント側でファイアウオールの設定を変更するには、`root` でログインして YaST ファイアウオールモジュールを起動し、［インタフェース］を選択します。表示された一覧からインターフェイスを選択して、［ゾーンの変更］を押します。あとは［internal］を選択して［OK］を押し、設定を反映させてください。

Active Directory アカウント

Active Directory の管理者がドメイン用の正しいアカウントを提供しない限り、Active Directory ドメインに参加することはできません。Linux クライアントから Active Directory ドメインのユーザ名とパスワードを入力してログインし、ドメインに参加してください。

8.3.1 Active Directory に接続する際に使用する YaST モジュールの選択

YaST には Active Directory との接続を行うにあたって、複数のモジュールが提供されています:

- ［ユーザのログイン管理］: 識別情報提供サービス (通常は LDAP) とユーザ認証サービス (通常は Kerberos) の 両方を併用して設定を行います。この設定機能は SSSD をベースにした仕組みで、Active Directory ドメインへの参加を行う場合には最適な選択肢となります。
  このモジュールについては 8.3.2項「［ユーザのログイン管理］を利用した Active Directory への参加」で説明しています。

- ［Windows ドメインメンバーシップ］: Kerberos と LDAP をそれぞれ利用して Active Directory ドメインへの参加を行います。この設定機能は `winbind` をベースにした仕組みで、NTLM での認証が必要な Active Directory ドメインに参加する必要がある場合や、フォレストを 跨いだ認証が必要な場合に最適な選択肢となります。
  このモジュールについては 8.3.3項「［Windows ドメインメンバーシップ］を利用した Active Directory ドメインへの参加」で説明しています。

8.3.2 ［ユーザのログイン管理］を利用した Active Directory への参加

YaST の［ユーザのログイン管理］モジュールでは、Active Directory 認証に対応しています。これに加えて、下記の識別子データを提供するサービスと、認証を処理するサービスに対応しています:

識別子データを提供するサービス

- ［サードパーティ製のソフトウエアライブラリに委任する］: プロキシを介して古い NSS プロバイダに対応します。
- ［FreeIPA］: FreeIPA および Red Hat Enterprise Identity Management のプロバイダです。
- ［汎用ディレクトリサービス (LDAP)］: LDAP のプロバイダです。LDAP に関する詳細は、`man 5 sssd-ldap` をお読みください。
- ［ローカルの SSSD ファイルデータベース］: ローカルユーザ向けの SSSD 内部プロバイダです。

認証を処理するサービス

- ［サードパーティ製のソフトウエアライブラリに委任する］: プロキシサービスを介して、他の PAM ターゲットに認証を委任します。
- ［FreeIPA］: FreeIPA および Red Hat Enterprise Identity Management のプロバイダです。
- ［汎用 Kerberos サービス］: LDAP プロバイダです。
- ［汎用ディレクトリサービス (LDAP)］: Kerberos 認証です。
- ［ローカルの SSSD ファイルデータベース］: ローカルユーザ向けの SSSD 内部プロバイダです。
- ［ドメインは認証サービスを提供していません］: 明示的に認証を無効化します。

SSSD と YaST の［ユーザのログイン管理］モジュールを利用して Active Directory ドメインに参加するには、下記の手順を実施します:

手順 8.1: ［ユーザのログイン管理］を利用した ACTIVE DIRECTORY ドメインへの参加

1. YaST を開きます。

2. 後から DNS の自動検出機能を利用できるようにするため、Active Directory のドメインコントローラ (Active Directory サーバ) をクライアント側のネームサーバとして設定します。

   a. YaST で［ネットワークの設定］を押します。

   b. ［ホスト名／DNS］を選択して、［ネームサーバ 1］のテキストボックス内に、Active Directory ドメインコントローラの IP アドレスを入力します。
   ［OK］を押して設定を保存します。

3. YaST のメインウインドウに戻って、今度は［ユーザのログイン管理］を押します。
お使いのコンピュータに対するネットワーク設定の概要と、現在使用されている認証方式が表示されます。



図 8.2: ［ユーザのログイン管理］のメインウインドウ

4. 編集を行うため、［設定の変更］を押します。

5. ここからドメインに参加します。

   a. ［ドメインの追加］を押します。

   b. 表示されたダイアログで、まずは［ドメイン名］に正しい値を入力します。その後、識別子データとユーザ認証でそれぞれ［Microsoft Active Directory］を選択します。
   このとき、［ドメインの有効化］を選択していることをご確認ください。
   ［OK］を押します。

c. 通常は、後続のダイアログの内容を変更する必要はありません。既定値のままでかまいません。ですが、場合によっては設定を変更する必要がある場合もあります:

   - **ドメインコントローラ内で設定しているホスト名とローカルのホスト名が異なる場合:** お使いのコンピュータのホスト名と Active Directory のドメインコントローラ内に登録されているホスト名が同じになっているかどうかを確認します。端末を開いて `hostname` コマンドを実行し、出力された内容が Active Directory のドメインコントローラ内にあるかどうかを確認してください。
     ホスト名が異なる場合は、［AD ホスト名］の欄に、Active Directory 内に登録されているホスト名を入力してください。それ以外の場合は、左記のテキストボックスには何も記入しないでください。

   - **DNS Auto-Discovery を使用したくない場合:** ［AD サーバのホスト名］内に、使用したいサーバを指定します。複数のドメインコントローラが存在する場合は、カンマ区切りで指定してください。

d. ［OK］を押して続行します。
   この時点で必要なソフトウエアがインストールされていない場合は、ここで必要なソフトウエアをインストールします。その後 Active Directory のドメインコントローラが利用できるかどうかを確認します。

e. 全ての設定が正しければ、後続のダイアログ内の［Active Directory Server］に検出されたサーバが表示されます。ただし、この時点では［まだ登録されていません］と表示されます。
   ダイアログ内にある［ユーザ名］と［パスワード］の欄に、Active Directory の管理者アカウント (通常は `Administrator`) の情報を入力します。
   現在のドメインが Samba 向けに有効化されていることを確認したい場合は、［この AD で動作するように Samba の設定を上書きする］を選択します。
   登録を行うには［OK］を押します。



図 8.3: ドメインへの登録

f. 登録が正常に行われたことを表すメッセージが表示されるはずです。メッセージを確認したら、［OK］を押します。

6. 登録完了後は、［ドメインユーザのログオンの管理］ウインドウ内で、クライアント側の設定を行います。

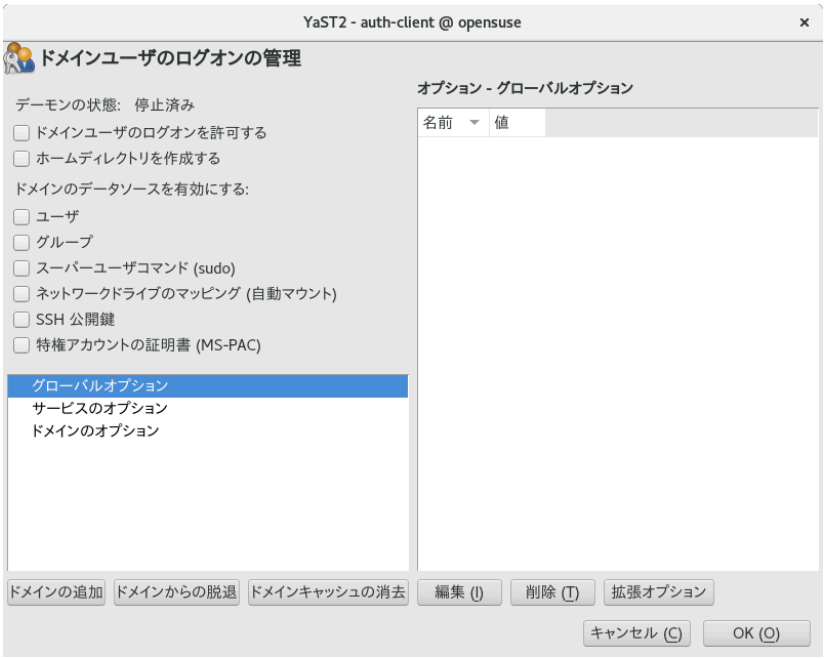


図 8.4: ［ユーザのログオンの管理］の設定ウインドウ

a. Active Directory 側に設定されているログイン情報を利用して、コンピュータにログインできるようにするには、［ドメインユーザのログオンを許可する］を選択します。

b. 必要であれば、［ドメインのデータソースを有効にする］以下にある項目を選択します。たとえば `sudo` コマンドを使用する許可を与えるユーザや、ネットワークドライブを利用させるかなどの設定を行うことができます。

c. Active Directory のユーザに対してホームディレクトリを作成するには、［ホームディレクトリを作成する］を選択します。ホームディレクトリのパスは、クライアントやサーバ、もしくはその両方など、様々な方法で設定することができます:

   - ドメインコントローラ側でホームディレクトリを設定したい場合は、それぞれのユーザに対して、`UnixHomeDirectory` という属性を設定してください。これに加えて、この属性がグローバルカタログ内で複製されるように設定する必要があります。Windows 側での設定方法について、詳しくは https://support.microsoft.com/kb/248717 をお読みください。

   - クライアント側に対して、ドメインコントローラで設定したホームディレクトリのパスを優先するように設定したい場合は、`fallback_homedir` オプションを設定してください。

   - クライアント側に対して、ドメインコントローラで設定したホームディレクトリの設定を上書きしたい場合は、`override_homedir` オプションを設定してください。

   ドメインコントローラ側の設定については、本文書内では説明していません。下記では、クライアント側の設定項目のみを説明しています。
   左側の枠内で［サービスのオプション］›［名前の切り替え］を選び、［拡張オプション］を押します。表示されたウインドウ内で、`fallback_homedir` もしくは `override_homedir` のいずれかを選択して、［追加］を押します。
   値を指定します。ホームディレクトリは `/home/ユーザ名` の形式で、たとえば `/home/%u` のように指定します。設定可能な値に関する情報は、`sssd.conf` のマニュアルページ（`man 5 sssd.conf`）内にある override_homedir の章をお読みください。
   ［OK］を押します。

7. ［OK］を押して設定を保存します。あとは表示された値が正しいことを確認して、［OK］を押して閉じます。

8.3.3 ［Windows ドメインメンバーシップ］を利用した Active Directory ドメインへの参加

`winbind` と YaST の［Windows ドメインメンバーシップ］を利用して Active Directory ドメインに参加したい場合は、下記の手順で行います:

手順 8.2: ［WINDOWS ドメインメンバーシップ］を利用した ACTIVE DIRECTORY ドメインへの参加

1. `root` でログインし、YaST を起動します。
2. ［ネットワークサービス］›［Windows ドメインメンバーシップ］を選択します。
3. ［Windows ドメインメンバーシップ］の画面 (図8.5「Windows ドメインメンバーシップの設定」をご覧ください) が表示されたら、［ドメインまたはワークグループ］内に、参加するドメインを入力します。お使いのマシンでの DNS 設定が Windows 側の DNS サーバに設定されていれば、`mydomain.mycompany.com` のような DNS ホスト名を入力することができます。ドメイン名を短い形式 (Windows 2000 以前のドメイン名) で入力した場合は、ドメインコントローラを検出する際に DNS ではなく NetBIOS の名前解決機構を使用するようになります。



図 8.5: WINDOWS ドメインメンバーシップの設定

4. Linux 側での認証時に SMB を使用する場合は、［Linux の認証に SMB の情報を使用する］を選択します。
5. Linux マシン側で Active Directory のユーザのホームディレクトリを自動的に作成したい場合は、［ログイン時にホームディレクトリを作成］を選択します。

6. Active Directory サーバとの通信が一時的に途切れるような場合や、ネットワークの接続を持たないような環境の場合は、［オフライン認証］を選択します。

7. Samba ユーザやグループに対する UID や GID の範囲を変更したい場合は、［熟練者向け設定］を押します。なお、必要な場合にのみ、DHCP で WINS サーバのアドレスを取得するように設定してください。これは、WINS システムでのみ名前を解決することのできるマシンが存在する場合にのみ設定します。

8. Active Directory 環境で NTP の時刻同期を設定するには、［NTP の設定］を押して、サーバ名もしくは IP アドレスを指定します。この手順は、通常の NTP 設定モジュールを呼び出すだけの仕組みであるため、ここまでで設定済みであれば特に何も行う必要はありません。

9. ［OK］を押してウインドウを閉じます。確認メッセージが表示されたら、確認に応答します。

10. Active Directory サーバ内での Windows 管理者のパスワードを入力して［OK］を押します (図8.6「管理者での認証」をご覧ください) 。

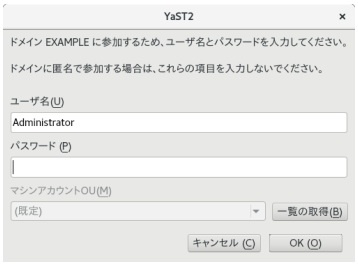


図 8.6: 管理者での認証

Active Directory ドメインへの参加を行ったあとは、お使いのデスクトップのディスプレイマネージャやコンソールなどから、ログインができるようになります。

重要: ドメイン名について

ドメイン名が `.local` で終わるドメインに対しては、ドメインへの参加が成功しないことがあります。`.local` で終わるドメインは、マルチキャスト DNS (MDNS) でリンクローカルのホスト名として予約されている名前であるためです。

注記: 管理者のみがコンピュータをドメインに参加させることができる件について

ドメインの管理者権限を持つアカウント (例: `Administrator`) のみが Active Directory に参加することができます。

8.3.4 Active Directory の接続状態の確認

Active Directory ドメインに正しく参加できているかどうかを確認したい場合は、下記のコマンドを実行してください:

- `klist` を実行すると、現在のユーザが正しい Kerberos チケットを保持しているかどうかを表示することができます。
- `getent passwd` を実行すると、LDAP データとして公開されている全てのユーザを一覧表示することができます。

8.4 Active Directory ドメインへのログイン

お使いのコンピュータで Active Directory に対する認証設定を完了し、かつ Windows 側のユーザアカウントをお持ちの場合は、Active Directory の認証情報で Linux マシンにログインすることができるようになります。ログインは GNOME のほか、コンソールや SSH 、その他の PAM 対応アプリケーションで行うことができます。

重要: オフライン認証について

openSUSE Leap ではオフライン認証にも対応しているため、オフラインの状態でもクライアントマシンにログインできるようになります。詳しくは 8.2.3項「オフラインサービスとポリシーサポート」をお読みください。

8.4.1 GDM

Active Directory サーバに対して GNOME のクライアントマシンから認証を行いたい場合は、下記の手順を実施します:

1. [アカウントが見つかりませんか?] を押します。
2. [ユーザー名] の欄に、Windows 側のドメイン名とユーザ名を `ドメイン名\ユーザ名` の書式で指定します。
3. Windows 側のパスワードを入力します。

ホームディレクトリを作成する設定がされていれば、openSUSE Leap は初回のログイン時に、ローカルマシン内にホームディレクトリを作成します。これにより、openSUSE Leap における機能を損なうことなく、Active Directory へのサポートを利用できるようになります。

8.4.2 コンソールログイン

グラフィカルなフロントエンドだけでなく、Active Directory のユーザはテキストベースのコンソールも、SSH 経由でのログインも行うことができます。

コンソールで Active Directory のユーザを利用してログインするには、`login:` のプロンプトで `ドメイン名\ユーザ名` の形式でユーザ名を入力し、パスワードを入力します。

SSH を利用して Active Directory のユーザがログインする場合は、下記の手順で行います:

1. 下記のように入力します:

   ```
   > ssh ドメイン名\\ユーザ名@ホスト名
   ```

   `\` はエスケープ処理を行う必要があることから、2 回入力していることに注意してください。

2. あとはユーザのパスワードを入力します。

8.5 パスワードの変更

openSUSE Leap では、企業内のセキュリティポリシーに適合する新しいパスワードを入力するよう、ユーザに求める機能が用意されています。対応する PAM モジュールがドメインコントローラ内の現在のパスワードポリシーを取得して、ログイン時にユーザアカウントに対して要求されているパスワード品質に関するメッセージを表示することができます。Windows 側のモジュールと同様に、openSUSE Leap でも下記の情報を含むメッセージを出力することができます:

- パスワードの履歴設定
- 最小パスワード長の要件
- パスワードの変更禁止期間
- パスワードの複雑さに関する要件

パスワードの変更処理は、全ての要件にきちんと適合していない限り、成功することはありません。パスワードの状態に関するフィードバックは、ディスプレイマネージャとコンソールの両方で提供されています。

GDM ではパスワードの有効期限に関する情報を表示することができるほか、対話的にパスワードを変更するように求めることもできます。ディスプレイマネージャ内でパスワードを変更する場合は、求められたとおりにパスワードを指定してください。

Windows のパスワードを変更したい場合は、サーバ内でパスワードを変更したりする必要はなく、標準的な Linux ユーティリティである `passwd` コマンドを使用することができます。

1. コンソールにログインします。
2. `passwd` と入力します。
3. 必要に応じて現在のパスワードを入力します。
4. 新しく設定するパスワードを入力します。
5. 確認のため、新しく設定するパスワードを再入力します。新しく設定するパスワードが Windows サーバ内のポリシーに適合していない場合は、その旨を表すメッセージが表示され、異なるパスワードを使用するように求められます。

GNOME デスクトップで Windows のパスワードを変更したい場合は、下記のようにして行います:

1. パネルの左上にある［コンピュータ］アイコンを押します。
2. ［コントロールセンター］を選択します。
3. ［詳細］内にある［ユーザー］›［パスワード］を押します。
4. 現在のパスワードを入力します。
5. 新しく設定するパスワードを 2 回入力します。
6. ［変更］を押して設定を適用し、ダイアログを閉じます。

8.6 Active Directory での証明書自動登録

証明書の自動登録機能は、ネットワークに接続されている openSUSE Leap などのデバイスが Active Directory の証明書サービスに対して証明書を自動登録する機能を指します。この機能はユーザ側での操作無しに実施することができます。これは Active Directory のグループポリシーで管理されている機能で、Samba では `samba-gpupdate` コマンドで提供されています。

8.6.1 サーバ側での証明書自動登録設定

Windows サーバ側では、**証明機関**，**証明書の登録ポリシー Web サービス**，**証明書の登録 Web サービス**，**ネットワーク デバイス登録サービス** という各名称の役割サービス 全てをインストールし、設定しておかなければなりません。

グループポリシーでの自動登録の設定は、下記に示す Microsoft 文書に説明があります: https://docs.microsoft.com/ja-jp/windows-server/networking/core-network-guide/cncg/server-certs/configure-server-certificate-autoenrollment#configure-server-certificate-auto-enrollment

8.6.2 クライアント側での証明書自動登録の有効化

クライアント側で証明書自動登録を有効化するには、下記の手順を実施してください:

1. まずは `samba-gpupdate` パッケージをインストールします。これにより `certmonger` , `cepces` , `sscep` の各パッケージを依存関係としてインストールするはずです。Samba では `sscep` コマンドを利用して証明機関のルートチェインをダウンロードし、`certmonger` と `cepces` を併用してホスト側の証明書テンプレートを監視します。
2. 次に Active Directory ドメインに参加します (8.6.1項「サーバ側での証明書自動登録設定」で説明している、証明機関の設定されたドメインに参加します) 。
3. Winbind で参加したマシンの場合は、`smb.conf` のグローバルパラメータ内に `apply group policies = yes` という行を追記します。
4. SSSD で参加したマシンの場合は、https://github.com/openSUSE/oddjob-gpupdate から `oddjob-gpupdate` をインストールします。
5. あとは証明書の自動登録が正しく設定されていることを確認します。具体的には、クライアント側で下記を実行します:

   ```
   > /usr/sbin/samba-gpupdate --rsop
   ```

   正しく設定できていれば、下記のような出力が現れるはずです:

   ```
   Resultant Set of Policy
   Computer Policy
   GPO: Default Domain Policy
   ============================================================
   CSE: gp_cert_auto_enroll_ext
   ------------------------------------------------------------
   Policy Type: Auto Enrollment Policy
   ------------------------------------------------------------
   [ <CA NAME> ] =
   [ CA Certificate ] =
   ----BEGIN CERTIFICATE----
   <CERTIFICATE>
   ----END CERTIFICATE----
   [ Auto Enrollment Server ] = <DNS NAME>
   ```

6. さらに下記のコマンドを実行して、インストール済みの証明書を表示します:

   ```
   > getcert list
    Number of certificates and requests being tracked: 1.
    Request ID 'Machine':
    status: MONITORING
    stuck: no
   ```

```
key pair storage: type=FILE,location='/var/lib/samba/private/certs/Machine.key'
certificate: type=FILE,location='/var/lib/samba/certs/Machine.crt'
CA: <CA NAME>
issuer: CN=<CA NAME>
subject: CN=<HOSTNAME>
expires: 2017-08-15 17:37:02 UTC
dns: <hostname>
key usage: digitalSignature,keyEncipherment
eku: id-kp-clientAuth,id-kp-serverAuth
certificate template/profile: Machine
```

証明書は `/var/lib/samba/certs` に、機密鍵は `/var/lib/samba/private/certs` にそれぞれインストールされます。

詳しくは `man samba-gpupdate` をお読みください。

9 FreeRADIUS サーバの構築

改訂履歴

2024-06-26

RADIUS (Remote Authentication Dial-In User Service; リモート認証ダイヤルインユーザサービス) プロトコルは、ネットワークへの接続を管理するための標準的なサービスです。認証／認可／アカウンティングの各機能 (AAA と略します) を、インターネットサービスプロバイダや携帯電話網のキャリアのような大規模ビジネスから、小規模なネットワークに至るまで、様々な環境に向けて提供しています。このプロトコルはユーザやデバイスを認証し、特定のネットワークサービスに対する認可を与え、課金や監査のための使用状況の監視を行います。もちろん AAA 全ての機能を使用する必要はなく、必要な機能だけを選ぶことができます。たとえばクライアントの認証機能だけを使用して使用状況の監視は行わないような場合にも対応できますし、逆に使用状況の監視だけを使用しておいて、認可機能は別のサービスを使用するようなこともできます。

このプロトコルは効率的に動作しますので、最近のハードウエアであれば数千ものリクエストを処理できます。もちろんネットワークプロトコルの種類にも依存しませんので、ダイヤルアップ以外の環境でもそのまま使用することができます。

RADIUS は分散型のアーキテクチャで動作していて、ネットワークに接続するためのアクセスサーバ (NAS) とは別に用意するのが一般的です。ユーザのアクセスデータは中央の RADIUS サーバ内に保管され、多数の NAS に対して提供されます。逆に NAS は、イーサネットスイッチや無線 LAN のアクセスポイントなど、ネットワークへの物理的なアクセス機能を提供します。

FreeRADIUS はオープンソースで開発されている RADIUS 実装で、最も広く使用されている RADIUS サーバでもあります。本章では FreeRADIUS サーバのインストールとテストの方法について説明しています。なお、RADIUS はさまざまな用途に対応できることから、初期設定を行ったあとは公式のドキュメンテーション (詳しくは https://freeradius.org/documentation/) をお読みになって、残りの必要な設定を行ってください。

9.1 openSUSE Leap でのインストールとテスト

下記の手順では単純なテスト用のシステムを構築します。サーバが正しく動作していることを確認したら、必要な残りの設定を行って本番環境を構築してください。なお、用途によっては手順を巻き戻す必要があるかもしれません。

まずは `freeradius-server` と `freeradius-server-utils` のパッケージをインストールします。インストールが終わったら、`/etc/raddb/certs` ディレクトリ内に移動して `bootstrap` を実行し、テスト用の証明書を作成してください:

```
# zypper in freeradius-server freeradius-server-utils
# cd /etc/raddb/certs
```

```
# ./bootstrap
```

`certs` ディレクトリ内にある README ファイルには、たくさんの有益な情報が書かれています。`bootstrap` スクリプトの実行が完了したら、あとはデバッグモードでサーバを起動します:

```
# radiusd -X
[...]
Listening on auth address * port 1812 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on auth address :: port 1812 bound to server default
Listening on acct address :: port 1813 bound to server default
Listening on auth address 127.0.0.1 port 18120 bound to server inner-tunnel
Listening on proxy address * port 54435
Listening on proxy address :: port 58415
Ready to process requests
```

`Listening` と `Ready to process requests` と書かれた行が表示されたら、サーバが問題なく起動したことになります。これらが表示されない場合は、出力された内容をよくお読みのうえ、何が問題なのかを調べてください。なお、出力された内容を調べる場合は、`tee` コマンドでテキストファイルに出力しておくと便利です:

```
> radiusd -X | tee radiusd.text
```

次にテスト用のクライアントとユーザを用意して、認証のテストを行います。クライアントは通常、無線 LAN のアクセスポイントやスイッチなど、RADIUS サーバのクライアントとして動作するものを使用します。サーバ側では、クライアントは `/etc/raddb/client.conf` ファイルに、ユーザは `/etc/raddb/mods-config/files/authorize` にそれぞれ設定します。

`/etc/raddb/mods-config/files/authorize` ファイルを開いて、下記のような行を入力します:

```
bob   Cleartext-Password := "hello"
Reply-Message := "Hello, %{User-Name}"
```

テスト用のクライアントである `/etc/raddb/client.conf` は `/etc/raddb/client.conf` 内に既に設定されていて、`testing123` というシークレット値が設定されています。これを利用して認証のテストを行います。一般ユーザで端末をもう 1 つ開いてから `radtest` コマンドを実行して、bob でのログインを試してみます:

```
> radtest bob hello 127.0.0.1 0 testing123
Sent Access-Request Id 241 from 0.0.0.0:35234 to 127.0.0.1:1812 length 73
        User-Name = "bob"
        User-Password = "hello"
        NAS-IP-Address = 127.0.0.1
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "hello"
```

```
Received Access-Accept Id 241 from 127.0.0.1:1812 to 0.0.0.0:0 length 20
```

`radius -X` を実行しているほうの端末には、ログインの成功を表す下記のような出力が現れているはずです:

```
(3) pap: Login attempt with password
(3) pap: Comparing with "known good" Cleartext-Password
(3) pap: User authenticated successfully
(3)     [pap] = ok
[...]
(3) Sent Access-Accept Id 241 from 127.0.0.1:1812 to 127.0.0.1:35234 length 0
(3) Finished request
Waking up in 4.9 seconds.
(3) Cleaning up request packet ID 241 with timestamp +889
```

次にネットワーク内にある異なるコンピュータからログインテストを行います。サーバ側の `clients.conf` は下記のようなクライアント設定が含まれていますので、こちらのコメント文字 (#) を外して IP アドレスなどの必要な修正を行います:

```
client private-network-1 }
  ipaddr          = 192.0.2.0/24
  secret          = testing123-1
  {
```

クライアント側では、さまざまなテストコマンドを提供している `freeradius-server-utils` パッケージをインストールしておいてください。インストールが終わったら `radtest` コマンドを利用し、`bob` としてログインを試してみます。なお、ホスト名でなく IP アドレスで指定しておいたほうが高速に処理されます:

```
> radtest bob hello 192.168.2.100 0 testing123-1
```

ログインが失敗するような場合は、まず全ての出力をよく読んで原因を調査してください。設定ファイルにはいくつかのテスト用のユーザやクライアントが書かれていますので、こちらを試してみてもかまいません。また、設定ファイル内にはコメントでさまざまな情報が書かれています (ただし英語です) ので、ここからさまざまなことを学べます。テストに問題がなく、本番で使用するための設定を始めたい場合は、まず `/etc/raddb/certs` ディレクトリ内にあるテスト用の証明書を全て削除して、実際に使用するものに置き換えてください。また、テスト用のユーザやクライアントの設定もコメントアウト (#) してください。なお、テスト用に動作させた `radiusd` を停止するには、`Ctrl` – `C` を押してください。本番環境では `systemctl` の `radiusd.service` で管理を行います。

お使いのネットワーク内で FreeRADIUS を適用するための方法について、詳しくは https://freeradius.org/documentation/ や https://networkradius.com/freeradius-documentation/ (いずれも英語) をお読みください。こちらにはさまざまな情報や手順などが示されています。

II ローカルセキュリティ

10 物理的なセキュリティ 136

11 ソフトウエア管理 146

12 ファイルの管理 150

13 パーティションやファイルの暗号化 158

14 cryptctl を利用したアプリケーション向けのストレージ暗号化 165

15 ユーザ管理 178

16 cron と at の制限 198

17 Spectre/Meltdown チェッカー 201

18 YaST を利用したセキュリティの設定 204

19 Polkit 認可フレームワーク 209

20 Linux でのアクセス制御リスト 217

21 AIDE を利用した侵入検知 229

10 物理的なセキュリティ

改訂履歴

2024-06-26

物理的なセキュリティ確保はとても重要です。Linux の本番サーバは、セキュリティチェックを通過した人間のみがアクセスできる、鍵のかかったデータセンター内に配置されるべきです。環境や状況にもよりますが、ブートローダに対してパスワードを設定することも検討すべきです。

上記に加えて、下記を考慮する必要もあります:

- ホストに対して直接アクセスできるのは誰か?
- その彼らがすべきことは?
- ホストを改ざんから保護することができるか?

特定のシステムにおける物理的なセキュリティは、そのシステムが配置されている状況に依存して決まるほか、利用可能な予算によっても大きく異なります。

10.1 システムの施錠

データセンター内のほとんどのサーバラックには施錠の機能が備わっています。通常はラックの前面に掛け金やシリンダー錠の形式で用意され、施錠や解錠を行うことで内部へのアクセスを許可したり禁止したりすることができます。かご形のラックであればサーバからデバイスを不正に差し替えたり抜きとったりすることを防げますし、筐体を開けたり妨害したりすることも防ぐことができます。また、システムを不正に再起動されたり、代替デバイス (例: CD/DVD/USB メモリなど) から起動できたりしないようにすることも重要です。

サーバによってはケースロック (施錠) を備えているものもあります。これらのロックはシステムの製造元や構造によって異なるもので、ほとんどのケースロックは筐体を開けなくする目的で提供されていますが、キーボードやマウスなどを接続できないようにするものもあります。施錠は便利な仕組みではありますが、安全性という観点では低く、悪意のある人間であれば難なく突破できてしまう程度の仕組みでもあります。

10.2 BIOS のロックダウン



ヒント: Secure Boot

本章では起動処理の安全性を高めるための基本的な方式のみを説明しています。UEFI を利用したより高度な起動保護方法や Secure Boot 機能そのものの説明については、『リファレンス』、第14章「UEFI (Unified Extensible Firmware Interface)」、14.1項「Secure Boot」をお読みください。

BIOS (Basic Input/Output System) やその後継である UEFI (Unified Extensible Firmware Interface) は PC システムで最も低い位置にあるソフトウエア／ファームウエアです。Linux の動作する他のハードウエアタイプ (POWER, IBM Z) にも、PC BIOS と似たような機能を持つ低レベルのファームウエアがあります。この文書内で "BIOS" と記述した場合、通常は BIOS と UEFI の両方を意味します。BIOS はシステムの設定を構成して一定の状態にするほか、低レベルなハードウエア機能も提供します。BIOS はその後、ホストを起動するための Linux ブートローダ (例: GRUB 2) を開始します。

ほとんどの BIOS 実装には、システムの不正な操作や不正な起動設定を防止するための機能が用意されています。一般には BIOS 管理者パスワードや起動パスワードなどと呼ばれています。管理者パスワードはシステムの設定を変更する際に求められるパスワード、起動パスワードは通常の起動を行う際に求められるパスワードです。ほとんどの場合、管理者パスワードのみを設定し、内蔵のハードディスクからのみ起動するよう設定すれば十分です、これにより、Linux のライブ CD や USB メモリなどを挿入もしくは接続して不正に起動するのを防ぐことができます。ただし、BIOS は高度なセキュリティを提供するものではなく、筐体を開くだけで容易に BIOS をリセットできたりパスワードを削除もしくは変更できたりしてしまうため、他の仕組みと組み合わせて使用する必要があります。

多くの BIOS ファームウエアには、それ以外のセキュリティ関連の設定が用意されています。システムの製造元や提供されているマニュアル、もしくは BIOS のメニューなどを参照して、どのような設定ができるのかをご確認ください。



重要: BIOS の起動パスワードを設定した場合の起動について

システムに対して起動パスワードを設定すると、そのシステムは無人では起動しなくなってしまいます (たとえばシステムの再起動や電源障害などの場合、自動では起動しなくなってしまいます。これは安全性と利便性のトレードオフとなります。

重要: BIOS 管理者パスワードを紛失した場合について

システムを初めて設定したような場合、BIOS の管理者パスワードはそれほど頻繁に入力する必要はありません。そのため、パスワードを忘れてしまわないように注意してください。パスワードを忘れてしまった場合、再度アクセスできるようにするには、ハードウエア側の操作を行って BIOS の設定を消去する必要があります。

10.3 ブートローダでのセキュリティ

openSUSE Leap の既定で使用される Linux のブートローダ GRUB 2 には、起動パスワードを設定する機能が用意されています。パスワード機能を利用することで、管理者のみが対話的な操作 (例: メニュー項目の編集やコマンドラインインターフェイスでの入力など) を行うことができるようにすることができます。またパスワードを指定した場合、 `C` および `E` を押して正しいパスワードを入力するまで、いかなる対話的な操作も禁止されるようになります。

詳しい設定については、GRUB 2 のマニュアルページをお読みください。

なお、パスワードを設定するにあたっては、設定したパスワードを忘れてしまわないように注意してください。また、パスワードを設定しても、侵入への時間を稼ぐことができるだけであって、完全に防げるわけではないことにも注意してください。たとえばリムーバブルデバイスから起動できる環境であれば、リムーバブルメディアを挿入してルートパーティションをマウントできてしまいます。このような場合は、BIOS レベルのセキュリティとブートローダのセキュリティの両方を使用することで、リムーバブルメディアからの起動を禁止して、かつ BIOS と起動の両方をパスワードで保護することができるようになります。

このほか、ブートローダの設定ファイルには適切なパーミッション `600` (`root` のみが読み書きできる) を設定して保護する必要もあります。保護を行わないと、パスワードやハッシュを読み取られてしまいます。

10.4 機密データを含む Linux サーバの廃棄

セキュリティポリシーには通常、廃棄予定のストレージメディアの取り扱いに関する規定も記述されています。ディスクやメディアの消去手順は、メディアの完全破壊としてもよく知られています。インターネット上にはさまざまなツールが提供されていますので、「ディスク 消去 ユーティリティ」などで検索を行うと、それらのツールに関する説明を読むことができます。機密データを含むサーバを廃棄する場合、ディスク内に書かれていたデータを修復できないようにすることが重要です。全てのデータの痕跡を消去するには、`scrub` などのユーティリティをお使いください。多くの消去ツールは、データ領域に上書きで何度も書き込むことによって消去を行います。これにより、高度な方法でデータにアクセスしようとしても、データを全く読み取れなくなります。ツールによっては単独で起動できる媒体が提供されているものや、アメリカ国防総省 (U.S. Department of Defense) 標準に準拠してデータを消去する機能を

備えているものもあります。なお、それぞれの政府機関では独自に消去標準を策定しています。標準によってはそれより強力なものもありますが、強力であればあるほど時間のかかるものであることに注意してください。

重要: ウエアレベリング機能を持つデバイスでの消去について

SSD などのデバイスにはウエアレベリングと呼ばれる仕組みがあり、データの上書きを行っても同じ場所に書き込まれるとは限らないものがあります。このようなデバイスの場合は、独自の機能を利用して消去する必要があります。

10.4.1 scrub: ディスクの上書きユーティリティ

`scrub` コマンドはハードディスクやファイルのほか、その他のデバイスに対しても利用可能なユーティリティで、データの復元を難しくするために繰り返しパターンを書き込むことで上書きを行います。このユーティリティは 3 種類のいずれかのモードで動作させることができます。モードはキャラクタ／ブロックデバイス、ファイル、ディレクトリのいずれかです。詳しくは `man 1 scrub` で表示されるマニュアルページをお読みください。

注記

ただし、`scrub` は SSD や NVMe ストレージ、もしくは CoW (Copy-on-Write; コピーオンライト) 型のファイルシステムに対しては、完全な上書きを保証するものではないことに注意してください。SSD や NVMe ストレージの場合、ドライブ内部に搭載されたウエアレベリングと呼ばれる仕組みにより、既存のデータを上書きしようとしても、別の場所に書き込まれてしまうことがあります。また ZFS や btrfs のような CoW 型のファイルシステムの場合、データの上書きは同じ場所への書き込みではなく、新しく確保した領域への書き込みになりますので、こちらの場合も上書きにはならなくなります。このように様々な抽象化の仕組みが存在していることから、ソフトウエアレベルでの上書きは、必ずしも既存のデータを消去できるとは限りません。

対応する消去方式

nnsa

国家核安全保障局 (NNSA) が NAP-14.1-C (XVI-8) として規定される方式で、リムーバブルメディアや非リムーバブルメディア (ハードディスク) に対応しています。全ての場所に対して擬似乱数パターンを 2 回、その後既知のパターンを書き込む 4 パス型の方式です: 乱数 (x2), 0x00, 書き込み検証

dod

DoD 5220.22-M section 8-306 procedure (d) として規定される方式で、リムーバブルメディアや非リムーバブルメディア (ハードディスク) に対応しています。書き込み可能な全ての範囲に対して、特定の文字、その反転パターン、ランダムな文字をそれぞれ書き込んで、最後に検証を行う 4 パス型の方式です。注意: scrub では書き込み検証を容易に実施できるよう、乱数のパスを最初に実行します: 乱数, 0x00, 0xff, 書き込み検証

bsi

German Center of Security in Information Technologies (https://www.bsi.bund.de) が推奨する 9 パス型の方式です: 0xff, 0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f

gutmann

Gutmann 氏の論文で示された方式で、合計 35 パスの手順を踏む方式です。

schneier

Bruce Schneier 氏の "Applied Cryptography" (1996) で示された 7 パス型の方式です: 0x00, 0xff, 乱数 (x5)

pfitzner7

Roy Pfitzner 氏による 7 乱数パス型の方式です: 乱数 (x7)

pfitzner33

Roy Pfitzner 氏による 33 乱数パス型の方式です: 乱数 (x33)

usarmy

アメリカ陸軍の AR380-19 方式です: 0x00, 0xff, 乱数 (注意: 磁気コアメモリを消去する場合は、DoD 522.22-M section 8-306 procedure (e) と等価になります)

fillzero

1 パス型のパターンです: 0x00

fillff

1 パス型のパターンです: 0xff

random

1 パス型のパターンです: 乱数 (x1)

random2

2 パス型のパターンです: 乱数 (x2)

old

バージョン 1.7 以前の scrub と同じ 6 パス型の方式です: 0x00, 0xff, 0xaa, 0x00, 0x55, 書き込み検証

fastold

5 パス型の方式です: 0x00, 0xff, 0xaa, 0x55, 書き込み検証

custom=(文字列)

1 パス型の独自パターンです。(文字列) には C 形式の数値エスケープ (¥nnn (8 進数) もしくは ¥xnn (16 進数)) を記述することができます。

10.5 リムーバブルメディアへのアクセス制限

環境によっては、USB ストレージや光学メディアなどのリムーバブルメディアへのアクセスを制限する必要があることがあります。このような場合は、`udisks2` パッケージを利用することで、設定を行うことができます。

1. まずはリムーバブルメディアをマウントしたり取り出したりすることができるユーザグループを作成します。下記の例では、`mmedia_all` というグループを作成しています:

```
> sudo groupadd mmedia_all
```

2. 次に `tux` を新しいグループに追加します:

```
> sudo usermod -a -G mmedia_all tux
```

3. `/etc/polkit-1/rules.d/10-mount.rules` ファイルを作成して、下記のような内容を記述します:

```
> cat /etc/polkit-1/rules.d/10-mount.rules
polkit.addRule(function(action, subject) {
 if (action.id =="org.freedesktop.udisks2.eject-media"
  && subject.isInGroup("mmedia_all")) {
   return polkit.Result.YES;
  }
});

polkit.addRule(function(action, subject) {
 if (action.id =="org.freedesktop.udisks2.filesystem-mount"
  && subject.isInGroup("mmedia_all")) {
   return polkit.Result.YES;
  }
});
```

重要: ルールファイルの命名について

ルールファイルのファイル名は必ず数字で始まっていなければなりません。数字で始まっていないファイル名の場合、そのファイルは無視されます。

ルールファイルはアルファベット順に処理されます。関数はいずれかの関数が値を返すまで、処理された順序で実行されます。そのため、特定のルールよりも前に処理させたい認可ルールがある場合は、そのルールファイルよりも前に並ぶファイル名を設定して /etc/polkit-1/rules.d 内に配置してください。たとえば `/etc/polkit-1/rules.d/10-mount.rules` のようになります。また、それぞれの関数は `polkit.Result` の値を返す必要があります。

4. `udisks2` を再起動します:

```
# systemctl restart udisks2
```

5. `polkit` を再起動します

```
# systemctl restart polkit
```

10.6 USBGuard による USB デバイスの認可とシステムの保護

USBGuard ソフトウエアフレームワークは、USB デバイスの認可を強制することによって、お使いのシステムを保護する機能を提供します。このフレームワークは、デバイスの属性情報をベースにした許可／拒否リストを使用します。

USBGuard には下記のような機能があります:

- 動作中の USBGuard デーモンとの対話を行うためのコマンドラインインターフェイス
- 動的な処理やポリシー強制を行うためのプロセス間通信 (IPC) 機能を持ったデーモンコンポーネント
- USB デバイスの認可ポリシーを記述するためのルール言語
- 共有ライブラリとして提供された、デーモンコンポーネントとの対話を行うための C++ API

10.6.1 USBGuard のインストール

USBGuard デーモンは、ポリシー内に規定されたルールをもとに、USB デバイスの使用可否を判断します。USBGuard をインストールして設定するには、下記のコマンドを使用してください:

1. USBGuard をインストールするには、下記の手順を実施します:

```
> sudo  zypper install usbguard
```

   これで USBGuard と必要な依存関係がインストールされます。USBGuard サービスとの対話を行うには、`usbguard-tools` も合わせてインストールしてください。

2. 現在接続されている USB デバイスを元にしてルールセットを生成したい場合は、`root` に切り替えて下記のコマンドを実行します:

```
# usbguard generate-policy > /etc/usbguard/rules.conf
```

   注記

   あとは必要に応じて `/etc/usbguard/rules.conf` ファイルを編集し、USBGuard のカスタマイズを行ってください。

3. USBGuard の開始やシステム起動時の自動開始を設定したい場合は、`root` に切り替えて下記のコマンドを実行します:

```
# systemctl enable --now usbguard.service
```

4. なお、USBGuard ではデバイスの許可か拒否をそれぞれ設定することができます。これは `usbguard-daemon.conf` ファイル内の `ImplicitPolicyTarget` オプションの値によっても影響を受けますが、これはポリシー内のどのルールにも該当しなかった場合の取り扱いを指定します。

```
usbguard allow-device 6
```

```
usbguard block-device 6
```

   システムへの接続を拒否してデバイスを取り外したい場合は、`reject-device` オプションを設定してください。



注記

なお、全てのオプションを確認したい場合は、`usbguard --help` コマンドを実行してください。

10.6.2 USBGuard の使用方法

デバイスの属性をベースにして allow (許可) または block (拒否) を設定することで、お使いのシステムに対する USB デバイスの接続ポリシーを設定することができます。

10.6.2.1 USBGuard の設定ファイル

USBGuard はコマンドラインオプションの解釈とデーモンへの設定適用が完了すると、`usbguard-daemon.conf` ファイルの読み込みを行います。このファイルは `/etc/usbguard/usbguard-daemon.conf` に配置します。このファイルには下記のようなオプションが含まれています:

オプション

`RuleFile=パス`

USBGuard デーモンは、このファイルをポリシールールセットの読み込み元として使用するほか、IPC (プロセス間通信) で指示された新しいルールの書き込み先としても使用します。既定値は `%sysconfdir%/usbguard/rules.conf` になります。

`ImplicitPolicyTarget= ターゲット`

ポリシー内のどのルールにも該当しないデバイスの取り扱い方法を指定します。たとえば下記のようになります:

- allow - デバイスの接続を許可します
- block - デバイスの接続を拒否します
- reject - システムからデバイスノードを論理的に削除します

`PresentDevicePolicy= ポリシー`

デーモンの起動時に既に接続されていたデバイスの取り扱い方法を指定します。

- allow - デバイスの接続を許可します
- block - デバイスの接続を拒否します

- reject - デバイスを削除します
- keep - 内部状態を同期させます
- apply-policy - 全ての既存デバイスに対して、ルールセットを適用します

`IPCAllowedUsers= ユーザ名`
: デーモンに対して IPC 経由でコマンドの送信を許可するユーザ名を、スペース区切りで指定します。

`IPCAllowedGroups= グループ名`
: デーモンに対して IPC 経由でコマンドの送信を許可するグループ名を、スペース区切りで指定します。

`IPCAccessControlFiles= パス`
: IPC のアクセス制御設定ファイルとしてデーモンが解釈すべきファイルパスを指定します。

例 10.1: 設定

```
IPCAllowedUsers=root joe
IPCAllowedGroups=wheel
```

たとえば上記のような例の場合、`root`，`joe` の各ユーザ、および `wheel` グループに属するユーザからの全ての IPC を許可します。

10.6.3 さらなる情報

USBGuard に関するさらに詳しい情報を参照したい場合は、それぞれ下記をご覧ください:

- 提供元のドキュメンテーション: https://usbguard.github.io/
- man usbguard
- man usbguard-rules.conf
- man usbguard-daemon
- man usbguard-daemon.conf

11 ソフトウエア管理

改訂履歴

2025-03-31

11.1 不要なソフトウエアパッケージ (RPM) の削除

Linux システムの安全性を高める際に重要なことは、Linux サーバの主な機能や役割を決めることです。機能や役割がはっきりしていないと、システムのどの場所を主に守るべきなのかがはっきりせず、作業の効率も落ちてしまいます。まずはインストールされているソフトウエアパッケージの一覧を確認して、不要なパッケージやセキュリティポリシーに合致しないパッケージを削除することが重要です。

一般的には RPM ソフトウエアパッケージには下記のものが含まれています:

- インストール時に RPM データベース内に書き込まれる、パッケージのメタデータ。
- 同梱されるファイルとディレクトリ。
- インストールやアンインストールの前後に動作するスクリプト。

また、下記のいずれかの条件に合致するパッケージについては、セキュリティ上のリスクとなる可能性があります:

1. setuid ビットもしくは setgid ビットの設定されたファイルを含むもの
2. グループもしくは全てのユーザが書き込み可能なファイルやディレクトリを含むもの
3. インストール時や既定で自動的にサービスが有効化されるもの。

上記 3 つの条件のいずれにも該当しなければ、パッケージは単なるファイル集であるとみなしてかまいません。このようなパッケージをインストールしてもアンインストールしても、システムのセキュリティには影響がありません。

しかしながら、お使いのシステムにインストールするパッケージはできる限り少なくしておくことをお勧めします。パッケージを減らすことで更新の手間を減らすことができますし、それによって発生しうる問題も減らすことができるからです。また本番サーバには開発用のパッケージやデスクトップ環境のソフトウエア (例: X サーバ) 等をインストールしないようにするのが最適です。もちろん Apache Web サーバや Samba ファイル共有サーバなど、サービスが不要であればインストールしないのが望ましいです。

重要: サードパーティ製のインストーラの要件について

Oracle 社や IBM 社等の提供するサードパーティ製のインストーラは、インストーラ自身を動作させるのにデスクトップ環境を求めるものがあります。本番サーバにデスクトップ環境をインストールしないようにするため、多くの組織では開発環境で無人インストール用のファイル (応答ファイル) を作成して回避しています。

それ以外にも、どうしても必要となる場合を除いて、FTP や Telnet デーモンもインストールしないことをお勧めします。これらは `ssh` , `scp` , `sftp` 等で代替可能です。

また、一般に Linux システムを構築するにあたっては、システムやアプリケーションで必要となるパッケージのほか、メンテナンスやトラブルシューティングで必要なパッケージ のみ をインストールするようにすべきです。まずは RPM の最小リストから出発して、必要なパッケージを追加していくとよいでしょう。

インストールされている全てのパッケージを一覧表示するには、下記のようなコマンドを入力して実行します:

```
# zypper packages -i
```

特定のパッケージに関する詳細を表示するには、下記のようなコマンドを入力して実行します:

```
# zypper info パッケージ名
```

パッケージを削除する際に発生しうる、矛盾関係や依存関係の問題を表示したい場合は、下記のようなコマンドを入力して実行します:

```
# zypper rm -D パッケージ名
```

このコマンドは便利な仕組みです。事前にテストを行っておかないと、ほとんどの場合、多くの問題点に直面し、複雑な依存関係の問題に悩まされる結果になります。

重要: 必須パッケージの削除について

パッケージを削除する場合は、システムで必須となるパッケージを削除してしまわないように注意してください。必須のパッケージを削除してしまうと、システムは全く起動できなくなってしまうばかりか、修復もできなくなってしまいます。不安な場合は、削除を行う前にシステムの完全バックアップを作成しておき、その後にパッケージを削除してください。

1 つまたは複数のパッケージを削除する場合は、`zypper` コマンドに「-u」オプションを追加することをお勧めします。これはパッケージを削除することによって不要となったパッケージを併せて削除するためのものです:

```
# zypper rm -u パッケージ名
```

11.2 Linux システムへの修正の適用

修正を管理するためのインフラストラクチャの構築は、Linux の本番環境を積極的に維持していくために重要な項目です。

なお、明示的なセキュリティポリシーを作成しておき、Linux のセキュリティ更新や問題に対する処理方針を定めておくことをお勧めします。たとえばセキュリティポリシーで修正の評価やテストの方法、そして修正の適用について時間枠を設定しておくなどのことが考えられます。また、ネットワーク関連のセキュリティ脆弱性の場合には、より高い優先度を設定し、より短い時間での解決を目指すなどの方針も考えられます。このほか評価フェーズではテスト環境内でのテストを行い、まずは開発環境から修正を適用するなどの方針をとるのもよいでしょう。

また、個別のセキュリティ記録を残しておき、受け取った Linux のセキュリティアナウンスと修正の検証状況や問題点の解決状況、そして修正の適用日時などを残しておくとよいでしょう。

SUSE では修正を 3 種類に分類しています。それぞれセキュリティ、推奨、オプションと呼んでいます。また、システムに対して修正を適用し、最新の状態に保つにあたってはいくつかの方式が提供されています。それぞれのシステムには同梱されている YaST ツールである YaST オンライン更新を利用して、SUSE Update の Web サイトに登録を行ってダウンロードすることができるようになっています。また、SUSE では リポジトリミラーリングツール (RMT) と呼ばれる、修正や更新などをローカルの環境にコピーし、インターネット側への通信量を減らすためのツールも用意されています。このほか SUSE では、SUSE Multi-Linux Manager と呼ばれる Linux システムのメンテナンス／修正適用／レポートシステムも用意されています。こちらは SUSE だけでなく、他のディストリビューションを管理することもできます。

11.2.1 YaST オンライン更新

サーバごとに重要な更新や改善を適用したい場合は、YaST オンライン更新ツールをお使いください。openSUSE Leap に対する更新は、修正を含む製品ごとの更新カタログを介して提供されています。修正の適用は YaST から［ソフトウエア］［オンライン更新］を選択してください。これを実行すると、オプション指定であるものを除き、未適用の全ての修正を自動的に選択します。後は［了解］ボタンを押すだけで、適用が行われます。

11.2.2 自動オンライン更新

YaST では自動更新にも対応しています。［ソフトウエア］›［自動オンライン更新］を選択してください。あとは毎日もしくは毎週を選択するだけです。カーネル更新など、ユーザ側での操作を必要とする修正の場合は、自動でインストールしようとしても止まってしまいますので、このような場合は［対話操作が必要な修正を飛ばす］を選択してください。

上記を選択した場合は、適宜手動でのオンライン更新を行い、対話操作が必要な修正を忘れずにインストールしておいてください。

また、［修正のダウンロードのみ］を選択すると、指定したタイミングで修正をダウンロードするものの、インストールは行われなくなります。ダウンロードした修正は、`rpm` や `zypper` を利用してインストールする必要があります。

12 ファイルの管理

改訂履歴

2024-05-10

12.1 ディスクのパーティション

サーバには一般に、`/` , `/boot` , `/var` , `/tmp` , `/home` などの個別のファイルシステムが設定されています。それぞれ別々のパーティション内に配置することで、`/var` や `/tmp` のディレクトリ内にあるログや一時ファイルなどが容量を大きく浪費して、ルートパーティションに書き込めなくなってしまうような事態を防ぐことができます。場合によっては、サードパーティ製のアプリケーションを配置する `/opt` ディレクトリを別のパーティションに配置する必要がある場合もあります。

このようにファイルシステムを分けることにより、もう 1 つの利点が生まれます。別々のマウントとして処理されることから、それぞれ別々のマウントオプションを設定できる、という点です。たとえば下記のようなマウントオプションがあります:

- `noexec` : プログラムの実行を禁止します。
- `nodev` : キャラクタデバイスやブロックデバイスを使用できないようにします。
- `nosuid` : `set-user-ID` や `set-group-ID` の効果を無効化します。
- `ro` : ファイルシステムを **読み込み専用** にします。

それぞれのオプションの設定は注意して行ってください。アプリケーションによっては動作しなくなってしまうことがあるほか、保守を受けられなくなってしまう場合もあります。マウントオプションは、適切な設定を行うことで、セキュリティに関わる攻撃を防いだり、設定の誤りによる予期しない動作を防いだりすることができます。たとえば `/tmp` に保存されるファイルには `set-user-ID` フラグの付いたプログラムを配置する必要は無いはずです。

また、システムの動作に合わせてパーティションを設計することも重要です。たとえば `/var/log` ディレクトリ内にログが大量に出力されるような環境の場合、`/` パーティションと `/var` パーティションを別々にしたほうがよいことになります。もう 1 つ知っておくべきこととしては、一般的な PC システムではプライマリパーティションが最大 4 つまでに制限されるため、LVM やその他のボリューム管理の仕組みを活用する必要があるかもしれません。

openSUSE Leap ではそのほかに、パーティションや単一のディレクトリを暗号化することができほか、ファイルをコンテナとして利用して暗号化することもできます。詳しくは 第13章「パーティションやファイルの暗号化」をお読みください。

12.2 特定のシステムファイルに対するパーミッションの変更

特に `/etc` ディレクトリ内に存在するファイルがそれに該当しますが、システム内の多数のファイルが読み取り可能な状態になっています。これは言い換えると、特権を持たないユーザでも自由に内容を読むことができる、ということを意味しています。一般的に、これは特に大きな問題とはなりませんが、システムの運用上の都合等の様々な理由で、機密情報を持つファイルを保護しておきたいという要件が存在することがあります。

openSUSE Leap では `permissions` パッケージで、ファイルのパーミッションを簡単に制御できるようになっています。このパッケージには 3 種類のシステムプロファイルが用意されています:

easy (簡易)
: ユーザにとって扱いやすいグラフィカルユーザシステムを構築するためのプロファイルです。こちらが既定のプロファイルです。

secure (厳格)
: グラフィカルユーザインターフェイスを持たない、サーバシステム向けのプロファイルです。

paranoid (偏執)
: 最大限にセキュリティを確保するためのプロファイルです。`secure` (厳格) プロファイルに加えて、setuid/setgid のようなフラグやケーパビリティフラグなどの特殊なパーミッションが 全て 削除されます。



警告: 特権を持たないユーザには使用できないシステムになる問題について

パスワードの変更などの簡単な作業を除き、特権を持たないユーザには使い物にならないシステムになってしまうことに注意してください。

そのため、`paranoid` (偏執) プロファイルはそのまま使用するのではなく、独自の許可設定を追加するためのテンプレートとしてのみお使いください。詳しい情報は `permissions.paranoid` ファイル内に英語にて記述があります。

独自のファイルアクセス許可を設定したい場合は、`/etc/permissions.local` ファイルを編集するか、もしくは `/etc/permissions.d/` ディレクトリ内に新しくファイルを作成してください。

```
# 追加の設定
/etc/security/access.conf       root:root       0400
/etc/sysctl.conf                root:root       0400
/root/                          root:root       0700
```

最初の列はファイル名を表しています。ただし、ディレクトリを表したい場合は、必ずスラッシュ (/) で終わらせてください。2 つめの列には所有者とグループを、3 つめの列にはモードを指定します。設定ファイルの書式についての詳細は、`man permissions` をお読みください。

あとは `/etc/sysconfig/security` ファイル内で、使用したいプロファイルを選択します。`easy` プロファイルと `/etc/permissions.local` ファイルの内容を適用したい場合は、下記のように指定します:

```
PERMISSION_SECURITY="easy local"
```

設定を適用するには、`chkstat --system --set` コマンドを実行します。

パーミッションは `zypper` によるパッケージ更新でも適用されます。そのため、`cron` もしくは `systemd` タイマーを利用して、定期的に `chkstat` を実行するようにしてください。

重要: 独自のファイルパーミッションのサポートについて

システムが提供するプロファイルは細部に至るまでよくテストされて構築されていますが、独自のファイルパーミッションを指定してしまうと、アプリケーションによっては正しく動作しなくなってしまう場合があります。SUSE ではそのような構成でご利用の場合、サポートを提供できませんのであらかじめご了承ください。

また、`chkstat` で独自のファイルパーミッションを適用するにあたっては、事前に必ずテストを実施して、普段の作業が想定通りに動作することを確認しておいてください。

12.3 ホームディレクトリに対する 755 から 700 へのパーミッション変更

既定では、ユーザのホームディレクトリはシステム内の誰にでもアクセス (読み込み、実行) 可能な状態になります。これは潜在的な情報漏洩につながることから、所有者自身だけがホームディレクトリにアクセスできるようにする必要があります。

下記のようなコマンドを入力して実行すると、`/home` ディレクトリ以下にある既存のホームディレクトリに対して、`700` というパーミッション (所有者のみがアクセスできる) を設定します:

```
> sudo chmod 755 /home
> sudo bash -c 'for dir in /home/*; do \
echo "Changing permissions of directory $dir"; chmod 700 "$dir"; done'
```

なお、このパーミッションを新しく作成するホームディレクトリに対して適用したい場合は、`/etc/login.defs` ファイルを編集して、`HOME_MODE` を `700` に設定してください。

```
# HOME_MODE is used by useradd(8) and newusers(8) to set the mode for new
# home directories.
# If HOME_MODE is not set, the value of UMASK is used to create the mode.
HOME_MODE      0700
```

`HOME_MODE` を指定しない場合、パーミッションは既定の umask 値から自動的に計算されます。なお、`HOME_MODE` は設定するパーミッションを指定するものであって、umask のように与えたくないフラグを指定するものではないことに注意してください。umask に関する詳細は、12.4項「既定の umask」をお読みください。

この設定変更をテストするには、`useradd -m testuser` のようにして新しいユーザを作成してみてください。あとは `ls -l /home` を実行して、作成されたディレクトリのパーミッションを確認してください。テストが終わったらユーザは削除してかまいません。

重要: パーミッション変更のテスト

ここまでの手順を実行することで、一般ユーザは他のユーザのホームディレクトリにアクセスできなくなります。ユーザやソフトウエアによっては、これが予期しない結果となる可能性もあります。

そのため、これを本番環境で適用する場合は、あらかじめテストをしておく必要があるほか、ユーザに対しても変更を周知しておく必要があります。

12.4 既定の umask

`umask` (ユーザがファイルを作成する際のモードマスク値) コマンドはシェルの内蔵コマンドで、新しく作成するファイルやディレクトリに対して、そのパーミッションを制御するためのものです。この値はシステムコールを利用することで変更することができますが、多数のプログラムやユーティリティでは `umask` コマンドを呼び出しています。

既定では、`umask` の値は `022` に設定されています。この umask の値は「許可を与えたくない」ビットを指定する仕組みです。

現在の umask 値を表示するには、`umask` コマンドに何もパラメータを付けずに実行します:

```
> umask
022
```

既定の umask 値はほとんどのユーザにとって十分な値になっています。たとえば下記のようになります。

```
> touch a
> mkdir b
```

```
> ls -on
合計 16
-rw-r--r--. 1 17086    0 11月29 15:05 a
drwxr-xr-x. 2 17086 4096 11月29 15:05 b
```

必要であれば umask 値を変更することもできます。

```
> umask 111
> touch c
> mkdir d
> ls -on
合計 16
-rw-rw-rw-. 1 17086    0 11月29 15:05 c
drw-rw-rw-. 2 17086 4096 11月29 15:05 d
```

不用意なデータ漏洩を防ぐため、さらに厳しい制限を行いたい場合は、`037` のような値を設定することもできます。

```
> umask 037
> touch e
> mkdir f
> ls -on
合計 16
-rw-r-----. 1 17086    0 11月29 15:06 e
drwxr-----. 2 17086 4096 11月29 15:06 f
```



ヒント: 最大限の保護について

最大限の保護を設定したい場合は、`077` という umask 値を設定してください。これにより、新しく作成されたファイルやディレクトリは、自分自身しかアクセスできなくなり、グループ内に対してもその他のユーザに対しても、全てのアクセスが禁止されるようになります。

ただし、このような設定を行うことで、他のユーザに不便を強いる可能性があったり、ソフトウエアが予期しない動作をしてしまったりすることがあるかもしれません。また、サポートチームに余計な負担をかけてしまう可能性にも注意してください。

12.4.1 既定の umask 値の調整

全てのユーザに対する umask 値を調整したい場合は、`/etc/login.defs` ファイル内にある `UMASK` の値を変更してください。

```
# Default initial "umask" value used by login(1) on non-PAM enabled systems.
# Default "umask" value for pam_umask(8) on PAM enabled systems.
# UMASK is also used by useradd(8) and newusers(8) to set the mode for new
# home directories.
```

```
# 022 is the default value, but 027, or even 077, could be considered
# for increased privacy. There is no One True Answer here: each sysadmin
# must make up their mind.
UMASK           022
```

個別のユーザに対して設定したい場合は、`/etc/passwd` ファイル内の 'gecos' フィールドに umask 値を設定します。たとえば下記のようになります:

```
tux:x:1000:100:Tux Linux,UMASK=022:/home/tux:/bin/bash
```

上記の設定は `yast users` でも行うことができます。ユーザの［詳細］›［追加のユーザ情報］内に、`UMASK=022` のような形式で設定してください。

なお、`/etc/login.defs` や `/etc/passwd` に設定した値は、`pam_umask.so` という PAM モジュールが読み込んで適用します。その他の設定オプションについての詳細は、`man pam_umask` で表示されるマニュアルページをお読みください。

また、設定値を反映させるためには、ユーザがログインし直す必要もあります。ログインし直したあと、`umask` コマンドで正しく設定されたかどうかを確認してください。

12.5 SUID/SGID フラグの付いたファイル

SUID (set user ID) や SGID (set group ID) のビットを実行可能なファイルに設定すると、そのプログラムを起動したユーザではなく、その実行可能なファイルを所有しているユーザもしくはグループとしてプログラムを起動することができます。たとえば SUID ビットが設定され、`root` が所有するプログラムがあった場合、誰がそのプログラムを起動しても、プログラムは `root` の UID で実行されることになります。よくある例は `passwd` コマンドで、`root` が所有する `/etc/shadow` ファイルのパスワード欄に書き込みを行うために設定されています。

SUID/SGID のビットを誤って他のプログラムに設定したりしてしまうと、それはセキュリティホールとなってしまいます。そのため、システム全体を調べて、予期せず SUID/SGID のビットが設定されているものが無いかどうかを調べる必要があります。具体的には、下記のようなコマンドを入力して実行します:

```
# find /bin /boot /etc /home /lib /lib64 /opt /root /sbin \
  /srv /tmp /usr /var -type f -perm '/6000' -ls
```

ファイルシステムの構造によっては、上記にさらにディレクトリを追加する必要があるかもしれません。

SUSE では、どうしても必要なプログラムにのみ SUID/SGID ビットを設定しています。また、コードの開発者に対しても、どうしても必要な理由がある場合を除いて、SUID/SGID のビットを設定しないように求めています。また、全てのユーザからは実行できないようにすることで、SUID/SGID ビットの悪影響を防ぐこともできます。もちろんソフトウエア側の設計改善やケーパビリティの使用などで、SUID/SGID ビットの使用を回避できるほうが望ましいです。

openSUSE Leap ではファイルのケーパビリティ機能に対応し、`root` に与えられる権限の一部のみをプログラムに許可することもできます:

```
# getcap -v /usr/bin/ping
/usr/bin/ping = cap_new_raw+eip
```

上記の例では、`ping` を実行するユーザに対して `CAP_NET_RAW` のケーパビリティを許可しています。この場合、たとえ `ping` に脆弱性が存在していても、攻撃者は CAP_NET_RAW のケーパビリティのみが許可され、`root` の全権限を得ることはありません。また可能であれば、SUID ビットとともにファイルのケーパビリティを設定しておくことをお勧めします。ただし、これは `root` が所有者の SUID プログラムにのみ当てはまるもので、それ以外の `news`, `lp` などのシステムユーザが所有するプログラムには当てはまりません。

12.6 全てのユーザが書き込むことのできるファイル

誰にでも書き込み可能なファイルは、システム内の任意のユーザが書き込めるファイルであることから、セキュリティリスクとなりうるものです。これに加えて、誰にでも書き込めるディレクトリは、誰にでもファイルの作成や削除ができてしまいます。これらのファイルやディレクトリを検索するには、下記のようなコマンドを入力して実行します:

```
# find /bin /boot /etc /home /lib /lib64 /opt /root /sbin \
  /srv /tmp /usr /var -type f -perm -2 ! -type l -ls
```

ファイルシステムの構造によっては、上記にさらにディレクトリを追加する必要があるかもしれません。

なお、上記では `! -type l` パラメータを指定していますが、これはシンボリックリンクを読み飛ばすための設定で、シンボリックリンクは通常全てのユーザが書き込めるファイルになっているためです。シンボリックリンクが指し示すファイルやディレクトリへの書き込みは、その示した先のファイルやディレクトリのパーミッションに従って処理されますし、そのファイルやディレクトリは上記のコマンドでチェック対象となるので、特に問題はありません。

また、`/tmp` ディレクトリなどのようにスティッキー (sticky) ビットが設定されたディレクトリに対しては、全てのユーザへの書き込みが許可されていても、そのファイルやディレクトリの所有者のみが削除もしくは名前変更を行うことができます。スティッキービットは、そのファイルやディレクトリを作成したユーザに "ベタベタ貼り付く" (sticky) ことで、他のユーザからの削除や名前変更を防ぐ意味を持ちます。そのため、ディレクトリの用途によっては、スティッキービットが設定されていれば、全てのユーザからの書き込みを許しても問題ない場合があります。

```
> ls -ld /tmp
drwxrwxrwt 18 root root 16384 Dec 23 22:20 /tmp
```

上記の `t` 表示がスティッキービットを表しています。

12.7 孤立したファイル／所有者のいないファイル

どのユーザやグループにも所有されていないファイルは、現時点ではセキュリティ上の問題となることはありません。ですが、このようなファイルは将来的にセキュリティ上の問題となる場合があります。たとえば新しいユーザを作成した際、偶然にもその UID を持つファイル (つまり、以前はどのユーザも所有していなかったファイル) があると、そのファイルに対する所有権を得てしまうことになります。

所有者や所有グループのいないファイルを検索するには、下記のコマンドを入力して実行します:

```
# find /bin /boot /etc /home /lib /lib64 /opt /root /sbin /srv /tmp /usr /var -nouser -o -
nogroup
```

ファイルシステムの構造によっては、上記にさらにディレクトリを追加する必要があるかもしれません。

もう 1 つ発生しうる問題としては、パッケージシステム経由でインストールしていないプログラムがある場合、そのプログラムに対しては自動では更新できないという問題があります。このようなファイルを検索するには、下記のようなコマンドを入力して実行します:

```
> find /bin /lib /lib64 /usr -path /usr/local -prune -o -type f -a -exec /bin/sh -c "rpm -
qf {} &> /dev/null || echo {}" \;
```

なお、上記のコマンドは権限を持たないユーザ (例: nobody) で実行してください。これは、ファイル名をうまく加工することで、コマンドそのものを実行してしまうことがあり得るためです。通常、これらのディレクトリは `root` のみに書き込み権限がありますので、特に問題となることはありませんが、念のため注意してください。

上記のコマンドは、`/bin`，`/lib`，`/lib64`，`/usr` の各ディレクトリ内にある全てのファイル (ただし `/usr/local` を除く) を列挙し、パッケージマネージャで管理されていないファイルを出力するものです。これらのファイルはすぐにセキュリティ問題になるとは言い切れませんが、どのファイルが追跡対象外なのかを把握しておき、あらかじめ注意しておくことが重要です。

13 パーティションやファイルの暗号化

改訂履歴

2024-05-10

ファイルやパーティション、ディスク全体の暗号化を行うことで、お使いのデータに対する不正なアクセスを保護し、ファイルや文書に対する機密を保護することができます。

暗号化を行うにあたっては、下記のいずれかの方法をとることができます:

ハードディスク内のパーティションの暗号化

YaST を利用することで、インストール中でもインストール後でも暗号化されたパーティションを作成することができます。詳しくは 13.1.1項「インストール時における暗号化パーティションの作成」および 13.1.2項「動作中のシステムにおける暗号化パーティションの作成」をお読みください。この方式は外付けのハードディスクなど、リムーバブルメディアに対しても適用することができます。こちらに関する詳細は 13.1.3項「リムーバブルメディアの内容の暗号化」をお読みください。

GPG を利用したファイル単独での暗号化

1 つまたは複数のファイルを単純に暗号化したいだけであれば、GPG ツールをお使いになるとよいでしょう。詳しくは 13.2項「GPG を利用したファイルの暗号化」をお読みください。

Rage を利用したファイル単独での暗号化

1 つまたは複数のファイルを単純に暗号化したいだけであれば、Rage 暗号化ツールをお使いいただくこともできます。詳しくは 13.3項「Rage によるファイルの暗号化」をお読みください。



警告: 暗号化による保護範囲について

本章で説明している暗号化は、動作中のシステムでの保護を行う仕組みではありません。たとえ暗号化されているボリュームであっても、マウントが成功してしまえば、アクセス許可での制限がない限り、アクセスができてしまいます。本章での暗号化は、コンピュータをどこかに置き忘れてしまった場合や、盗難にあった場合に役に立つ仕組みであり、このような場合に機密データが読み取られることのないように保護を実現するものです。

13.1 YaST を利用した暗号化ファイルシステムの設定

YaST を使用することで、インストール中であってもインストールが終わった後からであっても、暗号化されたパーティションや仮想ハードディスクを作成することができます。しかしながら、インストール済みのシステムに暗号化されたパーティションを追加したい場合、既存のパーティションのサイズを変更したり

する作業が必要になることがありますので、比較的難しい作業になります。このような場合は、必要なサイズで暗号化されたファイルを作成して、ファイルやディレクトリなどをその中に書き込むように設定したほうが便利な場合があります。それに対して、パーティション全体を暗号化する場合は、パーティションレイアウト内に専用のパーティションを作成して対応する必要があります。また、YaST で表示される標準のパーティション提案では、既定では暗号化されたパーティションが含まれません。インストール時に暗号化されたパーティションを設定したい場合は、パーティションダイアログにて手作業で設定をしてください。

13.1.1 インストール時における暗号化パーティションの作成



警告: パスワードの入力について

暗号化されたパーティションを作成する際は、パスワードを忘れることがないようにしてください。パスワードを忘れてしまうと、暗号化されたデータにアクセスできなくなってしまいます。

YaST の熟練者向けパーティション設定を使用することで、暗号化パーティションを作成することができます。新しい暗号化パーティションを作成するには、下記の手順で行います:

1. [システム] › [パーティション設定] を選択して、YaST 熟練者向けパーティション設定を起動します。
2. ハードディスクを選択して [追加] を押します。プライマリもしくは論理パーティション のいずれかを選択します。
3. パーティションに対して設定するサイズ、もしくは範囲を指定します。
4. パーティションに対して設定するファイルシステムと、マウントポイントを選択します。
5. このとき、[デバイスの暗号化] にチェックを入れておきます。

   #### 注記: 追加のソフトウエアが必要となる件について

   [デバイスの暗号化] にチェックを入れると、追加のソフトウエアをインストールするよう促すポップアップウインドウが表示されることがあります。暗号化パーティションを動作させるには、これら全てのパッケージをインストールする必要があります。

6. 暗号化されたパーティションを自動ではマウントしたくない場合は、[マウントのオプション] で [デバイスをマウントしない] を選択します。自動的にマウントする場合は、[デバイスのマウント] を選択してマウントポイントを入力または選択してください。

7. ［次へ］を押して、このパーティションを暗号化するためのパスワードを入力します。入力されたパスワードは表示されません。入力ミスを防止するため、パスワードは 2 回入力してください。

8. あとは［完了］を押して処理を完了します。暗号化パーティションが新しく作成されます。

`/etc/fstab` 内で暗号化されているパーティションを自動的にマウントするように設定している場合、システムの起動時に暗号化パーティションに対するパスワード入力を求めるようになります。パスワードの入力を行うことでマウント処理が実行され、全てのユーザからアクセスできるようになります。

システムの起動時に暗号化されたパーティションをマウントしたくない場合は、パスワード入力の際に `Enter` を押してください。再度パスワードの入力を求められた場合も、同様に入力を拒否してください。これにより、暗号化されたパーティションがマウントされなくなりますが、オペレーティングシステムの起動処理はそのまま続けられるようになります。この場合、暗号化されたパーティション内のデータにはアクセスができなくなります。

システムの起動時にマウントしなかった暗号化パーティションをマウントしたい場合は、ファイルマネージャを開いて［他の場所］を選び、表示された一覧の中から暗号化されたパーティションを選択してください。これにより、パスワードの入力が求められますので、パスワードを入力してください。

既にパーティションが存在するマシンに対してシステムをインストールしている場合は、インストール時に既存のパーティションを暗号化するように設定することもできます。この場合は、13.1.2項「動作中のシステムにおける暗号化パーティションの作成」の手順に従って作業を行ってください。ただし、既存のパーティション内にある全てのデータが削除されることに注意してください。

13.1.2 動作中のシステムにおける暗号化パーティションの作成



警告: 動作中のシステムにおける暗号化の有効化について

動作中のシステムで暗号化されたパーティションを作成することもできます。ただし、既存のパーティションを暗号化すると、その中のデータは全て削除されてしまうほか、場合によっては既存のパーティションのサイズを変更したり、再構築したりする作業が必要になってしまうこともあります。

動作中のシステムで暗号化されたパーティションを作成するには、YaST コントロールセンターから［システム］›［パーティション設定］を選び、表示された警告メッセージには［はい］を選びます。あとは［熟練者向けパーティション設定］で暗号化したいパーティションを選び、［編集］を押します。残りの作業は13.1.1項「インストール時における暗号化パーティションの作成」で示している手順と同じです。

13.1.3 リムーバブルメディアの内容の暗号化

YaST では、リムーバブルメディア (外付けハードディスクや USB フラッシュメモリなど) についても、通常のストレージデバイスと同じように扱います。これまでに説明してきたものと同じ手順で、リムーバブルメディア内に仮想ディスクやパーティションを作成することができます。ただし、システム起動時の自動マウントについては、無効化するようにしてください。これは一般に、リムーバブルメディアはシステムの起動時には接続されていないことがあるためです。

YaST でリムーバブルデバイスを暗号化した場合、GNOME デスクトップでは暗号化されたパーティションを自動的に認識することができますので、デバイスへのアクセス時にパスワード入力を求めるようになっています。また、GNOME の動作時に FAT 形式でフォーマットされたリムーバブルデバイスを接続すると、デスクトップを利用しているユーザがデバイスの所有者になります。FAT 以外のファイルシステムを使用しているデバイスの場合は、デバイスに対して読み書きができるようにするため、`root` でパーミッションを変更するなどして、所有権を明示的に設定してください。

13.2 GPG を利用したファイルの暗号化

GNU Privacy Guard (GPG) 暗号化ソフトウエアを使用することで、ファイルや文書を個別に暗号化することができます。

GPG でファイルを暗号化するには、まず鍵対を生成する必要があります。鍵対を生成するには、`gpg --gen-key` を実行して、表示される手順に従って作業を行ってください。鍵対を作成する際、GPG はユーザ ID (UID) を作成して鍵対を識別するようになります。ユーザ ID には本名やコメント、電子メールアドレスを含めることができます。この UID (もしくは名前の一部や電子メールアドレス) は、ファイルを暗号化する際に指定します。既存の鍵の一覧を取得するには、`gpg --list-keys` コマンドを実行します。鍵対を生成したあと、ファイルを暗号化するには、下記のコマンドを実行します:

```
> gpg -e -a --cipher-algo AES256 -r UID ファイル名
```

ここで、`UID` には UID の一部 (たとえば名前など) を、`ファイル名` には暗号化したいファイルを指定します。たとえば下記のようになります:

```
> gpg -e -a --cipher-algo AES256 -r Tux secret.txt
```

このコマンドは、指定したファイルのファイル名に `.asc` という新しい拡張子を追加して、暗号化されたファイルを作成します (今回の例では、`secret.txt.asc` というファイルが作成されます) 。

`-a` は ASCII テキスト形式でファイルを出力するための指定です。これにより、ファイルをコピー&ペーストで貼り付けることができるようになります。`-a` を省略した場合はバイナリ形式のファイルが作成され、ファイル名も `secret.txt.gpg` のような名前になります。

暗号化されたファイルを復号するには、下記のコマンドを実行します:

```
> gpg -d -o 復号化したファイル 暗号化されたファイル
```

ここで、復号化したファイル には復号後のファイルのファイル名 (出力先) を、暗号化されたファイル には暗号化されたファイルのファイル名を指定します。

ただし、この手順でファイルの暗号化を行った場合、暗号化で使用したものと同じ鍵を使用しないと、復号できないことに注意してください。他のユーザに対して暗号化したファイルを送りたい場合は、その相手の公開鍵を取得して、この公開鍵で暗号化を行わなければなりません。

13.3 Rage によるファイルの暗号化

Rage はファイルを暗号化するための暗号化ソフトウエアです。他のユーザとの間で容易に鍵を交換できる仕組みのほか、不用意な操作ミスや機密データの漏洩の防止機能を備えています。ファイルの暗号化を行う場合は、Rage の使用をお勧めします。

Rage のインストールは下記の手順で行います:

```
> sudo zypper install rage-encryption
```

暗号化したファイルを受信する側のユーザは、まず Rage で鍵対を生成しなければなりません:

```
> rage-keygen -o ~/rage.key 2 ~/rage.pub
```

上記で 2 つのファイルが作成されます; 上記の例では、それぞれ rage.pub と rage.key というファイル名になっています。

`rage.pub` の例

```
> cat file.pub
    Public key: age17e4g67cs07jk3lmylyq6gduv26uf7tz7nm9jrsaxn8xxx9uc9amsdg4a5e
```

`rage.key` の例

```
>  cat file.key
    # created: 2023-05-30T16:29:20+05:30
    # public key: age17e4g67cs07jk3lmylyq6gduv26uf7tz7nm9jrsaxn8xxx9uc9amsdg4a5e
```

重要

file.key は機密鍵であるため、他者に開示してはなりません。

暗号化

ファイルを暗号化するには、コマンドラインで公開鍵を指定して実行します:

```
> rage -e -r 公開鍵 -o 暗号化されたファイル 元の_(暗号化前の)_ファイル
```

たとえば下記のようになります:

```
> rage -e -r age17e4g67cs07jk3lmylyq6gduv26uf7tz7nm9jrsaxn8xxx9uc9amsdg4a5e -o
 test.txt.age test.txt
```

復号化

暗号化したファイルは、公開鍵に対応する機密鍵を持つユーザのみが復号化できます。つまり、誰かと暗号化したファイルをやりとりしたい場合は、相手の公開鍵を指定して暗号化する必要があります。

```
> rage -d -i ~/rage.key -o 復号化したファイル 暗号化されたファイル ファイル
```

たとえば下記のようになります:

```
> rage -d -i ~/rage.key -o test.txt.decrypted test.txt.age
```

パスフレーズ

`-p` や `--passphrase` のパラメータを指定することで、暗号化したファイルにさらにパスフレーズを設定することもできます。既定では Rage は独自にパスフレーズを設定しますが、パスフレーズを指定しておくこともできます。

```
> rage -e -p -o 暗号化されたファイル ファイル
```

たとえば下記のようになります:

```
> rage -e -p -o test.txt.age test.txt
```

SSH

Rage では、暗号化を行う際、Rage の鍵対ではなく SSH (Secure Socket Shell) の鍵対を使用することもできます。Rage ではそれぞれ ssh-rsa と ssh-ed25519 の公開鍵に対応し、復号化もそれに対応する機密鍵で実施することができます。なお、ssh-agent と ssh-sk(FIDO) には対応していません。

```
> rage -e -p -o 暗号化されたファイル ファイル
```

たとえば下記のようになります:

```
> rage -e -p -o test.txt.age test.txt
```

たとえば下記のようになります:

```
>  ssh-keygen -t ed25519
```

暗号化するには、下記のようなコマンドを実行します:

```
>  rage -e -a -R 公開鍵ファイル -o 暗号化されたファイル ファイル
```

たとえば下記のようになります:

```
>  rage -e -a -R id_ed25519.pub -o test.txt.age test.txt
```

復号化するには、下記のようなコマンドを実行します:

```
>  rage -d -i SSH_機密鍵ファイル -o 復号化したファイル暗号化されたファイル
```

たとえば下記のようになります:

```
>  rage -d -i id_ed25519 -o test.txt.decrypted test.txt.age
```



重要

なお、鍵とファイルはそれぞれパスで指定しなければなりません。

複数の宛先への対応

Rage は同時に複数の宛先で復号化できる暗号ファイルを作成することもできます。この場合、指定した複数の公開鍵のうち、いずれかの機密鍵を利用して復号化することができます。

```
rage -e -a -R 最初の_SSH_公開鍵-r 最初の_Rage_公開鍵... -o 暗号化されたファイル ファイル
```

たとえば下記のようになります:

```
rage -e -a -R id_ed25519.pub -r
 age1h8equ4vs5pyp8ykw0z8m9n8m3psy6swme52ztth0v66frgu65ussm8gq0t -o -r
 age1y2lc7x59jcqvrpf3ppmnj3f93ytaegfkdnl5vrdyv83l8ekcae4sexgwkg test.txt.age test.txt
```



ヒント

Rage のコマンドラインパラメータの一覧を取得したい場合は、`-h` または `--help` を指定して実行してください。

13.3.1 その他の情報

- https://github.com/str4d/rage Rage 暗号化の GitHub リポジトリ
- https://github.com/C2SP/C2SP/blob/main/age.md Age 暗号化の GitHub リポジトリ

14 cryptctl を利用したアプリケーション向けのストレージ暗号化

改訂履歴

2024-05-10

データベースなどのアプリケーションは、サードパーティ製の製品として提供される専用のサーバの形態で提供されることがあります。そのような環境を使用する場合、メーカー側のスタッフと連携してデータセンターのメンテナンスタスクを実行する必要があります。また、プライバシーの要件を満たすためにディスクの暗号化を行う必要があるかもしれません。

`cryptctl` は LUKS を利用して機密情報を含むディレクトリを暗号化するほか、下記のような追加機能も提供します:

- 暗号鍵を中央のサーバに配置することができます。通常は中央のサーバは顧客側の施設に設置します。
- 不用意な再起動が発生したような場合でも、暗号化されたパーティションを自動的に再マウントすることができます。

`cryptctl` は下記の 2 つのコンポーネントから構成されています:

- 1 つもしくは複数の暗号化されたパーティションをマウントするクライアントマシン。この場合、パーティションの暗号を解読するのに必要な鍵を恒久的な場所 (別のディスクなど) に保存しません。クライアントは、クラウドなどのホスティング環境が該当します。
- クライアント側からの要求に応じて、パーティションを解読するための暗号鍵を提供するサーバ。`cryptctl` サーバを設定することで、KMIP (Key Management Interoperability Protocol; 鍵管理相互運用プロトコル) 1.3 互換のサーバを構築することもできます。この場合、`cryptctl` サーバにはクライアント向けの暗号鍵は保存されず、その代わり KMIP 互換のサーバ内に保存することになります。



警告: `cryptctl` サーバのメンテナンスについて

`cryptctl` サーバは暗号化されたディスクに対するタイムアウトを管理するほか、必要であれば暗号鍵そのものを保持することもできます。暗号鍵そのものを保持させる場合は、不正にアクセスされることがなく、信頼できるユーザからのみアクセスできるように設定する必要があります。

これに加えて、サーバのバックアップは定期的に採取する必要があります。サーバのデータが失われてしまうと、クライアントがマウントする暗号化パーティションにもアクセスができなくなってしまうためです。

暗号化を処理するため、`cryptctl` は aes-xts-256 と 512 ビット鍵による LUKS を使用しています。暗号鍵は証明書の検証機能のある TLS を利用して転送されます。

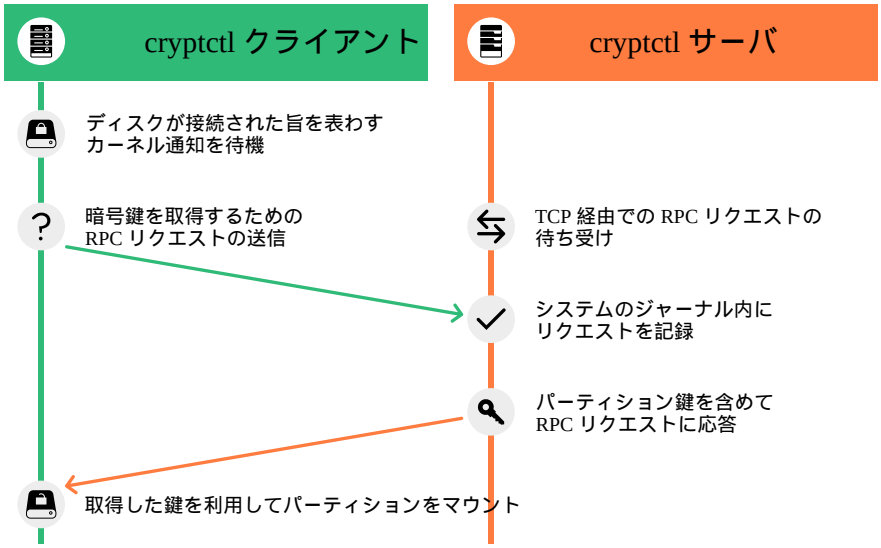


図 14.1: `cryptctl` による鍵の取得 (KMIP サーバへの接続を行わない場合の例)

注記: `cryptctl` のインストール

下記の手順を実行する前に、まずはサーバやクライアントとして動作させる全てのマシン内に `cryptctl` パッケージがインストールされていることをご確認ください。

14.1 `cryptctl` サーバの設定

`cryptctl` のクライアントを構築するよりも前に、`cryptctl` サーバを設定する必要があります。

まずはサーバとクライアントとの間で機密を保持するために使用する証明書を作成します。自己署名型の証明書であれば、一般的な手順で証明書を作成してください。公的な証明書を使用する場合は、サーバ向けの証明書を作成して証明機関に提出し、署名を受けてください。

クライアント側の証明書も同様に、公的な証明機関で署名してもらうことができます。このような追加のセキュリティ機能を使用する場合は、あらかじめ証明機関の証明書を手元に用意してください。

1. `root` で下記を実行します:

```
# cryptctl init-server
```

2. それぞれ表示された質問に対して回答を入力し、`Enter` を押していきます。このとき、既定値は質問の末尾に角括弧で括られて表示されます。

a. まずは強力なパスワードを生成し、そのパスワードが漏洩することの無いよう十分な保護を設定してください。このパスワードはサーバ内に登録されている全てのパーティションの解読を行うためのパスワードになります。

b. 次に PEM エンコードされた TLS 証明書、もしくは TLS 証明書のチェインのパスを指定します。自己署名型の証明書を使用する場合は、何も指定しないでください。また、パスを指定する場合は、絶対パスで指定してください。

c. 既定で表示されるホスト名以外の名前を使用したい場合は、ここでホスト名を指定してください。`cryptctl` では、指定したホスト名を含む証明書を生成します。

d. クライアント側からの解読要求を受け付けるネットワークインターフェイスの IP アドレスを指定し、続けてポート番号を指定します (既定のポートは 3737 です) 。
IP アドレスの既定値である `0.0.0.0` を使用すると、`cryptctl` は IPv4 を利用する全てのネットワークインターフェイスを介して、クライアントからの要求を受け付けるようになります。

e. クライアントに対する暗号鍵を保存するための、サーバ内のディレクトリを指定します。

f. 次にクライアントとの通信を行う際、クライアントに対して認証を要求するかどうかを指定します。ここで［No］(いいえ) を選択すると、クライアント側からはディスクの UUID のみを認証として使用します (なお、いずれを選択した場合でも、サーバの証明書を利用して通信が暗号化されます) 。
［Yes］(はい) を選択した場合は、クライアントの証明書に署名する証明機関を PEM 形式で指定してください。

g. 次に KMIP 1.3 互換のサーバ (1 つまたは複数) を使用するかどうかを選択します。
KMIP 1.3 互換のサーバを使用すると、クライアントの暗号鍵をそこに保存することができるようになります。こちらを選択した場合は、1 つまたは複数の KMIP 互換サーバのホスト名およびポートを指定してください。
このほか、KMIP サーバに対して使用するユーザ名やパスワード、証明機関の証明書などを設定します。また、`cryptctl` のサーバで使用するクライアントの識別証明書を設定することもできます。

重要: KMIP の再設定の難しさについて

KMIP サーバを使用する場合、後からの再設定は簡単ではありません。この設定を変更するには、`cryptctl` サーバだけでなく、クライアントについても設定を修正する必要があるためです。

h. 最後に電子メールによる暗号化／解読要求の通知を行うため、SMTP サーバを設定します。電子メールによる通知機能は、設定しなくてもかまいません。



注記: パスワードで保護されたサーバについて

`cryptctl` では、パスワードによる認証が必要な SMTP サーバを指定して、電子メールを送信することはできません。認証が必要な場合は、ローカルの SMTP プロキシを設定してください。

i. `cryptctl` サーバの起動を行うかどうかを尋ねられた場合は、`y` (はい) と入力してください。

3. 設定が終わったら、`cryptctl-server` サービスの状態を確認します:

```
# systemctl status cryptctl-server
```

サーバを後から再設定する場合は、下記のいずれかを行ってください:

- `cryptctl init-server` を再度実行します。`cryptctl` は既存の設定値を既定値として表示しますので、必要な箇所のみを変更することになります。
- 設定ファイルである `/etc/sysconfig/cryptctl-server` ファイルを直接編集してもかまいません。
  ただし直接編集する場合は、`AUTH_PASSWORD_HASH` と `AUTH_PASSWORD_SALT` の値を変更してはなりません。これらのオプションは、適切な手段で計算する必要があるためです。

14.2 `cryptctl` クライアントの設定

下記に示す `cryptctl` の対話型の設定が、唯一の設定手段となります。

まずは下記の要件を全て満たしていることを確認します:

- ネットワーク内に `cryptctl` サーバが既に存在していること。
- 暗号化の対象となるディレクトリが存在すること。
- 暗号化対象のディレクトリを保持するのに十分なサイズのある、未使用のパーティションがクライアントマシン内に存在すること。

- 自己署名型の証明書を使用する場合、サーバ側で生成された証明書 (`*.crt` ファイル) をクライアント側にも配置する必要があります。クライアント側で信頼している証明機関で署名された証明書であれば、この必要はありません。
- サーバ側でクライアントを認証する際、証明書を利用して行うように設定した場合は、サーバ側で選択した証明機関の署名が付いた TLS 証明書を用意してください。

1. `root` で下記を実行します:

```
# cryptctl encrypt
```

2. それぞれ表示された質問に対して回答を入力し、 `Enter` を押していきます。このとき、既定値は質問の末尾に角括弧で括られて表示されます。

   a. 接続先となる `cryptctl` サーバのホスト名とポートを指定します。

   b. サーバ側で TLS 証明書を利用してクライアントの認証を行うように設定している場合は、対応する証明書とその鍵を指定します。証明書はサーバ側で選択した証明機関による署名が必要となります。

   c. 次にサーバの証明書 (`*.crt` ファイル) の絶対パスを指定します。

   d. サーバを設定する際に指定した暗号化パスワードを入力します。

   e. 暗号化の対象となるディレクトリを指定し、その後にそのディレクトリを暗号化したあとのデータを保存するための空きパーティションのパスを指定します。

   f. パーティションを同時に解読することのできるマシンの台数を指定します。
   続いてクライアントからの通信が途絶えた際、追加でパーティションの解読を許可するようにするまでの時間を、秒単位で指定します。
   マシンが何らかの理由で停止して再起動した場合には、パーティションを再度解読できるようにする必要があります。そのため一般には、クライアント側の再起動にかかる時間よりも少し短めの値を、タイムアウトとして設定する必要があります。

> **重要: タイムアウトの設定値について**
>
> タイムアウトとして設定する時間が長すぎると、初回の試行では暗号化されたパーティションの解読が失敗するようになります。`cryptctl` では定期的に暗号鍵の取得を試すようになっていますが、次の試行までには時間が空きますので、遅延を発生させることになってしまいます。
>
> 逆にタイムアウトが短すぎると、暗号化されたパーティションをそのままコピーするだけで、不正なマシンが暗号を解読できることになってしまいます。

3. 暗号化を開始するには、`yes` (はい) と入力します。
   すると、`cryptctl` は指定したディレクトリの暗号化を行い、結果を空きパーティション内に保存するようになります。保存が完了すると、暗号化されたパーティションのマウントが行われます。ファイルシステムの種類は、暗号化前のファイルシステムと同じ種類になります。
   暗号化されたパーティションを作成する前に、`cryptctl` は暗号化前のディレクトリ内にある内容を、`cryptctl-moved-` という接頭辞の付いたディレクトリ内に移動します。

4. ディレクトリが正しく暗号化されたことを確認するには、下記のように入力して実行します:

```
> lsblk -o NAME,MOUNTPOINT,UUID
NAME                        MOUNTPOINT          UUID
[...]
sdc
└─sdc1                                          パーティションの_UUID
  └─cryptctl-unlocked-sdc1  /secret-partition   解読後の_UUID
```

   `cryptctl` では、暗号化されたパーティションを UUID で識別します。上記の例では、`sdc1` の UUID がそれにあたります。
   サーバ側でディレクトリが解読されているかどうかを確認するには、`cryptctl` コマンドを使用します:

```
# cryptctl list-keys
```

   正しく解読されていれば、下記のような出力が現れるはずです:

```
2019/06/06 15:50:00 ReloadDB: successfully loaded database of 1 records
Total: 1 records (date and time are in zone EDT)
Used By     When                UUID  Max.Users  Num.Users  Mount Point
IP_ADDRESS  2019-06-06 15:00:50  UUID  1          1          /secret-partition
```

   解読できていないパーティションの場合は、下記のような出力になります:

```
2019/06/06 15:50:00 ReloadDB: successfully loaded database of 1 records
Total: 1 records (date and time are in zone EDT)
Used By      When                UUID  Max.Users  Num.Users  Mount Point
             2019-06-06 15:00:50  UUID  1          1          /secret-partition
```

   `Used by` 列の違いに着目してください。
   表示された UUID が暗号化されたパーティションのものであることを確認してください。

5. パーティションの暗号化が正しく動作していることを確認したら、あとはクライアント側にある元のディレクトリ (暗号化されていないディレクトリ) を削除してください。たとえば `rm` などを使用して削除します。より安全に削除したい場合は、削除を行う前にファイルに対して `shred -u` などを使用して、ファイルの内容を上書きしてもかまいません。



重要: shred コマンドはデータが完全に消すことを保証するわけではない問題について

ストレージメディアの種類によっては、`shred` コマンドを使用してもデータが完全には消えない場合があります。特に SSD などではウエアレベリングの仕組みが存在することから、`shred` を実行しても同じ場所には書き込んでくれないことがあります。

クライアントからサーバに対する接続は、`/etc/sysconfig/cryptctl-client` ファイルで設定します。このファイルは手作業で編集してもかまいません。

クライアントのパーティションに対する暗号鍵は、サーバ内の `/var/lib/cryptctl/keydb/パーティションの_UUID` 内に保存されています。

14.3 LUKS ボリューム向け /etc/fstab の設定

LUKS で暗号化した新しいファイルシステムに対してマウントポイントを設定する場合、YaST では既定で `/etc/fstab` 内に暗号化デバイスの名称を設定します (たとえば `/dev/mapper/cr_sda1` のようなデバイス名を使用します) 。UUID やボリュームラベルではなくデバイス名を使用するのは、systemd やその他の関連ツールでの取り扱いにあたって都合がよいためです。

このような既定の動作を変更することもできます。具体的には熟練者向けパーティション設定か AutoYaST を使用します。

なお、設定を変更した場合でも、既に設定されている `/etc/fstab` には影響がありません。新しくシステムにインストールする場合や、動作中のシステムに新しいパーティションを作成する場合など、新しくマウントするファイルシステムにのみ影響があります。

14.4 サーバ側のコマンドでパーティションの解読状態を確認する方法

`cryptctl` のクライアントが動作している場合、クライアント側からは 10 秒間隔で `cryptctl` のサーバに「生存通知」(ハートビート) を送信します。クライアント側で設定した時間以上にクライアント側からの生存通知がサーバに届かない場合、サーバはクライアントがオフラインになったものと判断します。これにより、別のクライアント (もしくは再起動後の同じクライアント) から接続ができるようになります。

全ての鍵に対する使用状況を確認するには、下記のコマンドを入力して実行します:

```
# cryptctl list-keys
```

`Num. Users` 以下に、どの鍵が使用されているのかが表示されます。それぞれの鍵に対する詳細を表示したい場合は、下記のようなコマンドを入力して実行します:

```
# cryptctl show-key UUID
```

このコマンドを実行すると、マウントポイントやマウントオプション、使用時のオプションや鍵の最終取得時刻、そしてクライアントからの直近 3 回分の生存通知が表示されます。

これに加えて、`journalctl` コマンドを使用することで、鍵をいつ取得したのかのログを確認することができます。

14.5 手作業での暗号化パーティションの解読

パーティションを手作業で解読するにあたっては、2 種類の方法が存在しています。いずれもクライアント側で実行します:

- **オンライン解錠:** オンライン解錠を利用すると、タイムアウトの問題を回避したりユーザ制限を行ったりすることができます。この方式は、クライアントとサーバとの間がネットワークで繋がっているものの、クライアント側で自動マウントができないような場合に使用することができます。また、この方式を使用することで、全ての暗号化されたパーティションを解読することができます。
  オンライン解錠を実行するには、`cryptctl online-unlock` コマンドを実行します。サーバ側で設定したパスワードを入力する必要があることに注意してください。

- **オフライン解錠:** この方式は、クライアント側からサーバに接続できない状態や、接続すると何らかの理由で困ってしまうような場合に使用します。この方式を実行するには、サーバ側に暗号鍵を用意しておく必要があることに注意してください。また、この方式は最終手段としてのみ使用されるべきものであるほか、同時に 1 つのパーティションしか解錠できないことにも注意してください。
  オフライン解錠を実行するには、`cryptctl offline-unlock` コマンドを実行します。これを実行するには、あらかじめ対象のパーティションに対応するサーバ側の鍵ファイル (`/var/lib/cryptctl/keydb/パーティションの_UUID`) を用意しておく必要があります。

14.6 メンテナンス時の停止手順

メンテナンス中にパーティションが不正に解読されてしまわないようにするため、メンテナンス時にはクライアント側を停止し、`cryptctl` サーバを無効化しておくことをお勧めします。具体的には、下記のいずれかを実行します:

- `cryptctl-server` サービスを停止します:

```
# systemctl stop cryptctl-server
```

- `cryptctl` サーバをネットワークから切り離します。

14.7 cryptctl-server での HA 環境の構築

メンテナンスの際には cryptctl-server を停止しておかないと必要なデータが正しく保存されなくなってしまう一方、これによって暗号鍵を利用しているシステムが動作できなくなり、システムの停止が発生してしまいます。このような問題を避けるため、cryptctl-server で HA (高可用性) 環境を構築しておくことを強くお勧めします。この場合、少なくとも 2 台のノードから構成される高可用性クラスタを構築することになります。下記では、自己署名型の証明書を使用して、cryptctl-server 向けの HA クラスタを構築するための手順を説明しています。

まずは下記の要件を全て満たしていることを確認します:

- 少なくとも 2 台以上のサーバを用意します。これらは SUSE Linux Enterprise Server と High Availability 拡張の両方のほか、cryptctl パッケージをインストールしておきます。また、全てのサーバは SSH を介して相互にアクセスできる必要があります。
- 新しいクラスタを構築する場合は、クラスタの HA Web Konsole 向けの追加 IP アドレス (`AdminIP`) を用意します。
- また、cryptctl-server 向けに別途の IP アドレス (`CrypServerIP`) を用意します。
- さらに cryptctl-server 向けの DNS ホスト名 (`CrypServerHostName`) を用意します。このホスト名は、上述の IP アドレスを指し示すものとします。
- 鍵を保存するための HA が有効化されたブロックデバイスか、NFS 共有を用意します。下記の例では、`nfs-server.example.org/data/cryptctl-keys` という名前の NFS 共有を使用します。この共有は `/var/lib/cryptctl/keydb` という標準的なディレクトリにマウントします。
- このほか、SBD デバイスを使用しておくことを強くお勧めします。

手順 14.1: 2 ノードの CRYPTCTL HA クラスタの構築

1. `Node1` (ノード#1) に `root` でログインします。
2. 14.1項「`cryptctl` サーバの設定」に書かれた手順で cryptctl-server を構築します。このとき、下記に注意してください:
   a. 証明書を作成する場合は、cryptctl サーバでの `CrypServerHostName` に対応する証明書を用意してください。ホスト名に対応する証明書 ではありません 。
   b. cryptctl サーバに用意した個別の IP アドレス `CrypServerIP` を使用してください。既定の IP アドレス設定をそのまま使用しては なりません 。
   c. KMIP サーバを構築してはなりません。
   d. `cryptctl` サーバの起動を行うかどうかを尋ねられた場合は、`n` (いいえ) と入力してください。
3. 2 ノードの HA クラスタを構築します:
   a. **重要**
   ここでは、cryptctl サーバを構築したマシンが `Node1` というホスト名であるものとします。

   cryptctl サーバを設定したマシンで、最初のノードを起動します:

   ```
   # crm cluster init -i ネットワークデバイス名 -A AdminIP -n クラスタ名
   ```

   b. あとは `Node2` (ノード#2) にログインして、クラスタへの参加を実施します:

   ```
   # ssh Node2
   # crm cluster join -y Node1
   ```

   c. さらに詳しい情報については、Installation and Setup Quick Start (https://documentation.suse.com/sle-ha/html/SLE-HA-all/article-installation.html) をご覧ください。
4. cryptctl サーバに対するリソースグループを設定します:
5. a. cryptcl の crm-shell-script を利用することで、全てのノードで必要な全てのリソースエージェントを設定し、必要なファイルをコピーすることができます。まずは適切に構築できているかどうかを確認しておくことを強くお勧めします:

   ```
   # crm script verify cryptctl  \
   ```

```
cert-path=/etc/cryptctl/servertls/CertificateFileName \
cert-key-path=/etc/cryptctl/servertls/CertificateKeyFileName \
virtual-ip:ip=CrypServerIP \
filesystem:device=DevicePath
filesystem:fstype=FileSystemType
```

b. チェックが成功したら、下記のようにスクリプトを実行してクラスタグループの構築を行います:

```
# crm script verify cryptctl  \
cert-path=/etc/cryptctl/servertls/CertificateFileName \
cert-key-path=/etc/cryptctl/servertls/CertificateKeyFileName \
virtual-ip:ip=CrypServerIP \
filesystem:device=DevicePath
filesystem:fstype=FileSystemType
```

表 14.1: CRYPTCTL CRM スクリプトでリソースグループを設定するためのパラメーター一覧

名前	必須かどうか	既定値	説明
id	いいえ	cryptctl	リソースグループ名
cert-path	はい		作成した証明書のフルパス
cert-key-path	はい		作成した証明書に対応する機密鍵のフルパス
virtual-ip:id	いいえ	cryptctl-vip	cryptctl サーバの仮想 IP リソース ID
virtual-ip:ip	はい		cryptctl サーバの仮想 IP アドレス
virtual-ip:nic	いいえ	(virtual-ip リソースエージェントによって検出されます)	cryptctl が接続を待ち受けるべきネットワークデバイス名 (IP アドレスからデバイス名が検出できない場合にのみ必要)
virtual-ip:cidr_netmask	いいえ	(virtual-ip リソースエージェントによって検出されます)	cryptctl の IP アドレスに対応するネットマスク (IP アドレスからネッ

名前	必須かどうか	既定値	説明
			トマスクが検出できない場合にのみ必要)
virtual-ip:broadcast	いいえ	(virtual-ip リソースエージェントによって検出されます)	cryptctl サーバの IP アドレスに対応するブロードキャストアドレス (IP アドレスから検出できない場合にのみ必要)
filesystem:id	いいえ	cryptctl-filesystem	ディスクの暗号鍵とレコードを含むファイルシステムリソースの ID
filesystem:device	はい		ファイルシステムを含むデバイス名。`/dev/sda...` のようにブロックデバイス名のほか、`server:/path` のように NFS 共有を指定することができます。
filesystem:directory	いいえ	/var/lib/cryptctl/keydb	ファイルシステムのマウント先ディレクトリ名。`/dev/sda...` のようにブロックデバイス名のほか、`server:/path` のように NFS 共有を指定することができます。
filesystem:fstype	はい		ファイルシステムの種類 (NFS, XFS, EXT4 等)
filesystem:options	いいえ	(選択したファイルシステムに対応する既定値と同じ)	ファイルシステムのマウントオプション。

14.8 さらなる情報

さらに詳しい情報については、プロジェクトのホームページ https://github.com/SUSE/cryptctl/ をご覧ください。

15 ユーザ管理

改訂履歴

2024-05-10

2026-04-01

グローバル環境変数の説明を追加

15.1 様々なアカウントのチェック

15.1.1 ロック (施錠) されていないアカウント

ログインに使用していないシステムアカウントやベンダーのアカウントについては、ロック (施錠) を行っておくのが重要です。`/etc/shadow` ファイル内のパスワード欄が `!` や `*` で始まって いない ものが、ロックされていないアカウントになります。また `passwd -l` コマンドや `usermod -L` コマンドでアカウントをロックすると、同ファイルのパスワード欄の冒頭に `!` が追加されますので、これによって簡単にロックを行うことができます。また、多くのシステムアカウントや共有アカウントは、既定で `*`, `!!`, `!*` のいずれかがパスワード欄に設定されていますが、これらも暗号化されたパスワードとしては使用できない文字列となることから、これによってログインを防止しています。このような背景から、ロックされていないアカウントを一覧表示したい場合は、下記のようなコマンドを入力して実行します:

```
# egrep -v ':\*|:\!' /etc/shadow | awk -F: '{print $1}'
```

また、`/etc/passwd` ファイルのパスワード欄に `x` が設定されていることも確認しておいてください。下記のコマンドは、パスワード欄に `x` が書かれていないアカウントの一覧を表示します:

```
# grep -v ':x:' /etc/passwd
```

パスワード欄に `x` が書かれているアカウントは、そのパスワードが shadow で暗号化されていることを表します。この場合、実際のパスワードは `/etc/shadow` 内に暗号化されて記録されます。また、`/etc/passwd` ファイルのパスワード欄が空である場合は、shadow ファイルの参照は行われず、ログイン時のパスワード入力も行われない意味になります。

15.1.2 使用されていないアカウント

ユーザやアプリケーション、システムやデーモンが使用していないシステムアカウントやベンダーアカウントについては、システムから削除しておくべきです。下記のコマンドを使用することで、それぞれのアカウントがどのファイルを所有しているのかを調べることができます:

```
# find / -path /proc -prune -o -user アカウント -ls
```

上記では `-prune` オプションを設定していますが、これは /proc ファイルシステム以下を読み飛ばす設定です。上記を実行した結果、アカウントを削除しても問題がないと判断できれば、下記のコマンドを実行してアカウントを削除してかまいません:

```
# userdel -r アカウント
```

`userdel` コマンドに `-r` オプションを指定しないと、ユーザのホームディレクトリやメールスプールファイル (`/var/spool/mail/アカウント`) が削除されなくなります。なお、多くのシステムアカウントに対しては、ホームディレクトリの設定はありません。

15.2 パスワードの有効期限の設定

パスワードに対して有効期限を設定することは、一般に推奨されるセキュリティ設定ですが、システムアカウントや共有アカウント (例: Oracle) については、このルールから除外しておく必要があります。これは、パスワードの期限が切れたタイミングで、そのアカウントを使用するアプリケーションが動作しなくなってしまうからです。

一般に企業内のポリシーでは、システムアカウントや共有アカウントのパスワード変更に関する条項を作成しておくべきです。それ以外の一般ユーザのアカウントについては、自動的に期限が切れるように設定しておくとよいでしょう。下記の例では、ユーザアカウントごとにパスワードの有効期限を設定するための方法を示しています。

下記の表には、`useradd` コマンドで新しいアカウントを作成する際に使用できる、各種のファイルとパラメータが示されています。これらの設定は、アカウントの作成時に `/etc/shadow` ファイル内に書き込みが行われます。新しいユーザを作成する際に YaST のツール ([ユーザとグループの管理]) を使用すると、ユーザごとに設定を行うことができます。また、設定内容によってはシステム全体に適用されるものもあります (たとえば `/etc/login.defs` ファイルや `/etc/default/useradd` ファイルの変更がそれにあたります):

`/etc/login.defs`	`PASS_MAX_DAYS`	パスワードの最長有効期間 (日) を指定します。

`/etc/login.defs`	`PASS_MIN_DAYS`	パスワードの最短有効期間 (日) を指定します。これより短い間隔では、パスワードの変更ができなくなります。
`/etc/login.defs`	`PASS_WARN_AGE`	前回のパスワード変更から、次のパスワード変更通知までの日数を指定します。
`/etc/default/useradd`	`INACTIVE`	パスワードの有効期限が切れてから、アカウントが無効化されるまでの日数を指定します。
`/etc/default/useradd`	`EXPIRE`	アカウントそのものの有効期限を YYYY-MM-DD の形式で指定します。



注記: 既存のユーザに対する影響について

上記のファイルは、いずれもアカウントの作成時に適用されるもので、ファイルの変更前に作成されたアカウントについては効果がありません。

上記の `/etc/login.defs` ファイルや `/etc/default/useradd` ファイルを変更したことを確認したら、あとは実際にユーザを作成して、`/etc/shadow` ファイルにどのような設定が行われるのかを確認します。

新しいユーザを作成するには、下記のようなコマンドを入力して実行します:

```
# useradd -c "テストユーザ" -g グループ ユーザ名
```

`-g` オプションは、追加するアカウントのプライマリグループを指定するためのオプションです:

```
# id ユーザ名
uid=509(test) gid=100(users) groups=100(users)
```

`/etc/login.defs` と `/etc/default/useradd` に設定した内容が、新しく作成した test ユーザに対してどのように反映されたのかを確認します。`/etc/shadow` ファイル内を下記のようにして表示します:

```
# grep ユーザ名 /etc/shadow
test:!!:12742:7:60:7:14::
```

パスワードの有効期限については、`chage` コマンドを使用することでいつでも変更できます。システムアカウントや共有アカウントに対してパスワードの有効期限を無効化したい場合は、`chage` コマンドで下記のように入力して実行します:

```
# chage -M -1 システムアカウント名
```

パスワードの有効期限に関する情報を表示するには、下記のように入力して実行します:

```
# chage -l システムアカウント名
```

たとえば下記のようになります:

```
# chage -l TEST
Minimum: 7
Maximum: 60
Warning: 7
Inactive: 14
Last Change: Jan 11, 2015
Password Expires: Mar 12, 2015
Password Inactive: Mar 26, 2015
Account Expires: Never
```

15.3 複雑なパスワードの強制

セキュリティの実践の一環として、ユーザに対し単純なパスワードを禁止するような措置を取っておくことが重要です。パスワードが安全に保管されている限り、複雑なパスワードを設定すれば、容易には侵入できなくなるからです。また、複雑なパスワードを設定しておくことで辞書攻撃からアカウントを保護することができますし、ロックアウト (施錠) と併用すれば、さらに侵入を困難にすることができます。しかしながら、必ずしもこの方式が適切とは限りません。ロックアウトを併用すれば誰もアプリケーションやシステムに入れなくなってしまうことになりますので、サービス拒否 (Denial of Service; DoS) 攻撃というもう 1 つの問題を引き起こしてしまいます。

このような事情から、まずはパスワードの管理をより現実的かつ効率的に行うことが重要です。多くの企業ではパスワードに対して、1 文字以上の数字と小文字、大文字を必要とするようにしています。ポリシーは企業によって異なりますが、パスワードの複雑さの強制と効率性の両立は、いくぶん難しいものになります。

15.4 PAM を利用したパスワードとログインの管理

Linux-PAM (Pluggable Authentication Modules for Linux) は共有ライブラリから成るパッケージスイートで、ローカルのシステム管理者に対してアプリケーションからのユーザ認証方法を設定する機能を提供します。

PAM が提供する機能のほか、この仕組みによって提供される環境ごとの最適な構成方式については、よく知っておくことをお勧めします。この設定作業は一度だけ実施すればよく、全てのシステムに対して同じ設定を行っておく (設定の標準化) こともできますし、そこからさらにホスト固有の設定を追加する (特定のホストやサービス、アプリケーションに対するセキュリティ強化) こともできます。これはつまり、この仕組みが非常に柔軟な構造であることを理解することにもなります。

PAM のアーキテクチャを詳しく知りたい場合は、`/usr/share/doc/packages/pam` ディレクトリ内にある PAM のドキュメンテーション (英語) をお読みになることをお勧めします。ここには様々な形式が用意されています。

下記の説明では、特にパスワードの強度ポリシーに関する設定など、既定の PAM 設定の変更方法の例を示しています。これにはたとえばパスワードそのものの複雑性のほか、パスワードの再利用やアカウントのロックアウト (一時的な無効化) も含まれます。これらは機能全体のうちのごく一部ではありますが、PAM の構造や柔軟性を理解するには最適な例です。

重要: `pam-config` の制限について

`pam-config` ツールは PAM の設定ファイルのうち、common-{account,auth,password,session} の各設定ファイルを変更するために使用します。これらのファイルにはグローバルオプションのほか、下記のようなコメントが書かれています:

```
# This file is autogenerated by pam-config. All changes
# will be overwritten.
# (このファイルは pam-config で自動生成されたものです。直接ここで編集を
# 行っても、上書きされてしまう可能性があります)
```

それ以外の個別のサービスファイル、たとえば `login, password, sshd` , `su` などのファイルについては、手作業で編集しなければなりません。もちろん `pam-config` を全く使用せず、全てのファイルを手作業で編集してもかまいません。ですが `pam-config` には古い設定の変換や現在の設定の更新、そして正当性のチェックなど、様々な機能が用意されています。詳しくは `man 8 pam-config` で表示されるマニュアルページをお読みください。

15.4.1 パスワードの強度

openSUSE Leap ではパスワードの強度を調べるための `pam_cracklib` ライブラリが提供されています。このライブラリは、パスワードが弱すぎるような場合、より強いパスワードの使用を提案する機能も備えています。下記では、このライブラリが提供する、パスワードポリシーの設定や監査目的で使用する各種のパラメータについて説明しています。

PAM ライブラリは指定された順序で処理を行います。完璧に期待通りの動作になるよう、要件とポリシーをフローチャートの形にまとめてから設定することをお勧めします。

表 15.1: パスワードの強度に関するルール例

pam_cracklib.so	minlen=8	パスワードの文字数の最小値を指定します。ここでは 8 文字を指定しています
pam_cracklib.so	lcredit=-1	パスワードのうち、小文字の文字数の最小値を指定します。ここでは 1 文字を指定しています
pam_cracklib.so	ucredit=-1	パスワードのうち、大文字の文字数の最小値を指定します。ここでは 1 文字を指定しています
pam_cracklib.so	dcredit=-1	パスワードのうち、数字の文字数の最小値を指定します。ここでは 1 文字を指定しています
pam_cracklib.so	ocredit=-1	パスワードのうち、その他の文字数の最小値を指定します。ここでは 1 文字を指定しています

これらのパスワード制限を設定するにあたっては、`pam-config` ツールを利用することができます。たとえばパスワードの長さの最小値を設定したい場合は、下記のように入力して実行します:

```
> sudo pam-config -a --cracklib-minlen=8 --cracklib-retry=3 \
--cracklib-lcredit=-1 --cracklib-ucredit=-1 --cracklib-dcredit=-1 \
--cracklib-ocredit=-1 --cracklib
```

これで新しいパスワードに対する強度チェックが行われるようになります。あとは root 以外のアカウントでログインしなおし、`passwd` コマンドでパスワードを設定して試してみるとよいでしょう。なお上記の要件は、root で `passwd` コマンドを実行した場合には適用されないことに注意してください。

15.4.2 過去に使用したことのあるパスワードの制限

pam_pwhistory モジュールを使用することで、過去に使用したことのあるパスワードを禁止するように設定することもできます。下記の例では、少なくとも過去 6 ヶ月間は、以前のパスワードを使用できないようにシステムを設定します:

```
> sudo pam-config -a --pwhistory --pwhistory-remember=26
```

15.2項「パスワードの有効期限の設定」では `PASS_MIN_DAYS` に `7` を設定しています。これはパスワードを変更する際の最小間隔を指定する項目です。上記のように `pam_unix` に `26` を設定すると、過去 26 回分 (つまり、最低でも 26*7 日分 = 約 6 ヶ月分) のパスワードを記憶して、使用できないようにすることができます。

ここまでの `pam-config` コマンドの設定で生成された PAM 設定ファイル (`/etc/pam.d/common-password`) の内容は、下記のようになるはずです:

```
auth      required   pam_env.so
auth      required   pam_unix.so     try_first_pass
account   required   pam_unix.so     try_first_pass
password  requisite   pam_cracklib.so
password  required   pam_pwhistory.so        remember=26
password  optional   pam_gnome_keyring.so    use_authtok
password  required   pam_unix.so     use_authtok nullok shadow try_first_pass
session   required   pam_limits.so
session   required   pam_unix.so     try_first_pass
session   optional   pam_umask.so
```

15.4.3 ログイン失敗時のユーザアカウントのロックアウト

ssh, login, su, sudo などで一定回数以上のログイン失敗が繰り返された場合、対象のアカウントをロックアウトする措置は、一般的なセキュリティ対策として実施されるものです。しかしながら、これを実施してしまうとアプリケーションや管理者、`root` ユーザでのログインができなくなってしまいます。

重要: サービス拒否攻撃について

パスワードの入力失敗回数は、故意にログインを失敗させるだけで増やすことができてしまいますから、サービス拒否攻撃として容易に悪用されてしまいます。

そのため、パスワードの入力失敗回数によるロックアウトは、どうしても必要な場合だけに留めるものとし、対象のアカウントは少なく、かつ必須のアカウントについては除外するようにしてください。また、ロックアウトは人間が使用するアカウントだけでなく、サービスを提供するシステムアカウントに対しても発生しうることにも注意してください。

openSUSE Leap の既定では、アカウントのロックアウトは行いません。ただし、`pam_tally2` モジュールを使用することで容易にロックアウトを設定することができます。`/etc/pam.d/login` 内の冒頭に下記の内容を記述することで、`root` を除く全ユーザに対し、6 回のログイン失敗時に 10 分間のロックアウトを設定することができます:

```
auth required pam_tally2.so deny=6 unlock_time=600
```

下記は `/etc/pam.d/login` ファイルに設定した場合の例です:

```
#%PAM-1.0
auth     requisite      pam_nologin.so
auth     include        common-auth
auth     required       pam_tally2.so deny=6 unlock_time=600
account  include        common-account
account  required       pam_tally2.so
password include        common-password
session  required       pam_loginuid.so
session  include        common-session
#session  optional       pam_lastlog.so nowtmp showfailed
session  optional       pam_mail.so standard
```

`root` ユーザに対してもロックアウトを設定することができますが、注意してお使いください:

```
auth required pam_tally2.so deny=6 even_deny_root unlock_time=600
```

root に対して異なるロックアウトを設定することもできます:

```
auth required pam_tally2.so deny=6 root_unlock_time=120  unlock_time=600
```

ロックアウトの解除を自動では行わず、管理者の操作を必要としたい場合は、`unlock_time` オプションを削除してください。下記の 2 つのコマンド例では、ログイン失敗回数を表示して、ロックアウト解除を行う例を示しています:

```
> sudo pam_tally2 -u username
Login           Failures Latest failure     From
username            6    12/17/19 13:49:43  pts/1

> sudo pam_tally2 -r -u username
```

ログインを試した記録は、既定で `/var/log/tallylog` 内に保存されます。

ロックアウト時間の経過後や管理者によるロックアウト解除を実施すると、上記のカウンタは 0 にリセットされます。

その他のログインサービスで `pam_tally2` を使用するには、`/etc/pam.d/` 内にあるそれぞれの設定ファイル (`sshd, su, sudo, sudo-i, su-l`) に設定を追加してください。

15.5 root ログインの制限

既定では `root` にはパスワードを設定するため、さまざまな方法でログインができることになります。たとえばローカルの端末やグラフィカルなセッション、SSH 経由でのログインなどがあります。これらはセキュリティ上の理由から、できる限り制限しておくことが望ましいものです。もちろん root アカウントを

共有して使用するようなことも避けるべきです。このようなことから管理者は、`su` や `sudo` (詳しくは `man 1 su` コマンドもしくは `man 8 sudo` コマンドで表示されるマニュアルページをお読みください) などのツールを使用して特権を得るべきです。これにより、管理者でのログインを特定のユーザに結びつけることができるようになります。この方式では `root` のパスワードに加え、もう 1 つの安全性も提供します。それは root の特権を得るには `root` のパスワード だけでなく 、管理者 (一般ユーザ) のパスワード認証も突破しなければならないようにできる、という点です。本章では、システムのさまざまなレベルにおいて、直接的な root ログインを制限するための方法を説明しています。

15.5.1 ローカルのテキストコンソールへのログイン制限

TTY デバイスはコンソール経由でのテキストモードアクセスを提供するデバイスです。デスクトップシステムの場合は直接接続されたキーボード経由で、サーバシステムの場合は KVM スイッチやリモートマネージメントカード (例: ILO, DRAC) 経由でも使用することができます。既定では、Linux は 6 つの異なるコンソールを提供します。これらは `Alt` – `F1` から `Alt` – `F6` までを押すことで切り替えることができます。グラフィカル環境が動作している場合は、それぞれ `Ctrl` – `Alt` – `F1` から `Ctrl` – `Alt` – `F6` までを押してください。それぞれの端末は `tty1` から `tty6` に対応します。

下記の手順では、最初の TTY のみを使用できるよう root アクセスを制限しています。この方式は緊急時の管理者アクセスを残しておくためのもので、通常は TTY を使用すべきではありません。



注記

また、ここで示している手順は一般的な PC アーキテクチャ (x86 および AMD64/Intel 64) 向けの設定手順となります。POWER 等のアーキテクチャの場合は、`tty1` ではないデバイス名になります。また、端末デバイス名は誤って指定することのないように注意してください。現在使用している端末デバイス名を知りたい場合は、`tty` コマンドをお使いください。また、SSH 経由でのログインやグラフィカルなセッションを使用している場合 (表示されるデバイス名が `/dev/pts/N` のようになります) は設定しないでください。実際のログイン端末 (`Alt` – `FN` でアクセスできるもの) にのみ設定してください。

手順 15.1: ローカル TTY への ROOT ログインの制限

1. まずは PAM スタックの設定ファイル `/etc/pam.d/login` 内の `auth` ブロックに、`pam_securetty` モジュールを設定します:

```
auth     requisite      pam_nologin.so
 auth     [user_unknown=ignore success=ok ignore=ignore auth_err=die default=bad]
 pam_securetty.so noconsole
 auth     include        common-auth
```

これにより、ローカルコンソールでの認証時に、`pam_securetty` モジュールが使用されるようになります。このモジュールは `root` に対し、ログイン可能な TTY デバイスを制限する機能を提供します。ログイン可能な TTY デバイスの一覧は、`/etc/securetty` ファイルで設定します。

2. `/etc/securetty` ファイルの内容を編集して、1 つだけ残すようにします。これにより、root ユーザが使用できる TTY デバイスを制限できることになります。

```
#
# This file contains the device names of tty lines (one per line,
# without leading /dev/) on which root is allowed to login.
#
tty1
```

3. 設定を行ったら、実際に `root` でログインを試してみて、指定した TTY 以外からログインできなくなっていることを確認します。上記の例ではたとえば `tty2` などがそれにあたります。この場合、パスワードを尋ねられることなく即時に拒否されるはずですので、こちらもご確認ください。また、指定した TTY (`tty1`) からログインできることも確認しておいてください。これにより、ロックアウトされているのではなく、端末の制限でログインできなくなっていることが確定できるためです。

重要: pam_securetty モジュールの追加について

`/etc/pam.d/common-auth` に対しては `pam_securetty` を設定してはなりません。設定してしまうと `su` や `sudo` のコマンドにも影響が及ぶことになり、これらのコマンドでも TTY が制限されることになるためです。

重要

上記のように設定変更を行うと、シリアルコンソール (例: `/dev/ttyS0`) からのログインも制限対象となり、ログインができなくなります。シリアルコンソールからのログインを許可したい場合は、`/etc/securetty` ファイル内にシリアルのデバイス名を追加してください。

15.5.2 グラフィカルセッションへのログインの制限

サーバのセキュリティをより高めたい場合は、グラフィカル環境そのものの使用を避けることをお勧めします。グラフィカル環境のプログラムはいずれも `root` での使用を想定しておらず、コンソールプログラムよりもセキュリティ上の問題となりうる可能性があるためです。どうしてもグラフィカルログインが必要な場合は `root` 以外でログインするものとし、`root` ではグラフィカルログインができないように設定してください。

`root` に対してグラフィカル環境へのログインを禁止したい場合も、15.5.1項「ローカルのテキストコンソールへのログイン制限」と同じ手順を使用することができます。ディスプレイマネージャに属するPAM スタックの設定ファイル (たとえば GDM であれば `/etc/pam.d/gdm`) に、`pam_securetty` モジュールを追加してください。また、グラフィカルなセッションは TTY デバイス (既定では `tty7`) で動作していますので、`root` のログインを `tty1` だけに制限すれば、`root` はグラフィカル環境も使用できないようになります。

15.5.3 SSH ログインの制限

既定では、`root` ユーザは SSH ネットワークプロトコル経由でログインできるようになっています (SSH ポートがファイアウオールでブロックされていない限り) 。`root` でのログインを禁止したい場合は、OpenSSH の設定を変更することで実現することができます:

1. `/etc/ssh/sshd_config` ファイルを編集して、下記のようなパラメータを設定します:

   ```
   PermitRootLogin no
   ```

2. あとは `sshd` サービスを再起動して、設定を適用します:

   ```
   systemctl restart sshd.service
   ```



注記

OpenSSH の場合、`pam_securetty` モジュールでの設定は不適切です。これは SSH の仕様上の理由で、PAM スタック経由でログインするとは限らない (たとえば SSH は公開鍵認証にも対応しており、この場合は PAM を経由しない) ためです。これに加えて SSH では、パスワードの入力が誤っている場合と、ポリシーによって制限されている場合とで応答が異なるため、悪意のある攻撃者であれば容易に識別されてしまう問題もあります。

15.6 `sudo` を利用できるユーザの制限

`sudo` コマンドは、他のユーザ (一般的には `root`) に成り代わってコマンドを実行するためのコマンドです。`sudo` コマンドの設定には、成り代わり元や成り代わり先のユーザやグループ、そしてそれらのユーザやグループに対して許可するコマンド等のルールセットを記述します。設定は `/etc/sudoers` というファイルで指定します。

SUSE システムの既定の設定では、`sudo` コマンドは `root` のパスワードを尋ねるようになっています。ただし、`su` とは異なり、`sudo` にはパスワードを記憶する機能が用意されていて、5 分間以内の再実行であれば、パスワードを再度尋ねたりすることはありません。そのため、`sudo` は管理者として信頼できるユーザにのみ提供すべきものです。

手順 15.2: 一般ユーザに対する `sudo` の制限

1. `visudo` コマンドを実行するなどして、`/etc/sudoers` ファイルを編集します。
2. 下記のような行をコメントアウトして、それぞれのユーザがコマンドを実行する際、常に成り代わり先のユーザに対するパスワード入力を求めるように設定します。コメントアウトした後は、下記のようになるはずです:

   ```
   #ALL ALL=(ALL) ALL # WARNING! Only use this together with 'Defaults targetpw'!
   ```
3. 逆に、下記の行のコメントを外します:

   ```
   %wheel ALL=(ALL) ALL
   ```

   上記の行は `wheel` というグループのメンバーにのみ上述の機能を提供するための設定です。お使いの環境で `wheel` グループに対して許可を与えると問題があるとお考えの場合は、他のグループでもかまいません。
4. あとは `sudo` を特定のグループにのみ許可するよう、グループへのユーザ追加を行います。たとえば `tux` を `wheel` グループに追加したい場合は、下記のように入力して実行します:

   ```
   usermod -aG wheel tux
   ```

   なお、グループへの追加を反映させるには、対象のユーザにログインし直してもらう必要があります。
5. あとはグループに追加していないユーザでコマンドを実行してみて、実行が拒否されることを確認します。拒否されると、下記のようなエラーメッセージが表示されるはずです:

   ```
   wilber は sudoers ファイル内にありません。この事象は記録・報告されます。
   ```

   同様にグループに追加したユーザでコマンドを実行してみます。設定が正しければ、`sudo` でコマンドを実行できるはずです。

なお、ここまでの設定は `sudo` の機能に対する制限を設定しただけであり、`su` コマンドの機能は制限していないことに注意してください。こちらのコマンドは従来どおり、誰にでも利用できてしまいます。また、システムへのアクセス方式として別経路の方式が存在する場合、`root` のパスワードを知っていれば、その経路でログインして実行できる可能性があることにも注意してください。

15.7 対話型シェルセッションに対する無操作タイムアウトの設定

対話型のシェルに対しては、何も操作していない場合のタイムアウト時間を設定して、一定の時間が経過すると自動的に切断するように設定するとよいでしょう。この設定を行うことでセッションの盗用を未然に防ぐことができるほか、システムリソースの浪費を防ぐことにも繋がります。

既定では、シェルに対する無操作タイムアウトは設定されていません。シェルを開いて何も操作を行わなくても、何日も何年も開いたままになります。ほとんどのシェルには無操作タイムアウトの設定が存在し、一定時間の経過後自動的にセッションを終了させることができます。下記では、一般的に使用されるシェルを対象に、無操作タイムアウトの設定を示しています。

無操作タイムアウトはログインシェルにのみ適用するか、全ての対話型シェルに適用するかを選択することができます。全ての対話型シェルに設定した場合、タイムアウトはそれぞれのシェルで別々に動作し、タイムアウト値は累積して処理されます。言い換えると、サブシェルや子シェルを起動した場合、そのシェルに対しては新しいタイムアウト値が設定されるため、サブシェルもしくは子シェルが時間切れになっても親のシェルはそのタイミングでは終了せず、動作し続けることになります。

下記の表には、openSUSE Leap に同梱されている各種のシェルと、設定方法に関する一覧が記述されています:

パッケージ	使用できるシェル	シェル変数	時間単位	読み込み専用にするための設定方法	設定パス (ログインシェルの場合)	設定パス (全てのシェルの場合)
`bash`	`bash, sh`	`TMOUT`	秒	`read-only TMOUT=`	`/etc/profile.local`, `/etc/profile.d/`	`/etc/bash.bashrc`
`mksh`	`ksh, lksh, mksh, pdksh`	`TMOUT`	秒	`read-only TMOUT=`	`/etc/profile.local`, `/etc/profile.d/`	`/etc/ksh.kshrc.local`
`tcsh`	`csh, tcsh`	`autologout`	分	`set -r autologout=`	`/etc/csh.login.local`	`/etc/csh.cshrc.local`

パッケージ	使用できるシェル	シェル変数	時間単位	読み込み専用にするための設定方法	設定パス (ログインシェルの場合)	設定パス (全てのシェルの場合)
`zsh`	`zsh`	`TMOUT`	秒	`readonly TMOUT=`	`/etc/profile.local`, `/etc/profile.d/`	`/etc/zsh.zshrc.local`

上記のシェルはいずれも内部タイムアウト値の設定が存在し、これに値を設定することで無操作タイムアウトを実現しています。ユーザ側でタイムアウト設定を上書きさせたくない場合は、シェル変数を読み込み専用にしてください。具体的な設定方法も上記で示しています。

なお、シェル固有の設定ファイルを利用してシステム全体のタイムアウト設定を行ってしまうと、一貫性を損なうことになってしまいます:

- たとえば `/etc/bash.bashrc` ファイルの場合、`.local` の拡張子は bash の場合にのみ読み込まれますので、標準の `/bin/sh` の場合は、これらのタイムアウト設定を読み込まなくなってしまいます。
- また `/etc/ksh.kshrc.local` は `mksh` でのみ動作し、派生したソフトウエアである `lksh` や `pdksh` では無視されます。

これらの設定に一貫性を持たせるためには、グローバルな設定ファイル を作成して、様々なシェルの起動時に必ず読み込まれるように設定してください。具体的には、`/etc/profile.local` ファイルを編集して `$ENV` 変数を設定し、全てのログインセッションで指定したファイルを読み込むようにします。

1. まずはファイルを開きます:

```
> sudo  /etc/profile.local
```

2. ファイルの末尾に下記の行を追記します:

```
# 全ての POSIX シェルに対して汎用の rc ファイルを読み込ませる
export ENV=/etc/sh_common_rc
```

3. あとは汎用のリソースファイルを作成して、パーミッションを設定します:

```
> sudo  touch /etc/sh_common_rc
```

```
> sudo  chmod 644 /etc/sh_common_rc
```

4. あとはファイルを開いて、必要なタイムアウト設定を追加します:

```
# /etc/sh_common_rc
# セッションタイムアウトの設定（15 分 = 900 秒）
# 'readonly' を指定することで、ユーザ側での上書きを禁止しています
readonly TMOUT=900
export TMOUT
```



注記: 不正行為に対する保護にはならない件について

この機能は、ユーザがセッションの存在を忘れてしまいがちな状況や、セキュリティに配慮していない使い方に対する対策として設定できる項目であり、悪意に対する対策にはならないことに注意してください。また、このタイムアウト値はシェルの待機状態に対してのみ働くもので、悪意を持ってすれば容易にタイムアウトを無効化できてしまいます。

無操作タイムアウトを設定するには、それぞれのシェルの設定ファイルに対して変数を設定します。ログインシェルのみに設定する場合と、全てのシェルに対して設定する場合とでファイルが異なることにも注意してください。下記の例は `bash` と `ksh` に対して働く設定で、ユーザ側からはタイムアウト値を書き換えることができないよう、読み込み専用の設定を行っています。下記の内容は `/etc/profile.d/timeout.sh` ファイルを作成して記述してください:

```
# /etc/profile.d/timeout.sh for SUSE Linux
#
# Timeout in seconds until the bash/ksh session is terminated
# in case of inactivity.
# 24h = 86400 sec
readonly TMOUT=86400
```



ヒント

ログアウト後にもセッションを残しておきたい場合は、`screen` ツールをお使いになることをお勧めします。`screen` セッションはログアウト時にも終了させられることがなく、必要に応じて再接続できるためです。また、ログアウトせずにセッションをロック (施錠) することもできます (`man screen` で表示されるマニュアルページで、`Ctrl` – `A` – `X` / `lockscreen`) に関する説明をお読みください) 。

15.8 不用意なサービス拒否攻撃の防止

Linux ではユーザやグループに対して、システムリソースの制限を設定することができます。これはプログラム内にバグ (たとえばメモリリーク) があったような場合に非常に有用な仕組みで、これによってシステムの動作が遅くなったり、場合によっては全く使えなくなってしまったりするようなことを防ぐことができます。また、この制限を設定することにより、プログラムがリソースを使用しすぎてしまい (たとえばプログラムが利用可能なファイルハンドル全てを占有してしまったりなど) 、新しい接続を受け付けられなくなってしまったり、ローカルログインができなくなってしまったりする問題を防ぐこともできます。その反面、リソースの制限はサービス拒否 (Denial of Service; DoS) 攻撃を容易にしてしまうことにも注意してください。一般的にはユーザやグループに対してリソース制限を設定すると、その用途にもよりますが、システムの保護に役立てることができます。

15.8.1 システムリソースの制限例

下記の例では Oracle というユーザアカウントに対してリソース制限を設定しています。リソース制限の設定項目については、`/etc/security/limits.conf` ファイルを読むか、`man limits.conf` で表示されるマニュアルページをお読みください。

bash などのようなほとんどのシェルでは、さまざまなリソース (最大ファイルディスクリプタ数や最大プロセス数など) の制限を行うことができます。これらはユーザ単位で設定することができます。現在のシェルでのリソース制限設定を表示するには、下記のようなコマンドを入力して実行します:

```
# ulimit -a
```

bash シェルにおける `ulimit` の説明については、bash のマニュアルページをお読みください。

重要: SSH セッションに対する制限の設定について

SSH セッションに対しては、「hard」や「soft」の制限が期待通りにならないことがあります。正しく動作させたい場合は、いったん root でログインしたのち、対象のユーザ (たとえば `Oracle`) に `su` をして切り替えるようにしてください。またリソースの制限は、システムの起動時に開始されたアプリケーションに対しても作用させることができます。SSH 経由でリソース制限が正しく動作していないように見える場合は、`/etc/ssh/sshd_config` ファイル内の `UsePrivilegeSeparation` オプションを `no` に設定して、SSH デーモンを再起動 (`systemctl restart sshd`) してください。ただし、このような設定は、別の意味でシステムのセキュリティを弱める結果になることに注意してください。

ヒント: ssh でのパスワードログインの禁止について

SSH 経由でのログインに際しては、パスワードでのログインそのものを禁止することで、さらに安全性を高めることができます。なお、パスワードでのログインを禁止する場合は、あらかじめ公開鍵でのログインが正しく動作することを確認した後にしてください。パスワードでのログインを禁止するには、`/etc/ssh/sshd_config` ファイルに下記のような行を追加します:

```
UseLogin no
UsePAM no
PasswordAuthentication no
PubkeyAuthentication yes
```

下記の例では `oracle` ユーザが開くことのできるファイルおよびファイルハンドルの数を変更しています。この変更は `root` で `/etc/security/limits.conf` ファイルを編集することで行います:

```
oracle           soft    nofile          4096
oracle           hard    nofile          63536
```

上記の "soft" の行は `oracle` ユーザがログインしてからのファイルハンドル (開いているファイル) の数の上限を設定していて、この値を超過すると警告メッセージが表示されるようになります。絶対的な上限は "hard" の行で設定し、この例では 65536 個までを利用できることになります。コマンドで設定する場合は、下記のように入力して実行します:

```
# ulimit -n 63536
```

上記の "soft", "hard" の値は必要に応じて調整してください。

注記: ulimits を使用する場合の注意について

ulimit を設定する場合は、よく注意して設定してください。「nofile」に対して「hard」の上限をカーネルの規定する最大値 (`/proc/sys/fs/file-max`) に設定してしまうと、もしもそのユーザが全てのファイルハンドルを使用してしまった場合、PAM モジュールにアクセスする際にもファイルハンドルを使用しますので、ログインさえもできなくなってしまうことになります。

リソース制限は `pam_limits` を使用して設定することもできます。全てのサービスで共通で設定する場合は `/etc/pam.d/common-auth` ファイルを、SSH, su, login, telnet など、サービスごとに設定する場合は下記のファイルをそれぞれ編集してください:

`/etc/pam.d/sshd` (SSH の場合)
`/etc/pam.d/su` (su の場合)
`/etc/pam.d/login` (ローカルログインや telnet の場合)

全てのログインに対して適用したくない場合は、`/etc/security/limits.conf` ファイルを読み込む PAM モジュールがあります。PAM の設定ディレクティブは下記のようになります:

```
session     required      /lib/security/pam_limits.so
session     required      /lib/security/pam_unix.so
```

なお、上記で変更を行っても即時には反映されず、次回のログインから適用されることに注意してください:

```
# su - oracle
> ulimit -n
4096
```

また、ここでの説明はいずれも bash シェルを使用した場合の例であり、それ以外のシェルを使用する場合は `ulimit` オプションが異なることに注意してください。たとえば `oracle` ユーザに対する制限が `4096` であった場合、これを `63536` まで増やすには、下記のように入力して実行します:

```
# su - oracle
> ulimit -n
4096
> ulimit -n 63536
> ulimit -n
63536
```

`ulimit -n 63536` のような設定 (この例では bash) を恒久的に保存したい場合は、openSUSE Leap が提供する bash シェル向けのスタートアップファイル (`~/.bashrc` ファイルもしくは `~/.profile` ファイル) にそれを記述します (使用しているシェルを確認したい場合は、`echo $SHELL` を実行してください) 。下記のようなコマンドを実行することで、`oracle` ユーザに対する bash シェルのリソース制限を設定することができます:

```
# su - oracle
> cat >> ~oracle/.bash_profile << EOF
ulimit -n 63536
EOF
```

15.9 ログインバナーの表示

ポリシーや監査の理由から、もしくはユーザに対してセキュリティ手順を示しておく理由から、ログイン時にバナーを表示させることができます。

SSH やローカルコンソールなどからユーザがログインした 後 にログインバナーを表示させたい場合は、`/etc/motd` (motd = message of the day の略) ファイルにそれを記述します。このファイルは openSUSE Leap の既定で提供されていますが、特に何も書かれていません。表示したい内容をそのまま記述してください。

注記: バナーの長さについて

バナーとして表示するメッセージは、長くても端末 1 ページ以内に収まるようにしてください。端末によってはスクロールして戻ることができない場合があるため、長すぎると読めなくなってしまうことがあるためです。

ログインバナーはテキストベースの端末であれば、ログイン 前 に表示することもできます。ローカルコンソールへのログイン前に表示したい場合は `/etc/issue` ファイルに内容を記述してください。これはログインプロンプトよりも前に表示されます。SSH 経由でのログインの場合は、`/etc/ssh/sshd_config` ファイルの「Banner」パラメータを設定してください。こちらも SSH のログインプロンプトよりも前に表示されます。

GDM 経由でのグラフィカルログインの場合は、GNOME 管理者ガイド (英語) (https://help.gnome.org/admin/system-admin-guide/stable/login-banner.html.en) にある手順でログインバナーを設定することができます。これに加えて、[はい] か [いいえ] を選択するようなバナーを表示することもできます。これを行うには、`/etc/gdm/Xsession` ファイルを編集して、スクリプトの 最初のほう に下記のような内容を記述します:

```
if ! /usr/bin/gdialog --yesno '
This system is classified...
' 10 10; then
    /usr/bin/gdialog --infobox 'Aborting login'
    exit 1;
fi
```

上記の `This system is classified...` の部分に、表示したいバナーテキストを指定します。なお、このダイアログはログイン処理そのものを防ぐ仕組みではありません。GDM のスクリプト処理について、詳しくは GDM のドキュメンテーション (英語) (https://gitlab.gnome.org/GNOME/gdm) をお読みください。

15.10 各種のアカウント関連ユーティリティ

下記はユーザログインに関するデータを取得できるコマンドの一覧です:

`who`: 現時点でログインしているユーザを一覧表示します。

`w`: 誰がログインしているのかと、何をしているのかを表示します。

`last`: 直近のユーザログインを一覧で表示します。ログイン日時のほかログアウト日時、IP アドレスなどの情報が含まれます。

`lastb`： `last` と同じ出力を行いますが、こちらはログインの失敗を記録する `/var/log/btmp` の内容を出力します。

`lastlog`： このコマンドは、各ユーザの最終ログイン日時を記録する `/var/log/lastlog` 内のデータを出力します。

`ac`： `acct` パッケージをインストールすると利用できるようになるコマンドです。このコマンドは、ユーザ単位もしくは日単位の接続時間を表示します。このコマンドは `/var/log/wtmp` を読み込んで処理します。

`dump-utmp`： `/var/run/utmp` もしくは `/var/log/wtmp` の内容を読み込んで、これをテキストとして処理可能な形式に変換します。

そのほかにも `/var/log/messages` や `journalctl` コマンドの出力を確認してもかまいません。前者は何らかのログ記録 (syslog) アプリケーションが動作している場合、後者は動作していない場合に使用します。`journalctl` で使用する `systemd` のジャーナル機能については、**『リファレンス』、第11章「`journalctl`：systemd ジャーナルへの問い合わせコマンド」**をお読みください。

16 cron と at の制限

改訂履歴

2024-05-10

本章では、システムのセキュリティを改善するため、`cron` や `at` デーモンへのアクセスを制限する方法について説明しています。

16.1 cron デーモンの制限

`cron` は指定した時間に裏でコマンドを自動的に実行させるためのデーモンです。`cron` そのものに関する詳細は、**『リファレンス』、第15章「特殊なシステム機能」、15.1.2項「cron パッケージ」**をお読みください。

`cron.allow` ファイルは `cron` を利用してジョブを実行することのできるユーザを指定するためのファイルです。このファイルは既定では存在していないため、`cron.deny` ファイルに記載されている場合を除いて、全てのユーザがジョブを作成できるようになっています。

`root` 以外のユーザが `cron` ジョブを作成できないようにするには、下記の手順を実施します。

1. まずは空の `/etc/cron.allow` ファイルを作成します:

```
tux > sudo touch /etc/cron.allow
```

2. `cron` ジョブの作成を許可するユーザ名を、下記のようにして追加していきます:

```
tux > sudo echo "tux" >> /etc/cron.allow
```

3. 動作を確認したい場合は、root 以外でかつ `cron.allow` 内に追加したユーザになってから `cron` ジョブを作成してみます。下記のようなメッセージが (一瞬だけですが) 表示されるはずです:

```
tux > crontab -e
no crontab for tux - using an empty one
```

crontab のエディタを終了し、今度は上記のファイル内に追加していないユーザで同じことを実施します (もちろんユーザを追加する前に行ってもかまいません):

```
wilber > crontab -e
You (wilber) are not allowed to use this program (crontab)
See crontab(1) for more information
```

重要: 既存の cron ジョブについて

`cron.allow` はユーザに対して、新しい `cron` ジョブの作成を許可するための設定です。ユーザが `cron.deny` 内に登録されていた場合でも、既にそのユーザのジョブが存在していれば、それらは実行されてしまいます。既存のジョブについても禁止したい場合は、上述の手順でファイルを作成したあと、`/var/spool/cron/tabs` ディレクトリ内にあるユーザの crontab ファイルを削除してください。

注記: systemd タイマーユニットへの切り替えについて

`systemd` タイマーユニットへの切り替えについても検討しておくことをお勧めします。`systemd` タイマーユニットではさらに高度な設定が可能になっているほか、タスクの実行に対しても信頼性が高められています。既定では、一般ユーザはログインしていない限り、コードを実行することができないように設定されています。これにより、ユーザが接続していない場合のシステムとの対話を制限しています。

`systemd` のタイマーユニットに関する詳細は、『リファレンス』、第10章「systemd デーモン」、10.7項「systemd タイマーユニット」をお読みください。

16.2 at スケジューラの制限

`at` ジョブ実行システムは、ユーザに対して 1 回限りのジョブ実行の機能を提供します。`at.allow` ファイルは `at` を利用してジョブを実行することのできるユーザリストを指定するためのファイルです。このファイルは既定では存在していないため、`at.deny` ファイルに記載されている場合を除き、全てのユーザがジョブを作成できるようになっています。

`at` ジョブを `root` 以外のユーザが利用できないようにするには、下記の手順を実施します。

1. まずは空の `/etc/at.allow` ファイルを作成します:

```
tux > sudo touch /etc/at.allow
```

2. `at` ジョブを作成できるユーザのユーザ名を、下記のようにして追加していきます:

```
tux > sudo echo "tux" >> /etc/at.allow
```

3. 動作を確認したい場合は、root 以外でかつ `at.allow` 内に追加したユーザになってからジョブを作成してみます。下記のようなメッセージが表示されるはずです:

```
tux > at 00:00
```

```
at>
```

`Ctrl` – `C` を押して `at` プロンプトを抜け、今度は上記のファイル内に追加していないユーザで同じことを実施します (もちろんユーザを追加する前に行ってもかまいません):

```
wilber > at 00:00
You do not have permission to use at.
```



注記: at のアンインストールについて

`at` は現在、一般的には使用されていません。明確な用途がある場合を除き、使用を制限するのではなくデーモンをアンインストールしておくことをお勧めします。

17 Spectre/Meltdown チェッカー

改訂履歴

2024-06-27

`spectre-meltdown-checker` は、直近の 20 年間に製造されたほぼ全ての CPU 内に存在する投機的実行機能に関する脆弱性が、お使いのシステムに存在しているかどうかをテストするためのスクリプトです。この脆弱性はハードウエアの欠陥に起因するものであり、攻撃者はこの脆弱性を悪用することで、システム内に存在する全てのデータを読み取ることができる可能性があります。クラウド型のコンピューティングシステムの場合、1 台の物理ホスト内に複数の仮想マシンが動作する仕組みであることから、この脆弱性を悪用できてしまうと、全ての仮想マシン内のデータを読み取ることができる可能性があることになります。この脆弱性を根本から解決するには、CPU の再設計と入れ替えが必要になってしまいますが、それができるまでの間の回避策として、これらの脆弱性を緩和するためのソフトウエア修正が提供されています。お使いの SUSE システムを最新の状態に維持しておくことで、これらの修正は全てインストールされることになります。

`spectre-meltdown-checker` は詳細なレポートを生成します。お使いのシステムに脆弱性が存在しないことを保証する仕組みではありませんが、緩和策が正しく適用され、潜在的な脆弱性が存在していないことを確認することができます。

17.1 `spectre-meltdown-checker` の使用

スクリプトをインストールして、オプションを指定せずに root で実行します:

```
# zypper in spectre-meltdown-checker
# spectre-meltdown-checker.sh
```

すると、図17.1「spectre-meltdown-checker の出力」に示しているようなカラフルな出力が現れるはずです:

```
dreamer:/home/carla # spectre-meltdown-checker.sh
Spectre and Meltdown mitigation detection tool v0.40

Checking for vulnerabilities on current system
Kernel is Linux 4.12.14-lp151.28.13-default #1 SMP Wed Aug 7 07:20:16 UTC 2019 (0c09ad2) x86_64
CPU is Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz

Hardware check
* Hardware support (CPU microcode) for mitigation techniques
  * Indirect Branch Restricted Speculation (IBRS)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates IBRS capability: YES (SPEC_CTRL feature bit)
  * Indirect Branch Prediction Barrier (IBPB)
    * PRED_CMD MSR is available: YES
    * CPU indicates IBPB capability: YES (SPEC_CTRL feature bit)
  * Single Thread Indirect Branch Predictors (STIBP)
    * SPEC_CTRL MSR is available: YES
    * CPU indicates STIBP capability: YES (Intel STIBP feature bit)
  * Speculative Store Bypass Disable (SSBD)
    * CPU indicates SSBD capability: YES (Intel SSBD)
  * L1 data cache invalidation
    * FLUSH_CMD MSR is available: YES
    * CPU indicates L1D flush capability: YES (L1D flush feature bit)
  * Enhanced IBRS (IBRS_ALL)
    * CPU indicates ARCH_CAPABILITIES MSR availability: NO
    * ARCH_CAPABILITIES MSR advertises IBRS_ALL capability: NO
```

図 17.1: SPECTRE-MELTDOWN-CHECKER の出力

`spectre-meltdown-checker.sh --help` のように入力して実行すると、全てのオプションを表示します。また、色づけせずにテキストのみで出力したい場合、たとえばファイルに書き込みたいような場合は、下記のように実行します:

```
# spectre-meltdown-checker.sh --no-color| tee filename.txt
```

上記の例では動作中のシステムで実行してきました (これが既定の動作となります) 。この方法以外にも、`spectre-meltdown-checker` スクリプトに対してカーネルや設定、System.map をそれぞれ指定することで、オフラインで実行することもできます:

```
# cd /boot
# spectre-meltdown-checker.sh \
--no-color \
--kernel vmlinuz-6.4.0-150600.9-default \
--config config-6.4.0-150600.9-default \
--map System.map-6.4.0-150600.9-default| tee filename.txt
```

その他の便利なオプションは下記のとおりです:

--verbose, -v

冗長性を増します。`-v -v -v` のように繰り返し指定すると、さらに詳しい出力を行うようになります。

--explain
: より分かりやすい説明を出力します

--batch [short] [json] [nrpe] [prometheus]
: スクリプトなど機械処理向けの出力形式を指定します

重要: --disclaimer オプションについて

`spectre-meltdown-checker.sh --disclaimer` のように入力して実行すると、スクリプトの動作範囲などの重要な情報を表示します。

17.2 さらなる情報

さらに詳しい情報については、それぞれ下記をお読みください:

- SUSE ナレッジベース記事 #7022937: Security Vulnerability: Spectre Variant 4 (Speculative Store Bypass) aka CVE-2018-3639 (セキュリティ脆弱性: CVE-2018-3639 としても知られている Spectre Variant 4 (投機的ストアバイパス) について): https://www.suse.com/support/kb/doc/?id=7022937
- GitHub 内で公開されている speed47/spectre-meltdown-checker スクリプト、および対応する Common Vulnerabilities and Exposures (CVE) へのリンク: https://github.com/speed47/spectre-meltdown-checker
- SUSE ブログ記事: Meltdown and Spectre Performance (Meltdown および Spectre と性能について): https://www.suse.com/c/meltdown-spectre-performance/
- SUSE ナレッジベース記事 #7022512: アーキテクチャや CVE, 緩和策などの情報: https://www.suse.com/support/kb/doc/?id=7022512

18 YaST を利用したセキュリティの設定

改訂履歴

2023-06-06

YaST のモジュールである［セキュリティセンター］を利用することで、openSUSE Leap に対するセキュリティ関連の設定を一括で変更することができます。ログイン処理やパスワードの生成に関する設定のほか、起動の許可に関する設定やユーザの作成、既定のファイルアクセス許可などの設定が含まれています。このモジュールは、YaST コントロールセンターから［セキュリティとユーザ］›［セキュリティセンター］を選択することで、起動を行うことができます。起動を行うと、［セキュリティセンター］ダイアログは、［セキュリティの概要］が選択された状態で始まりますが、それ以外の設定分野については、左側のペインで選択して表示させてください。

18.1 ［セキュリティの概要］

［セキュリティの概要］には、お使いのシステムで最も重要なセキュリティ設定について、その一覧が表示されています。一覧内の各項目には、現時点でのセキュリティ状態が表示され、緑色のチェックマークであれば安全な設定が適用されていることを、赤色のバツ印であれば危険な設定であることを示しています。設定に関する概要のほか、安全にする方法を知りたい場合は、それぞれの項目内にある［ヘルプ］を押してください。設定を変更したい場合は、状態の列にある各リンクを押してください。設定の項目により、下記のような表示が現れます:

［有効］/［無効］
: このリンクを押すことで、有効／無効を直接切り替えることができます。

［設定］
: このリンクを押すと、別の YaST モジュールを起動します。起動したモジュール側で設定を行ったあと、このモジュールに戻って再確認してください。

［不明］
: 対応するサービスがインストールされていないため、設定の状態が不明であることを表しています。明確なセキュリティリスクを表しているものではありません。



図 18.1: YAST セキュリティセンターとセキュリティの強化: セキュリティの概要

18.2 ［事前定義済みのセキュリティ設定］

openSUSE Leap には 3 種類の［事前定義済みのセキュリティ設定］が用意されています。これらの設定は、［セキュリティセンター］モジュール内の全ての設定項目に対して影響を及ぼします。左側のペインで［事前定義済みのセキュリティ設定］を選択して画面を表示させ、適用したいものを選んで閉じてください。ここからさらに細かく設定を行いたい場合は、再度［セキュリティセンター］モジュールを開いて、［事前定義済みのセキュリティ設定］を選択したあと、右側のペインで［カスタム設定］を選択してください。このようにすることで、事前定義済みの設定からさらに細かい調整を施すことができます。

［ワークステーション］
: 任意のネットワーク接続 (インターネット接続を含む) を利用する、ワークステーション向けの設定です。

［ローミングデバイス］
: 様々なネットワークに接続するラップトップやタブレット向けの設定です。

［ネットワークサーバ］
: Web サーバやファイルサーバ、ネームサーバなど、ネットワークサービスを提供するマシン向けのセキュリティ設定です。この設定を選択することで、最大限に安全を確保した設定になります。

［カスタム設定］

3 種類の事前定義済みの設定を適用したあと細かい調整を行いたい場合は、［カスタム設定］を選んでください。

18.3 ［パスワード設定］

容易に推測できるパスワードを使用してしまうと、大きなセキュリティ問題となります。［パスワード設定］のダイアログでは、安全なパスワードのみを使用するように設定することができます。

［新しいパスワードのチェック］

このチェックボックスにチェックを入れると、辞書内に存在するような単語や固有名詞などが存在した場合、警告メッセージを表示するようになります。

［最小パスワード文字数］

ユーザがここで指定した長さよりも短いパスワードを設定しようとすると、警告メッセージを表示します。

［記録するパスワードの数］

パスワードの有効期限の設定 (［パスワード有効日数］) が有効化されている場合、ここで指定した数だけ過去のパスワードを保存することができるようになります。これにより、パスワードの再利用を防ぐことができます。

［パスワードの暗号化方法］

パスワードの暗号化方法を選択します。通常は既定値 (SHA-512) のままでかまいません。

［パスワード有効日数］

最小値と最大値をそれぞれ日数で指定することで、パスワードの有効期間を設定することができます。最小日数を `0` 日以上に設定すると、いったんパスワードを変更したあとは、すぐにパスワードを変更できなくなります (これにより、パスワードの有効期間の制限を迂回できなくすることができます) 。なお、最小値と最大値を `0` と `99999` に設定することで、有効日数の制限を無効化することができます。

［パスワード失効警告日数］

パスワードの有効期間が設定されている場合、事前にユーザに対して警告を表示することができます。ここでは、警告メッセージを表示する日数 (実際に有効期間が切れる前の日数) を指定します。

18.4 ［起動設定］

このダイアログでは、グラフィカルなログインマネージャでどのユーザがシャットダウンできるようにするのかを設定します。このほか、 Ctrl – Alt – Del の解釈や休止状態を誰に許可するのかなども設定することができます。

18.5 ［ログイン設定］

このダイアログでは、ログインに関わるセキュリティ設定を行うことができます:

［ログイン失敗時の待機時間］
: 繰り返しログインを試してみることで、ユーザのパスワードを推測できることがないよう、パスワードの入力を誤った場合には、ログインプロンプトの表示を遅らせるように設定しておくことをお勧めします。ここでは秒単位で値を設定しますが、あまりにも大きな値を設定してしまうと、パスワードの入力を誤った際に長い時間待たされることにもなってしまいます。

［リモートのグラフィカルログインを許可］
: この項目にチェックを入れると、グラフィカルなログインマネージャ (GDM) がネットワーク経由でアクセスできるようになります。これは潜在的なセキュリティリスクとなります。

18.6 ［ユーザ追加］

ユーザ ID やグループ ID の最小値と最大値を設定します。ほとんどの場合、既定値を変更する必要はありません。

18.7 ［その他の設定］

ここには、上述の分類に当てはまらなかったその他のセキュリティ設定が用意されています:

［ファイルのアクセス権］
: openSUSE Leap には 3 種類のファイルアクセス権セットが用意されています。これらのアクセス権は、一般ユーザがログファイルを読めるかどうかや、特定のプログラムを起動できるかどうかに影響するものです。［簡易］はスタンドアロンのマシンに最適な設定で、たとえば一般ユーザからほとんどのシステムファイルを読み込むことができるようになります。詳しい設定内容については、 /etc/permissions.easy をご覧ください。［厳格］はネットワーク経由でアクセスされる

マルチユーザ型のマシンに適切な設定で、詳細は `/etc/permissions.secure` に書かれています。[偏執] は最も制限の厳しい設定で、注意して使用しなければならないものです。詳しくは `/etc/permissions.paranoid` をご覧ください。

[updatedb を実行するユーザ]

`updatedb` プログラムはシステム内を検索するプログラムで、あとから `locate` コマンドで問い合わせることができるよう、全てのファイルの場所に関するデータベースを作成します。`updatedb` コマンドを nobody で実行すると、全てのユーザから読み込むことのできるファイルのみをデータベースに書き込むようになります。このプログラムを `root` で実行すると、ほぼ全てのファイル (ただし `root` から読み込むことができないものは除きます) を追加することになります。

[マジック SysRq キー]

マジック SysRq キーは、システムがクラッシュした場合などにいくつかの操作を行うことができる、キーの入力の組み合わせです。詳しい説明については、https://www.kernel.org/doc/html/latest/admin-guide/sysrq.html (英語) もしくは https://ja.wikipedia.org/wiki/%E3%83%9E%E3%82%B8%E3%83%83%E3%82%AFSysRq%E3%82%AD%E3%83%BC をお読みください。

19 Polkit 認可フレームワーク

改訂履歴

2024-06-26

Polkit はグラフィカルな Linux デスクトップ環境で使用される認可フレームワークで、システムでの権限管理をより精密に行うための仕組みです。従来の Linux システムでは、最高位のシステム権限を持つユーザとして `root` を規定し、それ以外のユーザやグループと分離していましたが、サウンドハードウエアにアクセスするための `audio` グループなどのように、`root` とは別の権限構成も存在していました。このような構成では、特定の状態にある場合にのみ許可したり、一時的に許可したりするようなことが実現できませんでした。

このような場合、通常は `sudo` コマンドなどで `root` に成り代わり、特権を得て作業を行うことになりますが、Polkit では要件に応じて特定のユーザやグループに対して、必要な権限のみを与えることができます。権限は処理と権限内容を記した設定ファイル内で動的に対応することができます。

19.1 考え方の概要

Polkit は複数のコンポーネント (部品) から構成されています。まず `polkitd` は裏で動作する特権サービスで、Polkit の設定ファイルを読み込んで必要な認証チェックを実施します。Polkit 対応のアプリケーションは認証要求を `polkitd` に送信しますが、このとき特権を持たない Polkit 認証エージェントが動作して、`polkitd` が求める認証要求を表示します。ここでユーザが対話的に入力した値は、エージェントを介して `polkitd` に送信されます。

Polkit での アクション は、Polkit の認可ルールに記された 1 つの処理を意味します。このアクションはたとえば、コンピュータを再起動したいなどの意図を表します。それぞれのアクションには唯一の ID が設定され、たとえばコンピュータの再起動であれば `org.freedesktop.login1.reboot` という名称が割り当てられています。

19.1.1 認証エージェント

ユーザが完全機能のデスクトップ環境内でグラフィカルなセッションを開始すると、認可エージェントが裏で自動的に起動されます。このエージェントは何らかの認証が必要となった場合にのみ表に現れるだけで、それ以外の状況では表には全く現れません。なお、テキストモードで起動した場合や SSH 経由でログインした場合は自動で起動されませんので、以降はグラフィカルなセッションを想定して記しています。

19.1.2 Polkit の設定

Polkit の設定は、アクション と 認可ルール を元にして作られています:

アクション (ファイルの拡張子 `*.policy`)
: XML ファイル形式で記述され、`/usr/share/polkit-1/actions` 内に配置されます。それぞれのファイルは 1 つ以上のアクションを定義していて、それぞれのアクションには説明と既定のアクセス許可が設定されます。システム管理者側では独自のルールを作成することができますが、Polkit 側が提供するファイルについては、編集してはなりません。

認可ルール (ファイルの拡張子 `*.rules`)
: ルールは JavaScript プログラミング言語で記述され、2 箇所に配置されます。システムが提供するパッケージの場合は `/usr/share/polkit-1/rules.d` に、独自に作成するファイルの場合は `/etc/polkit-1/rules.d` にそれぞれ配置します。それぞれのルールファイルには既定の認可設定のほか、複雑なロジックで権限を判断するようなことができます。たとえばルールファイルでは制限の厳しいアクセス許可を上書きすることができるほか、特定のユーザには認証 無しで実行できるようにしたりすることができます。

19.1.3 Polkit のユーティリティ

Polkit には、特定の作業を行うためのコマンドが用意されています (より詳しい説明については、それぞれのマニュアルページをお読みください):

`pkaction`
: 設定済みのアクションに対して詳細を取得します。詳しくは 19.3項「権限の問い合わせ」をお読みください。

`pkcheck`
: 特定の Polkit のアクションに対して、プロセスが認可されているかどうかを確認します。

`pkexec`
: Polkit の認可設定に従って、異なるユーザでプログラムを実行できるようにします。`su` や `sudo` に似た仕組みです。

`pkttyagent`
: テキスト型の認証エージェントを起動します。このエージェントは、デスクトップ環境側に独自の認証エージェントが存在しない場合に使用されます。

19.2 認可の種類

Polkit を有効化したプロセスが権限操作を行うと、このプロセスに対して権限が設定されているかどうかを Polkit に尋ねる処理を行います。Polkit はそのプロセスに対して設定されているポリシーを元にして、応答を返します。応答は `yes` (はい), `no` (いいえ), `authentication needed` (要認証) のいずれかとなります。`authentication needed` (要認証) の場合はユーザに対して認証ダイアログを表示し、必要な認可を得る処理を行います。

19.2.1 暗黙の認可

特定のアクションに対して専用の Polkit JavaScript ルールが記述されていない場合、その結果は、Polkit のポリシーファイルの各アクションに対して設定されている暗黙の認可設定に従って決定されます。この暗黙の認可設定には `allow_active` , `allow_inactive` , `allow_any` の 3 種類があります。`allow_active` は動作中のセッション内にいるユーザに適用されるもので、テキストモードやグラフィカルユーザインターフェイスでログインしているユーザに適用されます。なお、ログイン中であっても、ユーザがコンソールを切り替えるなどして動作中でなくなると、`allow_inactive` が適用されます。また、`allow_any` は SSH や VNC などでログインしているリモートのユーザなど、それ以外の状況下で適用されます。これらそれぞれに対して、下記に示すいずれかの認可を割り当てることができます:

no
: そのアクションに対して決して認可を与えないようにします。

yes
: 認証情報を入力することなく、そのアクションに対して常に認可を与えるようにします。

auth_self
: 権限を要求した際、自分自身のパスワードを入力させるようにします。

auth_self_keep
: `auth_self` に似ていますが、認証情報を一定時間だけ保持します。つまり、同じアプリケーションから繰り返し同じアクションを実行した場合、パスワードの入力を求めなくなります。

auth_admin
: 権限を要求した際、管理者 (`root`) のパスワードを入力させるようにします。

auth_admin_keep
: `auth_self_keep` と同様に、管理者 (`root`) のパスワードを一定時間だけ保持します。

19.2.2 SUSE 既定の権限

ここまでに説明してきた Polkit のポリシーファイル内の暗黙の認可設定は、アプリケーションの開発者が規定した既定の暗黙ポリシーセットが元になっています。このようなポリシーは「提供元の既定値」と呼ばれます。提供元の既定値で設定された権限は、既定の SUSE システムで有効化されていない場合もあります。openSUSE Leap では、提供元の既定値を上書きする独自の権限セットを用意していますが、これには 3 種類のセットが含まれています:

`/etc/polkit-default-privs.easy`
: 利用者が管理者自身であり、それ以外には誰も使用しないような、単一のデスクトップシステム向けの権限セットです。使いやすさを重視した設定であるため、セキュリティという観点では弱い設定です。

`/etc/polkit-default-privs.standard`
: ほとんどのシステムに対して適用できる均衡の取れた権限セットです。

`/etc/polkit-default-privs.restrictive`
: 使いやすさを犠牲にして、考えられる攻撃手法をできる限り無効化することを目指した、より保守的な権限セットです。

既定の権限を切り替えたい場合は、`/etc/sysconfig/security` ファイル内にある `POLKIT_DEFAULT_PRIVS` の値を、`easy`, `standard`, `restrictive` のいずれかに設定してください。設定が終わったら、`root` で `set_polkit_default_privs` を実行します。

上述のファイルについては変更してはなりません。独自の権限セットを構築したい場合は、`/etc/polkit-default-privs.local` で設定してください。詳しくは 19.4.3項「SUSE 既定の権限設定の変更」をお読みください。

19.3 権限の問い合わせ

権限を問い合わせたい場合は、Polkit に含まれている `pkaction` コマンドを使用します。

Polkit には権限を変更したり、他のユーザでコマンドを実行したりするためのコマンドラインツールが付属しています (概要は 19.1.3項「Polkit のユーティリティ」で説明しています) 。またそれぞれの既存のポリシーには、識別のための名前が設定されています。利用可能なポリシーを一覧で表示したい場合は、`pkaction` コマンドをお使いください。詳しくは `man pkaction` をお読みください。

特定のポリシー (例: `org.freedesktop.login1.reboot`) に対して、必要な認可情報を表示したい場合は、`pkaction` を下記のように実行します:

```
> pkaction -v --action-id=org.freedesktop.login1.reboot
org.freedesktop.login1.reboot:
```

```
  description:       Reboot the system
  message:           Authentication is required to allow rebooting the system
  vendor:            The systemd Project
  vendor_url:        http://www.freedesktop.org/wiki/Software/systemd
  icon:
  implicit any:      auth_admin_keep
  implicit inactive: auth_admin_keep
  implicit active:   yes
```



注記: openSUSE Leap における `pkaction` の制限事項について

`pkaction` は提供元の既定値に対してのみ作用するもので、提供元の既定値を上書きする SUSE の既定値に対しては作用しないことに注意してください。そのため、出力結果は注意して解釈する必要があります。

19.4 Polkit の設定変更

たとえば特定のチーム内に配布するコンピュータをセットアップするような場合など、複数のマシンに同じポリシーセットを設定したいような場合、Polkit の設定を修正する必要があります。このほか、Polkit の認可設定をカスタマイズすることで、特定のアクションに対するセキュリティを強化したり、よく使用されるアクションに対して繰り返しのパスワード入力を省略したりなどを行うこともできます。ただし、認証無しで Polkit のアクションを不用意に実行できるように設定してしまうと、設定によっては完全な `root` 権限を得る結果になってしまうこともあります。そのため、Polkit で認証設定を弱くする場合は、お使いの環境下でシステムのセキュリティを侵さない範囲でのみ設定するようにしてください。

19.4.1 Polkit ポリシーファイルの上書き

お使いのシステムでどのようなアクションが利用できるのかは、どのパッケージをインストールしているのかによって決まります。概要を表示したい場合は、`pkaction` を実行して、定義済みの全ルールを一覧で表示させてください。

下記では、例として `gparted` コマンド (「GNOME パーティションエディタ」) における Polkit との連携例を示しています。

`/usr/share/polkit-1/actions/org.opensuse.policykit.gparted.policy` ファイルには、下記のような内容が書かれています:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE policyconfig PUBLIC
 "-//freedesktop//DTD PolicyKit Policy Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/PolicyKit/1.0/policyconfig.dtd">
```

```
<policyconfig> ❶

  <action id="org-opensuse-polkit-gparted"> ❷
    <message>Authentication is required to run the GParted Partition Editor</message>
    <icon_name>gparted</icon_name>
    <defaults> ❸
      <allow_any>auth_admin</allow_any>
      <allow_inactive>auth_admin</allow_inactive>
     < allow_active>auth_admin</allow_active>
    </defaults>
    <annotate ❹
      key="org.freedesktop.policykit.exec.path">/usr/sbin/gparted</annotate>
    <annotate ❹
      key="org.freedesktop.policykit.exec.allow_gui">true</annotate>
  </action>

</policyconfig>
```

❶ ポリシーファイルの XML ルート要素です。

❷ このポリシー内で唯一設定されたアクションの開始位置です。

❸ ここには暗黙の認可が設定されています。

❹ `annotate` 要素には、Polkit がアクションを実行する際の方法に関わる設定が書かれています。この例では、gparted 実行ファイルのパスのほか、グラフィカルな表示を許可するための設定が書かれています。この要素は Polkit のツールである `pkexec` を使用する際に必要な設定です。

独自のポリシーを追加したい場合は、上記のような構造で `.policy` ファイルを作成し、`id` 属性に適切な値を設定して、既定のアクセス許可を記述してください。



注記: 以前の名称 PolicyKit について

Polkit 認可フレームワークは以前、PolicyKit という名称で呼ばれていました。このことから、XML 文書の属性値などには古い名称が残っている場合があります。

19.4.2 JavaScript 認可ルールの追加

独自の認可ルールを作成することで、暗黙の認可を上書きして動作させることができます。独自のルールを作成する場合は、`/etc/polkit-1/rules.d/` ディレクトリ内にファイルを作成してください。

このディレクトリにファイルを配置する場合、ファイル名は 2 桁の数字ではじめ、ハイフンで区切ってその後ろにわかりやすい名前を続けて、末尾は `.rules` で終わらせるようにしてください。これらのファイル内の関数は、並べ替えられた順に実行されます。たとえば `00-foo.rules` ファイルは `60-bar.rules` や `90-default-privs.rules` ファイルの前に実行されます。

ルールファイル内はスクリプト形式で、まずアクション ID (`.policy` ファイルで定義されている値) が特定のものであるかどうかをチェックします。たとえば `gparted` というコマンドを `admin` グループ内の任意のユーザに対して許可したい場合は、`org.opensuse.policykit.gparted` のアクション ID に対して、下記のような処理を記述します:

```
/* Allow users in admin group to run GParted without authentication */
polkit.addRule(function(action, subject) {
    if (action.id == "org.opensuse.policykit.gparted" &&
        subject.isInGroup("admin")) {
        return polkit.Result.YES;
    }
});
```

Polkit API 内で提供されている全てのクラスとメソッドについて、詳しくは https://www.freedesktop.org/software/polkit/docs/latest/ref-api.html をお読みください。

19.4.3 SUSE 既定の権限設定の変更

19.2.2項「SUSE 既定の権限」で説明しているとおり、SUSE では、提供元の開発者が設定した Polkit の暗黙の認可設定を上書きする設定を提供しています。それ以外の独自の権限については、`/etc/polkit-default-privs.local` ファイル内に記述してください。ここで定義された権限は、その他の設定ファイル内に書かれているものよりも常に優先して適用されます。独自の権限セットを定義したい場合は、下記のようにして行います:

手順 19.1: 既定の権限設定の変更

1. `/etc/polkit-default-privs.local` ファイルを編集します。権限を定義するには、下記のような形式で各アクション向けの行を記述します:

   ```
   <アクション-ID>      <任意のセッションに対する認可指定>:<非動作中のセッションに対する認可指定>:<動作中のセッションに対する認可指定>
   ```

   3 種類全てのカテゴリに対して同じ設定を適用したい場合は、下記のように 1 つだけ値を指定してもかまいません:

   ```
   <アクション-ID>      <全てのセッションに対する認可指定>
   ```

   たとえば、下記のように指定します:

   ```
   org.freedesktop.color-manager.modify-profile      auth_admin_keep
   ```

2. あとは設定を反映させるため、`root` で下記を実行します:

   ```
   # /sbin/set_polkit_default_privs
   ```

SUSE での Polkit 既定値に関する詳しいドキュメンテーションについては、`man polkit-default-privs` をお読みください。

19.5 SUSE 既定の権限設定の復元

認可設定を SUSE 提供の既定値に戻したい場合は、下記の手順を実施します:

手順 19.2: OPENSUSE LEAP の既定値の復元

1. まずは 19.2.2項「SUSE 既定の権限」で説明している設定値のいずれかを選択します。
2. `/etc/polkit-default-privs.local` に記述している全ての上書き設定を削除します。
3. あとは `set_polkit_default_privs` を実行すると、既定値に戻すことができます。

20 Linux でのアクセス制御リスト

改訂履歴

2023-06-06

POSIX ACL (アクセス制御リスト) は、ファイルシステム内のオブジェクト (ファイルやディレクトリなど) に対して設定することのできる、従来型のパーミッション (アクセス許可) の拡張です。ACL を使用することで、従来型のパーミッションよりも柔軟な構成をとることができるようになります。

POSIX ACL という名称からもおわかりのとおり、こちらは正式な POSIX (Portable Operating System Interface) 標準として規定されています。関連するドラフトは POSIX 1003.1e と POSIX 1003.2c で、いずれも諸事情により撤回されています。しかしながら、Unix ファミリに属する多くのオペレーティングシステムと同様に、ACL はこれらのドラフトをベースにしているほか、ファイルシステム側の ACL (本章で説明しています) 実装は、これら 2 つの標準に従って行われています。

20.1 従来型のファイルアクセス権

openSUSE Leap で提供されている全てのファイルに対するアクセス権 (パーミッション) は、よく考えられて設定されています。そのため、追加のソフトウエアやファイルをインストールする場合は、パーミッションの設定に注意してください。`ls` コマンドに `-l` オプションを指定することで、正しく設定されていないパーミッションを即時に見つけることができます。なお、パーミッションの設定は、ファイルの内容の変更や削除に影響するだけではありません。`root` が実行するプログラム自身を書き換えることもできてしまいますし、設定を書き換えて不正な動作を行わせることもできてしまいます。このように、パーミッションの設定は攻撃の可能性を増やすことにも繋がります。

openSUSE® Leap システム内の `/etc` ディレクトリには、`permissions` , `permissions.easy` , `permissions.secure` , `permissions.paranoid` の各ファイルが用意されています。これらのファイルは、全てのユーザが書き込みできるディレクトリやファイル、setuser ID (setuid) ビットなどの特殊なパーミッションを定義するためのファイルです。setuser ID ビットが設定されたプログラムは、そのプログラムを起動したユーザの権限で動作することはなく、ファイルの所有者 (通常は `root`) の権限で動作します。管理者側では、`/etc/permissions.local` というファイルを利用することで、独自の設定を行うことができます。

利用可能なプロファイルからいずれかを選択したい場合は、YaST の［セキュリティとユーザ］内にある［セキュリティセンターとセキュリティの強化］をお使いください。また、パーミッションに関する詳細については、`/etc/permissions` 内のコメント (英語) もしくは `man chmod` で表示されるマニュアルページをお読みください。

従来型のファイルパーミッションに関する詳しい説明については、GNU coreutils の info ページ内にある File permissions (`info coreutils "File permissions"`) (英語) の項をお読みください。setuid, setgid, sticky の各ビットに対する説明も用意されています。

20.1.1 setuid ビット

状況によっては、アクセスパーミッションの仕組みでは制限が大きすぎることがあります。そのため Linux には、特定の処理に対して現在のユーザやグループを一時的に変更することのできる機能が用意されています。たとえば `passwd` プログラムは `/etc/passwd` ファイルにアクセスすることから、通常であれば root の権限が必要となります。ただし、このファイルには各ユーザのホームディレクトリのほか、ユーザやグループの ID などの重要な情報が含まれています。このような構造から、一般ユーザに `passwd` ファイルへの書き込み許可を与えてしまうと、自分以外の情報も変更できてしまいますので、実質的には管理者権限を与えるのと同じになってしまい、セキュリティの意味がなくなってしまいます。このような状況下でも、自分自身のパスワードだけを書き換えることができるようにする必要があることから、setuid という仕組みが考えられました。setuid (set user ID) は特殊なファイル属性で、そのファイルの所有者に成り代わってプログラムを実行することができます。たとえば `passwd` コマンドは下記のようになっています:

```
-rwsr-xr-x  1 root shadow 80036 2004-10-02 11:08 /usr/bin/passwd
```

ユーザパーミッションの欄に `s` が書かれていることで、setuid ビットが設定されていることが分かります。setuid ビットの意味により、`passwd` コマンドは `root` の権限下で動作することになります。

20.1.2 setgid ビット

setuid ビットはユーザに対して適用される仕組みです。それと同様に、グループに対する仕組みも用意されています。これを setgid ビットと呼びます。このビットが設定されたプログラムは、どのユーザから実行された場合であっても、そのプログラムに設定されたグループ ID で実行されることになります。setgid ビットが設定されたディレクトリ内では、新しく作成されたファイルやサブディレクトリは、そのディレクトリに対して設定されているグループに割り当てられます。たとえば下記のようなディレクトリが存在するものとします:

```
drwxrws--- 2 tux archive 48 Nov 19 17:12  backup
```

グループパーミッションの欄に `s` が書かれていることで、setgid ビットが設定されていることが分かります。このディレクトリの所有者と `archive` グループのメンバーは、パーミッションに書かれているとおり、このディレクトリにアクセスすることができます。このグループのメンバーではないユーザは、対

応するグループに「マッピング」されます。さらに、書き込まれた全てのファイルの実効グループ ID は `archive` に設定されます。たとえば、グループ ID `archive` で実行されたバックアッププログラムは、root の権限を取得することなく、このディレクトリにアクセスすることができます。

20.1.3 sticky ビット

このほか、sticky ビット と呼ばれる仕組みも用意されています。これは実行可能なプログラムに対して設定されているのか、もしくはディレクトリに対して設定されているのかによって、異なる意味を持ちます。プログラムに対して設定されている場合、このプログラムはメモリに読み込まれ、使用されるごとにハードディスクから読み込む処理を行わなくなります。新しいハードウエアであれば、ハードディスクも十分に高速なものであることから、この仕組みはあまり使用されていません。ディレクトリに対して設定されている場合は、異なるユーザが作成したファイルを削除できないようにするフラグとして動作します。一般的には `/tmp` や `/var/tmp` ディレクトリに設定します:

```
drwxrwxrwt 2 root root 1160 2002-11-19 17:15 /tmp
```

20.2 ACL の利点

従来の Linux システムでは、3 種類のパーミッションセット (ファイルの所有者に対するセット／所有グループに対するセット／その他に対するセット) を各オブジェクトに設定していました。各セット内には読み込み (`r`), 書き込み (`w`), 実行 (`x`) のパーミッションが存在し、これらでアクセス許可を構築していました。これに加えて、set user id , set group id , sticky の各ビットが追加されて、従来型のパーミッションを構成していました。このような無駄のない仕組みは、ほとんどの動作環境で十分なものではありましたが、より複雑なシナリオや高度な用途では不足することもあります。このような場合は、システム管理者が様々な回避策を講じることで、従来型のパーミッションの考え方に収まるよう、うまく設計する必要がありました。

ACL (Access Control List) は従来型のファイルパーミションの考え方に対する拡張として使うことのできる仕組みです。この ACL の仕組みでは、所有者や所有グループ以外の任意のユーザやグループに対して、アクセス許可を設定することができるようになっています。ACL は Linux カーネルの機能であり、ext2, ext3, ext4, jfs, xfs などで使用することができます。この仕組みにより、システム管理者が複雑なパーミッションモデルを構築することなく、複雑なシナリオを構成することができるようになります。

ACL の利点は、Windows サーバを Linux サーバで入れ替えたりするような場合に顕著に働きます。移行前に Windows サーバに接続されていた作業端末は、移行が終わった後も引き続き使い続けることができなければなりませんが、Linux では Windows クライアント向けのファイルおよび印刷サービスとして、Samba が提供されています。Samba には ACL へのサポートが提供されているため、

Linux サーバ内でも Windows 側のグラフィカルなインターフェイスを使用した場合 (Windows NT もしくはそれ以降が必要です) でも、ユーザへのアクセス許可を自由に設定できるようになります。また、Samba スイートの一部として提供されている `winbindd` を使用することで、Linux サーバ内には存在していない Windows ドメインのユーザに対して、アクセス許可を設定することもできます。

20.3 定義

ユーザクラス

従来の POSIX 型のパーミッションでは、ユーザを 3 種類の クラス (所有者／所有グループ／その他) に分類して、それぞれに許可を設定していました。クラス内では、それぞれ読み込み (`r`), 書き込み (`w`), 実行 (`x`) の各権限を設定していました。

ACL

全ての種類のファイルシステムオブジェクト (ファイルやディレクトリ) に対するユーザおよびグループへのアクセス許可は、ACL によって決定されます。

既定の ACL

既定の ACL はディレクトリに対してのみ適用されるものです。この既定の ACL は、新しくファイルシステムオブジェクトが作成された際に、親ディレクトリのものを引き継いで設定するために使用されます。

ACL エントリ

それぞれの ACL には 1 つ以上の ACL エントリが含まれます。ACL エントリには種類のほか、そのエントリが参照するユーザやグループの修飾子、そして許可情報が含まれます。また、エントリの種類によっては、ユーザやグループに対する修飾子が未定義になることもあります。

20.4 ACL の処理

表20.1「ACL エントリの種類」には、ACL エントリとして存在しうる 6 つの種類が示されています。それぞれはユーザ自身やユーザのグループを表す仕組みで、所有者 はファイルやディレクトリを所有しているユーザの許可を、所有グループ はファイルを所有しているグループの許可をそれぞれ表しています。スーパーユーザは `chown` や `chgrp` のコマンドで所有者や所有グループを変更することができますが、これらのコマンドで変更を行うと、変更後の所有者や所有グループに対して作用することになります。また、特定ユーザ はエントリ内の修飾子内で指定されているユーザに対する許可を、特定グループ はエントリ内の修飾子内で指定されているグループに対する許可をそれぞれ表します。特定ユーザ および 特定グループ のエントリである場合にのみ、修飾子に値が設定されます。その他 のエントリでは、それ以外の全てのユーザに対する許可を表します。

マスク は特定ユーザや特定グループ、所有グループの各エントリに対して許可された内容のうち、どの項目を無効化して、どの項目を有効化するのかを表します。上述のいずれかのエントリ内で許可されていて、マスクエントリでも有効として設定されている場合、その許可はそのまま許可されます。逆に、マスクエントリ内にのみ存在した場合や、上述のいずれかのエントリ内にのみ存在した許可については、有効化されません。つまり、許可されなくなります。なお、所有者と所有グループに対するエントリは、マスクにかかわらず常に有効となります。詳しい仕組みについては、表20.2「アクセス許可のマスク」をご覧ください。

ACL は 2 種類に分類することができます。1 つは 最小 ACL と呼ばれ、所有者／所有グループ／その他に対するエントリのみを持ちます。これらはファイルやディレクトリに対する従来のパーミッションと同じ意味を持ちます。それに対して 拡張 ACL では、それより多くのエントリを持ちます。拡張 ACL ではマスクエントリを持たなければならないほか、必要に応じて特定ユーザや特定グループのエントリを持ちます。

表 20.1: ACL エントリの種類

タイプ	テキスト形式
所有者	`user::rwx`
特定ユーザ	`user:name:rwx`
所有グループ	`group::rwx`
特定グループ	`group:name:rwx`
マスク	`mask::rwx`
その他	`other::rwx`

表 20.2: アクセス許可のマスク

エントリの種類	テキスト形式	パーミッション
特定ユーザ	`user:geeko:r-x`	`r-x`
マスク	`mask::rw-`	`rw-`
	適用されるパーミッション:	`r--`

20.4.1 ACL エントリとファイルモード許可ビットの関係

図20.1「最小 ACL: ACL エントリと許可ビットとの比較」と 図20.2「拡張 ACL: ACL エントリと許可ビットとの比較」には、最小 ACL と拡張 ACL の例を示しています。この図は 3 つのブロックから構成され、左から順に「ACL エントリの種類指定」,「ACL の例」,「対応する従来型のパーミッションビット」(たとえば `ls -l` で表示されるもの) をそれぞれ示しています。いずれの場合であっても、所有者クラス のビットは ACL の所有者エントリに紐づけられ、その他のクラス はその他の種類のエントリに紐づけられます。ただし、グループクラス の部分については、この 2 つの例で対応付けが異なることに注意してください。

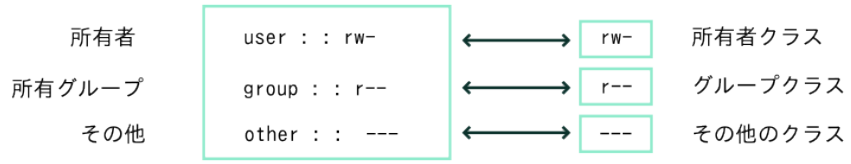


図 20.1: 最小 ACL: ACL エントリと許可ビットとの比較

最小 ACL の (マスクエントリがない) 場合、グループクラスのパーミッションは、図20.1「最小 ACL: ACL エントリと許可ビットとの比較」に示されているとおり、そのまま ACL の所有グループのエントリに対応します。拡張 ACL の (マスクエントリがある) 場合は、図20.2「拡張 ACL: ACL エントリと許可ビットとの比較」に示されているとおり、グループクラスのパーミッションはマスクエントリに対応することになります。

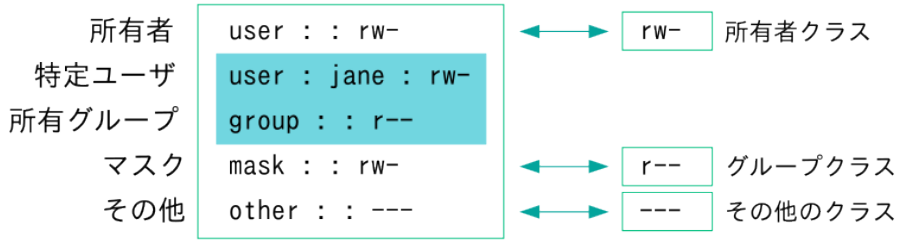


図 20.2: 拡張 ACL: ACL エントリと許可ビットとの比較

このようなマッピングの仕組みが存在することで、アプリケーション側が ACL に対応していない場合でも、問題なく動作できるようになっています。つまり、ACL は従来のパーミッションビットに対する「微調整」として働くことになります。なお、パーミッションビット側を変更することで ACL 側も反映されますし、逆に ACL を変更することでパーミッションビット側にも反映されます。

20.4.2 ACL の設定されたディレクトリ

コマンドラインで `getfacl` と `setfacl` の各ツールを使用することで、ACL を参照したり制御したりすることができるようになります。本章では、これらのコマンドの使用方法について説明しています。

まずはディレクトリを作成する前に、`umask` コマンドを利用してファイルオブジェクトの作成時におけるパーミッションのマスク値を設定してください。`umask 027` のように実行することで、所有者には全ての許可を (`0`)、グループに対しては書き込みビットを禁止し (`2`)、その他に対しては全てを禁止する (`7`) ように設定します。`umask` では、パーミッションのビットのうち、マスク (無効化) したいビットを 1 にします。詳しくは 12.4項「既定の umask」もしくは `umask` のマニュアルページをお読みください。

次に `mkdir mydir` を実行して、`mydir` ディレクトリを `umask` で設定したパーミッションビットで作成します。作成したディレクトリに対して `ls -dl mydir` のように実行し、まずはどのようなパーミッションが設定されているのかを確認します。このコマンドの出力は、たとえば下記のようになっているはずです:

```
drwxr-x--- ... tux project3 ... mydir
```

次に `getfacl mydir` を実行して、ACL の初期状態を確認します。出力は下記のようになっているはずです:

```
# file: mydir
# owner: tux
# group: project3
user::rwx
group::r-x
other::---
```

最初の 3 行には、ディレクトリの名前と所有者、および所有グループに関する情報が表示されています。次の 3 行には、それぞれ所有者と所有グループ、そしてその他に対する ACL エントリが表示されています。この時点では最小 ACL であることから、`getfacl` の出力で得られる情報と `ls` で得られる情報は、書式は違うものの違いがありません。

次に、`geeko` というユーザと `mascots` というグループに対して、読み込み／書き込み／実行の各許可を設定します:

```
# setfacl -m user:geeko:rwx,group:mascots:rwx mydir
```

`setfacl` コマンドで `-m` を指定することで、既存の ACL を修正しています。その後ろのパラメータは修正すべき ACL エントリを表していて、エントリ同士の間はカンマ区切りで指定しています。最後のパラメータは ACL の修正を適用したい先の名前 (ここではディレクトリ) を指定しています。上記のように実行したあとは、`getfacl` コマンドで再度 ACL を表示してみてください。

```
# file: mydir
# owner: tux
# group: project3
user::rwx
user:geeko:rwx
```

```
group::r-x
group:mascots:rwx
mask::rwx
other::---
```

すると、ユーザ `geeko` およびグループ `mascots` に対するエントリと、マスクエントリがそれぞれ追加されていることが分かります。マスクエントリでは、全てのアクセス許可がそのまま動作するように、全ての許可が指定されています。`setfacl` では、`-n` を指定しない限り、マスクエントリを自動調整する処理を行います。マスクエントリは特定ユーザや特定グループ、そして、所有グループに対する最大限の許可を指定する項目です。ここで再度 `ls -dl mydir` を実行してみます。すると、`mask` の設定に対応してパーミッションが表示されるようになります。

```
drwxrwx---+ ... tux project3 ... mydir
```

行頭には従来型のパーミッションビットが表示されていますが、ここに `+` という表示が追加されているのが分かります。これにより、この項目に対して 拡張 ACL が存在していることを表しています。

`ls` の出力にもあるとおり、マスクエントリの許可には書き込み許可が含まれています。従来型のパーミッションの仕組みでは、このビットが設定されていれば、所有グループ (上記の例では `project3`) が、そのディレクトリ (上記の例では `mydir`) に対して書き込み権限があることを表していました。

しかしながら、所有グループに対する実際の許可は、所有グループに対するアクセス許可とマスクエントリの許可に従って動作することになります。この例では `r-x` が適用されます (表20.2「アクセス許可のマスク」をお読みください) 。そのため、この例における所有グループに対する実際の許可については、ACL を追加したあとも特に変わっていないことになります。

さらに `setfacl` や `chmod` でマスクエントリを編集します。たとえば `chmod g-w mydir` のように実行します。こちらを実行後、`ls -dl mydir` を実行すると、下記のような表示になります:

```
drwxr-x---+ ... tux project3 ... mydir
```

`getfacl mydir` の出力は下記のようになります:

```
# file: mydir
# owner: tux
# group: project3
user::rwx
user:geeko:rwx          # effective: r-x
group::r-x
group:mascots:rwx       # effective: r-x
mask::r-x
other::---
```

`chmod` を実行してグループクラスのビットから書き込み権限を取り除くと、`ls` の出力でマスクビットが適切に変更されていることがわかるようになります。これにより、書き込み権限は `mydir` の所有者に限定されることになります。`getfacl` でも同じ内容を確認することができます。出力には上記のような

コメント部分 (#) が含まれるようになり、マスクエントリによって実際に作用するアクセス許可と元々のアクセス許可に差異があることを示すようになります。元のアクセス許可に戻したい場合は、`chmod g+w mydir` を実行してください。

20.4.3 既定の ACL が設定されたディレクトリ

ディレクトリには既定の ACL を設定することができます。既定の ACL は特殊な ACL で、ディレクトリ内に作成されたオブジェクト (サブディレクトリやファイル) に対して適用される、既定のアクセス権として動作します。

20.4.3.1 既定の ACL の効果

ディレクトリに対して設定された既定の ACL が、ファイルやサブディレクトリに適用される場合は、下記の仕組みで動作します:

- サブディレクトリに対しては、親ディレクトリ側での既定の ACL が自分自身の既定の ACL として適用され、自分自身の (通常の) ACL としても適用されます。
- ファイルが作成された場合、既定の ACL を自分自身の ACL として引き継ぎます。

ファイルシステムのオブジェクトを作成する際、全てのシステムコールは `mode` パラメータを使用します。このパラメータは、新しく作成するファイルシステムオブジェクトに対して、アクセス許可を設定するためのものです。親ディレクトリに既定の ACL が設定されていない場合、`mode` パラメータで指定された値から、`umask` で指定されたビットを取り除いてアクセス許可を生成し、新しいオブジェクトに適用します。親ディレクトリに既定の ACL が存在する場合は、既定の ACL と `mode` パラメータとの間で重複するビットのみを新しいオブジェクトに設定します。この場合、`umask` の設定は無視されます。

20.4.3.2 既定の ACL の使用例

下記に示す 3 つの例では、ディレクトリと既定の ACL に対する主な操作を説明しています:

1. 既存のディレクトリである `mydir` に対して、既定の ACL を追加するには、下記のように実行します:

   ```
   > setfacl -d -m group:mascots:r-x mydir
   ```

   `setfacl` コマンドに対して `-d` オプションを指定することで、既定の ACL 内での編集作業 (`-m` オプション) を実施することを表します。

このコマンドを実行すると、下記のような結果になります:

```
> getfacl mydir

# file: mydir
# owner: tux
# group: project3
user::rwx
user:geeko:rwx
group::r-x
group:mascots:rwx
mask::rwx
other::---
default:user::rwx
default:group::r-x
default:group:mascots:r-x
default:mask::r-x
default:other::---
```

`getfacl` は (通常の) ACL と既定の ACL の両方を返します。既定の ACL は `default` で始まる部分です。既定の ACL に対して、`mascots` グループに対するエントリを指定しただけですが、`setfacl` は通常の ACL から既定の ACL をコピーして作成しています。既定の ACL は、アクセスの際の許可に直接影響するものではなく、新しいファイルシステムオブジェクトを作成した場合にのみ影響があります。新しく作成したオブジェクトは、親ディレクトリの既定の ACL のみを引き継ぎます。

2. 次の例では、`mydir` 内に `mkdir` でサブディレクトリを作成しています。これにより、既定の ACL を引き継ぐようになります。

```
> mkdir mydir/mysubdir

getfacl mydir/mysubdir

# file: mydir/mysubdir
# owner: tux
# group: project3
user::rwx
group::r-x
group:mascots:r-x
mask::r-x
other::---
default:user::rwx
default:group::r-x
default:group:mascots:r-x
default:mask::r-x
default:other::---
```

期待のとおり、新しく作成したサブディレクトリである `mysubdir` は、親ディレクトリの既定の ACL を引き継いでいます。`mysubdir` の ACL は、`mydir` の既定の ACL に対する正確なコピーになっています。このディレクトリ内にさらにファイルやディレクトリを作成した場合も、この既定の ACL をコピーして適用することになります。

3. さらに `touch` コマンドを利用して `mydir` ディレクトリ内にファイルを作成します。今回は `touch mydir/myfile` と入力して実行します。`ls -l mydir/myfile` の結果は下記のようになります:

```
-rw-r-----+ ... tux project3 ... mydir/myfile
```

`getfacl mydir/myfile` の結果は下記のようになります:

```
# file: mydir/myfile
# owner: tux
# group: project3
user::rw-
group::r-x          # effective:r--
group:mascots:r-x   # effective:r--
mask::r--
other::---
```

`touch` コマンドは新しいファイルを作成する際、`mode` パラメータに `0666` を指定しますので、`umask` や既定の ACL (詳しくは 20.4.3.1項「既定の ACL の効果」をお読みください) で制限を設定しない限り、全てのユーザクラスに対して読み込みと書き込みを許可する形でファイルを作成します。つまり、`mode` パラメータに含まれていない全てのアクセス権限が、対応する ACL エントリから削除されていることになります。ただし、グループクラスの ACL エントリからはビットを削除せず、マスクエントリを変更して `mode` で指定されていないビットが設定されないようにします。
このような仕組みにより、ACL とアプリケーション (たとえばコンパイラ) を円滑に作用させることができるようになっています。たとえばコンパイラなどでは、不用意に実行されたりしないように、いったん厳しめのアクセス権を設定しておいて、あとから実行可能にすることがあります。`mask` の仕組みにより、必要なユーザやグループに対してだけ、実行できるようにすることができます。

20.4.4 ACL チェックアルゴリズム

ACL で保護されたファイルシステムオブジェクトに対して、プロセスやアプリケーションがアクセスを行う場合、事前にチェックアルゴリズムが適用されます。基本的な考え方としては、所有者エントリ, 特定ユーザエントリ, 所有グループエントリ, 特定グループエントリ, その他の順で確認が行われます。アクセスは、そのプロセスに最も適したエントリに従って処理されます。また、権限では複数のエントリを合成して判断するようなことはおこないません。

プロセスが複数のグループに属していて、複数のグループエントリに該当するような場合は、さらに複雑な処理になります。この場合は、該当する複数のエントリの中からランダムにエントリが選択され、必要なアクセス許可を得ることになります。この処理は、どのエントリが最終的に「アクセスを許可する」という結果を引き出すのかはわかりません。同様に、該当する複数のエントリグループの中のどれであってもアクセスが許可されない場合、ランダムに選択されたエントリから、最終結果である「アクセス拒否」が発行されます。

20.5 アプリケーション内での ACL サポート

ACL は新しいアプリケーションの要件を満たすため、複雑な許可シナリオを実装しています。従来型のパーミッションと ACL の両方の考え方は、賢く組み合わせることができます。基本的なファイルコマンド (`cp` , `mv` , `ls` など) も、Samba, Nautilus と同様に ACL に対応しています。

vi, vim, emacs はいずれも ACL に完全対応していて、バックアップを含め、ファイルへの書き込み時にアクセス許可を保持することができます。ただし、残念なことに多くのエディタやファイルマネージャには、ACL への対応がありません。その他のエディタでファイルを修正した場合は、エディタのバックアップ方式によって、ファイルの ACL が保持される場合と保持されない場合があります。エディタ側が元のファイルに対して書き込みを行うような場合であれば、ACL は保持されるものの、古いほうのファイルをバックアップファイルとしてリネームしてから、新しいファイルに更新後のものを書き込むような場合は、エディタ側が ACL に対応していない限り、ACL が失われてしまうことになります。また、`star` 以外のアーカイバを使用した場合は、ACL を保持したままバックアップを行うことはできません。

20.6 さらなる情報

ACL に関する詳細については、`getfacl(1)` , `acl(5)` , `setfacl(1)` の各マニュアルページをお読みください。

21 AIDE を利用した侵入検知

改訂履歴

2024-08-13

システムの機密を保持することは、代替のきかないシステムの管理者にとっては必須の課題となります。常にシステムが安全な状態にあることは誰も保証できませんので、定期的に (たとえば `cron` などで) 追加のチェックを行い、システムが正しく制御されていることを確認することが重要となります。このような要件を満たすことができるのが AIDE (Advanced Intrusion Detection Environment; 高度な侵入検知環境) です。

21.1 AIDE を使用する理由

RPM を使用すれば、不用意な変更が加わっているのかを簡易的に調べることができます。パッケージマネージャには、システム内に存在し、管理下にあるファイルに対して、チェックを行う機能が用意されています。全てのファイルをチェックしたい場合は、`rpm -Va` コマンドを実行してください。ただし、このコマンドは設定ファイルなどに対しても変更を検知してしまうため、想定通りに変更されているファイルについては、一覧から取り除いていく作業が必要となってしまいます。

また、RPM コマンドを使用して調べる方法には、根本的な問題も存在しています。それは、攻撃者が `rpm` コマンドそのものを書き換えることで、本来は変更されているものとして表示しなければならないものを隠蔽することができてしまう、という問題です。これにより、ルートキットなどの不正なソフトウエアを潜り込ませることができてしまい、不正に root 権限を得ることができてしまいます。この問題を解決するには、インストール済みのシステムからは完全に独立した、もう 1 つのチェックシステムが必要となります。

21.2 AIDE データベースの設定

重要: インストール後の AIDE データベースの準備について

システムをインストールする前に、まずはお使いのメディアのチェックサムを検証し、不正なソフトウエアを使用していないことをご確認ください (詳しくは『**スタートアップ**』、第4章「**トラブルシューティング**」、4.1項「**メディアの確認**」をお読みください) 。システムをインストールしたあとは、AIDE のデータベースの準備を行います。なお、インストール中やインストール後に問題なく

動作するよう、インストールはコンソールから直接実施するものとし、ネットワークには接続しないでください。また、AIDE でデータベースの作成が完了するまで、コンソールを誰にでもアクセスできる状態で放置したり、ネットワークに接続したりしないでください。

AIDE は openSUSE Leap の既定ではインストールされません。インストールを行うには、[Computer] › [Install Software] を使用するか、`root` で `zypper install aide` を実行してインストールしてください。

AIDE がファイルをチェックする際、どのような属性をチェック対象にしたいのかを設定したい場合は、`/etc/aide.conf` 設定ファイルを変更してください。最初のセクションでは、AIDE のデータベースファイルの場所など、一般的なパラメータを設定します。それに続く `Custom Rules` と `Directories and Files` の各セクションが重要な設定となります。既定のルールでは、下記のような値が設定されています:

```
Binlib      = p+i+n+u+g+s+b+m+c+md5+sha1
```

変数 `Binlib` の定義が終わったら、同じセクション内に存在する他の設定も、必要に応じて変更してください。主なオプションには下記のようなものがあります:

表 21.1: 主な AIDE のチェック項目

オプション	説明
p	選択したファイルやディレクトリのパーミッションのチェック。
i	inode 番号のチェック (それぞれのファイルには、必ずユニークな inode 番号が割り当てられていて、通常は変更されることはありません) 。
n	そのファイルに対して作成されているリンク数のチェック。
u	ファイルの所有者が変更されていないかどうかのチェック。
g	ファイルのグループ設定が変更されていないかどうかのチェック。
s	ファイルのサイズが変わっていないかどうかのチェック。

オプション	説明
b	このファイルが使用するブロックカウント数が変わっていないかどうかのチェック。
m	ファイルの修正日時が変わっていないかどうかのチェック。
c	ファイルのアクセス日時が変わっていないかどうかのチェック。
S	ファイルサイズが変わっていないかどうかのチェック。
I	ファイル名の変更を無視するかどうか。
md5	ファイルに対する md5 チェックサムが変わっていないかどうかのチェック。sha256 もしくは sha512 の使用をお勧めします。
sha1	ファイルに対する sha1 (160 ビット) チェックサムが変わっていないかどうかのチェック。sha256 もしくは sha512 の使用をお勧めします。
sha256	ファイルに対する sha256 チェックサムが変わっていないかどうかのチェック。
sha512	ファイルに対する sha512 チェックサムが変わっていないかどうかのチェック。

下記は `/sbin` ディレクトリ内にある全てのファイルをチェックするための設定で、`Binlib` オプション内で指定したチェックを実施するものの、`/sbin/conf.d/` ディレクトリについてはチェックを省略する指定です:

```
/sbin  Binlib
!/sbin/conf.d
```

AIDE データベースを作成するには、下記の手順を実施します:

1. `/etc/aide.conf` ファイルを開きます。

2. チェック項目を設定して、どのファイルをチェック対象とするのかを設定します。チェック項目全ての一覧については、`/usr/share/doc/packages/aide/manual.html` ファイルをお読みください。なお、ファイルを指定する際には、正規表現に関する知識が必要となります。設定が終わったら、ファイルを保存してください。

3. 設定ファイルの書式が正しいことを確認するには、下記のように実行します:

```
# aide --config-check
```

このコマンドの出力結果を読むことで、設定ファイルの誤っている箇所のヒントが得られます。たとえば下記のような出力が表示された場合を想定します:

```
# aide --config-check
35:syntax error:!
35:Error while reading configuration:!
Configuration error
```

上記のエラーメッセージからすると、`/etc/aide.conf` の 36 行目にエラーがあることになります。このとき、エラーメッセージの前の表示される行番号は、最後に正しく処理できた行の番号を示していることに注意してください。

4. AIDE のデータベースを準備します。下記のように実行します:

```
# aide -i
```

5. 生成されたデータベースを、CD-R や DVD-R 、もしくはリモートのサーバやフラッシュメモリなどにコピーしておいて、後から参照できるようにしておきます。

重要

このコピー作業は、データベースそれ自身を不正に書き換えられないようにするために重要な作業となります。データベースが不正に書き換えられたりしないよう、一度だけ書き込むことのできるメディアを使用しておくことをお勧めします。また、監視したいコンピュータ内には、データベースを 残してはなりません 。

21.3 ローカルの AIDE チェック

ファイルシステムのチェックを行うには、下記のようにして行います:

1. データベースのファイル名を変更します:

```
# mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db
```

2. 何らかの設定変更を行った場合は、AIDE のデータベースを再準備して、新しく生成されたほうのデータベースに移動を行う必要があります。この時点でのデータベースをバックアップしておいてもよいでしょう。詳しくは 21.2項「AIDE データベースの設定」をお読みください。

3. あとは下記のコマンドを実行すると、チェックを行うことができます:

```
# aide --check
```

何も出力が表示されない場合は、全く何も問題が見つからなかったことを表しています。AIDE が何らかの変更を検知すると、下記のように概要を表示します:

```
# aide --check
AIDE found differences between database and file system!!

Summary:
  Total number of files:        1992
  Added files:                  0
  Removed files:                0
  Changed files:                1
```

実際に変更されたファイル名の一覧を表示したい場合は、`-V` オプションをつけて冗長に出力するように指定してください。上記の出力例の場合、冗長に出力すると下記のようになります:

```
# aide --check -V
AIDE found differences between database and file system!!
Start timestamp: 2009-02-18 15:14:10

Summary:
  Total number of files:        1992
  Added files:                  0
  Removed files:                0
  Changed files:                1


---------------------------------------------------
Changed files:
---------------------------------------------------

changed: /etc/passwd

---------------------------------------------------
Detailed information about changes:
---------------------------------------------------


File: /etc/passwd
  Mtime    : 2009-02-18 15:11:02              , 2009-02-18 15:11:47
  Ctime    : 2009-02-18 15:11:02              , 2009-02-18 15:11:47
```

この例では、`/etc/passwd` ファイルのタイムスタンプが変わっていることが分かります。

21.4 システムから独立したチェック

AIDE 自身が不正に書き換えられたりする問題を避けるため、AIDE のプログラムを信頼できるソースから取得して、実行しておくこともできます。これにより、攻撃者が AIDE 自身を書き換えて、警告を出さないようにしてしまう問題を回避することができます。

AIDE 自身を信頼できるソースから実行するには、AIDE をインストール済みのシステム内で実行するのではなく、レスキューシステム内から実行しなければなりません。openSUSE Leap では、容易に拡張可能なレスキューシステムを提供していますので、ここから簡単に AIDE を実行することができます。

レスキューシステムを起動する前に、2 種類のパッケージを用意する必要があります。これらはシステムに対するドライバ更新ディスクと同じ書式で設定することができます。linuxrc の設定方法に関する詳細は、https://ja.opensuse.org/SDB:Linuxrc をお読みください。下記の手順では、実現可能な方法のうちの 1 つを説明しています。

手順 21.1: AIDE を利用したレスキューシステムの起動

1. まずは別のマシンに FTP サーバを準備します。
2. FTP サーバ内のディレクトリに、`aide` と `mhash` の各パッケージを配置します。たとえば `/srv/ftp/` のディレクトリに配置するものとします。下記の `ARCH` と `VERSION` の箇所は、実際のアーキテクチャ名とバージョン文字列に置き換えて実行してください:

   ```
   # cp DVD1/suse/ARCH/aideVERSION.ARCH.rpm /srv/ftp
   # cp DVD1/suse/ARCH/mhashVERSION.ARCH.rpm /srv/ftp
   ```
3. レスキューシステムで必要な起動パラメータを記述した、`/srv/ftp/info.txt` というファイルを作成します。下記の内容で保存してください:

   ```
   dud:ftp://ftp.example.com/aideVERSION.ARCH.rpm
   dud:ftp://ftp.example.com/mhashVERSION.ARCH.rpm
   ```

   ここで、FTP サーバのドメイン名はお使いの環境に合わせて変更してください。また、`ARCH` と `VERSION` の箇所は、実際のアーキテクチャ名とバージョン文字列に置き換えて実行してください。
4. まずはお使いのシステムを再起動して、DVD からレスキューシステムを起動して AIDE を開始します。このとき、起動パラメータには下記を指定します:

   ```
   info=ftp://ftp.example.com/info.txt
   ```

> このパラメータにより、`linuxrc` が `info.txt` ファイルを読み込んで、その内容をパラメータとして取り込むようになります。

レスキューシステムが起動したあとは、通常どおり AIDE プログラムを使用することができます。なお、レスキューシステムは日本語に対応していないため、表示は全て英語になりますので、あらかじめご了承ください。

21.5 さらなる情報

AIDE に関する情報は、それぞれ下記の場所にあります:

- AIDE の Web ページ: https://aide.github.io/
- 設定ファイル `/etc/aide.conf` 内のコメント文 (英語)
- `aide` パッケージをインストールしてある環境下であれば、`/usr/share/doc/packages/aide` 以下にもいくつかの文書があります。
- AIDE のユーザ向けメーリングリストが、https://www.ipi.fi/mailman/listinfo/aide で提供されています。

III ネットワークセキュリティ

22 X Window System と X 認証 237

23 OpenSSH によるネットワーク操作の機密保持 238

24 マスカレードとファイアウオール 269

25 VPN サーバの設定 289

26 X Window System で動作する PKI マネージャ XCA による管理 303

27 `sysctl` 変数によるネットワークセキュリティの改善 307

22 X Window System と X 認証

改訂履歴

2023-06-06

ネットワークを透過的に扱うことができるというのが Unix システムの特長です。Unix オペレーティングシステムにおけるウインドウシステムである X も同様に、ネットワーク経由で簡単に扱うことができます。X ではネットワーク上離れた場所からログインして、グラフィカルなプログラムをネットワーク経由で表示させ、操作することができるようになっています。

X クライアントがネットワーク上離れた場所にある X サーバとの間で通信を行う場合、サーバ側では不正なアクセスを行われないようにするため、保護を行う必要があります。もっとわかりやすい表現をすると、クライアントプログラム側に特定の許可を設定する必要があることになります。X Window System では、ホストベースのアクセス制御と、Cookie ベースのアクセス制御の 2 種類が用意されています。前者のアクセス制御は IP アドレスを元にした認証で、特定の IP アドレスのクライアントのみが X サーバにアクセスできるようにする仕組みです。この制御を行う際に使用するプログラムが `xhost` で、許可する IP アドレスをデータベースで管理することで、IP アドレスベースのアクセス制御を実現します。ただし、IP アドレスによる保護だけでは安全とは言えません。たとえば許可されている IP アドレスのクライアント内で、複数のユーザがログインしているような場合、異なるユーザにもアクセスを許可してしまうことになるからです。このようなセキュリティ面の問題から、この認証方法については詳しく説明していません。詳しく知りたい場合は、`man xhost` で表示されるマニュアルページをお読みください。

Cookie ベースのアクセス制御では、ID カードのように X サーバと正規のユーザの 2 者しか知り得ない文字列 (Cookie) を生成して、アクセスを制御します。Cookie はホームディレクトリ内の `.Xauthority` というファイル内に保存され、X サーバに対してウインドウを表示したい、様々なプログラムから使用することができます。`.Xauthority` ファイルは `xauth` というプログラムを使用することで、さまざまな処理を行うことができます。誤って `.Xauthority` を削除してしまったり、ファイル名を買えたりしてしまった場合は、新しいウインドウを表示することができなくなります。

SSH (Secure SHell) を使用することで、ネットワーク通信を暗号化して X サーバとの通信を透過的に扱うことができます。これを X Forwarding (転送) と呼びます。X Forwarding はサーバ側で擬似的な X サーバを動作させ、それにアクセスさせるために DISPLAY 環境変数を設定することによって成り立っています。SSH に関する詳細な説明については、第23章「OpenSSH によるネットワーク操作の機密保持」をお読みください。



警告: X Forwarding の機密性について

接続先のコンピュータが安全なホストではないとお考えの場合は、X Forwarding を使用してはなりません。X Forwarding が有効化されていると、攻撃者は SSH 接続を介して認証を行うことができてしまうためです。これにより、攻撃者はお使いの X サーバを乗っ取ることができてしまうため、たとえばキーボード入力を読み取るなどの行為が実現できてしまいます。

23 OpenSSH によるネットワーク操作の機密保持

改訂履歴

2024-06-26
2026-03-05

OpenSSH のバージョン変更についてと、設定に関する注意を追記

OpenSSH は openSUSE Leap に同梱される SSH (secure shell) 実装で、遠隔からの管理やファイル転送、機密を保持できないプロトコルへの機密トンネル設定など、さまざまな機能を提供するソフトウエアです。SSH は 2 つのホスト間での認証を含む全通信を暗号化しますので、盗聴や接続の乗っ取りなどの被害から通信を保護することができます。本章では基本的な操作のほか、ホスト鍵の切り替えや証明書認証など、大規模展開向けに便利な機能を説明しています。

23.1 OpenSSH の概要

SSH はエンドツーエンドで通信を保護するためのネットワークプロトコルで、お使いのネットワーク内同士だけでなく、外部のネットワークとの通信も保護することができます。SSH で他のコンピュータに接続するには、接続先でのユーザ名と適切な認証手段を設定する必要があります。

SSH はクライアント／サーバ型のプロトコルです。他のホストからの接続を受け付けるには `sshd` デーモンを動作させておく必要があります。また、各ホストで動作する `sshd` にはそれぞれ独自の設定を行うことができますので、アクセス可能なユーザを制限したり、許可する認証方式を制限したりすることもできるようになっています。

認証と暗号化の機能は暗号鍵対によって行われます。それぞれの鍵対には公開鍵と機密鍵という 2 つの鍵が存在し、公開鍵は暗号化を、機密鍵は暗号の復号化を行うために使用します。また、機密鍵がきちんと保護されている限り、公開鍵は誰にでも配布してかまいません。機密鍵が漏洩してしまうと、機密鍵の元の所有者になりすまして暗号化した通信を行うことができるようになってしまいます。

SSH ではサーバとクライアントの双方が互いに認証しなければならない仕組みにより、強力な保護を提供しています。クライアントが SSH での接続を開始すると、サーバは自身の持つ公開鍵を提示します。示された公開鍵が既知のものであった場合 (クライアント側の `~/.ssh/known_hosts` 内に保存されます) 、クライアントはサーバを信頼してそのまま処理を続行します。既知のものでなかった場合は、サーバを信頼して良いかどうか、下記のようにしてユーザに尋ねます:

```
The authenticity of host '192.168.22.219 (192.168.22.219)'
   can't be established. ECDSA key fingerprint is
   SHA256:yXf6pjV26NOfegvEYIt3HgG95s3Q1X6WYRhtHLF99pUo.
```

```
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

ユーザは `yes` (このまま接続を続ける) または `no` (接続を中止する) のいずれかを入力できるほか、比較のために鍵の指紋 (fingerprint) そのものを入力することもできます。



注記: ホスト鍵の指紋 (fingerprint) について

ホスト鍵の指紋 (fingerprint) を接続するユーザに配布することで、接続時にホスト鍵が正しいかどうかを確認できるようになります。上記の表示で指紋そのものを入力すると、`ssh` 側で指紋が一致しているかどうかを確認し、一致した場合にのみ接続を続けることができるようになります。これにより、目視で比較するよりずっと信頼性の高い確認を行うことができます。

これは、ユーザに検証を委ねても、セキュリティという観点では信頼性が低いためです。なお、指紋が一致していなかった場合でも、再度 `yes` を入力するか、表示された指紋をそのまま貼り付けることで、接続を続行することができます。また、より強力な保護として、証明書認証を使用する方法もあります。これは大規模向けの認証機構となるほか、上記のようなユーザへの手間を減らすことにもつながります (詳しくは 23.8項「OpenSSH での証明書認証」をお読みください) 。

接続先ホストの公開鍵が変更された場合、下記のようにセキュリティ警告を表示して接続が拒否されます:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:keNu/rJFWmpQu9BOSjIuo8NLjbeDY/x3Tktpl7oDJqo.
Please contact your system administrator.
Add correct host key in /home/geeko/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/geeko/.ssh/known_hosts:210
You can use following command to remove the offending key:
ssh-keygen -R 192.168.121.219 -f /home/geeko/.ssh/known_hosts
ECDSA host key for 192.168.121.219 has changed and you have requested strict
checking.
Host key verification failed.
```

このような場合、警告メッセージを表示せずに接続を行わせたい場合は、`~/.ssh/known_hosts` 内にある古い公開鍵を削除してから再度接続を行ってください。これにより、新しい鍵を受け付けるかどうかの確認が表示されるようになります。

`openssh` パッケージにはサーバとクライアントのほか、ファイル転送コマンドといくつかのユーティリティが含まれています。

OpenSSH では下記のような認証方式に対応しています:

パスワード認証
: ユーザ名とパスワードで認証する方式です。この方式は、SSH のセッションをどのマシンからでも受け付けることができますので、最も単純で柔軟な認証方式です。ただし、パスワードクラッキングには弱く、キーロガーなどによる窃取にも弱いため、機密性は低くなります。

公開鍵認証
: ユーザ名とパスワードではなく、個人単位で鍵を設定して認証を行う方式です。公開鍵に対応する機密鍵を持つホストからしかログインできない仕組みであるため、パスワード認証に比べると柔軟性が落ちますが、パスワードクラッキングに強く、キーロガーなどによる窃取も回避できますので、パスワード認証よりはずっと強力です。
  GNOME セッション内で公開鍵認証を自動化したい場合は、`gnome-keyring` を使用することができます。詳しくは 23.9項「gnome-keyring を利用した公開鍵ログインの自動化」をお読みください。
  また、コンソールセッション内で公開鍵認証を自動化したい場合は、`ssh-agent` を使用することができます。詳しくは 23.10項「ssh-agent を利用した公開鍵ログインの自動化」をお読みください。

パスフレーズ無しの公開鍵認証
: 機密鍵に対してパスフレーズを設定せずに公開鍵認証を使用する方式です。スクリプトや cron ジョブなどの自動化されたサービスで便利な方式です。ただし、機密鍵はきちんと保護しておかないと、容易にそのユーザになりすましてログインできてしまうので、注意が必要です。

証明書認証
: OpenSSH では証明書認証にも対応しています。これにより、鍵の管理の手間を省くことができるほか、より強力な認証手段となり、かつ大規模な SSH 展開にも対応できるようになります。

openSUSE Leap では既定で OpenSSH パッケージをインストールします。パッケージ内には、下記のようなファイルが含まれています:

`ssh`
: 遠隔のホストに対して SSH 接続を開始するためのクライアント側コマンドです。

`scp`
: 遠隔のホストとの間でファイルをやり取りするためのコマンドです。

`sftp`
: クライアントと SFTP サーバとの間でファイルを送受信するためのコマンドです (なお、SFTP (SSH FTP) プロトコルは FTPS や FTPES (FTP over SSL/TLS) と呼ばれるプロトコルとは異なり、独自に作成されているものです) 。

`ssh-add`

`ssh-agent` コマンドで動作する認証エージェントに対して、機密鍵の情報を追加するためのコマンドです。

`ssh-agent`

公開鍵認証で使用する、ユーザの機密鍵とパスフレーズを管理するためのエージェントです。`ssh-agent` はパスフレーズをメモリ内に保持し、必要に応じてパスフレーズを提供しますので、パスフレーズの再入力を省略することができるようになります。

`ssh-copy-id`

公開鍵認証を設定して、公開鍵を遠隔のホストにコピーするコマンドです。



重要

openSUSE Leap 15 SP6 およびそれ以降のバージョンでは、`openssh バージョン 9.6p1` をベースとした新しい OpenSSH パッケージが提供されるようになっています。このバージョンでは OpenSSH と crypto-policies フレームワークの統合が図られ、システム全体で暗号化の設定と暗号化標準の強制に関する設定を行うことができるようになっています。このような機能の追加に伴い、従来は `/etc/ssh/sshd_config` や `/etc/ssh/ssh_config` で設定していた鍵交換アルゴリズムや MAC (メッセージ認証コード) 、そして暗号化を制御していた設定については、直接的にアルゴリズムを指定しなくなっていることに注意してください。これらの設定は crypto-policies 側で実施することになります。既定で設定されている暗号化ポリシーは `DEFAULT` ですが、具体的に適用されるアルゴリズムの一覧を確認したい場合は、下記のディレクトリをご覧ください:

```
/etc/crypto-policies/back-ends/
```

なお、インストール処理では既存の設定ファイルを修正したり置き換えたりすることはありません。これは一般に、管理者側で最適な設定を決定するため、更新のたびにカスタマイズ内容を精査する必要があるためです。そのため、`sshd_config.rpmnew` ファイル内には、crypto-policies に関して下記のような設定が行われます:

```
# To modify the system-wide sshd configuration, create a "*.conf" file under
    # "/etc/ssh/sshd_config.d/" which will be automatically included below.
    # Don't edit this configuration file itself if possible to avoid update
    # problems.
    Include /etc/ssh/sshd_config.d/*.conf

    # The strategy used for options in the default sshd_config shipped with
    # OpenSSH is to specify options with their default value where
    # possible, but leave them commented.  Uncommented options override the
    # default value.
```

```
    Include /usr/etc/ssh/sshd_config.d/*.conf
```

通常は既存の設定ファイルの冒頭部に上記を追加すれば十分ですが、できれば変更前後のファイルを比較して、適切に適用されることを確認することをお勧めします。

23.2 サーバのセキュリティ強化

OpenSSH はインストール当初の状態でそれなりに強固な設定になっていますが、サーバのセキュリティを強化するための追加設定も用意されています。

重要: リモートの SSH サーバに対するアクセスの管理について

SSH サーバに対して何らかの設定変更を行う場合は、その変更が期待通りのものであることを確認できるまで、そのマシンに対して物理的なアクセス手段を残しておくか、もしくは root の SSH セッションを開いたままにしておく必要があります。これにより、期待通りの動作で無かった場合、設定をすぐに戻すことができるためです。

既定のサーバ設定は `/etc/ssh/sshd_config` で、ここには既定で使用される設定が書かれています。また、コメント内には全ての既定値が示されています。既定値を変更したい場合は、これらのコメント文字を外してから、必要な設定を記述してください。たとえば下記の例では、複数のネットワークインターフェイスを持つマシンで待ち受けるポートと IPv4 アドレスを指定しています:

```
#Port 22
Port 2022

#ListenAddress 0.0.0.0
ListenAddress 192.168.10.100
```

重要: /etc/services の更新について

非標準の待ち受けポートを使用する場合は、`/etc/services` ファイルを参照して、使用していないポートの中から選択してください。通常は 1024 以上の未使用ポートを使用します。設定後は、`/etc/services` ファイル内にポートが使用中である旨を記録しておくことをお勧めします。

root のログインは許可しないようにしておくことが最適です。その代わり、一般ユーザでのログインを行ったあと、`sudo` コマンドを利用して root 権限が必要な作業を行うようにしてください。どうしても root でのログインが必要な場合は、下記のサーバ設定例 (23.6項「公開鍵認証」) にある `PermitRootLogin prohibit-password` と `PasswordAuthentication` のオプションを利用して、公開鍵認証のみを受け付けるように設定してください。

なお、下記の `/etc/ssh/sshd_config` ファイル例は、アクセス制御をより強化した設定例です:

例 23.1: SSHD_CONFIG の例

```
# Check if the file modes and ownership of the user's files and# home directory are
 correct before allowing them to loginStrictModes yes# If your machine has more than one
 IP address, define which address or# addresses it listens onListenAddress 192.168.10.100
 # Allow only members of the listed groups to log inAllowGroups ldapadmins backupadmins
 # Or, deny certain groups. If you use both, DenyGroups is read firstDenyGroups users #
 Allow or deny certain users. If you use both, DenyUsers is read firstAllowUsers user1
 user2@example.com user3 DenyUsers user4 user5@192.168.10.10 # Allow root logins
 only with public key authenticationPermitRootLogin prohibit-password# Disable
 password authentication and allow only public key authentication# for all
 usersPasswordAuthentication no# Length of time the server waits for a user to log in and
 complete the# connection. The default is 120 seconds:LoginGraceTime 60 # Limit the number
 of failed connection attempts. The default is 6MaxAuthTries 4
```

`/etc/ssh/sshd_config` ファイルを変更したあとは、下記のようにして文法チェックを行ってください:

```
> sudo sshd -t
```

なお、文法チェック機能は文法のみをチェックする仕組みであり、設定の誤りは検出されないことに注意してください。チェック完了後は下記のように入力して実行することで、設定を再読み込みさせることができます:

```
> sudo systemctl reload sshd.service
```

また、サーバの鍵ディレクトリに適切なパーミッションが設定されていることを確認してください。

`/etc/ssh` ディレクトリは 0755/drwxr-xr-x のパーミッションで所有者が root:root でなければいけません。

機密鍵は 0600/-rw------- のパーミッションで所有者が root:root でなければいけません。

公開鍵は 0644/-rw-r--r-- のパーミッションで所有者が root:root でなければいけません。

23.3 パスワード認証

パスワード認証を行う場合は、接続先のマシン内にユーザアカウントとパスワードを設定し、`sshd` を設定してそれを動作させておくだけです。個人用の SSH 鍵を作成する必要はありません。下記は、sun というホストに対して suzanne というユーザ名で SSH ログインする場合の例です:

```
> ssh suzanne@sun
```

接続を行うと、まずは接続先で設定した suzanne のパスワード入力を求められます。また、`exit` と入力して `Enter` を押すと、SSH セッションを終了することができます。

両方のマシンでユーザ名が同じであれば、ユーザ名を指定する必要はありません。`ssh ホスト名` のように入力して実行するだけで済みます。認証が成功すると、相手側のコンピュータのコマンドラインを使用したり、YaST のテキストモードのような対話的なアプリケーションを使用したりすることができるようになります。

これらに加えて、`ssh ユーザ名@ホスト コマンド` の書式を使用することで、対話的でないコマンドを実行することもできます。この場合、ログインしてコマンドを実行したあと、そのまま接続を終了する動作になります。なお、`コマンド` では適切に引用符を指定する必要があります。また、複数のコマンドはローカルのシェルと同じ方法で繋げることができます:

```
> ssh suzanne@sun "df -h && du -sh  /home"
> ssh suzanne@sun "sudo nano /etc/ssh/sshd_config"
```

リモートのマシンで `sudo` を実行すると、`sudo` でのパスワード入力が求められます。

23.4 ユーザ鍵とホスト鍵の管理

生成できる鍵の種類としては、DSA, RSA, ECDSA, ECDSA-SK, Ed25519, Ed25519-SK があります。ただし、DSA は何年も前に廃止予定とされていて、OpenSSH 7.0 では無効化されていて使用すべきではありません。RSA は最も一般的に使用されている種類で、古い仕組みではあるものの一般的に使用されています (なお、OpenSSH 8.2 およびそれ以降のバージョンでは、RSA はホスト鍵としては廃止予定になっています。ホスト鍵には ECDSA もしくは Ed25519 をお使いください) 。

Ed25519 と ECDSA はより強力で高速な鍵です。Ed25519 が最も強力な鍵として提供されていますが、Ed25519 や ECDSA に対応していない古いクライアントを受け付ける必要がある場合は、3 種類全てのホスト鍵を作成してください。



注記: 古いクライアントの安全性について

古い SSH クライアントは ECDSA や ED25519 に対応していませんが、ECDSA と ED25519 への対応は 2014 年に公開された OpenSSH 6.5 から提供されているものです。どうしても接続を受け付ける必要がある場合は、必ずセキュリティ更新を適用しておいてください。また、可能であれば古いクライアントからの接続は許可しないでください。

SSH の鍵は 2 種類の目的で使用されます。クライアントがサーバを認証するためのホスト鍵と、サーバがクライアントを認証するためのクライアント鍵です (詳しくは 23.6項「公開鍵認証」をお読みください) 。サーバのホスト鍵は `/etc/ssh` ディレクトリ内に保存され、ユーザ鍵はユーザごとのディレクトリである `/home/ユーザ名/.ssh` ディレクトリ内に保存されます。

なお、ユーザ側で新しい SSH 鍵を作成すると、`/home/ユーザ名/.ssh` ディレクトリが作成されます。

また、ホスト鍵にはパスフレーズを設定してはいけません。

ほとんどの場合、ユーザ鍵には強力なパスフレーズを設定します。

23.4.1 ユーザ SSH 鍵の生成

下記に示す手順では、OpenSSH でのユーザ鍵の作成方法を説明しています。

手順 23.1: 既定の鍵と独自の鍵について

1. 既定のパラメータ (RSA, 3072 ビット) でユーザ鍵対を生成するには、オプションを何も指定せず `ssh-keygen` と入力して実行します。なお、機密鍵を保護するため、強力なパスフレーズを設定しておいてください:

```
> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tux/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:z0uJIuc7Doy07bFTe1ppZHLVrkD/bWWlBAF/PcHjblU user@host2
The key's randomart image is:
+---[RSA 3072]----+
|          ..o... |
|           o . +E|
|        . . o +.=|
|       . o . o o+|
|  .   . S . . o +|
| . =  .= * + . = |
|  o *.o.= * . +  |
|   ..Bo+.. . .   |
|    oo==  .      |
+----[SHA256]-----+
```

2. より長いビット長で RSA 鍵対を作成したい場合は、下記のように入力して実行します:

```
> ssh-keygen -b 4096
```

   OpenSSH での RSA 鍵は、最大で 16,384 ビットまでとなっています。しかしながら、ビット長を大きくしても、かかる負荷に見合うだけの効果を得られません。詳しくは GnuPG の FAQ https://www.gnupg.org/faq/gnupg-faq.html#no_default_of_rsa4096 (英語) をお読みください。

3. ユーザ鍵は必要な数だけ作成することができます。これらを異なるサーバへの接続に使用することができます。鍵対には重複しない名前を指定しなければならないほか、必要であればコメントを含めることもできます。これらは鍵の用途を記しておくのに最適です。たとえば下記のようにして名前とコメントを指定して作成します:

```
> ssh-keygen -f backup-server-key -C "infrastructure backup server"
```

4. 独自のファイル名やコメントを指定して Ed25519 鍵対を作成する場合は、下記のようにします:

```
> ssh-keygen -t ed25519 -f ldap-server-key -C "Internal LDAP server"
```

> Ed25519 の鍵は 256 ビットの固定長です。暗号強度は RSA 4096 ビットと同程度になります。

23.4.2 SSH サーバのホスト鍵作成

ホスト鍵はクライアント鍵とは違った管理方法になります。ホスト鍵にはパスフレーズを設定してはならないほか、鍵対は `/etc/ssh` ディレクトリ内に保存します。OpenSSH では、必要に応じて自動的にホスト鍵を生成します。たとえば下記のようになります:

```
> ls -l /etc/ssh
total 608
-rw------- 1 root root 577834  5月  6 2021 moduli
-rw-r--r-- 1 root root   2403  5月  6 2021 ssh_config
-rw-r----- 1 root root   3420  5月  6 2021 sshd_config
-rw------- 1 root root   1381  2月 10 2021 ssh_host_dsa_key
-rw-r--r-- 1 root root    604  2月 10 2021 ssh_host_dsa_key.pub
-rw------- 1 root root    505  2月 10 2021 ssh_host_ecdsa_key
-rw-r--r-- 1 root root    176  2月 10 2021 ssh_host_ecdsa_key.pub
-rw------- 1 root root    411  2月 10 2021 ssh_host_ed25519_key
-rw-r--r-- 1 root root     96  2月 10 2021 ssh_host_ed25519_key.pub
-rw------- 1 root root   2602  2月 10 2021 ssh_host_rsa_key
-rw-r--r-- 1 root root    568  2月 10 2021 ssh_host_rsa_key.pub
```

なお、`ssh-keygen` には新しいホスト鍵を作成するための特別なオプションである `-A` が用意されています。これはそれぞれの種類のホスト鍵対が存在していなかった場合、それらを自動的に生成することができます。保存先や空のパスフレーズ設定、鍵の種類ごとのビット長や空のコメントも自動的に設定されます。下記のように入力して実行すると、既存の鍵を削除して新しいホスト鍵対を生成することができます:

```
> sudo rm /etc/ssh/ssh_host*
> sudo ssh-keygen -A
```

なお、`ssh-keygen -A` では既存の鍵を上書きすることはありませんので、ホスト鍵を完全に入れ替える場合は、事前に削除しておく必要があります。

重要: DSA 鍵の使用禁止について

`ssh-keygen -A` を実行すると、既に何年も前から安全性に問題があると考えられている DSA 鍵も生成されます。OpenSSH 7.0 の時点でも未だ作成されますが、`sshd_config` 側で使用されないように設定されています。そのため、生成された DSA 鍵は削除して問題ありません。

ホスト鍵の切り替え (詳しくは 23.5項「ホスト鍵の切り替え」をお読みください) を検討している場合は、新しいホスト鍵を別のファイル名に保存しなければなりません。単純に新しい鍵に入れ替えるだけの作業では、ユーザ側でホスト鍵の正しさを検証できないためです。ホスト鍵を切り替えるには、サーバ側で古い鍵と新しい鍵の両方を送信するように設定し、古い鍵で新しい鍵を検証できるようにしてください。そのため、古い鍵とは別のファイル名に保存する必要があります。下記の例では、新しい RSA, Ed25519 のホスト鍵を作成して、年月を指定した新しいファイル名に保存しています。もちろん新しいホスト鍵にはパスフレーズを設定してはなりません:

```
> cd /etc/ssh
> sudo ssh-keygen -b 4096 -f "SSH_HOST_RSA_2022_02"
> sudo ssh-keygen -t ed25519 -f "SSH_HOST_ED25519_2022_02"
```

なお、新しい鍵に設定するファイル名は任意のものでかまいません。

23.5 ホスト鍵の切り替え

OpenSSH バージョン 6.8 およびそれ以降のバージョンでは、ホスト鍵を切り替えながら使用することのできるプロトコル拡張が利用できます。SSH サーバの管理者は、たとえばホスト鍵が漏洩したり、既存の鍵の強度では弱いと判断したりした場合、古いホスト鍵を廃棄して新しいホスト鍵に切り替えなければならなくなります。バージョン 6.8 以前では、ユーザ側のマシンの `ssh_config` で `StrictHostKeyChecking` が `yes` に設定されていると、ホスト鍵の変更が検出されると警告メッセージを表示して接続が行われなくなっていました。この場合、ユーザ側で対応する公開鍵を `known_hosts` ファイルから手作業で削除して接続し直し、新しい鍵を受け付ける作業が必要になります。SSH 経由でのスケジュールバックアップなど、自動化を行っている場合には面倒になってしまいます。

新しいホスト鍵の切り替え方式では、サービスの停止を伴うことなく新しい鍵を配布できる方式を提供しています。クライアントからの接続があると、サーバは手持ちの鍵を一覧で送信しますので、次回以降のログインで新しい鍵を自動的に受け付けるようにすることができます。これにより、鍵の移行のための猶予期間を設定することができますので、しばらく待ってから古い鍵を削除すれば、移行が完了することになります。ユーザ側の `known_hosts` ファイルも自動的に更新されますので、古い鍵から新しい鍵への切り替えも自動でできることになります。

ホスト鍵の切り替え機能を使用するには、新しい鍵の作成のほか、サーバ側では `/etc/ssh/sshd_config` ファイルの変更を、クライアント側では `/etc/ssh/ssh_config` ファイルの変更をそれぞれ行う必要があります。

まずは新しい鍵 (複数可) を作成します。下記の例では、ファイル名とコメントを指定して新しい RSA 鍵と Ed25519 鍵を作成しています。一般的には作成日などを含めておくとよいでしょう。また、ホスト鍵にはパスフレーズを設定してはいけません:

```
# ssh-keygen -t rsa -f ssh_host_rsa_2022-01 -C "main server"
```

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh_host_rsa_2022-01
Your public key has been saved in ssh_host_rsa_2022-01.pub
The key fingerprint is:
SHA256:F1FIF2aqOz7D3mGdsjzHpH/kjUWZehBN3uG7FM4taAQ main server
The key's randomart image is:
+---[RSA 3072]----+
|         .Eo*.oo |
|          .B .o.o|
|          o . .++|
|         . o ooo=|
|        S . o +*.|
|         o o.oooo|
|       .o ++oo.= |
|       .+=o+o + .|
|       .oo++..   |
+----[SHA256]-----+

# ssh-keygen -t ed25519 -f ssh_host_ed25519_2022-01 -C "main server"
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh_host_ed25519_2022-01
Your public key has been saved in ssh_host_ed25519_2022-01.pub
The key fingerprint is:
SHA256:2p9KOgiXv7WsRnLjwjs4hJ8EFcoX1FWR4nQz6fxnjxg main server
The key's randomart image is:
+--[ED25519 256]--+
|   .+o ...o+     |
| . .... o *      |
|  o..  o = o     |
|  ..   .. o      |
|   o. o S  .     |
|  . oo.*+   E o  |
|   + ++==..  = o |
|    = +oo= o. . .|
|     ..=+o=      |
+----[SHA256]-----+
```

表示されたフィンガープリント (key fingerprint) を、ユーザ側で鍵を検証する際の確認として記録しておいてください。

新しい鍵のファイル名を `/etc/ssh/sshd_config` で指定します。既存の鍵はコメントアウトせず、そのまま設定しておいてください:

```
## 古い鍵
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
HostKey /etc/ssh/ssh_host_ecdsa_key
```

```
## 新しい鍵
HostKey /etc/ssh/ssh_host_rsa_2022-01
HostKey /etc/ssh/ssh_host_ed25519_2022-01
```

ファイルを保存したら、`sshd` を再起動します:

```
# systemctl restart sshd.service
```

なお、ユーザ側の `/etc/ssh/ssh_config` では、下記のような設定をしている必要があります:

```
UpdateHostKeys ask
StrictHostKeyChecking yes
```

クライアント側からサーバに SSH で接続してテストしてみます。既に接続している場合はいったんログアウトして再接続してみてください。下記のようなメッセージが表示されるはずです:

```
The server has updated its host keys.
These changes were verified by the server's existing trusted key.
Deprecating obsolete hostkey: ED25519
SHA256:V28d3VpHgjsCoV04RBCZpLo5c0kEslCZDVdIUnCvqPI
Deprecating obsolete hostkey:
RSA SHA256:+NR4DVdbsUNsqJPIhISzx+eqD4x/awCCwijZ4a9eP8I
Accept updated hostkeys? (yes/no):yes
```

`UpdateHostKeys ask` を `UpdateHostKeys yes` に変更することで、変更された鍵を自動的に受け付けるように設定することもできます。

さらに詳しい情報を知りたい場合は、下記を参照してください:

- 23.4項「ユーザ鍵とホスト鍵の管理」
- http://blog.djm.net.au/2015/02/key-rotation-in-openssh-68.html (英語)
- man 5 ssh_config, man 5 sshd_config

23.6 公開鍵認証

公開鍵認証は、アカウントに対して設定されたパスワードではなく、ユーザが作成した個人用の鍵で認証を行う方法です。

下記は、コメント付きで個人用の RSA 鍵を作成する場合の例を示しています。まずは `~/.ssh` ディレクトリに移動 (存在していない場合は作成して移動) したあと、新しい鍵対を作成します。この場合、パスフレーズはできる限り複雑なものとし、パスフレーズは安全な場所に記録しておいてください:

```
> cd ~/.ssh
```

```
> ssh-keygen -C "web server1" -f id-web1 -t rsa -b 4096
```

あとは作成した鍵対のうち、公開鍵を接続先のホストにコピーします。接続先のマシンにユーザアカウントがあれば、SSH でアクセスして鍵を書き込みます:

```
> ssh-copy-id -i id-web1 user@web1
```

あとは作成した鍵でログインを試してみるだけです:

```
> ssh -i id-web1 user@web1
Enter passphrase for key 'id-web1':
Last login: Sat Jul 11 11:09:53 2022 from 192.168.10.122
Have a lot of fun...
```

この場合、パスフレーズの入力を求められますが、接続先のマシンのパスワードではなく、鍵のパスフレーズを入力する必要があることに注意してください。

なお、安全性を高めたい場合は、接続先のマシンで公開鍵認証を強制して、パスワード認証を拒否するように設定することをお勧めします (詳しくは 例23.1「sshd_config の例」) をお読みください) 。この場合、接続先のマシンに個人用の公開鍵を設定していないと、`ssh-copy-id` やその他の方法での公開鍵書き込みもまた実行できなくなることに注意してください。このような場合は、USB メモリなどに公開鍵を書き込んで、接続先のマシンの `~/.ssh/authorized_keys` にコピーする作業を実施しなければなりません。

23.7 パスフレーズ無しの公開鍵認証

これは機密鍵に対してパスフレーズを設定せずに公開鍵認証を使用する方式です。パスフレーズを指定せずに新しい鍵を作成したあとは、パスフレーズ付きの鍵と同じ手順で鍵を使用することができます。これは特にスクリプトや cron ジョブなどの自動化されたサービスで便利な方式です。ただし、機密鍵はきちんと保護しておかないと、容易にそのユーザになりすましてログインできてしまうので、注意が必要です。

なお、パスフレーズ無しで鍵を使用するもう 1 つの方法として、`gnome-keyring` があります。こちらは機密鍵のパスフレーズを記憶して、必要に応じて適用を行うことができます。`gnome-keyring` は GNOME デスクトップセッション向けの仕組みです (詳しくは 23.9項「gnome-keyring を利用した公開鍵ログインの自動化」をお読みください) 。

コンソールセッションの場合は `ssh-agent` を使用することができます (詳しくは 23.10項「ssh-agent を利用した公開鍵ログインの自動化」をお読みください) 。

23.8 OpenSSH での証明書認証

OpenSSH 5.4 およびそれ以降のバージョンでは、証明書による認証を行うことができます。証明書認証は公開鍵認証と似ていますが、ホストとユーザとの間で認証を行うにあたって、単純な暗号鍵ではなくデジタル署名された証明書を使用します。証明書認証はサーバとユーザとの間の認証を集中管理することができますので、いちいち公開鍵を接続先のホストにコピーするような手間も発生しません。また、管理者側での制御もできるようになりますので、より管理者側に権力を集中させることができます。

証明書には公開暗号鍵のほか、ユーザ定義の識別文字列やユーザ名、ホスト名などを必要に応じて含めることができます。ユーザ側とホスト側の公開鍵にはいずれも証明機関 (CA) による署名が付与され、これによって正当性を確認できるようになります。つまり、ユーザ側もホスト側も、CA の公開鍵を信頼しているだけで認証が成立しますので、わざわざユーザやホストごとに信頼を設定する必要もありません。

また、OpenSSH の公開鍵認証の場合、ユーザの公開鍵をそれぞれ接続したい SSH サーバにコピーしていく手間がある (通常は `~/.ssh/authorized_keys` ファイルに保存されます) ほか、ユーザ側でもホスト鍵を受け付ける (通常は `~/.ssh/known_hosts` ファイルに保存されます) 必要がありました。このような構成では操作ミスを引き起こすことがあるほか、管理上も面倒な存在でした。また、このような公開鍵認証では鍵の有効期限が設定できませんので、新しい鍵に入れ替えたい場合は、ネットワーク内に存在する鍵を削除していく手間もありました。

大企業では、このような手間を全体的に削減して自動化する仕組み (たとえば Ansible など) が必要でした。たとえば Meta 社 (詳しくは https://engineering.fb.com/2016/09/12/security/scalable-and-secure-access-with-ssh/ (英語) をお読みください) では、この仕組みを完全に自動化して、証明書を失効させて置き換えることができるようになっているほか、証明機関自身も自動化することで、業務の手間とならない仕組みを構成しています。

なお、事前に実施しておかなければならない作業として、ネットワーク内に存在する全てのホストに SSH 接続して設定ファイルを編集し、`sshd` を再起動する必要があります。

OpenSSH での証明機関構築は、下記のような手順で行います:

- 機密の保持できる信頼できるサーバを構築して、そこに証明機関を配置します。まずは証明機関自身の鍵対を作成して、この機密鍵でユーザ鍵やホスト鍵への署名を行うとともに、公開鍵はサーバへのアクセスを許可する全てのユーザに配布します。
- 次にホスト側の公開鍵を集めて、証明機関で署名を実施したあと、作成された証明書をそれぞれのホストに返します。発行されたホストの証明書は、ホスト鍵と同様に `/etc/ssh` 内に配置します。
- 次にユーザ側の公開鍵を集めて、証明機関で署名を実施したあと、作成された証明書を各ユーザに返します。ユーザ側の証明書は、ユーザ鍵と同様に `~/.ssh` 内に配置します。

- サーバ側、ユーザ側の各マシンで設定ファイルを編集して、必要であれば `sshd` を再起動します。
- なお、発行した証明書は、機密鍵が漏洩したり、ユーザが組織を脱退したり、サーバを退役させたりした場合に失効処理を実施します。証明書の失効処理は、ネットワーク内に配布した公開鍵を削除して回るよりは比較的簡単です。

なお、ユーザ側・サーバ側の管理者とも、それぞれの OpenSSH 鍵はきちんと保護してください。ただし、公開鍵は自由に公開してかまいません。電子メールなどの機密の保護できない方式で公開鍵を送信しても、機密鍵が保護されている限り問題はありません。

ただし、SSH での証明書は SSL/TLS ではなく OpenPGP の標準に従っていることに注意してください。これにより、証明書の形式は X.509 ではなく、OpenPGP 形式になっています。

23.8.1 新しい証明機関の構築

本章では、新しい証明機関 (CA) を構築する手順を説明しています。なお、証明機関を構築するにあたっては、管理の容易さや効率性を考えておくことをお勧めします。



重要: 証明機関自身の保護について

証明機関を設置するホストに対しては、注意深く構成するようにしてください。これは、証明機関は組織全体に対する認証の中枢となるためです。もしも証明機関に対して不正にアクセスできてしまうと、発行した証明書の安全性が損なわれるため、ネットワーク内のコンピュータに対して自由にアクセスを許す結果になってしまうばかりか、サーバや証明機関自身を騙して証明書を偽造することもできてしまいます。そのため、証明機関を設置するホストはそれ専用のコンピュータとして配置し、証明書の発行に必要なサービスのみを動作させるようにしてください。

また、サーバ向けとその他向けで別々の署名鍵を設定してもかまいません。管理すべき証明書の数が多い場合は、サーバ向けとクライアント向けで別々の証明機関のマシンを構築することもできます。同じマシン内に複数の証明機関を構築する場合は、別々のディレクトリ内でもかまいません。本章の例では、`/ca-ssh-hosts` と `/ca-ssh-users` というディレクトリ内にそれぞれの証明機関を構築しています。また、ホスト名は `ca.example.com` という名前を使用しています。

なお、セキュリティポリシー上の理由から、ユーザやサーバに対する公開鍵のコピーを保持しておく必要がある場合は、それぞれサブディレクトリを作成して保存するようにしてください。これにより、追跡が容易にできるほか、名前の衝突を防ぐことにも繋がります。

重要: RSA 署名鍵の廃止予定について

2020 年 2 月に公開された OpenSSH 8.2 もしくはそれ以降のバージョンでは、RSA による署名鍵は廃止予定とされています。Ed25519 もしくは ECDSA をお使いください。

下記の例では、それぞれホスト鍵に署名するための鍵と、ユーザ鍵に署名するための鍵をそれぞれ作成しています。それぞれには強力なパスフレーズを設定してください:

```
> sudo ssh-keygen -t ed25519 -f /ca-ssh-hosts/ca-host-sign-key -C "signing key for host
 certificates"
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ca-host-sign-key
Your public key has been saved in ca-host-sign-key.pub
The key fingerprint is:
SHA256:STuQ7HgDrPcEa7ybNIWOn6kPbj28X5HN8GgwllBbAtO
 signing key for host certificates
The key's randomart image is:
+--[ED25519 256]--+
|      o+o..      |
|   . . o.=E      |
|    = + B .      |
|   + O + = B     |
|  . O * S = +    |
|   o B + o .     |
|    =o=   .      |
|   o.*+  .       |
|   .=.o+.        |
+----[SHA256]-----+
```

```
> sudo ssh-keygen -t ed25519 -f /ca-ssh-users/ca-user-sign-key -C "signing key for user
 certificates"
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ca-user-sign-key
Your public key has been saved in ca-user-sign-key.pub
The key fingerprint is:
SHA256:taYj8tTnjkzgfHRvQ6HTj8a37PY6rwv96V1x+GHRjIk signing key for user certificates
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|             . +.|
|          . E o.o|
|         . + . ..|
|      . S * o .+.|
|     o + + = +..+|
```

```
|    . = * . O + o|
|     + = = o =oo+|
|      . o.o  oOX=|
+----[SHA256]-----+
```

鍵を作成したら、署名鍵 (公開鍵であることをよくお確かめください) を SSH サーバの動作する全てのホスト内の `/etc/ssh` 内にコピーします。あとは全てのホスト内の `/etc/ssh/sshd_config` ファイルを編集して、署名鍵を指定しておきます:

```
TrustedUserCAKeys /etc/ssh/ca-user-sign-key.pub
```

ファイルを保存したら、`sshd` を再起動します。

23.8.2 サーバ証明書の作成

下記の例では、データベースサーバ (`db-server`) のホスト公開鍵に対して、署名を付与しています:

```
> sudo ssh-keygen -s /ca-ssh-hosts/ca-host-sign-key \
   -n venus,venus.example.com -I "db-server host cert" \
   -h -V +4w /etc/ssh/ssh_host_ed25519_key.pub
Enter passphrase:
Signed host key /etc/ssh/ssh_host_ed25519_key-cert.pub: id
"db-server host cert" serial 0 for venus,venus.example.com
valid from 2022-08-08T14:20:00 to 2022-09-05T15:21:19
```

サーバ内に複数の鍵が存在する場合は、全ての鍵に対して署名を実施してください。

- `-s` では機密鍵を指定しています。
- `-n` ではプリンシパル名を指定しています。ホスト鍵の場合、ここには完全修飾ドメイン名によるマシンのホスト名を指定します。
- `-I` では識別用の文字列を指定しています。ここはコメント欄ですので、わかりやすい説明を記述してください。また、この文字列はログにも記録されますので、ログ内を検索したい場合にも有用です。
- `-h` ではホスト鍵の証明書であることを指定しています。
- `-V` では証明書の有効期限を指定しています。この例では、証明書は 4 週間後になっています (指定可能な期間については、`man 1 ssh-keygen` 内の "-V validity_interval" の章 (英語) をお読みください) 。

あとは発行された証明書が正しいことを確認します:

```
> ssh-keygen -Lf /etc/ssh/ssh_host_ed25519_key-cert.pub
```

```
/etc/ssh/ssh_host_ed25519_key-cert.pub:
        Type: ssh-ed25519-cert-v01@openssh.com host certificate
        Public key: ED25519-CERT SHA256:/
         U7C+qABXYyuvueUuhFKzzVINq3d7IULRLwBstvVC+Q
        Signing CA: ED25519 SHA256:
         STuQ7HgDrPcEa7ybNIWOn6kPbj28X5HN8GgwllBbAtO (using ssh-ed25519)
        Key ID: "db-server host cert"
        Serial: 0
        Valid: from 2022-08-08T14:20:00 to 2022-09-05T15:21:19
        Principals:
                venus
                venus.example.com
        Critical Options: (none)
        Extensions: (none)
```

あとは発行された証明書のフルパスを `/etc/ssh/sshd_config` で指定して、クライアント側に送信するようにします:

```
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub
```

ファイルを保存したら、`sshd` を再起動します:

```
> sudo systemctl restart sshd.service
```

ホストの証明書をクライアント側で受け付ける方法については、23.8.3項「ユーザ向けの CA 設定」をお読みください。

23.8.3 ユーザ向けの CA 設定

下記の例では、クライアント側で個別の鍵ではなく証明機関で署名された鍵を受け付ける設定を示しています。下記の例では単一のサーバに対してのみアクセスを許可しています。また、下記の設定は `~/.ssh/known_hosts` 内で改行せず 1 行で記述する必要があることに注意してください。また、既に `~/.ssh/known_hosts` ファイルが存在する場合は、ファイル名を変更して残しておき、下記の証明機関設定のみを含むようにしてください。なお、`/etc/ssh/ssh_known_hosts` ファイルに証明機関を設定する方法もありますが、こちらを編集するには管理者権限が必要となります:

```
@cert-authority db,db.example.com ssh-ed25519
 AAAAC3NzaC1lZDI1NTE5AAAAIH1pF6DN4BdsfUKWuyiGt/leCvuZ/fPu
 YxY7+4V68Fz0 signing key for user certificates
```

サーバを複数指定する場合は、カンマ区切りで指定してください。たとえば `venus,venus.example.com,saturn,saturn.example.com` のようになります。ワイルドカードを利用して `*.example.com,*.example2.com` のように指定し、ドメイン内の全てのサーバに対してアクセスを許可することもできます。

あとは実際にサーバに接続してみてください。ホスト鍵の受け付けを促すプロンプトが表示されず、そのままパスワード入力を求められれば成功です。

23.8.4 ユーザ証明書の作成

下記のようにしてユーザの公開鍵に署名を付与します:

```
> sudo ssh-keygen /ca-ssh-hosts/ca-user-sign-key -I "suzanne's cert" -n suzanne -V +52w
 user-key.pub
 Signed user key .ssh/ed25519key-cert.pub: id "suzanne's cert" serial 0
 for suzanne valid from 2022-09-14T12:57:00 to 2023-09-13T12:58:21
```

ここで、プリンシパル名は必ずユーザ名でなければならないことに注意してください。また、発行した証明書はユーザ側のマシンの `~/.ssh` 内に保存してください。

ユーザの証明書を発行すれば、サーバ側の `~/.ssh/authorized_keys` ファイルは不要になります。サーバ側のこのファイルを削除したあと、SSH で接続しなおしてみてください。パスワード入力を求められることなくログインできるようになるはずです (なお、サーバ側の `/etc/ssh/sshd_config` ファイルで、`TrustedUserCAKeys /etc/ssh/ca-user-sign-key.pub` を設定していることをご確認ください。この設定で、信頼すべき証明機関を設定しています) 。

また、サーバ内のログファイルに `Accepted publickey for suzanne` が記録されていることも確認しておくとよいでしょう。

23.8.5 ホスト鍵の失効処理

サーバが不正侵入を受けたり退役したりした場合は、証明書の失効処理を実施してホスト鍵の利用を停止する必要があります。このような場合は、各クライアント側の `/etc/ssh/revoked_keys` ファイルなどに対応する公開鍵を指定します:

```
ssh-ed25519-cert-v01@openssh.com
    AAAAIHNzaC1lZDI1NTE5LWNlcnQtdjAxQG9wZW5zc2guY29tAAAAIK6hyvFAhFI+0hkKehF/
    5O6fD1VdcW29ykfFJn1CPK9lAAAAIAawaXbbEFiQOAe5LGclrCHSLWbEeUauK5+CAuhTJyz0
    AAAAAAAAAAAAAAACAAAAE2RiLXNlcnZlciBob3N0IGNlcnQAAAAeAAAABXZlbnVzAAAAEXZl
    bnVzLmV4YW1wbGUuY29tAAAAAGMabhQAAAAAYz9YgQAAAAAAAAAAAAAAAAAAAAAADMAAAALc3No
    LWVkMjU1MTkAAAAgfWkXoM3gF2x9Qpa7KIa3+V4K+5n98+5jFjv7hXrwXPQAAABTAAAAC3Nz
    aC1lZDI1NTE5AAAAQI+mbJsQjt/9bLiURse8DF3yTa6Yk3HpoE2uf9FW/
    KeLsw2wPeDvOd6jv49Wgr5T3xHYPf+VPJQW35ntFiHTlQg= root@db
```

上記で作成したファイルは、`/etc/ssh/sshd_config` 内で下記のようにして指定します:

```
RevokedKeys /etc/ssh/revoked_keys
```

23.9 gnome-keyring を利用した公開鍵ログインの自動化

GNOME デスクトップ環境をインストールすると、`gnome-keyring` パッケージも自動的にインストールされ、有効化されます。`gnome-keyring` はシステムのログインに統合され動作する仕組みであるため、ログイン時に自動的に機密記憶領域の保護が解除されます。ログイン時のパスワードを変更しても、`gnome-keyring` はそれを自動的に認識して更新処理を行います。

`gnome-keyring` は `~/.ssh` ディレクトリ内にある全ての鍵対 (`*.pub` の形式の公開鍵があるもの) を自動的に読み込みます。その他のディレクトリ内にあるファイルを読み込みたい場合は、`ssh-add` コマンドで読み込みを行ってください。たとえば下記のようになります:

```
> ssh-add ~/.otherkeys/my_key
```

読み込まれた全ての鍵を表示するには、下記のコマンドを実行します:

```
> ssh-add -L
```

システムを起動して SSH セッションを開くと、機密鍵のパスフレーズの入力を促されます。

`gnome-keyring` はセッションを終了するまでの間パスフレーズを記憶します。そのため、システムを再起動しない限り、パスフレーズを再度入力する必要がなくなります。

23.10 ssh-agent を利用した公開鍵ログインの自動化

`openssh` パッケージには `ssh-agent` というユーティリティが付属しています。このユーティリティは、セッションが終了するまでの間だけ機密鍵とパスフレーズを記憶して、必要なときに自動的にそれを適用することができます。

`ssh-agent` をログイン時に自動的に起動して鍵を読み込むように設定したい場合は、`~/.profile` ファイル内に下記のような行を記述します:

```
eval "$(ssh-agent)"
ssh-add
```

上記の行のうち、最初の行は `ssh-agent` を起動するコマンドで、次の行は `~/.ssh` ディレクトリ内にある全ての鍵を読み込むコマンドです。このように起動を行うことで、公開鍵認証が必要な SSH セッションを開始すると、最初の 1 回だけはパスフレーズの入力を求められます。その後はシステムを再起動するまでパスフレーズを記憶しますので、再入力を行う必要が無くなります。

なお、特定の鍵のみを読み込むように設定したい場合は、下記のように `~/.profile` 内で `id_rsa` や `id_ed25519` のファイルを指定してください:

```
> ssh-add id_rsa id_ed25519
```

23.10.1 `ssh-agent` の X セッション内での利用

openSUSE Leap では、GNOME ディスプレイマネージャを起動すると `ssh-agent` が自動的に開始されます。X セッションの開始時に `ssh-add` で鍵を自動的に読み込むようにしたい場合は、下記のように実施します:

1. まずは対象のユーザでログインし、`~/.xinitrc` ファイルが存在しているかどうかを確認します。
2. ファイルが存在していない場合、既存のテンプレートを使用するか、`/etc/skel` 内にあるファイルをそのままコピーします:

```
if [ -f ~/.xinitrc.template ]; then mv ~/.xinitrc.template ~/.xinitrc; \
else cp /etc/skel/.xinitrc.template ~/.xinitrc; fi
```

3. テンプレートをコピーした場合は、下記のような行を探してコメント文字を削除します。`~/.xinitrc` ファイルが既に存在していた場合は、下記の内容を追記します (コメント文字は削除してください)。

```
# if test -S "$SSH_AUTH_SOCK" -a -x "$SSH_ASKPASS"; then
#       ssh-add < /dev/null
# fi
```

4. あとは新しく X セッションを開始すると、SSH のパスフレーズ入力を求められるようになります。

23.11 SSH 機密鍵のパスフレーズ変更

機密鍵に設定したパスフレーズを変更したり、パスフレーズを削除したりしたい場合は、`ssh-keygen` コマンドを使用します:

```
> ssh-keygen -pf ~/.ssh/server1
Enter old passphrase:
Key has comment 'shared videos server1'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

23.12 鍵の指紋 (fingerprint) の取得

`ssh-keygen` コマンドを使用することで、公開鍵の指紋を表示することができます。下記の例では、ED25519 鍵に対して SHA256 形式の指紋を表示しています:

```
> ssh-keygen -lf ldap-server
256 SHA256:W45lbmj24ZoASbrqW0q9+NhF04muvfKZ+FkRa2cCiqo comment (ED25519)
```

`-v` フラグを指定すると、ASCII アート形式での鍵表示を行うこともできます:

```
> ssh-keygen -lvf ldap-server
256 SHA256:W45lbmj24ZoASbrqW0q9+NhF04muvfKZ+FkRa2cCiqo comment (ED25519)
+--[ED25519 256]--+
|                 |
|                 |
|    .. .         |
|  .o..+ +        |
| ...o+ BSo+      |
|. ..o.o =X       |
|...o o..* =      |
|o.*.* =+ = .     |
|E*o*+O. o.o      |
+----[SHA256]-----+
```

23.13 ネットワーク上の離れたホストでの X11 アプリケーションの起動

SSH では、ネットワーク上離れた場所にある X アプリケーションも簡単に使用することができます。接続先のマシンの `/etc/ssh/sshd_config` で `X11Forwarding Yes` を設定しておかなければなりませんが、`ssh` コマンドに `-X` オプションを付けることで、接続先の `DISPLAY` 変数が自動的に設定され、全ての X 出力が SSH の接続を介してローカルマシンに表示されるようになります。それと同時に接続先で起動された X アプリケーションは、不正に傍受されたりできないようになります。

たとえば下記のように実行することで、リモートのマシンでシンプルなゲームである GNOME Mines を動かすことができます:

```
> ssh wilber@sun
Password:
Last login: Tue May 10 11:29:06 2022 from 192.168.163.13
Have a lot of fun...

wilber@sun>  gnome-mines
```

起動したアプリケーションはローカルにインストールされているかのように、ローカル側に表示されるはずです (ただしネットワーク側の遅延があると、その分だけ性能に影響があります) 。アプリケーションの終了も通常と同じで、閉じるボタンを押すだけです。ただし、アプリケーションを閉じても SSH のセッションは終了せず、そのまま動作し続けます。

重要: Wayland では X11 転送機能が使用できない問題について

X11 転送機能を利用するには、接続先で X Window System が動作している必要があります。これは、X Window System にはネットワーク機能が内蔵されているものの、Wayland にはそのような機能が内蔵されていないことによるものです。

X Window System で動作しているのか、それとも Wayland で動作しているかを確認するには、下記のように入力して実行します:

```
> echo $XDG_SESSION_TYPE
x11
```

Wayland で動作している場合は、下記のような出力になります:

```
> echo $XDG_SESSION_TYPE
wayland
```

systemd を利用して判断したい場合は、`loginctl` コマンドを使用します:

```
> loginctl show-session "$XDG_SESSION_ID" -p Type
Type=x11

> loginctl show-session "$XDG_SESSION_ID" -p Type
Type=wayland
```

23.14 エージェント転送

`-A` オプションを追加することで、ssh-agent 認証の仕組みが接続先のマシンに引き継がれます。このオプションを使用することで、公開鍵を様々なホストに登録して適切な場所に保存しておくだけで、それらのホストにパスワード無しでログインできるようになります (詳しくは 23.6項「公開鍵認証」をお読みください) 。

`/etc/ssh/sshd_config` では、既定で `AllowAgentForwarding yes` が設定されています。無効化したい場合は `No` に変更してください。

23.15 scp - 機密を確保したファイルのコピー

scp は、指定したファイルをリモートのマシンとの間でコピーするプログラムです。jupiter 内でのユーザ名と sun 内でのユーザ名が異なる場合は、ユーザ名@ホスト名 の形式で指定してください。また、ホームディレクトリ以外のディレクトリにファイルをコピーしたい場合は、sun: ディレクトリ の形式で指定してください。下記の例では、ローカルから接続先に、および接続先からローカルにそれぞれファイルをコピーしています。

```
> scp ~/MyLetter.tex tux@sun:/tmp ❶
> scp tux@sun:/tmp/MyLetter.tex ~ ❷
```

❶ ローカルから接続先に

❷ 接続先からローカルに



ヒント: -l オプションについて

ssh コマンドでは、-l オプションを指定することで、接続先のコンピュータ内でのユーザを指定することができます (ユーザ名@ホスト名 と同じ意味になります) 。scp コマンドの場合、-l は scp が消費する帯域を制限するためのオプションを意味しています。

正しいパスワードの入力が行われると、scp はデータ転送を開始します。進捗表示バーと、各ファイルの転送にかかる残り時間が表示されます。-q オプションを指定すると、それらの出力を省略することができます。

scp はディレクトリ全体を再帰的にコピーする用途でも使用することができます:

```
> scp -r src/ sun:backup/
```

上記のコマンドを実行すると、src ディレクトリ内にある全てのファイルとサブディレクトリを、sun 内の ~/backup ディレクトリにコピーします。サブディレクトリが存在していない場合は、それらは自動的に作成されます。

-p オプションは、scp に対してファイルのタイムスタンプを変更しないようにするオプションです。-C はデータ転送を圧縮するよう指示するオプションです。これにより転送の際のデータ量を減らすことができますが、送信側・受信側両方のホストでプロセッサにかかる負荷が上昇します。

23.16 sftp - 機密を保持したファイルの転送

23.16.1 sftp の使用

複数のファイルをコピーしたい場合や、異なる場所からコピーを行いたい場合は、scp の代わりに sftp を利用したほうが便利になることがあります。このコマンドは、通常の FTP プロトコルと同様に、様々なコマンドを実行できるシェルを開くことができます。利用可能なコマンドの一覧を取得したい場合は、sftp のプロンプトで help と入力してください。より詳しい詳細は、sftp のマニュアルページに書かれています。

```
> sftp sun
Enter passphrase for key '/home/tux/.ssh/id_rsa':
Connected to sun.
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
[...]
```

23.16.2 ファイルアップロードのパーミッション設定

通常の FTP サーバと同様に、ユーザは SFTP を利用することで、ダウンロードやアップロードを行うことができます。コマンドも通常の FTP サーバと同じで、たとえばアップロードであれば put コマンドを利用します。既定では、アップロードしたファイルのパーミッションは、ローカル側のパーミッションと同じに設定されます。ただし、自動的に設定されるパーミッションを変更したい場合は、下記のいずれかを実施してください:

umask の設定

umask はパーミッションに対するフィルタとして動作する仕組みで、元々のファイルとアップロード先のファイルとの間で、引き継いで欲しくないビットを 1 にします。なお、アクセス許可を追加することはできません。取り消すことのみ実現できます:

元々のパーミッション	umask の値	アップロードされたファイルのパーミッション
0666	0002	0664
0600	0002	0600

元々のパーミッション	umask の値	アップロードされたファイルのパーミッション
0775	0025	0750

SFTP サーバ側で umask を適用するには、`/etc/ssh/sshd_configuration` ファイルを編集します。まずは `Subsystem sftp` で始まる行を探し、行の末尾に `-u` オプションを追加して、必要な値を指定してください。たとえば下記のようになります:

```
Subsystem sftp /usr/lib/ssh/sftp-server -u 0002
```

明示的なパーミッションの指定

SFTP 経由でアップロードされた全てのファイルに対して、同じパーミッションを設定するように指定する方法です。`-u` オプションと同様に、`600`，`644`，`755` のような 3 桁のパーミッション値を指定してください。`-m` と `-u` の両方が指定された場合、`-u` は無視されます。

SFTP サーバ側で明示的にパーミッションを指定するには、`/etc/ssh/sshd_config` ファイルを編集します。まずは `Subsystem sftp` で始まる行を探し、行の末尾に `-m` オプションを追加して、必要な値を指定してください。たとえば下記のようになります:

```
Subsystem sftp /usr/lib/ssh/sftp-server -m 600
```



ヒント: SSH デーモンのログファイル表示

`sshd` が出力したログを表示したい場合は、下記のコマンドを実行します:

```
> sudo journalctl -u sshd
```

23.17 ポート転送 (SSH トンネリング)

`ssh` は TCP/IP 接続を転送するような用途でも使用することができます。この機能は `SSH トンネリング` とも呼ばれ、暗号化された通信を介して TCP の接続を別のマシン内にある特定のポートに転送することができます。

下記のコマンドを実行することで、ローカルの TCP ポート 25 (SMTP) への接続を行うと、jupiter への暗号化接続を介して sun の TCP ポート 25 (SMTP) に接続するようになります。これは特に、SMTP-AUTH や POP-before-SMTP のような機能を持たない SMTP サーバを使用するような場合に有用です。このようにポート転送を行うことで、電子メールを「自宅の」メールサーバに配送するような構成を構築することができます。

```
# ssh -L 25:sun:25 jupiter
```

上記と同様に、全ての POP3 接続 (ポート 110) を sun に転送したい場合は、下記のように実行します:

```
# ssh -L 110:sun:110 jupiter
```

いずれのコマンドも特権ポートを使用していることから、`root` で実行しなければなりません。いったんコマンドを実行すれば、一般ユーザからもそのポートを使用して電子メールを送受信できるようになります。この場合、電子メールソフトウエア側のサーバ設定は `localhost` を指定します。上記で説明したそれぞれのプログラムのオプションに関する説明は、`/usr/share/doc/packages/openssh` 内にある OpenSSH のパッケージドキュメンテーション内にあります。

23.18 さらなる情報

https://www.openssh.com
: OpenSSH の Web ページ

https://en.wikibooks.org/wiki/OpenSSH
: OpenSSH Wikibook (英語)

`man sshd`
: OpenSSH デーモンのマニュアルページ

`man ssh_config`
: OpenSSH のクライアント側設定ファイルのマニュアルページ

`man scp`,
`man sftp`,
`man ssh`,
`man ssh-add`,
`man ssh-copy-id`,
`man ssh-keygen`
: ファイルコピー (`scp` , `sftp`) やログイン (`slogin` , `ssh`) 、鍵管理などのコマンドに対するマニュアルページ。

`/usr/share/doc/packages/openssh-common/README.SUSE`
: SUSE 固有のパッケージドキュメンテーション; 提供元からの既定値の変更点などを説明しています。

23.19 fail2ban による SSH ブルートフォース攻撃の抑止

SSH ブルートフォース攻撃とは、様々なユーザ名とパスワードの組み合わせでログインを試すことにより、ネットワーク経由でサーバへの不正侵入を試みるものです。攻撃者は自動化されたツールを利用して侵入を試みるため、膨大な数のユーザ名とパスワードの組み合わせを試すことができてしまいます。

このような不正侵入を防ぐには、fail2ban と呼ばれるソフトウエアを使用します。fail2ban は定期的にシステムログを検索して攻撃を検知し、ファイアウオールで接続元の IP アドレスからのアクセスを禁止するなど、その攻撃内容に応じた処理を実施します。なお fail2ban は、ユーザ名とパスワードを使用するサービスを保護するためだけに使用することができます。

23.19.1 fail2ban とは?

fail2ban は `/var/log/apache/error_log` 等のログファイルを定期的に検索し、繰り返し何度もログインを試しているような不正アクセスの端緒を検出し、IP アドレス単位でのアクセス禁止を設定することができます。fail2ban ではファイアウオールルールを設定して、一定時間のアクセス禁止を実現しています。

fail2ban は Apache, SSH, Courier など、様々なサービスに対応するフィルタが提供されています。fail2ban を利用することで、不正アクセスを未然に防ぐことができるようになります。

23.19.2 SSH ブルートフォース攻撃を抑止するための fail2ban の使用

fail2ban をインストールして設定しておくことで、SSH ブルートフォース攻撃からサーバを保護することができます。

手順 23.2: SSH ブルートフォース攻撃を抑止するための FAIL2BAN の使用

1. fail2ban をインストールするには、下記のようなコマンドを入力して実行します:

```
# sudo zypper -n in fail2ban firewalld
```

2. fail2ban をインストールすると、既定の設定ファイルである `jail.conf` がインストールされますが、このファイルは fail2ban パッケージの更新で上書きされてしまう仕組みであることから、このファイルに直接設定を記述してはなりません。その代わりに、jail.local というファイルを作成して設定してください。fail2ban では両方のファイルを読み込むようになっています。

```
# cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

3. まずはエディタでファイルを開きます。

```
vi /etc/fail2ban/jail.local
```

4. 設定項目のうち、知っておかなければならない項目は 4 種類になります。

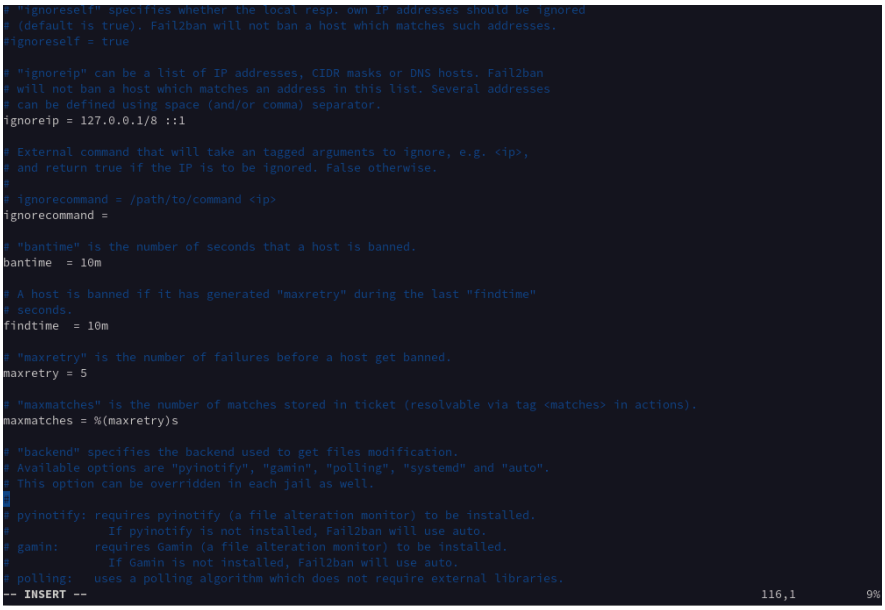


図 23.1: JAIL.LOCAL ファイルの設定

ignoreip
: アクセスを禁止したくない IP アドレスの一覧を指定します。既定では IPv4 アドレスの 127.0.0.1 と IPv6 アドレスの ::1 (いずれも自分自身を表すアドレスです) が設定されています。それ以外にもアクセスを禁止したくないアドレスがある場合は、ここに追加してください。

bantime
: 特定の IP アドレスからのアクセスを禁止する際、その禁止期間を指定します。設定値が m (分) または h (時間) で終わっていない場合、禁止期間は秒単位であるものとして解釈されます。また、-1 を設定すると、恒久的にアクセスを禁止することになります。

findtime
: 特定の IP アドレスからのアクセスを禁止するにあたっての持続時間を指定します。

maxretry
: 試行失敗の上限回数を指定します。

注記

同じ IP アドレスからのアクセスが findtime で指定した時間内に maxretry で指定した回数以上実施された場合、bantime で指定した時間だけアクセスを禁止することになります。なお、ignoreip に含まれる IP アドレスについては除外されます。

fail2ban では様々な種類の jail に対応しています。jail とは設定の単位で、それぞれが様々な種類の接続に対応しています。

5. fail2ban サービスを有効化や起動は、下記のようにして行います:
   サービスを有効化するには:

```
# systemctl enable fail2ban
```

   起動するには:

```
# systemctl start fail2ban
```

   サービスの状態を確認するには:

```
# systemctl status fail2ban.service
```

重要

なお、設定を変更した場合は fail2ban の再起動を実施しなければならないことに注意してください。

24 マスカレードとファイアウオール

改訂履歴

2025-06-12

Linux マシンがネットワーク内に配置されていれば、カーネルの機能を利用してネットワークパケットの制御を行い、内部ネットワークと外部ネットワークを区別して扱ったり、パケットそのものを書き換えたりすることができます。Linux では `netfilter` フレームワークが提供されていて、これによって様々なネットワークを個別に扱う効率的なファイアウオールを構築することができます。この `netfilter` フレームワークのフロントエンドが iptables で、ルールセットを定義するための汎用的なテーブル構造を作成することができます。これにより、ネットワークインターフェイスからの通過を許可するパケットを、正確に制御することができます。また、`firewalld` やそのグラフィカルインターフェイスである `firewall-config` を利用することでも、パケットフィルタを構築することができます。

openSUSE Leap 15.0 では新しいソフトウエアファイアウオールとして `firewalld` が提供されています。これは SuSEfirewall2 を置き換えるべく提供されているものです。本章では `firewalld` の設定方法のほか、古いバージョンの openSUSE Leap リリースからアップグレードしたユーザ向けに、SuSEfirewall2 から `firewalld` への移行方法をも説明しています。

24.1 iptables によるパケットフィルタリング

本章では、パケットフィルタリングに関してパケット寄りの詳細を説明しています。`netfilter` と `iptables` のコンポーネントは、ネットワークパケットのフィルタリングや内容の変更、そしてネットワークアドレス変換 (NAT) に関する制御を行う仕組みです。フィルタリングの判断条件とそれに付属する制御内容はチェインという形で保持され、パケットが届くごとにそれぞれ 1 つずつ判断を行います。`iptables` コマンドでは、これらのテーブルやルールの制御を行うことができます。

Linux カーネルでは 3 種類のテーブルを管理しています。これらはそれぞれ別々のフィルタ機能に対応しています:

filter
: このテーブルは、狭い意味での パケットフィルタリング を行うために提供されているテーブルで、様々なフィルタリングルールを設定して使用します。このテーブルでは、パケットの通過を許可する (`ACCEPT`) か拒否する (`DROP`) かを指定するなどして、ルールを構成します。

nat
: このテーブルは、パケットの発信元と送信先のアドレスをそれぞれ変更するために使用するテーブルです。これらの機能を利用することで、プライベートネットワークをインターネットに接続する際に使用する、マスカレード と呼ばれる処理を実現することができます。

mangle
: このテーブルは、IP ヘッダなどに含まれている様々な値 (Type Of Service など) を制御するために使用するテーブルです。

これらのテーブルに対して、それぞれ下記に示すチェインが用意されています:

PREROUTING
: このチェインは、届いた全てのパケットを処理するためのチェインです。

INPUT
: このチェインは、このシステムが送信先となっている全てのパケットを処理するためのチェインです。

FORWARD
: このチェインは、このシステムを通過していく全てのパケットを処理するためのチェインです。

OUTPUT
: このチェインは、このシステムから発信される全てのパケットを処理するためのチェインです。

POSTROUTING
: このチェインは、発信される全てのパケットを処理するためのチェインです。

図24.1「iptables: パケットの流通経路」の図には、Linux システム内におけるパケットの流通経路を示しています。単純化する目的でチェインの中にテーブルを記述していますが、実際にはこれらのチェインはテーブル内に含まれる形になっています。

最もわかりやすい例として、`eth0` のインターフェイスに対して、このシステム宛のパケットが届いた場合を考えてみます。パケットは最初、`mangle` テーブル内の `PREROUTING` チェインで処理され、その後 `nat` テーブル内の `PREROUTING` チェインで処理が行われます。その後、パケットの経路 (ルーティング) を判断する処理を行い、パケットの送信先が自分自身であるかどうかを判断します。今回は自分自身が宛先なので、`mangle` および `filter` の `INPUT` チェインで処理されたあと、`filter` テーブル内で許可される設定になっていれば、対応するプログラムに渡されることになります。

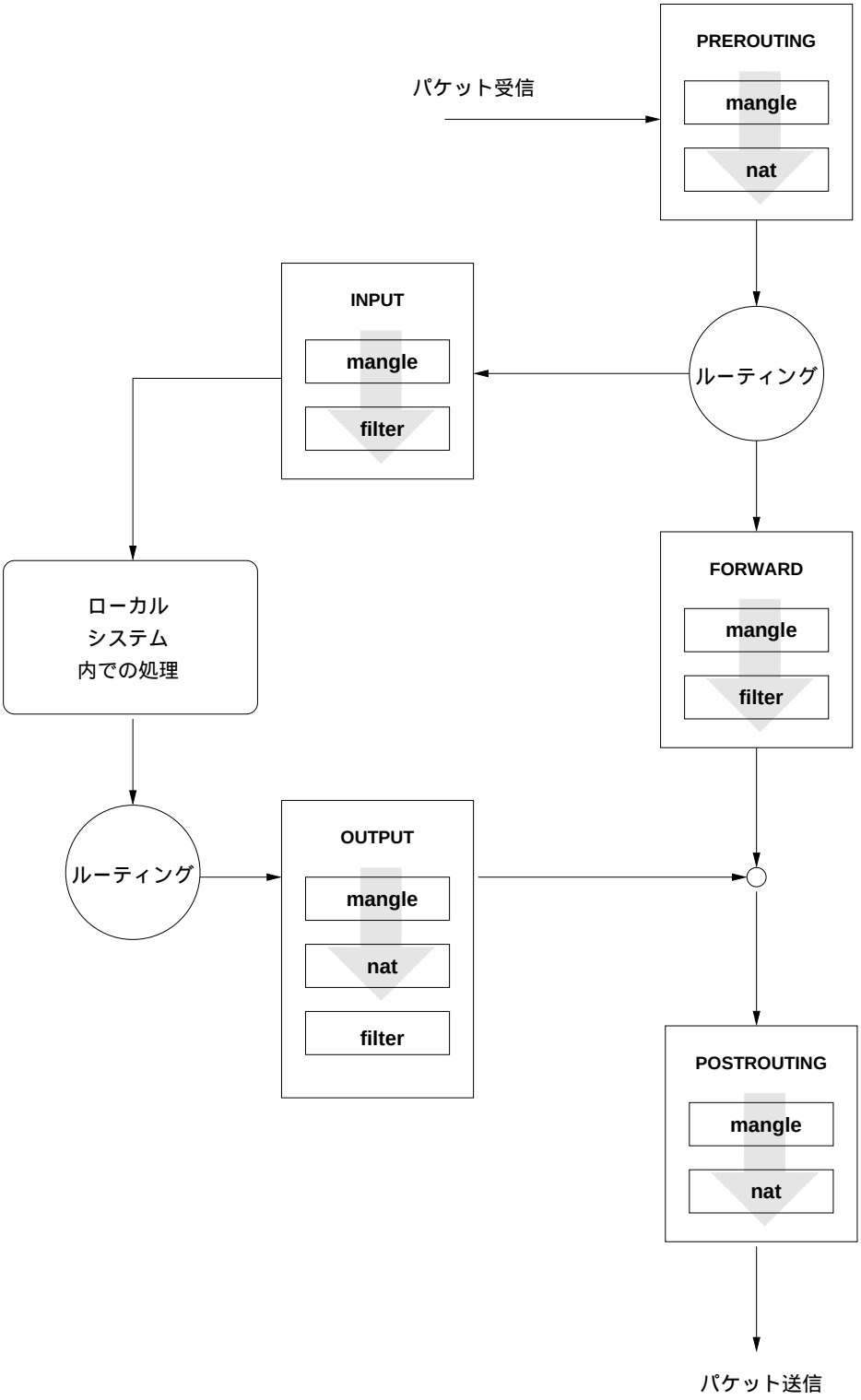


図 24.1: IPTABLES: パケットの流通経路

24.2 基本的なマスカレード処理

マスカレードとは Linux 固有の NAT (Network Address Translation; ネットワークアドレス変換) の一形態で、小規模な LAN をインターネットに接続する際に使用するものです。LAN 内のホストにはプライベート IP アドレスの範囲 (詳しくは『リファレンス』、第13章「ネットワークの基礎」、13.1.2項「ネットマスクとルーティング」をお読みください) 内にあるアドレスを割り当ててあり、インターネット側にはプロバイダなどから割り当てられたアドレスを使用している場合、インターネットに接続するには、LAN 側の IP アドレスをインターネット側のアドレスに変換する作業が必要となります。通常、この処理は LAN とインターネットを結ぶルータが行うべき作業です。一般的な構成は非常にシンプルで、ルータには複数のネットワークインターフェイス (通常はネットワークカード) があり、それとは別にインターネットに接続しているインターフェイスが用意されています。LAN 内のコンピュータは、このルータをデフォルトゲートウエイとして設定することで、外部 (つまりインターネット) の世界に出て行くことができるようになっています。

重要: 正しいネットマスクの設定について

ネットワークを設定する際、ブロードキャストアドレスとネットマスクは、LAN 内の全てのコンピュータで同じ値を設定する必要があります。設定の異なるコンピュータが存在すると、正しい経路を見つけられなくなってしまうことがあります。

上述のとおり、LAN 内のコンピュータがインターネットアドレス宛にパケットを送信する場合、デフォルトゲートウエイとして設定したルータに対して通信が行われます。ただし、ルータとして動作するにはあらかじめパケットを転送できるように設定しなければなりません。セキュリティ上の理由から、既定のインストールでは転送機能は有効化されません。有効化を行うには、`/etc/sysctl.conf` ファイル内に `net.ipv4.ip_forward = 1` という行を追加してください。YaST から設定を行いたい場合は、`yast routing ip-forwarding on` と入力して実行してください。

接続先の (インターネット側の) ホストからは、ルータまでは到達可能であるものの、本来のパケット送信元である内部ネットワーク内のコンピュータまでは到達できないようにしています。このような仕組みをマスカレード (仮装や変装などの意味) と言います。アドレス変換の仕組みにより、ルータは応答パケットに対する第一の送信先となります。ルータ側ではこのパケットがどれに紐付くものであるのかを探して、本来の (LAN 側の) 送信元に配送します。

内部側のネットワークパケットの配送はマスカレード処理用のテーブルに従って行われるため、外側 (インターネット側) から内側 (LAN 側) に接続を開くことはできなくなっています。これは、外部から内部に接続しようとしても、テーブル内に対応する項目が存在しないためです。これに加えて、既に確立されている接続に対しては、テーブル内でその状態を管理するようにもなっています。これにより、他の接続で使用されてしまうようなことがないようになっています。

このような構造になっていることから、ICQ, cucme, IRC (DCC, CTCP), FTP (PORT モード) などのプロトコルを使用した場合、接続がうまくいかない場合があります。Web ブラウザや標準的な FTP プログラムなど、多数のプログラムは PASV モードを使用しますので、フィルタリングやマスカレードという観点では問題が発生しにくくなっています。

24.3 基本的なファイアウオール処理

ファイアウオール とはおそらく、ネットワーク間を流れるデータを制御する仕組みとして最もよく使われている用語でしょう。厳密に言うと、本章で説明している内容は パケットフィルタ と呼ばれる仕組みです。パケットフィルタはプロトコルやポート、IP アドレスなどの条件を元に、データの流れを管理するための仕組みです。この仕組みによって、ネットワークに対して届いて欲しくないパケットを防ぐことができます。たとえば Web サーバなど、一般的に広くアクセスを許すサービスを提供するような場合、明示的に対応するポートを開くように設定する必要があります。しかしながら、一般にパケットフィルタではパケットの内容までは判断できませんので、アドレス情報などが間違っていない限り、Web サーバとの通信はできてしまいます。つまり、Web サーバ内で動作する CGI プログラムを不正に操作するようなパケットが届いたとしても、パケットフィルタではそれを防ぐことができず、通過させてしまうことになります。

より効率的で複雑な構成を取るとすれば、アプリケーションゲートウエイやプロキシなどとパケットフィルタを組み合わせて、多階層の防御を行う方法があります。この場合、パケットフィルタは許可していないポート宛の通信を拒否し、それを通過したパケットのみがアプリケーションゲートウエイやプロキシに渡されます。このゲートウエイやプロキシは、サーバとクライアントとの間の仲介役として動作し、アプリケーションが使用するプロトコルを直接解釈してやり取りを行います。このようなアプリケーションゲートウエイやプロキシとして代表的なものが Squid です。こちらは HTTP/FTP のプロキシサーバとして動作するもので、HTTP ページや FTP ファイルへのアクセスは、プロキシキャッシュ内に記憶していればその内容をそのまま提供し、キャッシュ内に存在していなければ直接アクセスしてページを提供することができます。

下記の章では、openSUSE Leap に付属しているパケットフィルタ機能について説明しています。パケットフィルタやファイアウオールに関する詳細は、Firewall HOWTO (https://tldp.org/HOWTO/Firewall-HOWTO.html) (英語) でも提供されています。

24.4 firewalld

注記: firewalld による SuSEfirewall2 の置き換えについて

openSUSE Leap 15.0 15 では SuSEfirewall2 の置き換えとして `firewalld` が新しい既定のファイアウオールになっています。openSUSE Leap 15.0 以前のバージョンからアップグレードしている場合は、SuSEfirewall2 が従来どおりインストールされたままの状態になりますので、手作業で `firewalld` に移行を行ってください (詳しくは 24.5項「SuSEfirewall2 からの移行」をお読みください) 。

`firewalld` はシステム内にある `iptables` のルールを管理し、D-Bus インターフェイス経由で操作することのできるデーモンです。対応するコマンドラインユーティリティ `firewall-cmd` と、グラフィカルユーザインターフェイス `firewall-config` がそれぞれ用意されています。`firewalld` は裏で動作する仕組みであることから、たとえば仮想マシンのネットワーク設定を行うなど、iptables のルールに対してアプリケーション側から変更要求を送信することができるようになっています。

`firewalld` ではセキュリティゾーンと呼ばれる仕組みを提供しています。既定の設定では、`internal` (内部) や `public` (公共) などのゾーンが用意されています。それ以外のゾーンが必要な場合でも、管理者側で自由に設定することができます。各ゾーンにはそれぞれ対応する iptables のルールを設定することができます。また、ネットワークインターフェイスはいずれか 1 つのゾーンに属するように設定するほか、発信元のアドレスをベースにして接続ごとにゾーンを割り当てることもできます。

それぞれのゾーンは信頼レベルを表していて、たとえば `public` はインターネットや無線ホットスポットに対して設定すべきゾーンで、最も信頼レベルの低いネットワークを表しています。逆に `internal` は、家庭内のネットワークや企業内のネットワークなど、最も信頼できるネットワークを表しています。このように信頼レベルを分けることで、接続されるネットワークごとに異なるルールを適用できるようにしています。

`firewalld` の既定で設定されているゾーンとその意味について、詳しくは https://www.firewalld.org/documentation/zone/predefined-zones.html (英語) にあるマニュアルをお読みください。

注記: ゾーンを割り当てない場合の動作について

ネットワークインターフェイスに対する設定を行っていない場合、そのインターフェイスには何もゾーンが割り当てられません。この場合、`firewall-cmd --get-default-zone` で表示される既定のゾーンが割り当てられます。既定のゾーンを何も設定していない場合、既定値は `public` (公共) になっています。

`firewalld` のパケットフィルタリングでは、接続の開始側については基本的に許可される設定になっています。接続の開始側とは、手元のコンピュータから相手のコンピュータに対して、接続を始める種類の接続を意味しています。逆に、接続を受け入れる側の場合は、ゾーンの設定で明示的に許可を指定しない限り、アクセスが拒否されるようになっています。そのため、何らかのサービスの接続を受け入れるインターフェイスに対しては、適切なゾーンを指定しておいて、ゾーン内で必要なサービスの許可を設定してください。

`firewalld` では、もう 1 つ重要な考え方があります。それは 一時的な 設定と 恒久的な 設定です。一時的な設定とは現在動作中のルールそのものを表していて、恒久的な設定とは次回の `firewalld` の再起動で適用される、保存済みのルールを表しています。これにより、次に `firewalld` を再起動するまでの間、一時的に適用するルールを設定することができますので、実験的にルールを追加してみて、うまくいかなければ元に戻すと言ったことが容易に実現できるようになっています。なお、設定を変更する際には、どちらの設定を編集しているのかをよくお確かめください。こちらについての詳細は、24.4.1.2項「一時的な設定と恒久的な設定の違い」で説明しています。

`firewalld` をグラフィカルユーザインターフェイス `firewall-config` で操作したい場合は、ドキュメンテーション (https://www.firewalld.org/documentation/utilities/firewall-config.html) (英語) をお読みになることをお勧めします。以下の章では、コマンドライン経由で利用する `firewall-cmd` で `firewalld` の制御を行っています。

24.4.1 コマンドラインからのファイアウオール設定

24.4.1.1 ファイアウオールの起動

`firewalld` は既定でインストールされ、有効化されます。`firewalld` は通常の `systemd` サービスとして設定されていますので、`systemctl` や サービスマネージャ で設定することができます。

重要: 自動的なファイアウオール設定について

インストールを行うと、YaST は `firewalld` を自動的に起動して、全てのインターフェイスを既定値の `public` ゾーンに割り当てます。何らかのサーバアプリケーションをインストールしていて、システム内で有効化している場合は、YaST 側で［ファイアウオールでポートを開くインターフェイス］でインターフェイスを選択するか、もしくは［ファイアウオールでポートを開く］を選択することで、自動的にファイアウオールの設定を行うことができます。また、サーバモジュールによっては、［ファイアウオールの詳細］ボタンを押すことで、追加のサービスやポートを開くように設定することもできます。

24.4.1.2 一時的な設定と恒久的な設定の違い

既定では、`firewall-cmd` は一時的な設定に対して作用し、恒久的な設定は変更しません。恒久的に設定を変更したい場合は、`--permanent` オプションを追加してください。なお、`--permanent` オプションを指定した場合、一時的な設定には反映されず、即時に反映されることもありません。現時点では両方に対して一回で設定する機能は用意されていませんが、一時的な設定を恒久的な設定にコピーすることは可能です。コピーを行うには、下記のように実行します:

```
# firewall-cmd --runtime-to-permanent
```

上記を実行することで、現時点での一時的な設定を恒久的な設定として書き込むことができます。これにより、コマンドラインやその他のプログラムなどで一時的な設定に書き込んだ場合でも、それを恒久的に保存することができるようになります。ただし、より安全に作業を行いたい場合は、恒久的な設定に新しいルールを追加して、`firewalld` を再起動することで有効化することもできます。



注記

なお、既定のゾーンなど一部の設定は、一時的な設定と恒久的な設定で異なる管理を行っていません。これらの設定を変更した場合は、両方の設定に対して反映されます。

逆に、一時的に設定を変更していて、その変更を取り消したい場合は、下記のいずれかを実行してください。前者は `firewalld` のコマンドラインインターフェイスを介して、後者は `systemd` を介して実施しています:

```
# firewall-cmd --reload
```

```
# systemctl reload firewalld
```

以下の章では、記述を簡略化する目的で、常に一時的な設定に対して作業を行っています。恒久的な設定に書き込みたい場合は、対応するオプションを付与して実行してください。

24.4.1.3 インターフェイスのゾーンへの割り当て

ゾーンが割り当てられている全てのネットワークインターフェイスを表示したい場合は、下記のように実行します:

```
# firewall-cmd --zone=public --list-interfaces
eth0
```

同様に、インターフェイス名を指定してゾーンを問い合わせることもできます:

```
# firewall-cmd --get-zone-of-interface=eth0
```

```
public
```

下記のコマンドラインは、インターフェイスをゾーンに割り当てるコマンドラインです。なお、`--add-interface` は `eth0` がどのゾーンにも割り当てられていない場合にのみ動作します。`--change-interface` は、どちらの場合でも動作します。この場合、既にゾーンへの割り当てが存在していると、そのゾーンに対する割り当てを外す動作になります:

```
# firewall-cmd --zone=internal --add-interface=eth0
# firewall-cmd --zone=internal --change-interface=eth0
```

なお、明示的に `--zone` オプションでゾーンを指定しない場合は、既定のゾーンが指定されたものとして扱われます。下記のコマンドは、それぞれ既定のゾーンが何であるのかを問い合わせ、既定のゾーンを特定のゾーンに設定するコマンドです:

```
# firewall-cmd --get-default-zone
dmz
# firewall-cmd --set-default-zone=public
```



重要

特定のゾーンに割り当てられていないネットワークインターフェイスは、自動的に既定のゾーンに割り当てられているものとして扱われます。また、既定のゾーンの変更は即時に反映され、かつ恒久的にも保存されます。そのため、`internal` などのセキュリティの低いゾーンを既定のゾーンに設定したりしてはなりません。これにより、お使いのコンピュータを不用意に危険にさらすことになってしまいます。たとえば USB イーサネットなどのホットプラグ型インターフェイスを接続した場合、既定のゾーンが `internal` ゾーンであるとすると、接続を開始した直後からネットワークの接続を信頼してしまうことになります。

また、ゾーンのインターフェイス一覧には、明示的にゾーンが指定されているもの以外は表示されないことにも注意してください。現時点では、ゾーンの割り当てのないインターフェイスを表示するコマンドはありません。このような理由から、通常使用するネットワークインターフェイスについては、必ずゾーンを割り当てるようにしておいてください。

24.4.1.4 ネットワークサービスにアクセスできるようにする設定

`firewalld` には サービス という考え方があります。サービスは 1 つ以上のポートやプロトコルの定義から構成されるもので、Web やメールサーバのプロトコルなどのように、まとめて使用することになるポートやプロトコルを一括で指定できるように構成しているものです。下記のコマンドでは、サービス名を指定してその詳細情報を取得しています:

```
# firewall-cmd --get-services
```

```
[...] dhcp dhcpv6 dhcpv6-client dns docker-registry [...]
# firewall-cmd --info-service dhcp
dhcp
  ports: 67/udp
  protocols:
  source-ports:
  modules:
  destination:
```

これらのサービス定義は、ゾーン内で特定の機能をまとめて有効化したい場合に使用します。たとえば下記のようにコマンドを入力して実行すると、`internal` ゾーンに対して HTTP (Web サーバ) のポートを開くことができます:

```
# firewall-cmd --add-service=http --zone=internal
```

特定のゾーンに対してサービスの許可を取り消したい場合は、逆のサブコマンドである `--remove-service` を使用します。新しくサービスを定義したい場合は、`--new-service` サブコマンドを使用します。詳しくは https://www.firewalld.org/documentation/howto/add-a-service.html (英語) をお読みください。

ポート番号を直接指定してポートを開きたい場合は、下記のように入力して実行します。下記の例では、`internal` ゾーンに対して TCP ポート 8000 を開きます:

```
# firewall-cmd --add-port=8000/tcp --zone=internal
```

ポートを閉じたい場合は、逆のサブコマンドである `--remove-port` をお使いください。



ヒント: 一時的にサービスやポートを開く設定について

`firewalld` には特定のサービスやポートを指定した時間内のみ開くことのできる、`--timeout` というパラメータが用意されています。これは簡易的なテストに有用な仕組みで、サービスやポートを閉じ忘れることを防止することができます。たとえば `internal` ゾーンに対して 5 分間だけ `imap` サービスを許可したい場合は、下記のように実行します:

```
# firewall-cmd --add-service=imap --zone=internal --timeout=5m
```

24.4.1.5 ロックダウンモード

`firewalld` には、そのファイアウォールのルールが有効な間は変更を禁止することのできる、ロックダウンモード が用意されています。アプリケーション側からは D-Bus インターフェイスを介して自動的にファイアウォールのルールを変更することができてしまうほか、PolicyKit のルールに従って一般

ユーザからも同じことを実施できてしまいますので、状況によっては変更の防止が有用となることがあります。詳しくは https://fedoraproject.org/wiki/Features/FirewalldLockdown をお読みください。

なお、ロックダウンモードは実際のセキュリティを提供するものではなく、不用意に設定を変更してしまったりする行為を防止するための仕組みです。`firewalld` でのロックダウンモードは、不正アクセスからの防御にはなっていないことに注意してください。詳しくは https://seclists.org/oss-sec/2017/q3/139 (英語) をお読みください。

24.4.1.6 独自の `iptables` ルールの追加

`firewalld` はコンピュータ内の `netfilter` ルールに対して、排他的にアクセスすることによって成り立っています。そのため、`iptables` のようなコマンドで、ファイアウオールのルールを直接書き換えたりすべきではありません。これにより、`firewalld` の動作が不安定になってしまったり、セキュリティや機能性が損なわれてしまったりする危険性があります。

`firewalld` ではカバーしていない独自のファイアウオールルールを追加する必要がある場合は、2 種類ある方法のうちのいずれかを実施してください。1 つは `iptables` で設定する内容をそのまま `firewalld` に投げる方法で、この場合は `--direct` オプションを指定します。このとき、`iptables` を実行する際のテーブルやチェイン、優先度に関する指定を行う必要があります。下記の例では、FORWARD チェインに対して独自の追跡 (tracking) ルールを追加しています:

```
# firewall-cmd  --direct --add-rule ipv4 filter FORWARD 0 -i eth0 -o eth1 \
    -p tcp --dport 80 -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
```

それ以外にも、`firewalld` では リッチルール と呼ばれる仕組みにも対応しています。この方法のほうが `iptables` のルールを簡単に指定することができます。詳しくは https://www.firewalld.org/documentation/man-pages/firewalld.richlanguage.html (英語) をお読みください。下記の例では、特定の発信元アドレスからの IPv4 パケットを drop (処理せず落とす) する設定を行っています:

```
# firewall-cmd --zone=public --add-rich-rule='rule family="ipv4" \
    source address="192.168.2.4" drop'
```

24.4.1.7 ルーティング／フォワード／マスカレード

`firewalld` は完全機能のルータとして動作するようには設計されていません。ですが、一般的な家庭用ルータ程度の設定であれば可能です。企業内の本番環境で使用されるようなルータの場合は、`firewalld` を使用すべきではありません。ルータやファイアウオールとして専用に設計された機器をお使いください。下記の章では、`firewalld` 内でルーティングを扱うにあたっての、いくつかの情報源を示しています:

- IP プロトコルの経路制御を有効化するのに必要な情報は、24.2項「基本的なマスカレード処理」に概要があります。
- たとえば `external` ゾーンに対して IPv4 マスカレードを有効化したい場合は、下記のコマンドを実行します。

```
# firewall-cmd --zone=external --add-masquerade
```

- `firewalld` ではポート転送の設定も行うことができます。たとえば下記のコマンドを実行すると、ポート 80 宛に届いた通信を他のホストに転送することができます:

```
# firewall-cmd --zone=public \
    --add-forward-port=port=80:proto=tcp:toport=80:toaddr=192.168.1.10
```

24.4.2 動的にポートを割り当てるタイプのサービスへのアクセス

ネットワークサービスによっては、特定のポート番号で待ち受けていないものがあります。この場合、`portmapper` や `rpcbind` のプロトコルをベースにして動作しています (以降は `rpcbind` と表現します) 。これらのサービスが開始されると、ローカル側のポートをランダムに選択したあと、`rpcbind` と通信を行って、このポートを既知のものとして使用するようにします。`rpcbind` それ自身は特定の (既知の) ポート上で動作します。ネットワーク上離れたシステムからは、`rpcbind` に対して問い合わせを行い、どのポートで動作しているのかを確認します。現在、この仕組みはそれほど多くのプログラムで使用していないものですが、Network Information Services (NIS; `ypserv` および `ypbind`) と Network File System (NFS) バージョン 3 で使用されています。



注記: NFSv4 について

新しい NFSv4 では、TCP ポート 2049 という既知のポート単独で動作するようになっています。プロトコルバージョン 4.0 では、カーネルのパラメータ `fs.nfs.nfs_callback_tcpport` で固定のポートを指定する必要があります (詳しくは 例24.1「`/etc/modprobe.d/60-nfs.conf` 内での `nfs` カーネルモジュールに対するコールバックポートの設定」をお読みください) 。プロトコルバージョン 4.1 以降であれば、この設定も不要となっています。

`rpcbind` プロトコルを使用する仕組みでは、動的にポートが設定されるため、ファイアウォール経由でサービスにアクセスできるようにすることが難しくなっています。`firewalld` では、それ単体でこれらのサービスに対応することができません。手作業で作業を行う場合は、24.4.2.1項「ポートを固定するための設定」をお読みください。それ以外にも、openSUSE Leap にはヘルパースクリプトが用意されています。詳しくは 24.4.2.2項「固定のポートを設定するための `firewall-rpcbind-helper` の使用」をお読みください。

24.4.2.1 ポートを固定するための設定

問題を解決するための方法のうちの 1 つとして、各ネットワークサービスの使用するポートを固定する方法があります。これを設定することで、`firewalld` でポートを開くように設定することができるようになり、全てが問題なく動作するようになります。ポート番号の指定は自由に決めることができますが、他のサービスで使用されてしまうことを防ぐため、既知の (いわゆる well-known な) ポート範囲以外から設定しておくことをお勧めします。NIS や NFSv3 のサービスに対する設定項目の一覧については、表24.1「ポートを固定させるための主な sysconfig 変数」をお読みください。ただし、NIS や NFS の設定によっては、これら全てのポートを設定する必要がない場合もあります。

表 24.1: ポートを固定させるための主な SYSCONFIG 変数

ファイルパス	変数名	値の例
`/etc/sysconfig/nfs`	MOUNTD_PORT	21001
	STATD_PORT	21002
	LOCKD_TCPPORT	21003
	LOCKD_UDPPORT	21003
	RQUOTAD_PORT	21004
`/etc/sysconfig/ypbind`	YPBIND_OPTIONS	-p 24500
`/etc/sysconfig/ypserv`	YPXFRD_ARGS	-p 24501
	YPSERV_ARGS	-p 24502
	YPPASSWDD_ARGS	--port 24503

設定を反映させるには、対応するサービスをそれぞれ再起動する必要があります。なお、`rpcinfo -p` と入力して実行することで、割り当てられた rpcbind のポートを表示することもできます。正しく設定がされていれば、ここに固定のポートが表示されるはずです。

ユーザスペース内で動作しているネットワーク向けのポート設定とは別に、Linux カーネルが NFS に対して直接使用するポートも存在しています。これらのポートのうちの 1 つは `nfs_callback_tcpport` と呼ばれます。このポートは、バージョン 4.1 以前の NFS プロトコルでのみ必要となります。ここで使用するポートは、sysctl の `fs.nfs.nfs_callback_tcpport` で設定することができます。この sysctl ノードは、NFS マウントが有効な場合にのみ動的に現れます。そのため、この設定はカーネルモジュールのパラメータとして設定しておくことが最適です。これは 例24.1「/etc/modprobe.d/60-nfs.conf 内での nfs カーネルモジュールに対するコールバックポートの設定」で示している設定で実現することができます。

例 24.1: /etc/modprobe.d/60-nfs.conf 内での nfs カーネルモジュールに対するコールバックポートの設定

```
options nfs callback_tcpport=21005
```

この設定を反映させるのに最も簡単な方法は、マシンの再起動です。マシンを再起動しないで適用するには、NFS 関連のサービスを全て停止させて `nfs` カーネルモジュールを読み込みなおす必要があります。なお、現時点での NFS コールバックポートを確認したい場合は、`cat /sys/module/nfs/parameters/callback_tcpport` と入力して実行してください。

このようにして RPC 関連のポートを固定化したら、あとは新しい `firewalld` サービス定義を作成して、使用するようにすると便利です。このサービス定義は関連する全てのポートを含むもので、これによって一括でポートを許可するように設定することができます。例24.2「NFS 向けの新しい firewalld RPC サービスの設定コマンド」には、新しくサービスを定義して、それらの固定ポートをサービス内に設定し、使用するまでの手順を示しています。

例 24.2: NFS 向けの新しい firewalld RPC サービスの設定コマンド

```
# firewall-cmd --permanent --new-service=nfs-rpc
# firewall-cmd --permanent --service=nfs-rpc --set-description="NFS related, statically
 configured RPC ports"
# TCP/UDP ポートを順序通りにサービスに追加する
# for port in 21001 21002 21003 21004; do
    firewall-cmd --permanent --service=nfs-rpc --add-port ${port}/udp --add-port ${port}/
tcp
done
# コールバックポートは TCP のみ
# firewall-cmd --permanent --service=nfs-rpc --add-port 21005/tcp

# 新しく定義したサービスの詳細表示
# firewall-cmd --info-service=nfs-rpc --permanent -v
nfs-rpc
  summary:
  description: NFS and related, statically configured RPC ports
  ports: 4711/tcp 21001/udp 21001/tcp 21002/udp 21002/tcp 21003/udp 21003/tcp 21004/udp
 21004/tcp
  protocols:
```

```
  source-ports:
  modules:
  destination:

# firewalld を再読み込みさせて、新しいサービス定義を利用できるようにする
# firewall-cmd --reload

# 新しいサービス定義を internal ゾーンに追加する
# firewall-cmd --add-service=nfs-rpc --zone=internal
```

24.4.2.2 固定のポートを設定するための firewall-rpcbind-helper の使用

上記の章で示した固定のポートを設定するための手順は、SUSE が提供する `firewall-rpc-helper.py` というツールを使用することで、より簡単に行うことができます。このツールは、`zypper in firewalld-rpcbind-helper` でインストールすることができます。

このツールは、上記の章で実施したサービス設定を対話に設定するツールです。このほか、現時点でのポートの割り当てを表示することができるほか、スクリプト内で使用することもできるようになっています。詳しくは `firewall-rpc-helper.py --help` を実行して出力された内容をお読みください。

24.4.2.3 既定のバックエンドとしての nftables の使用

`firewalld` では、既定のバックエンドとして nftables を使用します。nftables は Linux `netfilter` プロジェクトが提供するフレームワークで、パケットフィルタリングのほか、ネットワークアドレス変換 (NAT) 等の機能を提供します。また、nftables はコネクショントラッキングやユーザスペースキューイング、ロギングやフック機能などの既存の netfilter サブシステムを流用した仕組みで、iptables, ip6tables, arptables, ebtables, ipset 等の代替として機能するものとなっています。

nftables を使用することの利点としては、下記のようなものがあります:

- IPv4, IPv6 の両プロトコルに対して 1 つのフレームワークで対応できます。
- ルールはルールセット全体の再読み込みや更新、保存などを行うことなく適用することができます。
- イベントの追跡は nft ツールで、ルールセットのデバッグやトレースは nftrace でそれぞれ行うことができます。
- `nft` コマンドラインツールは、ルールを netlink 形式の VM バイトコードにコンパイルします。逆にルールセットを取り出す際は、VM バイトコードを元のルールセット表記に翻訳して出力します。

基本的な nftables の設定ファイル:

```
> cat /etc/nftables.conf
  #!/usr/sbin/nft -f

  flush ruleset

  # IPv4 と IPv6 の両方にマッチします
  table inet filter {
      # チェイン名は任意に設定できます。
      # どのトラフィックを処理するのかについては、
      # type 以下で指定します。
      chain input {
          type filter hook input priority 0; policy accept;
      }
      chain forward {
          type filter hook forward priority 0; policy accept;
      }
      chain output {
          type filter hook output priority 0; policy accept;
      }
  }
```

上記の例では、全てのチェインで accept ポリシーを設定していることから、全てのトラフィックを許可することになります。もちろん実際には全てのトラフィックが既定で拒否されますので、必要なトラフィックのみを許可するように設定するのが適切です。

マスカレードを使用する nftables の設定:

```
> cat /etc/nftables.conf
  #!/sbin/nft -f

  flush ruleset

  table ip nat {
    chain prerouting {
      type nat hook prerouting priority 0; policy accept;
    }

    # WAN 宛に送信される全てのトラフィックに対して、経路の判断後に
    # 発信元アドレスを WAN インターフェイスのプライマリ IP アドレスに置き換えます。
    chain postrouting {
      type nat hook postrouting priority 100; policy accept;
      oifname "wan0" masquerade
    }
  }
```

固定 IP の環境であれば、マスカレードではなくソース NAT (SNAT) を使用したほうが少しだけ高速になります。これは、それぞれの発信パケットの置き換え時にあらかじめ IP アドレスが分かっていたほうが、手っ取り早く処理できるためです。

いくつかのルールを適用した nftables の設定ファイル:

```
>  cat /etc/nftables.conf
    #!/usr/sbin/nft -f

flush ruleset

table inet filter {
    chain base_checks {
        ## 接続状態の確認を行う追加のルールセットです。
        # 確立済みの接続や関連する接続は自動的に許可します
        ct state {established, related} accept;

        # 不正な接続に対しては早めに廃棄します
        ct state invalid drop;
    }

    chain input {
        type filter hook input priority 0; policy drop;

        # ループバック接続を許可します
        iif "lo" accept;

        jump base_checks;

        # ICMP, IGMP を許可します
        ip6 nexthdr icmpv6 icmpv6 type { echo-request, echo-reply, packet-too-big, time-
exceeded, parameter-problem, destination-unreachable, packet-too-big, mld-listener-query,
 mld-listener-report, mld-listener-reduction, nd-router-solicit, nd-router-advert, nd-
neighbor-solicit, nd-neighbor-advert, ind-neighbor-solicit, ind-neighbor-advert, mld2-
listener-report } accept;
        ip protocol icmp icmp type { echo-request, echo-reply, destination-unreachable,
 router-solicitation, router-advertisement, time-exceeded, parameter-problem } accept;
        ip protocol igmp accept;

        # ログを記録して拒否するテストです
        counter log prefix "[nftables] input reject " reject;
    }
    chain forward {
        type filter hook forward priority 0; policy accept;
    }
    chain output {
        type filter hook output priority 0; policy accept;
    }
}
```

上記の例は、下記のようになっています:

- input チェインの既定値は drop (廃棄) で、末尾に reject (拒否) ルールが設定されているため、全ての到着トラフィックをブロックする設定になっています。
- `localhost` インターフェイスの全てのトラフィックを許可しています。
- `base_checks` チェインでは、全てのパケットに対して既存の接続に関係したものであるかどうかを確認しています。これにより、こちらからの接続に対する応答はブロックされないようになります。
- ICMP, IGMP の各パケットは、タイプ名を指定したセットを使用することでそれぞれ許可しています。
- カウンタは、直近にリセットされた時点からの合計パケット数や合計バイト数を保持する仕組みです。nftables の場合、カウンタを実装したいルールでそれぞれ指定しなければなりません。

24.4.2.3.1 さらなる情報

nftables に関する詳細については、下記をお読みください:

- 提供元のドキュメンテーション: https://www.netfilter.org/projects/nftables/index.html
- nftables のマニュアルページ: https://www.netfilter.org/projects/nftables/manpage.html

24.5 SuSEfirewall2 からの移行



注記: AutoYaST での `firewalld` 設定の作成について

AutoYaST で `firewalld` の設定を作成する方法について、詳しくは AutoYaST ガイド 内の ファイアウオール設定 の章をお読みください。

openSUSE Leap 15.0 以前の任意のバージョン からopenSUSE Leap 15.7 にアップグレードした場合は、SuSEfirewall2 はそのまま保持され、動作し続けるようになっています。SuSEfirewall2 から `firewalld` への自動的な移行の仕組みは提供されていないことから、移行作業は手作業で実施する必要があります。ただし、`firewalld` には `susefirewall2-to-firewalld` と呼ばれる移行支援ツールが用意されていますので、こちらをお使いいただくことができます。ですが、このツールはご利用いただいている SuSEfirewall2 の設定の複雑さによって、全く問題なく移行できる場合から、全く支援にならない場合までさまざまな結果になります。通常は部分的に移行が行われ、残りは手作業で `firewalld` の設定を調整して対応する必要が発生します。

このツールが生成した設定は、SuSEfirewall2 と同じような動作になるようになります。そのような背景から、`firewalld` の機能を完全に発揮するには、既存の設定の移行ではなく全く新しく設定を作成し直すことをお考えください。また、`susefirewall2-to-firewalld` に何もオプションを付けずに実行した場合は、恒久的な変更は行われなくなっていますが、ネットワーク上離れた場所から管理を行っているような場合は、移行によって締め出されてしまうこともあります。

`susefirewall2-to-firewalld` をインストールして実行するには、下記の手順を実施します:

```
# zypper in susefirewall2-to-firewalld
# susefirewall2-to-firewalld
INFO: Reading the /etc/sysconfig/SuSEfirewall2 file
INFO: Ensuring all firewall services are in a well-known state.
INFO: This will start/stop/restart firewall services and it's likely
INFO: to cause network disruption.
INFO: If you do not wish for this to happen, please stop the script now!
5...4...3...2...1...Lets do it!
INFO: Stopping firewalld
INFO: Restarting SuSEfirewall2_init
INFO: Restarting SuSEfirewall2
INFO: DIRECT: Adding direct rule="ipv4 -t filter -A INPUT -p udp -m udp --dport 5353 -m
 pkttype
  --pkt-type multicast -j ACCEPT"
[...]
INFO: Enabling direct rule=ipv6 -t filter -A INPUT -p udp -m udp --dport 546 -j ACCEPT
INFO: Enabling direct rule=ipv6 -t filter -A INPUT -p udp -m udp --dport 5353 -m pkttype
  --pkt-type multicast -j ACCEPT
INFO: Enable logging for denied packets
INFO: ############################################################
INFO:
INFO: The dry-run has been completed. Please check the above output to ensure
INFO: that everything looks good.
INFO:
INFO: ############################################################
INFO: Stopping firewalld
INFO: Restarting SuSEfirewall2_init
INFO: Restarting SuSEfirewall2
```

このスクリプトを実行するとさまざまな出力が生成されますが、これをファイルに保存しておくことで、後からゆっくり読むことができるようになります:

```
# susefirewall2-to-firewalld | tee newfirewallrules.txt
```

このスクリプトには、下記のオプションが用意されています:

`-c`

変更点を恒久的に保存します。提示された設定で問題が無い場合にのみ、このオプションを指定してください。また、既存の `firewalld` の設定はリセットされますので、必ずバックアップを採取しておいてください!

`-d`

動作に関する詳細な情報を出力します。このスクリプトのバグを報告したい場合などに有用ですが、機密情報が含まれることもあります。

`-h`

このメッセージを表示します。

`-q`

出力を抑制します。エラーメッセージも表示されなくなります！

`-v`

冗長モードを指定します。警告メッセージやその他の情報に関するメッセージが出力されるようになります。

24.6 さらなる情報

`firewalld` パッケージに対する最新の情報と、その他のドキュメンテーションについては、`/usr/share/doc/packages/firewalld` 内で提供されています。netfilter および iptables プロジェクトの Web ページは https://www.netfilter.org にあります。こちらには、多数の言語に対応した iptables 関連の文書が多数用意されています。

25 VPN サーバの設定

改訂履歴

2025-06-12

今やインターネット接続は手頃でどこでも利用できるようになっています。しかしながら、全ての接続で機密が守られる保証はありません。この場合、仮想プライベートネットワーク (Virtual Private Network; VPN) を構築することで、インターネットや Wi-Fi などの機密が守られないネットワークを経由して、社内や家庭内などにアクセスすることができるようになります。この VPN には様々な実装があり、様々な目的で使用されます。本章では OpenVPN (https://openvpn.net) と呼ばれる、インターネットと社内／家庭内を結ぶ実装について説明しています。

25.1 考え方の概要

本章では VPN に関するいくつかの用語を使用していますので、まずはその説明から行います。

25.1.1 用語

エンドポイント
: VPN の接続の「両端」を表す用語です。VPN サーバとクライアントの両方を意味しています。

TAP デバイス
: TAP デバイスはイーサネットデバイス (イーサネットフレームなどの OSI モデルにおけるレイヤ 2) を疑似するデバイスです。TAP デバイスはブリッジ型の VPN 接続で使用され、イーサネットのフレームを転送します。

TUN デバイス
: TUN デバイスは IP ネットワーク (IP パケットなどの OSI モデルにおけるレイヤ 3) を疑似するデバイスです。TUN デバイスはルーティング型の VPN で使用され、IP のフレームを転送します。

トンネル
: 2 つのエンドポイントを結ぶ通信経路を意味する用語です。もっと技術的な観点で言うと、クライアント側のデバイスとサーバ側のデバイスの接続を意味します。通常、トンネルは暗号化されていますが、暗号化されていなくてもトンネルと表現します。

25.1.2 VPN のシナリオ

VPN で接続を行うと、IP パケットは トンネル を介して伝送されるようになります。トンネルは TUN または TAP のいずれかを使用します。TUN/TAP は仮想ネットワークに対応するカーネルドライバで、イーサネットや IP フレーム (パケット) の伝送を行います。

ユーザスペース側のプログラム (OpenVPN) では、TUN/TAP のデバイスに接続して、オペレーティングシステムとの間でパケットの送受信を行います。

一般的に言うと、VPN には様々なソリューションが存在し、構成も様々なものが存在しています。本章では OpenVPN パッケージを利用した場合の説明を行っています。OpenVPN では、他の VPN ソフトウエアとは異なり、下記に示す 2 種類のモードに対応しています:

ルーティング型 VPN

ルーティング型 VPN は構築が簡単な方式です。ブリッジ型 VPN よりも効率的で大規模環境にも向きます。それに加えて、さらに効率を高めるために MTU (最大伝送単位; Maximum Transfer Unit) を調整することもできます。しかしながら、ゲートウエイ側で Samba サーバを動作させない限り、NetBIOS ブロードキャストが動作しないという仕様上の問題があり、異種混合型の環境では都合が悪いことがあります。なお、IPv6 を必要とする場合は、双方のエンドポイントで TUN デバイスに対して明示的に設定を行う必要があります。この方式の構成は 図 25.1「ルーティング型 VPN」 のようになります。

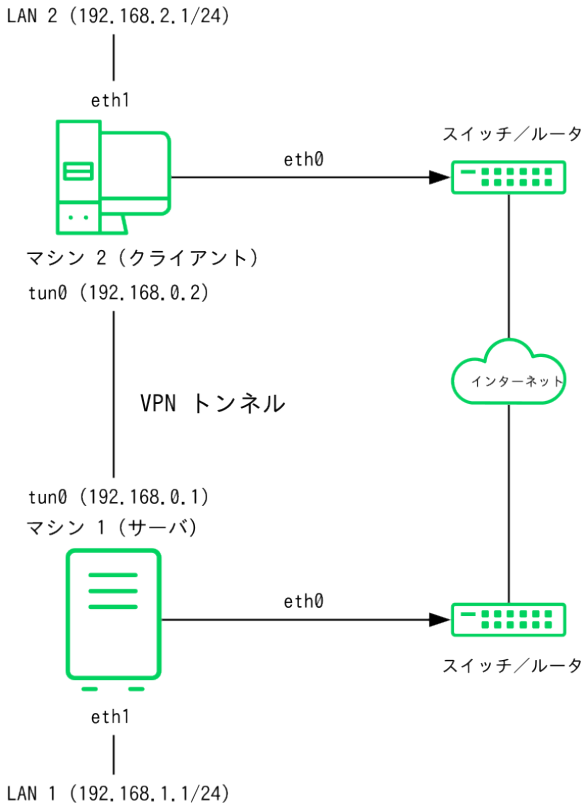


図 25.1: ルーティング型 VPN

ブリッジ型 VPN

ブリッジ型は少し複雑な方式です。Samba や WINS サーバなどの設定を行うことなく、VPN を介して Windows のファイル共有を参照できるようにしたい場合などに使用します。またブリッジ型の VPN では、レイヤ 2 のイーサネットフレームを転送するため、TAP と呼ばれるデバイスを使用します。これにより IP プロトコル以外 (IPX など) を使用できるほか、ブロードキャストパケットを必要とする様々なアプリケーションに有用です。しかしながら、ルーティング型 VPN と比べると効率は悪く、大規模環境にも向きません。この方式の構成を下記の図に示します。

シナリオ 1

サーバ (マシン 1) とクライアント (マシン 2) で直接 VPN を介して通信する形態です。

シナリオ 2

サーバ側 (マシン 1) の TAP デバイス `tap0` と LAN 1 に接続されている `eth1` がブリッジ接続されている場合の例です。`eth1` に何らかのレイヤ 2 パケットが届くと、ブリッジ `br0` および VPN を介してクライアント (マシン 2) に配送されます。

シナリオ 3

サーバ (マシン 1) とクライアント (マシン 2) の双方で、TAP デバイス `tap0` と `eth1` がブリッジ接続されている場合の例です。`eth1` に何らかのレイヤ 2 パケットが届くと、ブリッジ `br0` および VPN を介して相手側に配送されます。

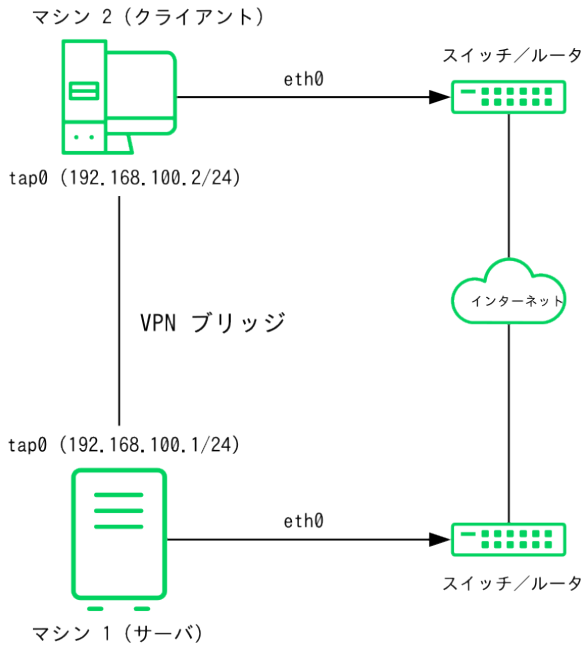


図 25.2: ブリッジ型 VPN - シナリオ 1

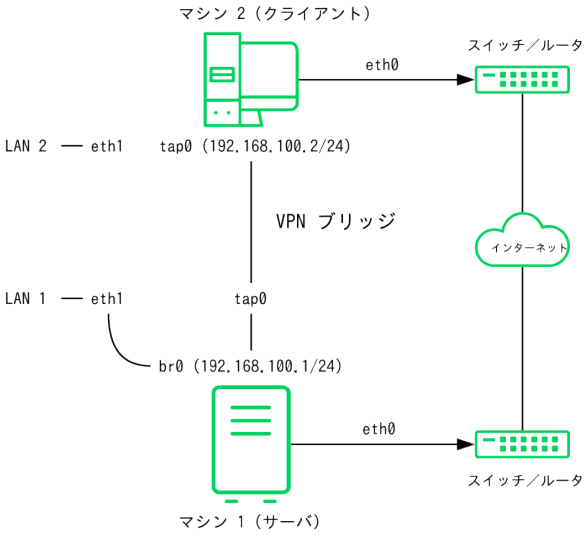


図 25.3: ブリッジ型 VPN - シナリオ 2

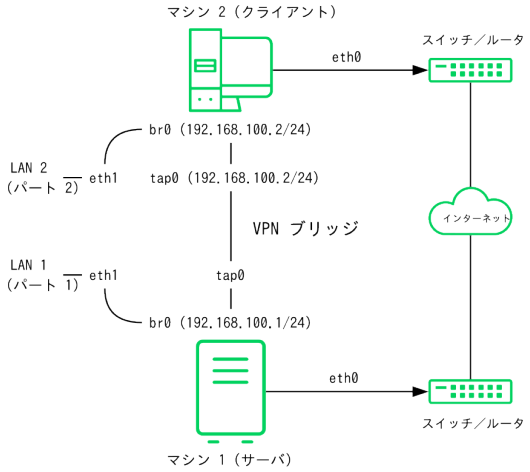


図 25.4: ブリッジ型 VPN - シナリオ 3

ブリッジ型とルーティング型の大きな違いは、ルーティング型の場合は IP ブロードキャストを通すことができないのに対して、ブリッジ型の場合はそれができるという点があります。

25.2 シンプルな構成の構築例

下記の例では、一対一の VPN トンネルを作成します。この例を読むことで、クライアントとサーバとの間で、どのようにして VPN トンネルが作られるのかを知ることができます。なお、VPN サーバ側のアドレスは `サーバの_IP` 、VPN クライアント側のアドレスは `クライアントの_IP` として、いずれもプライベート IP アドレスを使用しているものとします。なお、実際の構築にあたっては、他の IP アドレスと重複しないものを選んでください。

警告: セキュリティ面の考慮について

下記のシナリオは VPN 技術の仕組みを説明するために用意しているものであります。VPN 技術はその仕組み上、社外からのアクセスを受け入れる仕組みであるため、誤った設定を行ってしまうと、家庭内や社内のネットワークを危険にさらすことにも繋がります。実際に使用するにあたっては、よくご確認のうえ実施してください。



ヒント: 設定ファイルの名前について

OpenVPN の設定ファイルを簡単に扱うことができるようにするため、下記のような構成をお勧めします:

- OpenVPN の設定ファイルは、`/etc/openvpn` 以下に配置してください。
- 1 つのファイル内に全ての設定を書き込む場合は、`MY_CONFIGURATION.conf` のようなファイルを作成して書き込んでください。
- 複数のファイルを利用して 1 つの構成を作り上げる場合は、`/etc/openvpn/MY_CONFIGURATION` のようなサブディレクトリを作成して、その中にファイルを置いてください。

25.2.1 VPN サーバの設定

VPN サーバの構築は下記のようにして行います:

手順 25.1: VPN サーバの設定

1. VPN サーバとして動作させたいマシンに、`openvpn` パッケージをインストールします。
2. シェルを開いて `root` になり、下記のように実行して VPN の機密鍵を作成します:

```
# openvpn --genkey secret /etc/openvpn/secret.key
```

3. 機密鍵をクライアントにコピーします:

```
# scp /etc/openvpn/secret.key root@クライアントの_IP:/etc/openvpn/
```

4. 下記のような内容で、サーバ側の設定ファイル `/etc/openvpn/server.conf` を作成します:

```
dev tun
ifconfig サーバの_IP クライアントの_IP
secret secret.key
```

5. `/etc/sysconfig/network/ifcfg-tun0` というファイル名のファイル (TUN デバイスの設定) を作成して、下記の内容を記述します:

```
STARTMODE='manual'
BOOTPROTO='static'
TUNNEL='tun'
TUNNEL_SET_OWNER='nobody'
TUNNEL_SET_GROUP='nobody'
```

```
LINK_REQUIRED=no
PRE_UP_SCRIPT='systemd:openvpn@server'
PRE_DOWN_SCRIPT='systemd:openvpn@service'
```

途中、`openvpn@server` という記述がありますが、これは OpenVPN 内のサーバ設定 `/etc/openvpn/server.conf` を参照するための設定です。詳しい情報については、`/usr/share/doc/packages/openvpn/README.SUSE` をお読みください。

6. ファイアウオールを使用している場合は、YaST を開いて［セキュリティとユーザ］›［ファイアウオール］›［サービス］を選び、UDP ポート 1194 を開くように設定します。
7. TUN デバイスを起動して、OpenVPN のサービスを開始します:

```
> sudo wicked ifup tun0
```

下記のように表示されれば成功です:

```
tun0            up
```

25.2.2 VPN クライアントの設定

VPN クライアントの構築は下記のようにして行います:

手順 25.2: VPN クライアントの設定

1. VPN クライアントとして動作させたいマシンに、`openvpn` パッケージをインストールします。
2. 下記のような内容で、クライアント側の設定ファイル `/etc/openvpn/client.conf` を作成します:

```
remote サーバのグローバル_IP_もしくはドメイン名
dev tun
ifconfig クライアントの_IP サーバの_IP
secret secret.key
```

`サーバのグローバル_IP_もしくはドメイン名` の箇所には、サーバのグローバル IP アドレス、もしくはドメイン名とホスト名を入力します。

3. `/etc/sysconfig/network/ifcfg-tun0` というファイル名のファイル (TUN デバイスの設定) を作成して、下記の内容を記述します:

```
STARTMODE='manual'
BOOTPROTO='static'
TUNNEL='tun'
TUNNEL_SET_OWNER='nobody'
TUNNEL_SET_GROUP='nobody'
```

```
LINK_REQUIRED=no
PRE_UP_SCRIPT='systemd:openvpn@client'
PRE_DOWN_SCRIPT='systemd:openvpn@client'
```

4. ファイアウオールを使用している場合は、手順25.1「VPN サーバの設定」内の ステップ 6 で説明している手順に従って、UDP ポート 1194 を開くように設定します。
5. TUN デバイスを起動して、OpenVPN のサービスを開始します:

```
> sudo wicked ifup tun0
```

下記のように表示されれば成功です:

```
tun0            up
```

25.2.3 シンプルな構成のテスト

OpenVPN のサービスを正常に動作させることができたら、あとは下記のコマンドを利用して TUN デバイスが正常に動作しているかどうかを確認します:

```
ip addr show tun0
```

VPN の接続を確認するには、クライアントとサーバの両方で `ping` コマンドを使用し、互いに通信ができていることを確認します。サーバからクライアントに ping を行いたい場合は、下記のように実行します:

```
ping -I tun0 サーバの_IP
```

逆に、クライアントからサーバに対して ping を行いたい場合は、下記のように実行します:

```
ping -I tun0 クライアントの_IP
```

25.3 証明機関を利用した VPN サーバの構築

25.2項 で説明している構築例はテストには便利であるものの、本格的な使用に耐えるものではありません。本章では、同時に複数の接続を受け入れることのできる VPN サーバを構築する方法について、説明を行っています。これは公開鍵基盤 (PKI; Public Key Infrastructure) を利用して実現するもので、それぞれのサーバやクライアントに別々の鍵対 (公開鍵と機密鍵) を発行して、それを証明機関 (CA; Certificate Authority) で署名してもらうことによって、正当性を設定する方法です。

この構成で設定を行うには、下記の手順を実施する必要があります:

1. 25.3.1項「証明機関の作成」

2. 25.3.2項「VPN サーバの設定」
3. 25.3.3項「VPN クライアントの設定」

25.3.1 証明機関の作成

VPN の接続を確立する前に、クライアントはサーバが提示する証明書を認証し、サーバはクライアントが提示する証明書を認証しなければなりません。これを 相互認証 と呼びます。

証明書の作成を行うための直接的な機能は、openSUSE Leap には用意されていません。下記の手順では CA の証明書とサーバの証明書、そしてクライアントの証明書を他のシステムで作成したものとして説明しています。

サーバの証明書は PEM 形式で保存し、それに対応する機密鍵も暗号化することなく PEM 形式で保存しておきます。証明書は VPN サーバ内の `/etc/openvpn/server_crt.pem` に、(暗号化していない) 機密鍵は `/etc/openvpn/server_key.pem` にそれぞれ置いておきます。

クライアント証明書は PKCS12 (推奨) 形式、もしくは PEM 形式で保存しておきます。PKCS12 形式の場合は、その中に CA の証明書を含めておく必要があるほか、ファイルは `/etc/openvpn/クライアント名.p12` というファイル名で保存しておきます。PEM 形式で、CA の証明書が含まれているものの場合は、`/etc/openvpn/クライアント名.pem` というファイル名で保存しておきます。PEM 形式でクライアント証明書 (`*.ca`) とクライアント鍵 (`*.key`) 、そして CA の証明書 (`*.ca`) をそれぞれ別々に分割している場合は、それぞれのクライアント内の `/etc/openvpn/` ディレクトリにそれらを保存します。

また、CA の証明書は、サーバおよびクライアント内の `/etc/openvpn/vpn_ca.pem` にコピーしておく必要があります。

重要: クライアント証明書を分割した場合について

クライアント証明書と対応する機密鍵、そして CA の証明書をそれぞれ別々のファイルに分割している場合は、それらのファイル名を各クライアント内の OpenVPN の設定ファイルで指定する必要があります (詳しくは 例25.1「VPN サーバの設定ファイル」をお読みください) 。

25.3.2 VPN サーバの設定

サーバ側の設定に際しては、`/usr/share/doc/packages/openvpn/sample-config-files/server.conf` にある雛形を `/etc/openvpn/` にコピーしたあと、必要に応じて修正を行うことをお勧めします。

例 25.1: VPN サーバの設定ファイル

```
# /etc/openvpn/server.conf
port 1194 ❶
proto udp ❷
dev tun0 ❸

# Security ❹

ca    vpn_ca.pem
cert  server_crt.pem
key   server_key.pem

# ns-cert-type server
remote-cert-tls client ❺
dh   server/dh2048.pem ❻

server 192.168.1.0 255.255.255.0 ❼
ifconfig-pool-persist /var/run/openvpn/ipp.txt ❽

# Privileges ❾
user nobody
group nobody

# Other configuration ❿
keepalive 10 120
comp-lzo
persist-key
persist-tun
# status      /var/log/openvpn-status.tun0.log ⓫
# log-append  /var/log/openvpn-server.log ⓬
verb 4
```

❶ OpenVPN が待ち受ける TCP/UDP ポートを指定します。なお、ここで指定するポートは、ファイアウオール側で開いておく必要があります。ファイアウオールに関する詳細は 第24章「マスカレードとファイアウオール」をお読みください。標準ではポート 1194 を使用しますが、今回はそのまま使用するものとします。

❷ プロトコルを指定します。UDP もしくは TCP のいずれかを指定します。

❸ デバイスの種類を TUN もしくは TAP から選択して指定します。これらの違いについての詳細は 25.1.1項「用語」をお読みください。

❹ これ以降の行には、証明機関の証明書 (`ca`) と サーバ証明書 (`cert`)、そしてサーバ証明書の鍵 (`key`) をそれぞれ指定しています。これらのファイルは 25.3.1項「証明機関の作成」で生成したものです。

❺ 相手側に要求する証明書の種類を指定しています。この場合、RFC3280 TLS rules で規定されている key usage と extended key usage (鍵の用途) について、明確な署名のある証明書のみを受け入れる設定です。

❻ Diffie-Hellman パラメータを指定します。ここで指定するファイルは、下記のコマンドで生成することができます:

```
openssl dhparam -out /etc/openvpn/dh2048.pem 2048
```

❼ VPN サブネットを指定します。サーバ自身は `192.168.1.1` を使用します。

❽ クライアントと仮想 IP アドレスの対応付けに関する情報を、指定したファイルに記録する指定です。これは特に、サーバ側を再起動するような場合、再起動後もクライアントに対して同じアドレスを割り当てたい場合に有用です。

❾ セキュリティ上の理由から、OpenVPN に対する権限は制限しておくことをお勧めします。権限を制限するには、ユーザおよびグループを `nobody` に設定します。

❿ その他の設定です。詳しくは `/usr/share/doc/packages/openvpn/sample-config-files` にあるサンプル設定ファイル内のコメント (英語) をお読みください。

⓫ このオプションを有効にすることで、指定したファイルに状態に関する概要情報と統計情報 (「operational status dump」) を出力するようになります。既定では有効化されていません。全ての出力は、`journalctl` で表示することのできるシステムジャーナルに書き込まれます。複数の設定ファイルが存在するような場合 (たとえば一方は自宅用、他方は仕事用など) は、ファイル名にデバイス名を入れておくことをお勧めします。これにより、不用意にファイルが上書きされてしまうことが無くなります。この例の場合は `dev` に書かれているとおり、`tun0` がデバイス名となります。詳しくは ❸ をお読みください。

⓬ 既定では、ログメッセージは syslog に出力されます。この動作を変更したい場合は、この行にあるコメント文字 (#) を削除してください。コメント文字を削除すると、全てのメッセージが `/var/log/openvpn-server.log` に書き込まれるようになります。なお、logrotate サービスの設定を忘れずに実施するようにしてください。詳しくは `man 8 logrotate` をお読みください。

設定作業が終わったら、OpenVPN サーバのログが `/var/log/openvpn.log` に出力されます。初回の起動時には、下記のように出力されるはずです:

```
... Initialization Sequence Completed
```

上記のようなメッセージが表示されない場合は、ログファイルをよく読んで、設定ファイル内のどの箇所が誤っているのかをご確認ください。

25.3.3 VPN クライアントの設定

クライアント側の設定に際しては、`/usr/share/doc/packages/openvpn/sample-config-files/client.conf` にある雛形を `/etc/openvpn/` にコピーしたあと、必要に応じて修正を行うことをお勧めします。

例 25.2: VPN クライアントの設定ファイル

```
# /etc/openvpn/client.conf
client ❶
dev tun ❷
proto udp ❸
remote IP_OR_HOST_NAME 1194 ❹
resolv-retry infinite
nobind

remote-cert-tls server ❺

# Privileges ❻
user nobody
group nobody

# Try to preserve some state across restarts.
persist-key
persist-tun

# Security ❼
pkcs12 client1.p12

comp-lzo ❽
```

❶ このマシンをクライアントとして動作させる宣言です。

❷ ネットワークデバイスを指定します。クライアントとサーバの両方で、同じデバイスを使用しなければなりません。

❸ プロトコルを指定します。こちらもクライアントとサーバの両方で、同じプロトコルを設定しなければなりません。

❺ こちらはクライアント向けのセキュリティオプションで、接続先のコンピュータが目的のサーバかどうかを確認することができます。

❹ `IP_OR_HOST_NAME` の箇所を OpenVPN サーバのホスト名もしくは IP アドレスに置き換えてください。ホスト名もしくは IP アドレスの後ろには、サーバ側のポート番号を指定します。複数の VPN サーバのうちのいずれかを選択して接続したい場合は、`remote` を複数行指定してもかまいません。この場合、複数の VPN サーバを負荷分散として使用することになります。

❻ セキュリティ上の理由から、OpenVPN に対する権限は制限しておくことをお勧めします。権限を制限するには、ユーザおよびグループを `nobody` に設定します。

❼ クライアント向けに発行された鍵を指定します。セキュリティ上の理由から、クライアントごとに異なる鍵を使用してください。

❽ 圧縮を有効化しています。このパラメータは、サーバ側でも有効化しておかなければなりません。

25.4 YaST を利用した VPN サーバおよびクライアントの設定

YaST を使用することで、VPN サーバの設定を行うことができます。ただし、残念ながら YaST のモジュールは OpenVPN に対応していません。その代わり、IPsec プロトコル (StrongSwan ソフトウエアで実装されています) を使用することができます。OpenVPN と同様に、IPsec は幅広く使用されている VPN 方式です。

手順 25.3: IPSEC サーバの設定

1. YaST の VPN モジュールを起動するには、［ネットワークサービス］›［VPN ゲートウエイとクライアント］を選択します。
2. まずは［グローバル設定］内にある［VPN デーモンの有効化］を選択します。
3. 新しく VPN を作成するには、［新しい VPN］を押して、接続に対して設定する名前を入力します。
4. ［種類］では［ゲートウエイ (サーバ)］を選択します。
5. あとは用途に合わせて下記を実施します:

   - Linux クライアントを接続する場合は、［事前共有鍵による機密通信］と［証明書による機密通信］が最適な選択となります。
   - Android, iOS, macOS などの新しいバージョンでネイティブに対応している設定を行いたい場合は、［Android, iOS, MacOS X クライアントに対してアクセスを提供する］を選択します。この設定では、事前共有鍵をベースにした設定となり、必要であればユーザ名／パスワードによる認証も行うことができます。
   - Windows や BlackBerry デバイスでネイティブに対応する設定を行いたい場合は、［Windows 7, Windows 8 クライアントに対してアクセスを提供する］を選択します。こちらは証明書をベースにした設定となり、必要であればユーザ名／パスワードによる認証も行うことができます。

   この例では、［事前共有鍵による機密通信］を選択したものとします。

6. 鍵を指定するには、［認証情報の編集］を押します。あとは［鍵の表示］を押してから、共有鍵を入力します。入力が終わったら［OK］を押します。

7. ［VPN クライアントのアクセス方法］では、VPN を利用した場合のアクセス制限を設定することができます。特定の IP 範囲のみを許可したい場合は、［カンマ区切りでのアドレス制限］内に、CIDR 形式かつカンマ区切りで指定します。CIDR 形式に関する詳細は、https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing をお読みください。

8. ［クライアントのアドレスプール］では、VPN サーバからクライアントに対して提供する、IP アドレスを指定します。

9. 設定を完了するには、［OK］を押します。これにより、YaST VPN モジュールは自動的にファイアウオールのルールを追加して有効化し、クライアント側から接続できるように設定します。
接続の状態を確認したい場合は、表示される確認ウインドウで［はい］を押します。これにより、お使いの VPN 接続に対する `systemctl status` が表示され、VPN が有効化されているかと正しく設定されているかを確認することができます。

25.5 さらなる情報

NetworkManager を利用した VPN 接続の設定方法については、**『リファレンス』、第28章「NetworkManager の使用」、28.3.4項「NetworkManager と VPN」**をお読みください。

VPN に関する情報をご希望の場合は、下記をご覧ください:

- https://openvpn.net : OpenVPN の Web ページ
- `man openvpn`
- `/usr/share/doc/packages/openvpn/sample-config-files/` : 様々な用途に対応した設定ファイルの例があります。
- `/usr/src/linux/Documentation/networking/tuntap.txt` : TUN/TAP インターフェイスの説明 (英語) です。なお、このファイルは `kernel-source` パッケージ内に含まれています。

26 X Window System で動作する PKI マネージャ XCA による管理

改訂履歴

2024-06-26

従来は独自の公開鍵基盤を管理するのに `openssl` ユーティリティを使用してきました。openSUSE Leap 15.7 では、グラフィカルなツールを希望する管理者向けに、X Window System で動作する証明書／鍵管理ツール XCA (https://hohnstaedt.de/xca) を提供しています。

XCA では X.509 規格の証明書や署名要求のほか、RSA, DSA, EC などの機密鍵やスマートカード、証明書失効リスト (CRL) を作成したり管理したりすることができます。つまり、XCA では独自の証明機関を作成／管理するのに必要な全ての機能に対応していることになります。このほか、XCA には証明書や署名要求の生成にあたって、カスタマイズ可能なテンプレート (雛型) も含まれています。本章では、基本的な手順を説明しています。

26.1 XCA のインストール

XCA は `xca` というパッケージで提供されています:

```
> sudo zypper in xca
```

26.2 新しい PKI の作成

XCA は暗号に関するデータをデータベース内に保存します。XCA を初めて起動して新しい PKI を作成する場合、まずは［ファイル > 新しいデータベース］を選択します (図26.1「新しい XCA データベースの作成」) 。

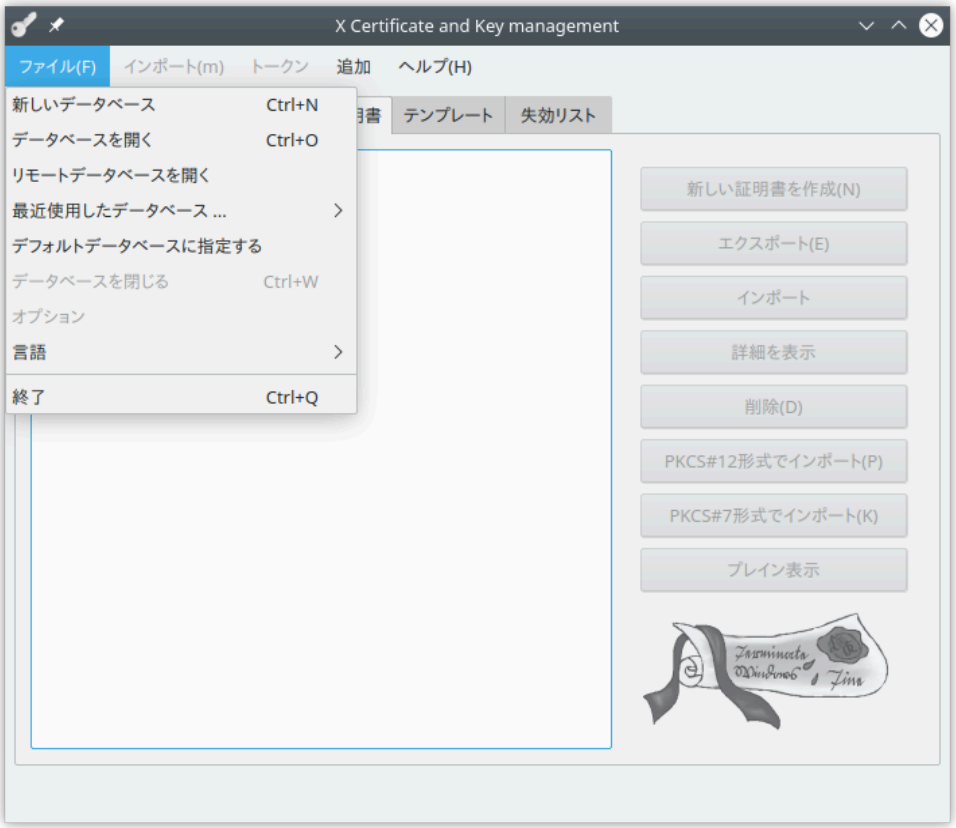


図 26.1: 新しい XCA データベースの作成

26.2.1 新しいルート証明機関の作成

新しいルート証明機関を作成するには、下記の手順を実施します。

1. [署名済み証明書] タブを選択します。
2. [新しい証明書を作成] ボタンを押します。
3. [ソース] タブに移動します。ウインドウの下部には [新しい証明書のテンプレート] という枠がありますので、この中で [[default] CA] を選択します。あとは [すべてに適用] を押します。
4. [サブジェクト] タブに移動します。ここでは [内部名] に名前を入力します。この名前は XCA 内部でのみ使用されるもので、識別のための名前です。
   [Distinguished Name] 以下の各項目に入力を行います。ここに記載されていない要素を指定したい場合は、[追加] を押して選択および入力を行ってください。
5. [秘密キー] のドロップダウンボックスでは、この証明書に結びつける機密鍵を指定します。機密鍵がない場合は、新しく作成することもできます。

6. ［拡張キー］タブに移動します。ここでは必要な項目をそれぞれ設定します。既定の［有効期間］は 10 年です。また、証明書失効リスト (CRL) の配布ポイント (［X509v3 CRL Distribution Points］) は証明書内に記載されますので、どこからでもアクセス可能な URL を指定しておくことをお勧めします (例: `http://www.example.com/crl/crl.der`) 。項目の選択および入力が終わったら、［OK］ボタンを押します。

26.2.2 新しいホスト証明書の作成

ここでは、新しく作成した証明機関を利用して、新しいホスト向けの証明書を作成します。

1. ［署名済み証明書］タブに移動して、［新しい証明書を作成］ボタンを押します。

2. ［ソース］タブでは［[default] TLS_server］を選択して［すべてに適用］ボタンを押します。これで［拡張キー］, ［キー使用法］, ［Netscape］の各タブの項目内に適切な値が設定されます。また、［署名］以下の選択では、26.2.1項「新しいルート証明機関の作成」で作成した証明機関を選択します。

3. ［サブジェクト］タブに移動します。まずは［内部名］に名前を入力します。こちらは XCA 内での表示用にのみ使用される項目で、一般的にはホスト名や FQDN (完全修飾ドメイン名) を指定しておくことをお勧めします。あとは［Distinguished Name］以下の項目にそれぞれ入力を行います。ホスト証明書の場合、［commonName］は証明書を使用するホストの FQDN でなければなりません (別名があればそれでもかまいません) 。たとえば jupiter.example.com に Web サーバが存在していて、その Web サーバに対して www.example.com という DNS CNAME エントリが存在する場合、［commonName］には www.example.com を指定します。この中に表示されていない項目を指定したい場合は、［追加］ボタンを押してドロップダウンから選択してください。また［秘密キー］では、対応付ける機密鍵を選択するか作成します。

4. ［拡張キー］タブを選択します。既定の［有効期間］は 1 年です。この期間を変更する場合は、忘れずに［適用］ボタンを押してください。

5. また、証明書の失効リスト (CRL) の場所を指定しておくこともお勧めします。この場所はルート証明書ごとに別々の場所でなければなりません。XCA では PEM 形式もしくは DER 形式で CRL を取り出すことができますので、失効リストの場所は `http://www.example.com/crl/crl.der` のような場所にしておくとよいでしょう。［CRL Distribution points］の右側にある［編集］ボタンを押し、［追加］を押して URI を指定してください。あとは［検証］および［適用］を押します。
   あとは［OK］を押します。

26.2.3 証明書の失効化

1. ［署名済み証明書］タブを選択します。
2. 失効させたい証明書をマウスの右ボタンで選択し、［失効］を選択します。
3. 失効を行った証明機関をマウスの右ボタンで選択し、［CA > CRL を作成］を選択します。［CRL の作成］ダイアログが表示されたら、［OK］ボタンを押します。
4. メインウインドウで［失効リスト］のタブを選択します。あとは作成した失効リストを選んで［エクスポート］ボタンを押し、必要な形式 (おそらく DER でしょう) を選択して［OK］を押します。これでファイルとして保存が行われますので、このファイルを［CRL Distribution Points］(CRL 配布ポイント) で指定した場所に配置します。

27 sysctl 変数によるネットワークセキュリティの改善

改訂履歴

2023-06-06

sysctl (system control; システム制御) 変数はカーネルのパラメータを制御するための変数で、オペレーティングシステム内の様々な箇所の動作に影響する変数です。これらのパラメータは、`proc` ファイルシステム内の `/proc/sys` 経由でアクセスすることができます。多くのカーネルパラメータは、このディレクトリ内の擬似ファイルに直接書き込むことで値を直接変更することができますが、指定した値は保存されませんので、システムを再起動してしまうと元の値に戻ってしまいます。このような構造から、システムの起動時に値を変更するよう、変更点を sysctl の設定ファイルに書き込んでおくことをお勧めします。

本章では、Linux のセキュリティ機能を改善するのに役立つ様々なネットワーク関連変数を設定しています。ファイアウオールの有効化やその設定内容によっては、ここに書かれている値のいくつかが自動的に設定されている場合があります。現在の値を確認したい場合は、`sysctl` ユーティリティを利用して下記のように入力して実行します:

```
> /sbin/sysctl net.ipv4.conf.all.rp_filter
net.ipv4.conf.all.rp_filter = 2
```

下記の設定を適用したい場合は、`/etc/sysctl.d/` ディレクトリ内に設定ファイルを作成してください。設定ファイルは `.conf` という接尾辞で終わっていなければならないことに注意してください (例: `/etc/sysctl.d/network.conf`)。詳しくは `man 5 sysctl.d` で表示されるマニュアルページをお読みください。

なお、下記の値は環境に合わせて変更してください。

- ```
  # これらの値の既定値: 2 (loose mode)
  net.ipv4.conf.default.rp_filter = 1
  net.ipv4.conf.all.rp_filter = 1
  ```

  上記の設定を適用すると、IPv4 における逆方向パスフィルタを厳格モードに設定します。これにより、IP パケットに対する応答を送信する際、必ず受信したインターフェイスから送信するように強制されます。なお、システム側で応答パケットを生成する際、ルーティングテーブルに従って受信パケットとは異なるインターフェイスを選択してしまうと、これらのパケットは廃棄されるようになります。この設定は、IP スプーフィング攻撃と呼ばれる分散型サービス拒否攻撃 (DDoS) 等を防ぐのに役立ちます。

- ```
  # これらの値の既定値: 0
  net.ipv4.conf.default.accept_source_route = 0
  net.ipv4.conf.all.accept_source_route = 0
  ```

上記の設定を適用すると、IPv4 のヘッダ内に SSR オプションが設定されているパケットの受信を無効化します。つまり、ソースルーティング を使用するパケットが拒否されることになります。これにより IP パケットの転送を防止することができるため、ファイアウオールの配下に存在するホストにパケットが届かないようになります。

- ```
  # この値の既定値: 1
  net.ipv4.tcp_syncookies = 1
  ```

  IPv4 と IPv6 の 両方 に対して、TCP SYN Cookie 保護の機能を有効化します。これは TCP プロトコルレベルでの特定のサービス拒否 (Denial Of Service; DoS) 攻撃を防ぐための機能で、少しだけ CPU の負荷が上がるものの、不正攻撃によって発生しうるメモリ浪費を防ぐことができます。この保護機構はフォールバック型のアルゴリズムとして作られていて、通常の TCP 接続を受け付けられないほど負荷が高い場合にのみ動作します。また、この仕組みは TCP プロトコルとの完全な互換性は無いため、相手側の TCP 実装によっては問題が発生する可能性があります。過負荷状態では新しい接続を全く受け付けなくする手法もありますが、この仕組みを利用することで、正当な TCP 接続と TCP によるサービス拒否攻撃を正しく分けて扱うことができるようになります。なお、お使いのシステムで TCP 接続の負荷が高いことが予想される場合、この設定によって逆効果となる場合もあります。

- ```
  # 既定値: 128
  net.ipv4.tcp_max_syn_backlog = 4096
  ```

  TCP SYN backlog 設定は、処理待ちの SYN パケットの数を設定するためのものです。指定した値よりも多くの SYN パケットが届いた場合、SYN パケットは廃棄され、新しい TCP 接続はできなくなります (もしくは、SYN Cookie 保護が動作するようになります) 。この値を増やすことで、TCP SYN 攻撃に対する保護を強化することができます。

- ```
  # 既定値: 1
  net.ipv4.icmp_echo_ignore_broadcasts = 1
  ```

  IPv4 ブロードキャストアドレスに対する ICMP エコー要求 (ping) は特定のネットワーク内に存在するホストや IP アドレスを検出するための仕組みですが、ネットワークセグメントに対する過負荷攻撃としても使用することができます。この設定を適用することで、ブロードキャストアドレスに送信される ICMP エコー要求を無視するようになります。

- ```
  # 既定値: 1
  net.ipv4.icmp_ignore_bogus_error_responses = 1
  ```

  この設定は、ブロードキャストフレームに対する不正な応答によって、不要なエラーメッセージを生成しないようにするためのもので、ログファイルが埋め尽くされることを防ぐ効果があるものです。詳しくは RFC 1122 インターネットホストの要件: 通信レイヤー (https://jprs.jp/tech/material/rfc/RFC1122-ja.txt) のセクション 3.2.2 をお読みください。

- ```
  # 既定値: 0
  net.ipv4.conf.default.accept_redirects = 0
  net.ipv4.conf.all.accept_redirects = 0
  net.ipv6.conf.default.accept_redirects = 0
  net.ipv6.conf.all.accept_redirects = 0
  ```

  ICMP redirect (転送) メッセージの受け付けを無効化します。これらのメッセージはゲートウエイから発信されるもので、宛先のホストに辿り着くためのより適切なルートを指し示すためのものです。転送メッセージを悪用することで、中間者攻撃が成立してしまうことから、無効化しておくべき値となります。

- ```
  net.ipv4.conf.default.secure_redirects = 0
  net.ipv4.conf.all.secure_redirects = 0
  ```

  ごく稀な用途ですが、'安全な' ICMP redirect (転送) を受け付けるための設定です。どうしても必要な場合を除いて、無効化しておくべきものです。

- ```
  net.ipv4.conf.default.send_redirects = 0
  net.ipv4.conf.all.send_redirects = 0
  ```

  IPv4 ICMP redirect (転送) 要求を送信しないようにする設定です。ルータとして動作させる場合を除いて、不要であるためです。

- ```
  # 既定値: 0
  net.ipv4.ip_forward = 0
  net.ipv6.conf.all.forwarding = 0
  net.ipv6.conf.default.forwarding = 0
  ```

  ルータとして動作させる場合を除いて、IP 転送機能は無効化しておくべきものです。

- ```
  net.ipv4.conf.all.log_martians = 1
  net.ipv4.conf.default.log_martians = 1
  ```

  上記の設定は Martian パケットをログに記録する設定です。Martian パケットとは、送信元もしくは送信先のアドレスが、そのマシンのあるネットワーク内では使用されないはずのアドレスになっているものを指す用語で、この設定を有効化することによって、設定ミスや外部からの攻撃を検知しやすくする効果があります。
  ```

IV AppArmor による権限の制限

28 AppArmor の紹介 311

29 入門 313

30 プログラムに対する予防接種 318

31 プロファイルのコンポーネントと文法 328

32 AppArmor のプロファイルリポジトリ 360

33 YaST を利用したプロファイルの構築と管理 361

34 コマンドラインからのプロファイル構築 370

35 チェンジハット機能による Web アプリケーションのプロファイル作成 399

36 pam_apparmor によるユーザの制限 410

37 プロファイルを作成したアプリケーションの管理 411

38 サポート 413

39 AppArmor 用語集 422

28 AppArmor の紹介

改訂履歴

2023-06-06

信頼できる プログラムを実行していても、バグによって引き起こされる様々な脆弱性に直面します。信頼できるプログラムのうち、権限を利用して動作しているプログラムについては、攻撃者からも格好の標的となります。プログラム内に何らかのバグがあれば、攻撃者はそれを悪用して必要な権限を奪取しようとしますので、プログラムに対する信頼も損なわれることになります。

AppArmor® はアプリケーション側でのセキュリティソリューションで、特にプログラムに対して与えられた権限を制限することによって、システムの安全性を守る仕組みです。AppArmor では、管理者がプログラムに対して プロファイル と呼ばれる動作許可範囲を作成して適用することができます。このプロファイルは、プログラムがアクセスする可能性のあるファイルや、プログラムが行う処理などを集めたもので、これによって不正な動作を防ぎ、未知の脆弱性に対して攻撃が加えられた場合であっても、その攻撃による影響を防いだり、軽減したりすることができます。

28.1 AppArmor のコンポーネント

AppArmor には下記のものが含まれています:

- 一般的な Linux* アプリケーション向けの AppArmor プロファイルライブラリ。プログラム側からアクセスする必要のあるファイルを示しています。
- DNS 参照やユーザ認証など、一般的なアプリケーション動作に必要な、AppArmor プロファイルファウンデーションクラス (プロファイル構築ブロック) ライブラリ。
- AppArmor のプロファイルを作成したり拡張したりする際に使用するツールスイート。これらのツールを使用することで、既存のツールを要件にあわせて調整したり、独自のアプリケーション向けに新しいプロファイルを作成したりすることができるようになります。
- ユニークなサブプロセス制約として強化されたセキュリティを提供する、AppArmor を有効化するために独自に修正されたプログラム類 (Apache など) 。
- お使いの openSUSE® Leap システムで使用される AppArmor 関連のカーネルコード、および AppArmor ポリシーを強制するための関連する制御スクリプト類。

28.2 AppArmor プロファイリングに関する背景となる情報

AppArmor の科学的およびセキュリティの観点における詳しい情報については、下記の書籍をお読みください (いずれも英語です):

SubDomain: Parsimonious Server Security (Crispin Cowan, Steve Beattie, Greg Kroah-Hartman, Calton Pu, Perry Wagle, Virgil Gligor 氏)

AppArmor 初期の設計と実装について説明しています。2000 年 12 月にアメリカのルイジアナ州ニューオーリンズで行われた USENIX LISA Conference の議事録として発表されたものです。この文書は今となっては古くなっていて、現在の AppArmor 製品とは文法も機能も異なるものになっています。そのため、本文書は技術文書としてではなく、背景となる情報を知るための資料としてのみお使いください。

Defcon Capture the Flag: Defending Vulnerable Code from Intense Attack (Crispin Cowan, Seth Arnold, Steve Beattie, Chris Wright, John Viega 氏)

短期間で深刻なセキュリティ問題を解決する必要がある場合、AppArmor を戦略的／戦術的に使用するのに適したガイドです。2003 年にアメリカのワシントン DC で行われた、DARPA Information Survivability Conference and Expo (DISCEX III) の議事録として発表されたものです。

AppArmor for Geeks (Seth Arnold 氏)

この文書には、AppArmor の技術的な側面について、理解を深めるための説明が書かれています。https://en.opensuse.org/SDB:AppArmor_geeks (英語) でも提供されています。

29 入門

改訂履歴

2024-05-10

お使いのシステムに AppArmor を問題なく配置するには、下記の項目について注意深く考慮する必要があります:

1. プロファイルを作成するアプリケーションの決定。詳しくは 29.3項「プロファイルを作成するアプリケーションの選択」をお読みください。
2. 必要なプロファイルの作成。概要は 29.4項「プロファイルの作成と修正」で説明しています。出力されたプロファイルは、必要に応じて調整する必要がある場合もあります。
3. 環境が変更された場合や、AppArmor のレポートツールで記録されたセキュリティイベントへの対応。詳しくは 29.5項「プロファイルの更新」をお読みください。

29.1 AppArmor のインストール

openSUSE® Leap では、インストール時に選択したパターンに関わらず、AppArmor は既定でインストールされ実行されます。下記に示すパッケージは、AppArmor の全ての機能を使用するために必要なパッケージの一覧です:

- `apparmor-docs`
- `apparmor-parser`
- `apparmor-profiles`
- `apparmor-utils`
- `audit`
- `libapparmor1`
- `perl-libapparmor`
- `yast2-apparmor`



ヒント

もしもお使いのシステムに AppArmor がインストールされていない場合、後からインストールを行うには `apparmor` という名称のパターンをインストールしてください。これにより、全ての機能を使用できるようになります。インストールにあたっては、YaST のソフトウエア管理モジュールか、zypper のコマンドラインを使用します。zypper の場合は下記のように入力して実行します:

```
> sudo zypper in -t pattern apparmor
```

29.2 AppArmor の有効化／無効化

AppArmor は openSUSE Leap を新規にインストールしている環境であれば、自動的に設定され動作しています。AppArmor の状態を切り替えたい場合は、下記のようにして行います:

YaST サービスマネージャを使用する方法

システムの起動時に実行されるスクリプト群から、AppArmor の起動スクリプトを追加したり削除したりすることで、有効かと無効を切り替えます。この場合、変更は再起動後に適用されることになります。

AppArmor の設定ウインドウを使用する方法

YaST の AppArmor コントロールパネルを使用して、動作中のシステム内から有効化と無効化を切り替えることができます。変更は即時に適用されます。コントロールパネルでは、AppArmor に対して起動もしくは停止のイベントを送信することができるほか、システムの起動時のスクリプト群から AppArmor の起動スクリプトを追加したり削除したりすることもできます。

AppArmor を恒久的に無効化したい (システムの起動時に実行されるスクリプト群から取り除きたい) 場合は、下記のようにして行います:

1. YaST を起動します。
2. [システム] › [サービスマネージャ] を選択します。
3. `apparmor` と書かれた行を探してマウスの左ボタンで選択し、ウインドウの下部にある [有効／無効] を押します。これで [有効] と書かれていた列が [無効] になります。
4. [OK] を押して閉じます。

これで AppArmor がシステムの起動時に開始されなくなり、再度 [有効] が設定されるまで、動作しなくなります。有効に戻したい場合は、YaST [サービスマネージャ] を利用して同様の手順を実施してください。

動作中のシステムで AppArmor の状態を切り替えたい場合は、AppArmor の設定ウインドウをお使いください。ここでの設定は即時に反映され、システムを再起動しても切り替え後の状態が保持されます。AppArmor の状態を切り替えるには、下記の手順を実施します:

1. YaST を起動して、［AppArmor の設定］を選択し、メインウインドウ内の［設定］を選択して［起動］を押します。
2. AppArmor を有効化したい場合は、［AppArmor を有効にする］のチェックボックスにチェックを入れます。無効化したい場合はチェックを外します。
3. ［終了］を押すと設定を保存して終了することができます。

29.3 プロファイルを作成するアプリケーションの選択

お使いの環境内で攻撃を受ける可能性のあるプログラムが、保護を必要とするプログラムとなります。それらに該当するアプリケーションのみに対してプロファイルを作成してください。一般的な例としては、下記のようなものがあります:

ネットワークエージェント
Web アプリケーション
cron ジョブ

どのプロセスがネットワークポートを開いて動作していて、プロファイルを作成する必要があるのかを知りたい場合は、`root` で `aa-unconfined` を実行します:

例 29.1: `aa-unconfined` の出力

```
19848 /usr/sbin/cupsd not confined
19887 /usr/sbin/sshd not confined
19947 /usr/lib/postfix/master not confined
1328 /usr/sbin/smbd confined by '/usr/sbin/smbd (enforce)'
```

上記の例では、いくつかの行に対して `not confined` と書かれていますが、これらのプログラムに対して独自のプロファイルを作成する必要があることを示しています。また、`confined by` と書かれているプログラムは、既に AppArmor による保護が適用されているプログラムであることを示しています。



ヒント: さらなる情報

プロファイルを作成するアプリケーションを選ぶための方法について、詳しくは 30.2項「予防接種を行うプログラムの決定」をお読みください。

29.4 プロファイルの作成と修正

openSUSE Leap での AppArmor には、あらかじめ設定済みのプロファイル集が提供されていて、主要なアプリケーションに対して保護が行われています。これに加えて AppArmor を使用することで、任意のアプリケーションに対してプロファイルを作成することができます。

プロファイルの管理方法には 2 種類のものがあります。一方は YaST の AppArmor モジュールが提供するグラフィカルなフロントエンドで、もう一方は AppArmor スイート自身が提供するコマンドラインツールです。YaST では AppArmor プロファイルに対する基本的な機能のみを提供するのに対して、コマンドラインツールはより細かくプロファイルを更新したり修正したりすることができます。

それぞれのアプリケーションに対してプロファイルを作成するには、下記の手順を実施します:

1. `root` になり、`aa-genprof` プログラム名 と入力して実行し、AppArmor に対してアプリケーションの大まかなプロファイルを作成させます。
   もしくは、[YaST] › [セキュリティとユーザ] › [AppArmor の設定] › [手作業でプロファイルを追加] を選択し、プロファイル対象のアプリケーションのパスを入力して、基本的なプロファイルを作成してもかまいません。
   基本的なプロファイルが新しく作成され、学習 (learning) モードに設定されます。学習モードとは、プログラムの動作がログに記録されるものの、プロファイルによる制限が課されないモードのことを指します。

2. あとはアプリケーションの動作を一通り行い、AppArmor に対して動作の細かい仕組みを学習させます。

3. その後、aa-genprof 内で `S` を押し、ステップ 2 で生成されたログファイルを AppArmor に分析させます。
   AppArmor はアプリケーションの動作時に記録されたログファイルを読み込み、記録されたそれぞれのイベントに対してアクセス権の設定可否を尋ねます。ここでは、ファイル名を直接指定するだけでなく、グロブ (ワイルドカード) も使用することができます。

4. アプリケーションの複雑さにも依存しますが、ステップ 2 と ステップ 3 を繰り返し実行する必要がある場合もあります。いったんプロファイルを作成したあと、その設定でさらにアプリケーションを動作させ、そこで出力されたログを元にさらにプロファイルを更新する、という作業を繰り返します。アプリケーションの動作範囲を正確に記録するには、何度も繰り返す必要があるかもしれません。

5. `aa-genprof` を終了すると、作成したプロファイルは強制 (enforce) モードに設定されます。これによりプロファイルが適用され、AppArmor はそれにあわせて制限されるようになります。

> 不平 (complain) モードにある既存のプロファイルを持つアプリケーションに対して `aa-genprof` を開始した場合、このプロファイルは学習サイクルを終了するまでの間、学習モードに設定されます。プロファイルのモード変更について、詳しくは 34.7.3.2項「aa-complain: 不平モード (学習モード) への突入」と 34.7.3.6項「aa-enforce: 強制モードへの突入」をお読みください。

制限の設定されたアプリケーションを実際に利用してみて、プロファイルが正しいことを確認してください。通常、制限の設定されたプログラムは全く問題なく動作し、AppArmor の制限が課されていることにも気がつかないレベルで動作します。しかしながら、アプリケーションの動作に何らかの異常が見つかった場合は、システムログを調べて AppArmor の制限が原因で発生したものでないかどうかをご確認ください。お使いのシステムで使用しているログの仕組みにもよりますが、一般的には下記の場所に AppArmor のログが記録されます:

`/var/log/audit/audit.log`
`journalctl | grep -i apparmor` コマンドの出力
`dmesg -T` コマンドの出力

プロファイルを調整したい場合は、対象のアプリケーションに関連するログメッセージを再度分析してください。詳しくは 34.7.3.9項「aa-logprof: システムログのスキャン」をお読みください。このとき、問い合わせが表示されたら、アクセス権を設定するか制限を設定するかを選択してください。



ヒント: さらなる情報

プロファイルの作成や修正に関する詳しい情報については、第31章「プロファイルのコンポーネントと文法」, 第33章「YaST を利用したプロファイルの構築と管理」, 第34章「コマンドラインからのプロファイル構築」の各章をお読みください。

29.5 プロファイルの更新

ソフトウエアの仕様やシステムの設定は、日々変わっていきます。そのような背景から、AppArmor のプロファイル設定についても、日々調整を加える必要があることがあります。AppArmor では、システムログを調査してポリシー違反やその他の AppArmor イベントを検出し、それにあわせてプロファイルを調整することができます。プロファイルの定義範囲を超えるアプリケーション動作に対応したい場合は、`aa-logprof` をお使いください。詳しくは 34.7.3.9項「aa-logprof: システムログのスキャン」をお読みください。

30 プログラムに対する予防接種

改訂履歴

2024-05-10

コンピュータシステムのセキュリティを効率的に強化するには、まず権限を仲介するプログラムをできる限り少なくした後、それらのプログラムをできるだけ強く保護する流れで行います。AppArmor は、お使いの環境下で攻撃にさらされるプログラムに対してプロファイルを作成するだけで保護が実現できる仕組みであることから、セキュリティを強化する際の手間を劇的に減らすことができます。AppArmor ではポリシーの仕組みによって、プログラムが想定通りの動作のみを実行するよう強制することができます。

AppArmor では予防接種技術により、アプリケーションが直面する可能性のあるプログラム固有の脆弱性から身を守ることができます。AppArmor をインストールしてプロファイルを構築し、コンピュータを再起動するだけで、AppArmor のセキュリティポリシーを強制することができますので、システムを保護することができるようになります。ちなみに、AppArmor によるプログラムの保護を 予防接種 (immunizing) と呼びます。

このような構造により、管理者は攻撃に対して脆弱なアプリケーションに対して関心を持っておいて、それらに対してプロファイルを作成するだけでよいことになります。つまりシステムの保護は、AppArmor のプロファイルの作成と運用、そしてそこから発生し AppArmor のレポート機能によって報告される、ポリシー違反や例外の監視を行うだけの作業となるわけです。

ユーザ側では AppArmor の存在に気がつくことはありません。AppArmor は「裏方で」動作している仕組みであり、ユーザ側での操作は何も必要としないためです。AppArmor での性能劣化もほとんど発生しません。AppArmor のプロファイルでカバーしていない動作や、AppArmor によって禁止されているアプリケーションの動作が発生したような場合でも、管理者がそれに気づいてプロファイルの調整を行うだけであるためです。

AppArmor には、様々な既定のアプリケーションプロファイル集が用意されていて、標準的な Linux サービスを保護しています。他のアプリケーションを保護したい場合は、AppArmor のツールを利用して、保護対象のアプリケーションに対してプロファイルを作成してください。本章では、プログラムに対する予防接種の考え方について説明しています。既に一通りの知識をお持ちで、AppArmor のプロファイルを作成したり管理したりしたい場合は、第31章「プロファイルのコンポーネントと文法」, 第33章「YaST を利用したプロファイルの構築と管理」, 第34章「コマンドラインからのプロファイル構築」などをお読みください。

AppArmor では、ネットワークサービスに対する直感的なアクセス制御の仕組みが提供されています。この仕組みでは、それぞれのプログラムが読み込んだり書き込んだり、実行したりすることを許すファイルのほか、どの種類のネットワークに接続できるのかを制御します。このような仕組みにより、それぞれのプログラムが期待通りの動作をするように強制することができます。また、AppArmor ではプログラムを検疫することで、不正な処理によるダメージからシステムを保護することができるようになっています。

AppArmor はホストへの侵入保護や強制アクセス制御の仕組みを提供するものです。従来のアクセス制御では、コンピュータを巨大なタイムシェアリングシステムとして使用してきた経緯から、ユーザを中心にして構成されてきました。現在のネットワークサーバでは、ユーザに対して直接的なログインを許可することはなく、ユーザに対して様々なネットワークサービス (Web, メール, ファイル, 印刷などの各サーバ) を提供する存在へと変遷しています。AppArmor では、ネットワークサービスやその他のプログラムに対するアクセス制御を設定し、脆弱性を突かれないように保護します。



ヒント: AppArmor に関する背景情報

AppArmor に関するより深い情報のほか、全体的な考え方については、28.2項「AppArmor プロファイリングに関する背景となる情報」をお読みください。

30.1 AppArmor フレームワークの紹介

本章では、「舞台裏」で何が起こっているのか (および YaST の裏では何を行っているのか) について、非常に基本的な部分を説明しています。

AppArmor のプロファイルは純粋なテキストファイル形式で、ここにはパス表記とアクセス許可が書かれています。プロファイルに関する詳細な書式情報については、31.1項「AppArmor プロファイル内の各部の意味」をお読みください。また、このテキストファイル内にはディレクティブも含まれていて、ここには AppArmor のルーチンがプロセスやプログラムを検疫する際に強制される仕組みも設定されています。

AppArmor のプロファイルやポリシーを構築したり強制したりするにあたって、下記のようなツールが提供されています:

`aa-status`

`aa-status` は、現時点での AppArmor の制限について、様々な内容をレポートします。

`aa-unconfined`

`aa-unconfined` はお使いのシステムで動作しているアプリケーションのうち、ネットワーク接続を待ち受けていながら、AppArmor のプロファイルで保護されていないものを検出します。このツールに関する詳細については、34.7.3.12項「aa-unconfined: 保護されていないプロセスの識別」をお読みください。

`aa-autodep`

`aa-autodep` はプロファイルに対する基本フレームワークで、本番環境で使用する前に肉付けを行う必要があるものです。生成されたプロファイルを読み込んで不平モードに設定すると、AppArmor のルールでカバーされていないアプリケーションの挙動をレポートします。このツールに関する詳しい使い方については、34.7.3.1項「aa-autodep: 近似プロファイルの作成」をお読みください。

`aa-genprof`

`aa-genprof` は基本的なプロファイルを作成して、そのプロファイルを元にしてアプリケーションを実行し、プロファイルを改良するよう求めて、AppArmor ポリシーで配慮する必要のある箇所をイベントとして報告します。アプリケーションの実行時に生成されたそれぞれのログイベントに対しては質問が表示され、どのように扱うべきかを判断することになります。プロファイルの生成が終わったあとは、そのまま強制モードに配置されます。このツールの詳しい使い方については、34.7.3.8項「aa-genprof: プロファイルの生成」をお読みください。

`aa-logprof`

`aa-logprof` は、アプリケーションが不平モードと強制モードの両方の制限で生成したログ項目を読み込んで確認することができるツールです。プロファイル内に新しい項目を生成する支援を行います。このツールの詳しい使い方については、34.7.3.9項「aa-logprof: システムログのスキャン」をお読みください。

`aa-easyprof`

`aa-easyprof` は AppArmor のプロファイル生成に関して使いやすいインターフェイスを提供します。`aa-easyprof` では雛形とポリシーグループを利用して、アプリケーションのプロファイルを素早く作成できるようにします。ただし、このツールはポリシーの生成支援までは行いますが、雛形とポリシーグループ、そして抽象化の仕組みに大きく依存していることに注意してください。`aa-genprof` や `aa-logprof` と比較すると、`aa-easyprof` では制限の緩いプロファイルを作成することになります。

`aa-complain`

`aa-complain` は AppArmor のプロファイルを強制モードから不平モードに切り替えます。プロファイル内に書かれたルールへの違反はログ内に記録されるものの、プロファイルは強制されません。このツールに関する詳しい使い方については、34.7.3.2項「aa-complain: 不平モード (学習モード) への突入」をお読みください。

`aa-enforce`

`aa-enforce` は AppArmor のプロファイルを不平モードから強制モードに切り替えます。プロファイル内に書かれたルールへの違反は記録され拒否されるようになり、結果としてプロファイルを強制することになります。この津ルールに関する詳しい使い方については、34.7.3.6項「aa-enforce: 強制モードへの突入」をお読みください。

`aa-disable`

`aa-disable` は、1 つもしくは複数の AppArmor プロファイルに対して、強制モードを無効化することができます。このコマンドはカーネルからプロファイルの読み込みを解除するだけでなく、AppArmor の起動時に読み込まれないようにもします。`aa-enforce` および `aa-complain` を使用することで、この動作を変更することができます。

`aa-exec`

`aa-exec` は指定した AppArmor プロファイルやネームスペースで制限した状態でプログラムを起動します。プロファイルとネームスペースの両方を指定すると、コマンドは新しいポリシーネームスペース内のプロファイルで制限されることになります。ネームスペースのみを指定した場合は、現在の制限のプロファイル名を使用します。プロファイルもネームスペースも指定しない場合は、標準的なプロファイル適用 (つまり `aa-exec` を使用せずに実行した場合) と同じ動作になります。

`aa-notify`

`aa-notify` はお使いのデスクトップ環境内に AppArmor の通知を表示することのできる、使いやすいユーティリティです。指定した日数の直近の通知に関して、概要を表示するように設定することもできます。詳しくは 34.7.3.13項「aa-notify」をお読みください。

30.2 予防接種を行うプログラムの決定

AppArmor に関するある程度の知識を得たら、次にプロファイルを作成するアプリケーションの選択を行いましょう。プロファイルを作成する必要のあるプログラムとは、権限を仲介するタイプのプログラムです。下記のようなプログラムでは、一般ユーザが保持していない資源へのアクセスを行うことがありますので、実質的に権限を仲介していることになります:

`cron` ジョブ

`cron` で定期的に実行されるタイプのプログラムです。これらのプログラムは様々な情報源から入力を受け付け、特別な (場合によっては `root` の) 権限で動作します。たとえば `cron` では `/usr/sbin/logrotate` を実行することがありますが、これはシステムログのローテーション (一定時間以上古いログを廃棄する) のほか、圧縮やメール送信などを行うことがあります。これらの種類のプログラムを見つける方法について、詳しくは 30.3項「cron ジョブへの予防接種」をお読みください。

Web アプリケーション

CGI スクリプトや PHP のページ、その他の複雑な Web アプリケーションなど、Web ブラウザ経由で実行されるプログラムです。これらの種類のプログラムを見つける方法について、詳しくは 30.4.1項「Web アプリケーションへの予防接種」をお読みください。

ネットワークエージェント

ネットワークポートを開いて通信するプログラム (クライアントやサーバ) です。電子メールクライアントや Web ブラウザなどのユーザクライアントでも、権限を仲介していることになります。これらのプログラムは、ユーザのホームディレクトリに書き込むことのできる権限を保有していて、場合によっては不正な Web サイトや電子メール内に埋め込まれた不正なコードにアクセスすることで、潜在的に攻撃を受ける可能性があるためです。これらの種類のプログラムを見つける方法について、詳しくは 30.4.2項「ネットワークエージェントに対する予防接種」をお読みください。

逆に、権限を使用しないプログラムに対しては、プロファイルを作成する必要はありません。たとえばシェルスクリプトでは `cp` プログラムを利用してファイルのコピーを行いますが、`cp` プログラムには既定のプロファイルやサブプロファイルが提供されていません。これは `cp` プログラムが単純にファイルコピーの機能しか提供しないためで、呼び出し元のシェルスクリプトのプロファイルを引き継ぐだけで十分であるためです。そのため、`cp` プログラムは、呼び出し元のシェルスクリプトのプロファイルに書かれた許可内容に応じてファイルをコピーすることになります。

30.3 cron ジョブへの予防接種

`cron` ジョブ内で実行されるプログラムを見つけるには、ご利用の環境内の `cron` の設定をご確認ください。残念ながら、`cron` の設定は比較的複雑で、様々なファイルを調べる必要があります。それぞれ下記のファイルをご覧ください:

```
/etc/crontab
/etc/cron.d/*
/etc/cron.daily/*
/etc/cron.hourly/*
/etc/cron.monthly/*
/etc/cron.weekly/*
```

`crontab` コマンドを実行すると、現在のユーザに対する cron ジョブの一覧を表示したり編集したりすることができます。`root` に対する `cron` ジョブの一覧を表示／編集したい場合は、`root` になってから `crontab -e` を実行することで編集を、`crontab -l` を実行することで一覧表示をすることができます。

30.4 ネットワークアプリケーションへの予防接種

`aa-unconfined` ツールを使用することで、プロファイルを作成すべきネットワークサーバデーモンを自動的に見つけることができます。

`aa-unconfined` ツールは `netstat -nlp` コマンドを利用して、コンピュータ内でポートを開いているプログラムを検出します。このツールでは、開いているポート番号と対応するプログラム、そして読み込まれている AppArmor のプロファイルセットをそれぞれ検出することができます。検出が完了すると、`aa-unconfined` ツールはそれぞれのプログラムと、対応する AppArmor のプロファイルが存在していればそれを、存在していなければ「none」(つまり制限が課されていない) を表示します。



注記

新しいプロファイルを作成した場合、AppArmor で正しく制限を受けるようにするため、プロファイルを作成したプログラムを再起動しなければなりません。

下記は `aa-unconfined` の出力例です:

```
3702❶ /usr/sbin/sshd❷ confined
   by '/usr/sbin/sshd❸ (enforce)'
4040 /usr/sbin/smbd confined by '/usr/sbin/smbd (enforce)'
4373 /usr/lib/postfix/master confined by '/usr/lib/postfix/master (enforce)'
4505 /usr/sbin/httpd2-prefork confined by '/usr/sbin/httpd2-prefork (enforce)'
646 /usr/lib/wicked/bin/wickedd-dhcp4 not confined
647 /usr/lib/wicked/bin/wickedd-dhcp6 not confined
5592 /usr/bin/ssh not confined
7146 /usr/sbin/cupsd confined by '/usr/sbin/cupsd (complain)'
```

❶ 最初に表示されるのが番号です。この番号は、接続を待ち受けているプログラムのプロセス ID 番号 (PID) です。

❷ 次に表示されるのが文字列で、接続を待ち受けているプログラムの絶対パスです。

❸ 最後に表示されるのが、プログラムの権限を制限しているプロファイル (もしあれば) です。



注記

`aa-unconfined` は `root` の権限が必要なプログラムであり、AppArmor のプロファイルによる制限が課されたシェルからは、実行すべきではありません。

`aa-unconfined` ではネットワークインターフェイスを区別せずに扱います。そのため、内部の LAN インターフェイスのみで待ち受けているプロセスに対しても、ポートを開いていれば一覧に記載されることになります。

ユーザが使用しているネットワーククライアントアプリケーションの検出は、お使いの環境によって異なる方法になります。`aa-unconfined` ツールは、クライアントアプリケーションが開いているネットワークポートに対しても報告を行いますが、`aa-unconfined` で分析している最中に動作していたアプリケー

ションのみが検出されます。一般に、ネットワークサーバアプリケーションは常に動作させておくものですが、クライアントアプリケーションは必要に応じて起動する仕組みであることから、常に検出されるとは限らないことになります。

AppArmor のプロファイルによるネットワーククライアントアプリケーションへの適用は、ユーザ側の希望に応じて行ってください。そのため、クライアントアプリケーションのプロファイル作成については、ここでは説明しません。

デスクトップアプリケーションを積極的に制限したい場合は、`aa-unconfined` コマンドに `--paranoid` オプションを追加して実行してください。これにより、動作中の全てのプロセスを一覧で表示し、それらに対して関連づけられている可能性のある AppArmor プロファイルを表示することができます。これにより、AppArmor のプロファイルを必要とするプログラムを判断することができるようになります。

新しくプロファイルを作成した場合やプロファイルを修正した場合、可能であれば、実施したアプリケーションの使用範囲を添えて `apparmor@lists.ubuntu.com` のメーリングリストに送信していただければ幸いです。AppArmor チームではその内容を精査して、openSUSE Leap の一部として組み込むかもしれません。ただし、お送りいただいたプロファイルを全て組み込むことは保証できませんが、できる限りの努力はさせていただきます。

30.4.1 Web アプリケーションへの予防接種

Web アプリケーションを見つけたい場合は、まず Web サーバの設定ファイルを調査することをお勧めします。Apache Web サーバであれば、非常に細かく設定することができるシステムであるため、設定によって Web アプリケーションを様々なディレクトリ内に配置することができます。openSUSE Leap の既定では、Web アプリケーションは `/srv/www/cgi-bin/` 内に配置されます。なお、可能な限りそれぞれの Web アプリケーションに AppArmor プロファイルを作成してください。

これらのプロクラムが見つかったら、あとは `aa-genprof` ツールと `aa-logprof` ツールを使用することで、それらに対する AppArmor のプロファイルを作成することができます。

CGI プログラムは Apache Web サーバが実行する仕組みであることから、openSUSE Leap 向けの Apache2 それ自身用に作成されている `usr.sbin.httpd2-prefork` というプロファイルは、CGI プログラムにあわせて実行許可を追加するよう変更しなければなりません。たとえば `/srv/www/cgi-bin/my_hit_counter.pl rPx` という行を追加すると、`my_hit_counter.pl` という Perl スクリプトを Apache から実行する許可を与え、`my_hit_counter.pl` 用のプロファイルを別途に用意することになります。`my_hit_counter.pl` に専用のプロファイルを用意していない場合は、`/srv/www/cgi-bin/my_hit_counter.pl rix` のように指定して、`my_hit_counter.pl` が `usr.sbin.httpd2-prefork` のプロファイルを引き継ぐように設定してください。

Apache が実行するそれぞれの CGI スクリプトに対していちいち実行許可を設定するのは、ユーザにとっては不便に感じるかもしれません。その代わりに管理者は、複数の CGI スクリプトに対してアクセスを制御することができます。たとえば `/srv/www/cgi-bin/*.{pl,py,pyc} rix` のように指定すると、`/srv/www/cgi-bin/` ディレクトリ内にあって `.pl` (Perl スクリプト) もしくは `.py` や `.pyc` (Python スクリプト) で終わるファイル名のスクリプト全てを含めることができるようになります。この設定では、`ix` と書かれていることから、Python スクリプトが Apache のプロファイルを引き継ぐようになります。このような設定は、それぞれの CGI スクリプトに対して、いちいち個別のプロファイルを作成したくない場合に便利です。



注記

Apache モジュール (`mod_perl` や `mod_php`) で Web アプリケーションを動作させていて、サブプロセスを制限するモジュール (`apache2-mod-apparmor`) の機能を使用したい場合は、YaST 内もしくはコマンドラインでプロファイルを追加する際に、ChangeHat の機能をお使いください。サブプロセス制限に関する利点について、詳しくは 35.2項「チェンジハット対応のアプリケーションの管理」をお読みください。

`mod_perl` や `mod_php` を使用する Web アプリケーションのプロファイルを作成する場合は、少し異なる作業が必要となります。この場合の「プログラム」は、Apache プロセス内のモジュールが直接解釈して実行するスクリプトとなりますので、プログラムの実行が発生しないためです。この場合は、AppArmor に対応した Apache が要求先の URI の名前からサブプロファイル (「ハット」) を判断して `change_hat()` を呼び出します。



注記

Apache がモジュールのスクリプトを探す際の設定にもよりますが、スクリプトに対する名前と URI とが無関係になってしまうことがあります。Apache 側で異なる場所にスクリプトを配置するように設定している場合は、AppArmor がアクセス違反を報告する際、ログファイル内に異なる名前で記録されてしまうことがあります。詳しくは 第37章「プロファイルを作成したアプリケーションの管理」をお読みください。

`mod_perl` や `mod_php` のスクリプトの場合、これはリクエストした Perl スクリプトや PHP ページの名前になります。たとえば下記のようなサブプロファイルを設定したとすると、`localtime.php` のページからローカルのシステム時刻とロケールファイルへのアクセスができるようになります:

```
/usr/bin/httpd2-prefork {
  # ...
  ^/cgi-bin/localtime.php {
```

```
    /etc/localtime                  r,
    /srv/www/cgi-bin/localtime.php  r,
    /usr/lib/locale/**              r,
  }
}
```

何もサブプロファイルを指定しない場合、Apache の AppArmor 対応版は `DEFAULT_URI` ハットを適用します。このサブプロファイルは Web ページを表示するだけであれば十分なサブプロファイルです。AppArmor が提供する `DEFAULT_URI` は、下記のようになっています:

```
^DEFAULT_URI {
    /usr/sbin/suexec2                   mixr,
    /var/log/apache2/**                 rwl,
    @{HOME}/public_html                 r,
    @{HOME}/public_html/**              r,
    /srv/www/htdocs                     r,
    /srv/www/htdocs/**                  r,
    /srv/www/icons/*.{gif,jpg,png}      r,
    /srv/www/vhosts                     r,
    /srv/www/vhosts/**                  r,
    /usr/share/apache2/**               r,
    /var/lib/php/sess_*                 rwl
}
```

Apache が提供する全ての Web ページと CGI スクリプトに対して、単一の AppArmor プロファイルを使用したい場合は、`DEFAULT_URI` サブプロファイルを編集することをお勧めします。Apache 内での Web アプリケーションの制限について、詳しくは 第35章「チェンジハット機能による Web アプリケーションのプロファイル作成」をお読みください。

30.4.2 ネットワークエージェントに対する予防接種

プロファイルを作成する必要のあるネットワークサーバデーモンやネットワーククライアント (`fetchmail` や Firefox など) を見つけるには、まずお使いのマシンで開いているポートを探してください。その後、これらのポートを使用しているプログラムを探して、できる限り多くのプログラムに対してプロファイルを作成してください。ネットワークポートを開いている全てのプログラムに対してプロファイルを設定すれば、攻撃者は AppArmor のプロファイルポリシーで許可されていない限り、マシン内のファイルシステムには全くアクセスできなくなります。

スキャナ (nmap など) を使用すれば、サーバの外側から開いているネットワークポートを調べることができますし、内部からであれば `root` で `netstat --inet -n -p` を実行することで、調べることができます。その後、それらのポートで待ち受けているプログラムが何であるのかを調査します。



ヒント

利用可能なオプションについて、詳しくは `netstat` コマンドのマニュアルページをお読みください。

31 プロファイルのコンポーネントと文法

改訂履歴

2024-06-26

アプリケーションに対する AppArmor のプロファイル作成はわかりやすく、直感的な仕組みになっています。AppArmor にはプロファイル作成を支援するツールが添付されているため、プログラミングの知識もスクリプトの処理も必要とはなりません。管理者に対して求められる作業は、セキュリティを強化する目的で、各アプリケーションに対して最も厳格なアクセス制御とプログラム実行のポリシーを設定することだけです。

アプリケーションに対するプロファイルの更新や修正は、ソフトウエアの設定を変更したり、必要な機能範囲の変化が起こったりした場合のみです。AppArmor には、プロファイルの更新や修正に対応した、直感的なツールが用意されています。

プロファイルを作成したいプログラムを選択したあとは、実際に AppArmor のプロファイルを作成するだけです。プロファイルの作成を行うにあたっては、コンポーネントやその文法の理解を深める必要があります。AppArmor には、シンプルで再利用性の高いプロファイルを作成するための、さまざまな機能が用意されています:

ファイル取り込み
: include と呼ばれるステートメントを使用することで、他の AppArmor プロファイル内の一部分を取り込むことができます。これにより、新しいプロファイルの構造を単純化することができます。

抽象
: 抽象機能は、一般的なアプリケーションの処理をまとめて取り込む機能を提供します。

プログラムチャンク
: プログラムチャンクとはファイル取り込み機能の一部で、特定のプログラムスイートに固有として設定されているプロファイルの一部を、取り込むことができる機能です。

ケーパビリティ項目
: ケーパビリティ項目とは、POSIX.1e (https://ja.wikipedia.org/wiki/POSIX#POSIX.1) として定められている Linux ケーパビリティ向けのプロファイル項目で、制限されたプロセスが権限の必要なシステムコールを行う際、より細かい制御を行うことのできる仕組みを指します。

ネットワークアクセス制御項目
: ネットワークアクセス制御項目とは、アドレスの種類とファミリをベースにして、ネットワークへのアクセスを制御するための項目です。

ローカル変数定義
: ローカル変数を定義することで、パスのショートカットを設定することができます。

ファイルアクセス制御項目

ファイルアクセス制御項目とは、アプリケーション側からのアクセスを許可するファイルの一覧を表す項目です。

rlimit 項目

rlimit 項目は、アプリケーションに対するリソース制限を設定し、制御するための項目です。

プロファイルを作成すべきプログラムの判断については 30.2項「予防接種を行うプログラムの決定」を、YaST を利用して AppArmor のプロファイルを作成する方法については 第33章「YaST を利用したプロファイルの構築と管理」を、AppArmor のコマンドラインインターフェイスを利用したプロファイルの作成については 第34章「コマンドラインからのプロファイル構築」を、それぞれお読みください。

AppArmor のプロファイルに関する詳細については、`man 5 apparmor` をお読みください。

31.1 AppArmor プロファイル内の各部の意味

プロファイルに何が含まれているのか、およびプロファイルを作成するにはどうしたらよいのかを説明するにあたって、もっとも簡単な方法は例を挙げる方法です。たとえば `/usr/bin/foo` という架空のアプリケーションに対して、下記のようなプロファイルが存在する場合を例にします:

```
#include <tunables/global>❶

# a comment naming the application to confine
/usr/bin/foo❷ {❸
   #include <abstractions/base>❹

   capability setgid❺,
   network inet tcp❻,

   link /etc/sysconfig/foo -> /etc/foo.conf,❼
   /bin/mount            ux,
   /dev/{,u}❽random      r,
   /etc/ld.so.cache      r,
   /etc/foo/*            r,
   /lib/ld-*.so*         mr,
   /lib/lib*.so*         mr,
   /proc/[0-9]**         r,
   /usr/lib/**           mr,
   /tmp/                 r,❾
   /tmp/foo.pid          wr,
   /tmp/foo.*            lrw,
   /@{HOME}❿/.foo_file   rw,
   /@{HOME}/.foo_lock    kw,
   owner⓫ /shared/foo/** rw,
   /usr/bin/foobar       Cx,⓬
   /bin/**               Px -> bin_generic,⓭
```

```
   # a comment about foo's local (children) profile for /usr/bin/foobar.

   profile /usr/bin/foobar 14 {
      /bin/bash          rmix,
      /bin/cat           rmix,
      /bin/more          rmix,
      /var/log/foobar*   rwl,
      /etc/foobar        r,
   }

  # foo's hat, bar.
   ^bar 15 {
    /lib/ld-*.so*         mr,
    /usr/bin/bar          px,
    /var/spool/*          rwl,
   }
}
```

1. ここでは、変数定義を含むファイルを読み込んでいます。
2. 制限対象となるプログラムに対して、正規化されたパスを指定しています。
3. この中括弧 (`{}`) は、ステートメントやサブプロファイル、パス項目やケーパビリティ項目、ネットワーク項目などのコンテナとして機能するものです。
4. このディレクティブにより、AppArmor プロファイルのコンポーネントを取り込んでいます。これにより、プロファイルを単純化できるようになっています。
5. capability (ケーパビリティ) 項目のステートメントでは、POSIX.1e ドラフトで規定されたケーパビリティを有効化しています。
6. アプリケーションに対して、どの種類のネットワークアクセスを許可するかを指定しているディレクティブです。詳しくは 31.5項「ネットワークアクセス制御」をお読みください。
7. リンク対ルールでは、リンクの元と宛先をそれぞれ指定することができます。詳しくは 31.7.6項「リンク対」をお読みください。
8. この中括弧 (`{}`) は、カンマ区切りで複数の項目を列挙するための仕組みです。この場合、「u」があるものと無いものに展開されます。
9. プログラム側から、ファイルシステム内のどの領域にアクセスを許可するのかを設定する、パス項目です。この項目では、最初に絶対パス表記でのファイル名 (ワイルドカード可) を記述し、次に許可されるアクセスモード (たとえば `r` であれば読み込み、`w` であれば書き込み、`x` であれば実行など) をそれぞれ指定します。スペースやタブなど、任意の種類のホワイトスペースをパス名の前に指定することができますが、パス名とアクセスモードの指定は区切らなければなりません。また、アクセスモードと行末のカンマとの間は、スペースを入れても入れなくてもかまいません。利用可能なアクセスモードに関する詳細は、31.7項「ファイルアクセス許可のアクセスモード」をお読みください。

10 この変数は、プロファイル本体を変更することなく内容を変更したい場合に使用するものです。

11 所有者条件ルールと呼ばれるルールで、自分自身で所有しているファイルに対して、読み込みや書き込みの権限を許可することができます。詳しくは 31.7.8項「所有者条件ルール」をお読みください。

12 この項目は、`/usr/bin/foobar` という名前のローカルプロファイルに遷移させるためのものです。利用可能な実行モードに関する概要は、31.12項「実行モード」をお読みください。

13 グローバルスコープ内に位置する bin_generic というプロファイルへの遷移を表す、名前付きプロファイル遷移です。詳しくは 31.12.7項「名前付きプロファイル遷移」をお読みください。

14 ここでは、ローカルプロファイル `/usr/bin/foobar` を定義しています。

15 ここでは、アプリケーションに対する「ハット」と呼ばれるサブプロファイルを参照しています。AppArmor のチェンジハット機能について、詳しくは 第35章「チェンジハット機能による Web アプリケーションのプロファイル作成」をお読みください。

プログラムに対してプロファイルを作成すると、プログラムはプロファイル内に指定されたモードでファイルへのアクセスが許可されるほか、同じくプロファイル内に指定された POSIX ケーパビリティの使用のみが許可されます。これらの制限は、Linux のアクセス制御に付加する形で動作します。

例: プログラム側でケーパビリティ `CAP_CHOWN` を使用できるようにするには、通常の Linux アクセス制御で `CAP_CHOWN` に対するアクセス許可を持つ (通常は `root` が所有するプロセスである必要があります) だけでなく、プロファイル内に `chown` のケーパビリティが設定されていなければなりません。また、プログラムが `/foo/bar` というファイルに書き込む場合、ファイルの所有者やパーミッションの設定で書き込みできる権限を持つだけでなく、プロファイル内に `/foo/bar w` という設定がなければなりません。

AppArmor のルールへの違反は、`audit` パッケージがインストールされていれば `/var/log/audit/audit.log` に、インストールされていない場合は `/var/log/messages` に記録されます (syslog パッケージさえもインストールされていない場合は、`journalctl` にのみ記録されます) 。このような AppArmor の仕組みにより、実際に攻撃を受けた場合であっても、攻撃による被害を軽減することができるようになっています。

31.2 プロファイルの種類

AppArmor ではプロファイルが 4 種類に分けられています。それぞれ標準プロファイルと未接続プロファイル、ローカルプロファイルとハットと呼ばれます。標準プロファイルと未接続プロファイルは単独で動作するプロファイルであり、`/etc/apparmor.d/` ディレクトリ内にファイルとして配置されます。ローカルプロファイルとハットは、親プロファイル内に組み込まれる子プロファイルで、アプリケーションのサブタスクに対してより厳しい、もしくは別途の制限を提供するために使用します。

31.2.1 標準プロファイル

既定の AppArmor プロファイルは、プログラムに対して名前で接続される仕組みであるため、プロファイル名は制限対象のアプリケーションパスに一致していなければなりません。

```
/usr/bin/foo {
...
}
```

上記のようなプロファイルが存在すると、制限を受けていないプロセスが `/usr/bin/foo` を実行すると、自動的に読み込まれて制限が適用されます。

31.2.2 未接続プロファイル

未接続プロファイルは、ファイルシステムのネームスペース内に存在しておらず、そのためにアプリケーションの実行時に自動的に読み込まれ適用されるものではありません。未接続プロファイルの場合、プロファイル名の前に `profile` というキーワードを指定します。プロファイル名には自由な名前を設定することができますが、`:` や `.` で始まる名前は指定できません。また、プロファイル名にはスペースを入れることもできますが、この場合は正しく引用符を指定しなければなりません。また、名前が `/` で始まる場合、そのプロファイルは標準プロファイルとして扱われます。たとえば下記の 2 つのプロファイルは、同じ名前の標準プロファイルを意味するものになります:

```
profile /usr/bin/foo {
...
}
/usr/bin/foo {
...
}
```

上述のように、未接続プロファイルは自動的に読み込まれませんし、`Px` ルールで遷移することもありません。この未接続プロファイルを使用するには、名前付きプロファイル遷移の機能 (詳しくは 31.12.7項「名前付きプロファイル遷移」をお読みください) を使用するか、もしくは `change_profile` ルール (詳しくは 31.2.5項「ルール変更」をお読みください) で適用する必要があります。

未接続プロファイルは一般に、システム全体のプロファイルとしては制限を加えるべきではないシステムユーティリティ (例: `/bin/bash`) などに対して、特別なプロファイルを割り当てるために使用します。また、役割の設定やユーザへの制限などを設定する際にも使用します。

31.2.3 ローカルプロファイル

ローカルプロファイルは、制限を受けているアプリケーションがユーティリティプログラムを起動する際、そのユーティリティに対する特殊な制限を設定する際に便利な機能です。この種類のプロファイルは標準プロファイルと同様に指定することができますが、親プロファイル内に組み込まれて使用され、かつ `profile` キーワードで始める点が異なります。

```
/parent/profile {
   ...
   profile /local/profile {
      ...
   }
}
```

ローカルプロファイルに遷移させたい場合は、`cx` ルール (詳しくは 31.12.2項「専用ローカルプロファイル実行モード (cx)」をお読みください) もしくは名前付きプロファイル遷移 (詳しくは 31.12.7項「名前付きプロファイル遷移」をお読みください) を使用します。

31.2.4 ハット

AppArmor では、"ハット" とは追加の制限を提供するローカルプロファイルを意味するほか、`change_hat` を利用して遷移することのできる、暗黙のルールを意味します。詳しくは 第35章「チェンジハット機能による Web アプリケーションのプロファイル作成」をお読みください。

31.2.5 ルール変更

AppArmor では、`change_hat` および `change_profile` と呼ばれるルールを利用して、制御ドメインの遷移を行うことができます。`change_hat` がプロファイル内で定義したハットに遷移するのに対して、`change_profile` は他のプロファイルを参照する処理を行います。`change_profile` では、下記のように `change_profile` キーワードを明記します:

```
change_profile -> /usr/bin/foobar,
```

`change_hat` と `change_profile` のいずれも、個別のアプリケーションを起動することなく、アプリケーション内でのプロファイル遷移を実現します。`change_profile` は一般に、読み込まれたプロファイル間で一方向の遷移を行います。それに対して、`change_hat` は戻ってくることを前提とした遷移となり、後から正しい機密鍵を指定することで、親プロファイルに戻ってくることができます。

`change_profile` は、アプリケーションが特定の設定フェーズを経由して起動し、後から権限レベルを落とすような仕組みである場合に、最適な仕組みです。起動フェーズ内でマッピングしたり開いたりしたリソースについては、プロファイルが変更されてもアクセスが許可され続けますが、新しいプロファイル

では新しいリソースを開く処理が制限されたり、リソースによっては切り替え前よりも制限を厳しくしたりするような動作に適切な仕組みです。特に、ケーパビリティやファイルのリソースについては制限ができる (ただしメモリマップされていなければ) ものの、メモリについては以前と同じく利用できるようになります。

`change_hat` はアプリケーションが仮想マシンやインタプリタを動作させるような場合に最適な仕組みで、それらの仮想マシンやインタプリタがアプリケーションのリソースへのアクセスを提供しないような場合に適切です (たとえば Apache の `mod_php` などがそれにあたります) 。なお `change_hat` は、元のプロファイルに戻るための機密鍵をアプリケーションのメモリ内に保存しますので、権限を縮小したフェーズではメモリに直接悪性できる権限を持つべきではありません。また、ハットではファイルハンドルへのアクセスを制限することはできますが、閉じることはできませんので、ファイルアクセスが正しく分離されていることも重要となります。もしもアプリケーションがバッファリングを行っていて、バッファを利用したファイルアクセスの機能を提供している場合、これらのファイルへのアクセスはカーネルからは検出することができず、新しいプロファイルでも制限がされないことがありうることにも注意してください。



警告: ドメイン遷移の安全性について

`change_hat` や `change_profile` によるドメイン遷移は、プロセスのメモリマッピングに影響を与えることができませんし、既に開いているリソースを閉じることもしませんので、exec によるドメイン遷移よりはずっと安全性が低いことに注意してください。

31.3 include ステートメント

include ステートメントは、プロファイルを単純化するため、他の AppArmor プロファイルのコンポーネントを取り込むことができるディレクティブです。一般に、プログラムからファイルへのアクセス許可を設定するための仕組みで、ファイルやディレクトリへのアクセスを別ファイルに切り出すことができますので、他のプログラムでも同様に必要となるようなファイルやディレクトリを列挙することができます。これにより、プロファイルのサイズも小さくすることができます。

include ステートメントは通常、ハッシュ (`#`) 記号から書き始めます。ただし、ハッシュ (`#`) 記号はプロファイル内のコメント文を記述する際にも使用するものであるため、プロファイルが少しわかりにくくなってしまいます。このような問題から、`#include` はその前にハッシュ記号がある場合には効果を失うように作られている (`##include` はコメントとして扱われます) ほか、`#` と `include` の間にスペースを入れた場合も、同様に効果を失うように作られています (`# include` もコメントとして扱われます) 。

このほか、冒頭のハッシュ記号を省略して、`include` だけで記述することもできます。

```
include "/etc/apparmor.d/abstractions/foo"
```

上記は、下記と同じ意味になります:

```
#include "/etc/apparmor.d/abstractions/foo"
```



注記: 末尾に ',' を付けてはならない件について

include ステートメントは C 言語のプリプロセッサの文法に従って作られているため、その他の AppArmor ルールのように、文末に ',' を入れることはありません。

また、文法を少し変えることで、`include` の動作を変更することができます。たとえばパスの指定の際、`""` で括って指定すると、絶対パスや相対パスとして読み込ませることができます。

```
include "/etc/apparmor.d/abstractions/foo"    # 絶対パス表記
include "abstractions/foo"    # 現在のディレクトリからの相対パス表記
```

なお相対パス表記の場合、現在読み込んでいるファイルからの相対パスとして扱われることに注意してください。たとえば現在のプロファイルが `/etc/apparmor.d/bar` である場合、下記のように include を記述したとします:

```
include "abstractions/foo"
```

すると、`/etc/apparmor.d/abstractions/foo` というファイルを取り込むことになります。また、

```
include "example"
```

のような include ステートメントが `/etc/apparmor.d/abstractions/foo` プロファイル内に存在したとすると、`/etc/apparmor.d/abstractions/example` というファイルを取り込むことになります。

また、`<>` を使用してファイルパスを指定すると、インクルードパス (`-I` で指定することができます。既定では `/etc/apparmor.d` ディレクトリです) 内を検索することができます。たとえばインクルードパスが下記のように指定されているものとします:

```
-I /etc/apparmor.d/ -I /usr/share/apparmor/
```

この場合、下記のような include ステートメントを実行したとします:

```
include <abstractions/foo>
```

すると、まずは `/etc/apparmor.d/abstractions/foo` にファイルが存在していないかどうかを調べ、存在していない場合は `/usr/share/apparmor/abstractions/foo` のファイルが存在していないかどうかを調べる動作になります。



ヒント

既定のインクルードパスは、`apparmor_parser` に `-I` オプションを指定するか、もしくは `/etc/apparmor/parser.conf` 内に指定することで、上書きすることができます:

```
Include /usr/share/apparmor/
Include /etc/apparmor.d/
```

このファイル内でも複数の項目を設定することができます。ただし、複数の項目を設定した場合は、`apparmor_parser` に対して、順序通りに `-I` や `--Include` を指定したものとして扱われます。

include ステートメントのパス指定が '/' で終わるものであった場合、これはディレクトリ内にある全てのファイルを取り込むものとして処理されます。

お使いのアプリケーションのプロファイル作成を支援する目的で、AppArmor には 3 種類の include ステートメント (抽象／プログラムチャンク／チューナブル) が用意されています。

31.3.1 抽象

抽象とは一般的なアプリケーション処理をまとめた include ファイルで、たとえば認証機構やネームサービスルーチンへのアクセス、一般的なグラフィック要件やシステムアカウンティングなどが含まれます。これらの抽象内に記述されたファイルは、それぞれの処理で必要となるファイルであり、これらのうちのどれか 1 つにでもアクセスが必要となった場合、同じ抽象内の他のファイルに対しても、おそらくアクセスが必要となるものとして作られています (もちろん設定内容やプログラム側の要件にも依存します) 。抽象は `/etc/apparmor.d/abstractions` ディレクトリ内に配置されています。

31.3.2 プログラムチャンク

プログラムチャンクディレクトリ (`/etc/apparmor.d/program-chunks`) には、プログラムスイートごとに固有となるプロファイルチャンクのほか、スイート外では一般に必要とされないチャンクが含まれています。そのため、プロファイルウイザード (`aa-logprof` および `aa-genprof`) では、使用するよう提案することはありません。現時点では、プログラムチャンクは postfix プログラムスイート向けにのみ提供されています。

31.3.3 チューナブル

チューナブルのディレクトリ (`/etc/apparmor.d/tunables`) には、グローバル変数の定義が含まれています。プロファイル内で使用すると、これらの変数は処理時に展開され処理されますので、プロファイルそのものを変更することなく、値を変更することができるようになります。全てのプロファイルで使用される全てのチューナブル定義を追加するには、`/etc/apparmor.d/tunables/global` に設定します。

31.4 ケーパビリティ項目 (POSIX.1e)

ケーパビリティルールは、単純に `capability` の後ろに、`capabilities(7)` のマニュアルページに書かれた POSIX.1e のケーパビリティ名を記述するだけです。単一のルール内に複数のケーパビリティを記述したり、全てのケーパビリティを指定 (`capability` とだけ記述する) したりすることもできます。

```
capability dac_override sys_admin,   # 複数のケーパビリティ
capability,                          # 全てのケーパビリティ
```

31.5 ネットワークアクセス制御

AppArmor では、アドレスの種類とファミリをベースにして、ネットワークアクセスの制限を行うことができます。下記にネットワークアクセス制御ルールの書式を示します:

```
network [[<ドメイン>❶][<種類❷>][<プロトコル❸>]]
```

❶ 指定できるドメイン: `inet` , `ax25` , `ipx` , `appletalk` , `netrom` , `bridge` , `x25` , `inet6` , `rose` , `netbeui` , `security` , `key` , `packet` , `ash` , `econet` , `atmsvc` , `sna` , `pppox` , `wanpipe` , `bluetooth` , `unix` , `atmpvc` , `netlink` , `llc` , `can` , `tipc` , `iucv` , `rxrpc` , `isdn` , `phonet` , `ieee802154` , `caif` , `alg` , `nfc` , `vsock`

❷ 指定できる種類: `stream` , `dgram` , `seqpacket` , `rdm` , `raw` , `packet`

❸ 指定できるプロトコル: `tcp` , `udp` , `icmp`

AppArmor ツールではファミリと種類の指定にのみ対応しています。また、AppArmor モジュールは「ACCESS DENIED」(アクセス拒否) メッセージ内に `network ドメイン 種類` のみを書き込みます。さらに、これらのメッセージはプロファイル生成ツール (YaST およびコマンドライン) のみに出力されます。

下記の例は、AppArmor プロファイル内で使用されるネットワーク関連のルール例を示しています。ただし、後ろの 2 つの書式は、現時点での AppArmor ツールでは対応していないことに注意してください。

```
network❶,
network inet❷,
network inet6❸,
network inet stream❹,
network inet tcp❺,
network tcp❻,
```

❶ 全てのネットワーク処理を許可します。ドメインや種類、プロトコルでの制限を適用しません。

❷ IPv4 のネットワーク処理を許可します。

❸ IPv6 のネットワーク処理を許可します。

❹ IPv4 TCP のネットワーク処理を許可します。

❺ IPv4 TCP のネットワーク処理を許可します (1 行上のルールと同じ意味です) 。

❻ IPv4 と IPv6 の TCP ネットワーク処理を許可します。

31.6 プロファイル名／フラグ／パス／グロブ

プロファイルは、プログラムの実行ファイルのフルパスを指定することで、プログラムに割り当てられます。たとえば標準プロファイル (31.2.1項「標準プロファイル」) の場合、プロファイルは下記のように記述します:

```
/usr/bin/foo { ... }
```

下記では、プロファイルの命名の際や他の既存のコンテキスト内にプロファイルを配置する際、そして単純にファイルパスを指定する際に、便利なテクニックを紹介しています。

AppArmor ではディレクトリパスとファイルパスを明示的に区別して扱います。ディレクトリとして明示したい場合は、末尾に `/` を指定します:

`/some/random/example/* r`
: `/some/random/example` 内のファイルに対して、読み込みアクセスを許可します。

`/some/random/example/ r`
: ディレクトリにのみ読み込みアクセスを許可します。

`/some/**/ r`
: `/some` 以下の任意のディレクトリに対して、読み込みアクセスを許可します (ただし `/some/` 自身には許可しません) 。

`/some/random/example/** r`

`/some/random/example` 以下の任意のファイルとディレクトリに対して、読み込みアクセスを許可します (ただし `/some/random/example/` 自身には許可しません) 。

`/some/random/example/**[^/] r`

`/some/random/example` 以下のファイルに対して、読み込みアクセスを許可します。このとき、明示的にディレクトリを除外します (`[^/]`) 。

グロブ (もしくは正規表現マッチング) はワイルドカードなどを利用して、ファイルやサブディレクトリをまとめて指定するための仕組みです。一般的なシェル、たとえば csh, Bash, zsh などで使用されるグロブ書式を利用して指定することができます。

`*`	任意の長さの文字列にマッチします (ただし `/` は含まないものとします) 。 例: 複数パスの一括指定。
`**`	任意の長さの文字列にマッチします (`/` を含みます) 。 例: サブディレクトリを含む全ファイルの一括指定。
`?`	1 文字にマッチします (ただし `/` は含まないものとします) 。
`[abc]`	`a` , `b` , `c` のいずれかの文字にマッチします。 例: `/home[01]/*/.plan` のように指定すると、`/home0` および `/home1` のディレクトリ内にある `.plan` ファイルにアクセスを許可します。
`[a-c]`	`a` , `b` , `c` のいずれかの文字にマッチします。
`{ab,cd}`	`ab` という文字列、もしくは `cd` という文字列のいずれかにマッチします。 例: `/{usr,www}/pages/**` のように指定すると、`/usr/pages` 以下の全てのファイルと、`/www/pages` 以下の全てのファイルに対して、アクセスを許可します。
`[^a]`	`a` を除く全ての 1 文字にマッチします。

31.6.1 プロファイルフラグ

プロファイルフラグは、対応するプロファイルの動作を制御するためのものです。プロファイルの定義内に手作業で指定することで、フラグを追加することができます。プロファイルフラグは下記の書式で設定します:

```
/path/to/profiled/binary flags=(フラグ) {
  [...]
}
```

複数のフラグを指定したい場合は、カンマ ',' またはスペース ' ' で区切って指定します。プロファイルフラグには 3 種類のものがあります。モードフラグ、相対フラグ、接続フラグです。

モード フラグには `complain` フラグ (不正なアクセスが記録されるだけで、拒否はされない) があります。このフラグを指定しない場合、プロファイルは `enforce` フラグ (実際にポリシーを強制する) が指定されたものとして扱われます。



ヒント

プロファイル全体を complain (不平) モードに設定する場合、より柔軟な方法として、`/etc/apparmor.d/force-complain/` ディレクトリ内にシンボリックリンクを作成する方法があります。

```
ln -s /etc/apparmor.d/bin.ping /etc/apparmor.d/force-complain/bin.ping
```

相対 フラグには `chroot_relative` フラグ (ネームスペースではなく chroot に対する相対として扱う) のほか、`namespace_relative` フラグ (こちらが既定値で、chroot の外側で相対として扱う) があります。これらは相互排他です。

接続 フラグには相互排他関係にある 2 対のフラグがあります。`attach_disconnected` フラグと `no_attach_disconnected` フラグ (パス名がネームスペース外に解決された場合、ルートに割り当てるかどうか (つまり、冒頭に '/' があるかどうか)) と、`chroot_attach` フラグと `chroot_no_attach` フラグ (chroot 環境外に存在していて、同じネームスペース内に存在するファイルに対して、chroot 環境内からのパス名生成の制御) があります。

31.6.2 プロファイル内での変数の使用

AppArmor ではプロファイル内でパスを指定するために変数を使用することができます。グローバル変数を使用することで、作成するプロファイルに可搬性を持たせたり、パスに対するショートカットを定義したりすることができます。

グローバル変数が便利になる例としては、たとえばユーザのホームディレクトリが異なる場所にマウントされるような環境があります。この場合、変数を使用すれば、影響する全てのプロファイルのホームディレクトリを書き換える代わりに、変数の値を書き換えるだけで済みます。グローバル変数は `/etc/apparmor.d/tunables` 内で定義し、include ステートメントを利用して参照してください。ホームディレクトリであれば、`/etc/apparmor.d/tunables/home` ファイル内に `@{HOME}` と `@{HOMEDIRS}` の変数定義が存在しています。

一方のローカル変数は、プロファイルの冒頭部に記述します。これは chroot されたパスのベースディレクトリを指定するような場合に有用です。たとえば下記のようになります:

```
@{CHROOT_BASE}=/tmp/foo
/sbin/rsyslogd {
...
# chrooted applications
@{CHROOT_BASE}/var/lib/*/dev/log w,
@{CHROOT_BASE}/var/log/** w,
...
}
```

下記の例では、@{HOMEDIRS} でユーザのホームディレクトリが存在するディレクトリを設定し、@{HOME} ではホームディレクトリの一覧を設定しています。その後、@{HOMEDIRS} に対して、ホームディレクトリとして使用するディレクトリを追加しています。

```
@{HOMEDIRS}=/home/
@{HOME}=@{HOMEDIRS}/*/ /root/
[...]
@{HOMEDIRS}+=/srv/nfs/home/ /mnt/home/
```



注記

現在の AppArmor ツールでは、変数を利用するには、プロファイルを手作業で編集および管理しなければなりません。

31.6.3 パターンマッチ

プロファイル名にグロブ表記を使用することで、複数のバイナリにマッチするプロファイルを作成することができます。

下記の例は、`/usr/bin` もしくは `/bin` 内にある `foo` というバイナリに対するプロファイル例です。

```
/{usr/,}bin/foo { ... }
```

下記の例では、`/bin/foo` という実行ファイルの場合、正確にマッチする `/bin/foo` というプロファイルを選択します。それ以外の、たとえば `/bin/fat` というプログラムの場合は、`/bin/foo` にマッチしませんので、`/bin/f*` という名前の、`/bin/**` よりも厳密にマッチするプロファイルを選択します。

```
/bin/foo { ... }

/bin/f*  { ... }

/bin/**  { ... }
```

プロファイル名におけるグロブの使用例について、詳しくは `man 5 apparmor.d` で表示されるマニュアルページ (英語) 内の `Globbing` というセクションをお読みください。

31.6.4 ネームスペース

ネームスペースは異なるプロファイルセットを提供するために使用します。一方をシステム、他方を chroot 環境やコンテナなどに適用するためのものです。また、ネームスペースは階層構造を持っていて、親は子を参照できるものの、子は親を参照できなくなっています。また、ネームスペースは `:` で始まり、その後ろに英数字の名前を指定したあと、末尾に `:` と、必要に応じてダブルスラッシュ `//` を付けます。具体的には下記のようになります:

```
:childNameSpace://
```

子ネームスペースに読み込まれたプロファイルには、ネームスペースの指定が冒頭に付与されます (親側からの見た目):

```
:childNameSpace://apache
```

ネームスペースは `change_profile` API のほか、名前付きプロファイル遷移でも指定することができます:

```
/path/to/executable px -> :childNameSpace://apache
```

31.6.5 プロファイル名と接続仕様

プロファイルには名前のほか、接続仕様を指定することができます。この仕組みにより、ユーザや管理者にとって分かりやすいプロファイル名を指定しながら、同時に複雑なパターンマッチング (詳しくは 31.6.3項「パターンマッチ」をお読みください) を設定することができます。たとえば下記のような既定のプロファイルがあるとします:

```
/** { ... }
```

上記のようなプロファイルに対して、下記のように名前を設定することができます:

```
profile default /** { ... }
```

パターンマッチングに対しても名前を設定することができます。たとえば:

```
/usr/lib64/firefox*/firefox-*bin { ... }
```

上記のようなプロファイルに対して、下記のように名前を設定することができます:

```
profile firefox /usr/lib64/firefox*/firefox-*bin { ... }
```

31.6.6 別名ルール

別名ルールは、プロファイルパスのマッピングをサイト固有のレイアウトに変換するためのもう 1 つの方法です。これらは変数を利用してパスを書き換える方式の代替となるものであるほか、変数解決よりも後に動作する仕組みです。別名ルールでは、ルールがターゲットのプレフィクス内に存在するかのように、同じソースプレフィクスを持つルールを扱うよう指定することができます。

```
alias /home/ -> /usr/home/
```

上記のように指定すると、`/home/` に対するマッチングが、`/usr/home/` に対しても作用するようになります。たとえば:

```
/home/username/** r,
```

上記のような設定が存在すると、下記の意味として使用されるようになります:

```
/usr/home/username/** r,
```

別名ルールは、プロファイル自身を書き換えることなく、ルールのマッピングを素早く変更する機能も提供します。もちろん元のパスも従来通り許可されます。上述の例では、別名ルールを設定した後でも、`/home/` は許可され続けます。

また `alias` ルールを複数指定することで、同時に複数のターゲットを指定することもできます。

```
alias /home/ -> /usr/home/
alias /home/ -> /mnt/home/
```



注記

現在の AppArmor ツールでは、別名を利用するには、プロファイルを手作業で編集および管理しなければなりません。



ヒント

グローバルな別名を設定したい場合は、`/etc/apparmor.d/tunables/alias` ファイル内に設定してください。

31.7 ファイルアクセス許可のアクセスモード

ファイルのアクセス許可モードを指定する場合、下記の中から組み合わせて指定してください:

`r`	読み込みモード
`w`	書き込みモード (`a` とは互いに排他関係にあります)
`a`	追記モード (`w` とは互いに排他関係にあります)
`k`	ファイル施錠 (ロック) モード
`l`	リンクモード
`link ファイル -> ターゲット`	リンク対ルール (他のアクセスモードとは共存不可)

31.7.1 読み込みモード (r)

指定したリソースに対して、プログラムからの読み込みアクセスを許可します。読み込みアクセスはシェルスクリプトなどのインタプリタ型のコンテンツで必要となるほか、コアダンプを実施できるかどうかの判断としても使われます。

31.7.2 書き込みモード (w)

指定したリソースに対して、プログラムからの書き込みアクセスを許可します。ファイルを削除する (アンリンクする) ような場合にも、この許可が必要となります。

31.7.3 追記モード (a)

指定したリソースに対して、プログラムからファイルの末尾への書き込みを許可します。`w` モードとは異なり、追記モードではファイル内容の上書きが許可されないほか、ファイル名の変更や削除なども行うことができなくなります。追記の許可は通常、アプリケーション側からログファイルへの書き込みを行うような場合に指定すべきもので、これにより既に記録されているログコンテンツの上書きを禁止することができます。追記の許可は書き込みモードのサブセットとして規定されているため、`w` と `a` のフラグの両方を指定することはできませんし、相互に排他関係にあるフラグとなります。

31.7.4 ファイル施錠 (ロック) モード (k)

指定したリソースに対して、プログラムからファイル施錠 (ロック) を許可します。従来のバージョンの AppArmor では、アプリケーション側からアクセスができれば、施錠も許可されていました。新しいバージョンの AppArmor では、個別のモードとして規定されるようになっていますので、どうしても施錠が必要なファイルにのみ許可を与えることができますので、サービス拒否 (Denial of Service; DoS) 攻撃のような攻撃シナリオを防ぐことができるようになっています。

31.7.5 リンクモード (l)

リンクモードはハードリンクに対するアクセスを許可します。リンクを作成すると、ターゲットとなるファイルと元のファイルは同じアクセス許可になります (ただしリンク先のファイルにはリンクモードの指定は不要です) 。

31.7.6 リンク対

リンクモードでは任意のファイルに対してリンクを作成する許可を与えますが、これはリンクがターゲットによって付与された許可のサブセットを持っている場合に限られます (サブセット許可テストと呼びます) 。

```
/srv/www/htdocs/index.html rl,
```

リンク元とリンク先を指定することで、リンク対ルールはハードリンクの作成方法に関して、より広い制御を提供します。リンク対ルールは、既定では標準ルールのリンク許可が必要とする、リンクのサブセット許可テストを強制しません。

```
link /srv/www/htdocs/index.html -> /var/www/index.html
```

テストを必要とするようルールを強制するには、`subset` キーワードを指定します。下記の 2 つのルールは同じ意味になります:

```
/var/www/index.html l,
link subset /var/www/index.html -> /**,
```



注記

現時点では、リンク対ルールは YaST やコマンドラインツールでサポートしていません。この種類のルールを使用したい場合は、プロファイルを手作業で編集してください。なお、ツール側ではリンク対ルールを操作することはありませんので、それらを含むプロファイルをツールで操作しても問題はありません。

31.7.7 任意指定の allow および file ルール

プロファイル内では通常、許可するファイルを列挙する仕組みであるため、`allow` キーワードを指定する必要はありません。逆に特定のファイルに対して明示的に禁止を設定したい場合にのみ、`deny` キーワード (詳しくは 31.7.9項「拒否ルール」をお読みください) を指定してください。

```
allow file /example r,
allow /example r,
allow network,
```

同様に、`file` キーワードも指定不要です。これは、`network` や `mount` などのルール種類のキーワードが現れない限り、自動的に `file` が指定されたものとして扱われるためです。

```
file /example/rule r,
```

上記は、下記と同じ意味になります。

```
/example/rule r,
```

下記のルールは全てのファイルに対するアクセスを許可します:

```
file,
```

上記は、下記と同じ意味になります。

```
/** rwmlk,
```

ファイルルールの場合、アクセス許可を先に記述することもできますが、基本的にはこれを使用すべきではありません。もしもこの方式を使用する場合は、ルールの冒頭で記述してください。これは特に、他の種類のルールと混在させるような場合に重要です。

```
/path rw,             # 古い形式
rw /path,             # 許可を先に記述した例
file rw /path,        # 明示的に 'file' キーワードを指定した例
allow file rw /path, # 明示的に 'allow' キーワードを指定した例
```

31.7.8 所有者条件ルール

ファイルルールでは、ファイルの所有者であった場合の特例を設定することができます (fsuid がファイルの uid と一致している必要があります) 。所有者である場合のルールを設定するには、ルールの前に `owner` キーワードを指定します。所有者条件ルールでも、通常のファイルルールと同様に設定を行うことができます。

```
owner /home/*/** rw
```

所有者条件ルールとリンクルールの両方を使用した場合、テストはターゲットのファイルに対して行われ、リンクができるようになるためには、ユーザがファイルを所有していなければならなくなります。



注記: 通常のファイルルールの優先順位について

所有者条件ルールは通常のファイルルールに対するサブセットとして扱われます。通常のファイルルールと所有者条件ルールが重複した場合、それらは一括で扱われます。たとえば下記のようなルールが存在したものとします:

```
/foo r,
owner /foo rw,  # もしくは単に "w,"
```

この場合ルールが合成され、全てのユーザに対して `r` が許可され、所有者にのみ `w` が許可されます。



ヒント

ファイルの所有者 以外の ユーザを明示的に指定したい場合は、`other` キーワードを指定してください。

```
owner /foo rw,
other /foo r,
```

31.7.9 拒否ルール

拒否ルールは既知のオブジェクトに対して明示的な禁止を設定するための仕組みです。プロファイル生成ツールでは、拒否ルールによる既知の拒否については問い合わせを行いません。また、監査ログにもこの拒否は記録されなくなりますので、ログファイル内に不要な出力を行わないようにすることができます。なお、拒否ルールで拒否された内容も監査ログに出力したい場合は、拒否ルールの指定の前に `audit` を指定してください。

拒否ルールと許可ルールを混在させることもできます。この場合、幅広く許可ルールを設定しておいて、特定のファイルにのみ明示的な拒否を設定することができるようになります。拒否ルールは所有者条件ルールと組み合わせることもできます。この場合、ユーザが所有するファイルへのアクセスを拒否することができます。下記の例では、ユーザのディレクトリ内にある全てに対して許可を与えるものの、 `.ssh/` 以下のファイルについては書き込みを禁止しています:

```
deny /home/*/.ssh/** w,
owner /home/*/** rw,
```

ただし、拒否ルールを数多く設定してしまうようなことは避けてください。これは、プロファイルの可読性を損なうことになってしまうためです。しかしながら、拒否ルールを賢く使用することで、プロファイルを単純化することができます。そのため、ツールでは特定のファイルのみを生成し、グロブを伴う拒否ルールは作成しないようになっています。グロブを利用した拒否ルールを作成したい場合は、手作業でプロファイルを編集してください。ツール側では拒否ルールを操作することはありませんので、それらを含むプロファイルをツールで操作しても問題はありません。

31.8 マウントルール

AppArmor では、マウントやマウント解除の操作を制限することもできます。この制限では、ファイルシステムの種類やフラグなどを設定することもできます。ルールの書式は `mount` コマンドの書式をベースにしていて、 `mount` (マウント時), `remount` (再マウント時), `umount` (マウント解除時) のキーワードのいずれかを指定します。その他の条件指定は任意で、指定しない場合は全てに対してマッチするものとして扱われます。たとえばファイルシステムを指定しなかった場合、全てのファイルシステムに対してルールが適用されることになります。

条件指定は `options=` もしくは `options in` で条件を指定します。

`options=` では、厳密に一致すべきオプションを指定します。

```
mount options=ro /dev/foo -E /mnt/,
```

上記は下記にマッチします:

```
# mount -o ro /dev/foo /mnt
```

ただし、下記にはマッチしません:

```
# mount -o ro,atime /dev/foo /mnt
# mount -o rw /dev/foo /mnt
```

`options in` では、使用されているマウントオプションが、ルール内で指定された値のうちのいずれかにマッチすることを求めます。

```
mount options in (ro,atime) /dev/foo -> /mnt/,
```

上記は下記にマッチします:

```
# mount -o ro /dev/foo /mnt
# mount -o ro,atime /dev/foo /mnt
# mount -o atime /dev/foo /mnt
```

ただし、下記にはマッチしません:

```
# mount -o ro,sync /dev/foo /mnt
# mount -o ro,atime,sync /dev/foo /mnt
# mount -o rw /dev/foo /mnt
# mount -o rw,noatime /dev/foo /mnt
# mount /dev/foo /mnt
```

複数の条件を設定したい場合は、`options` を複数並べて記述してください。

```
mount options=ro options=atime
```

上記は下記にマッチします:

```
# mount -o ro /dev/foo /mnt
# mount -o atime /dev/foo /mnt
```

ただし、下記にはマッチしません:

```
# mount -o ro,atime /dev/foo /mnt
```

複数の指定を行った場合、それらは個別に扱われます。

```
mount options=ro,
mount options=atime,
```

上記は下記とは異なります:

```
mount options=(ro,atime),
mount options in (ro,atime),
```

下記のルールでは、`/dev/foo` デバイスを `/mnt/` にマウントすることを許可し、その際に読み込み専用か、inode のアクセス日時のオプション指定を許すと共に、'nodev' と 'user' のオプションの指定も許可しています:

```
mount options=(ro, atime) options in (nodev, user) /dev/foo -> /mnt/,
```

上記を指定すると、下記が許可されるようになります:

```
# mount -o ro,atime /dev/foo /mnt
# mount -o nodev /dev/foo /mnt
# mount -o user /dev/foo /mnt
# mount -o nodev,user /dev/foo /mnt
```

31.9 ピボットルートルール

AppArmor ではルートファイルシステムの変更を制限することもできます。書式は下記のとおりです:

```
pivot_root [oldroot=古いルート] 新しいルート
```

'pivot_root' 内で指定するパスはディレクトリであるため、'/' で終わらなければなりません。

```
# 任意のルートディレクトリを許可する
pivot_root,

# 新しいルートディレクトリは任意で、古いルートディレクトリのみ
# /mnt/root/old/ に制限する
pivot_root oldroot=/mnt/root/old/,

# 新しいルートディレクトリのみ /mnt/root/ に制限する
pivot_root /mnt/root/,

# 古いルートディレクトリを /mnt/root/old/ に、
# 新しいルートディレクトリを /mnt/root/ に制限する
pivot_root oldroot=/mnt/root/old/ /mnt/root/,

# 古いルートディレクトリを /mnt/root/old/ に、
# 新しいルートディレクトリを /mnt/root/ に制限し、
# /mnt/root/sbin/init のプロファイルに遷移する
pivot_root oldroot=/mnt/root/old/ /mnt/root/ -> /mnt/root/sbin/init,
```

31.10 PTrace ルール

AppArmor では ptrace のシステムコールを制限することもできます。ptrace ルールは蓄積型のルールで、ptrace に対する許可は全ての ptrace ルールの和集合となります。ルール内でアクセスリストを指定していない場合は、暗黙に許可が発行されるようになります。

`trace` および `tracedby` の許可では ptrace(2) の制御を、`read` および `readby` の許可では proc(5) ファイルシステムへのアクセスや kcmp(2), futexes (get_robust_list(2)), パフォーマンストレースイベントの制御を、それぞれ行うことができます。

ptrace が許可されるようにするには、ptrace する側とされる側の両方のプロセスに対して、必要な許可を設定する必要があります。つまり、ptrace する側のプロセスには `trace` の許可が、ptrace される側のプロセスには `tracedby` の許可が必要となります。

AppArmor における PTrace ルールの例:

```
# 全ての ptrace アクセスを許可する
ptrace,

# 明示的に全ての ptrace アクセスを許可する
ptrace (read, readby, trace, tracedby),

# ptrace(2) の使用を明示的に禁止する
deny ptrace (trace),

# 無制限のプロセス（例: デバッガ）から自身に対する ptrace を許可する
ptrace (readby, tracedby) peer=unconfined,

# /usr/bin/foo 以下のプロファイルを利用している実行中プロセスに対して、 ptrace を許可する
ptrace (trace) peer=/usr/bin/foo,
```

31.11 シグナルルール

AppArmor ではプロセス間シグナルを制御することもできます。AppArmor の signal ルールは蓄積型のルールで、シグナルに対する許可は全ての signal ルールの和集合となります。ルール内でアクセスリストを指定していない場合は、暗黙に許可が発行されるようになります。

なお、シグナルの送信側と受信側のプロセスは、両方とも必要な許可を得る必要があります。

signal ルールの例:

```
# 全てのシグナルへのアクセスを許可する
signal,

# HUP と INT のシグナルのみ明示的に禁止する
deny signal (send) set=(hup, int),

# 無制限プロセスから自分自身へのシグナルを許可する
signal (receive) peer=unconfined,

# /usr/bin/foo 以下のプロファイルで動作するプロセスに対して、
# シグナルの送信を許可する
signal (send) peer=/usr/bin/foo,

# PID の存在チェックを許可する
signal (receive, send) set=("exists"),

# 組み込みの @{profile_name} 変数を使用して、自分自身へのシグナル送信を許可する
signal peer=@{profile_name},

# 2 種類のリアルタイムシグナルを許可する
signal set=(rtmin+0 rtmin+32),
```

31.12 実行モード

実行モードとは名前付きプロファイル遷移とも呼ばれ、下記のモードが用意されています:

`Px`	専用プロファイル実行モード
`Cx`	専用ローカルプロファイル実行モード
`Ux`	無制限実行モード
`ix`	継承実行モード
`m`	`mmap(2)` コールによる `PROT_EXEC` の許可

31.12.1 専用プロファイル実行モード (px)

このモードでは、特定のリソースを実行する際の AppArmor のドメイン遷移で、専用のセキュリティプロファイルを求めるようになります。プロファイルが何も定義されていない場合、リソースへのアクセスは拒否されます。

`Ux`，`ux`，`px`，`ix` とは非互換です。

31.12.2 専用ローカルプロファイル実行モード (cx)

`Px` と同様に専用のプロファイルを求めますが、`Cx` ではグローバルなプロファイルセットを検索するのではなく、現在のプロファイル内にあるローカルのプロファイルを検索します。このプロファイル遷移は、ヘルパーアプリケーションなどに対して個別のプロファイル設定を必要とするような場合に利用します。



注記: 専用ローカルプロファイル実行モード (cx) の制限について

現時点では Cx による遷移はトップレベルのプロファイルに限定され、ハットや子プロファイル内では使用できません。この制限は将来的に撤廃される予定です。

`Ux`，`ux`，`Px`，`px`，`cx`，`ix` とは非互換です。

31.12.3 無制限実行モード (ux)

プログラムに対して、実行時にどの AppArmor プロファイルも適用せずに実行することを指示します。このモードは、プログラム側からマシンの再起動など、特権処理を実行する必要がある場合に有用です。他の実行ファイル内に特権操作の機能を配置し、そこに対して実行権限を与えることで、制限を受けている全てのプロセスに課された制限を迂回できるようになります。なお、ルートプロセスに対して無制限処理を適用すると、AppArmor のポリシー自身も変更できることになります。制限の内容について、詳しくは `apparmor(7)` のマニュアルページをお読みください。

`ux`，`px`，`Px`，`ix` とは非互換です。

31.12.4 安全ではない実行モード

実行モードのうち、小文字版 (具体的には `px`，`cx`，`ux` がそれに該当します) は限られた条件下でのみお使いください。これらの実行モードでは `LD_PRELOAD` などの環境変数の洗浄処理を行いませんので、呼び出す側のドメインは呼び出された側のリソースに対して、影響を与えすぎてしまうことがあります。これらのモードは、`LD_PRELOAD` などの変数を使用しなければならないため、子プロファイル側でどうしても制限を取り外さなければならない場合に *のみ* 使用すべきです。これらのモードを指定したプロファイルでは、セキュリティが弱くなることに注意してください。また、ご自身の責任でお使いください。

31.12.5 継承実行モード (ix)

`ix` は指定されたプログラムを `execve(2)` で実行する際、AppArmor ドメイン遷移を防止する仕組みです。その代わり、現在のプロファイルを継承して実行するようになります。

このモードは、制限を受けているプログラムが他の制限付きのプログラムを呼び出していて、そのプログラム側で追加の許可を設定して欲しくない場合や、現在持っている許可を失いたくない場合に指定します。なお、`ix` の実行モードでは権限を変更しないため、環境変数の洗浄を行う版はありません。

`cx`，`ux`，`px` とは非互換です。また、`m` の許可を内包します。

31.12.6 実行可能なマッピングの許可 (m)

このモードはファイルをメモリにマッピングする際、`mmap(2)` の呼び出しで `PROT_EXEC` フラグを付けることを許可します。このフラグにより、メモリ内容を実行できるようにします。これは特定のアーキテクチャ向けに用意されている、実行不可データページの機能を利用するものです。これにより、ファイルを経由した不正なコード実行を防ぐことができます。AppArmor ではこのモードを利用し、行儀のよいプログラム (もしくは実行不可メモリへのアクセス制御を強制するアーキテクチャ上の全てのプログラム) に対して、ライブラリとして使用できるファイルを制限し、`ld(1)`，`LD_PRELOAD`，`LD_LIBRARY_PATH` に対して設定される、無効な `-L` フラグの効果を制限することができます。詳しくは `ld.so(8)` のマニュアルページをお読みください。

31.12.7 名前付きプロファイル遷移

既定では、`px` と `cx` (およびそれらのクリーン実行版) の遷移では、実行されるプログラム名に一致するプロファイルに遷移します。名前付きプロファイル遷移では、遷移先のプロファイル名を指定することができます。これは単一のプロファイルを複数のバイナリで共有する必要があるような場合や、名前以外の方法で異なるプロファイルを使用する必要があるような場合に有用です。名前付きプロファイル遷移は `cx`，`Cx`，`px`，`Px` と併用することができます。現時点では、プロファイルごとに 20 種類までの名前付きプロファイル遷移を設定することができます。

名前付きプロファイル遷移では、`->` 記号を利用して、遷移先のプロファイル名を指定します:

```
/usr/bin/foo
{
  /bin/** px -> shared_profile,
  ...
  /usr/*bash cx -> local_profile,
  ...
  profile local_profile
  {
    ...
```

```
  }
}
```



注記: 通常の遷移と名前付き遷移の違いについて

グロブ機能を利用することで、通常型のプロファイルでは「一対多」の関係を設定することができます。たとえば `/bin/** px` のように指定すると、`/bin/ping`，`/bin/cat` などのプログラムに対応したプロファイルに遷移することになります。

一方の名前付きプロファイル遷移では、「多対一」の関係を設定することができます。ルールに該当するものであれば、その名前にかかわらず常に同じプロファイルに遷移することになります。

名前付きプロファイル遷移は、ログ内では `Nx` モードとして記録されます。また、遷移先のプロファイル名は `name2` として示されます。

31.12.8 プロファイル遷移に対するフォールバックモード

`px` と `cx` はいずれも、強い依存関係を指定していることになります。つまり、指定したプロファイルが存在しない場合、実行は失敗することになります。継承フォールバックの場合は実行が成功しますが、現在のプロファイルを継承して動作することになります。継承フォールバックを指定するには、`cx`，`Cx`，`px`，`Px` に `ix` を加えて、`cix`，`Cix`，`pix`，`Pix` のように指定してください。

```
/path Cix -> プロファイル名,
```

もしくは、下記のように指定してもかまいません:

```
Cix /path -> プロファイル名,
```

なお、`-> プロファイル名` の指定は任意です。

同じことが無制限実行モードである `ux` にも適用できます。これらを組み合わせると、`cux`，`CUx`，`pux`，`PUx` になります。これらのモードは、指定したプロファイルが見つからない場合、「無制限に」実行するモードにフォールバックすることになります。

```
/path PUx -> プロファイル名,
```

もしくは、下記のように指定してもかまいません:

```
PUx /path -> profile_name,
```

なお、`-> プロファイル名` の指定は任意です。

フォールバックモードは、名前付きプロファイル遷移でも使用することができます。

31.12.9 実行モード内での変数設定

Px, Cx, Ux の実行モードのうちのいずれかを選択した場合、子プロセスが継承する環境変数のうち、下記の変数の内容が削除されることに注意してください。そのため、これらの変数に依存して動作するアプリケーションやプロセスが存在した場合、Px, Cx, Ux のフラグを指定すると、正しく動作しなくなってしまいます:

- GCONV_PATH
- GETCONF_DIR
- HOSTALIASES
- LD_AUDIT
- LD_DEBUG
- LD_DEBUG_OUTPUT
- LD_DYNAMIC_WEAK
- LD_LIBRARY_PATH
- LD_ORIGIN_PATH
- LD_PRELOAD
- LD_PROFILE
- LD_SHOW_AUXV
- LD_USE_LOAD_BIAS
- LOCALDOMAIN
- LOCPATH
- MALLOC_TRACE
- NLSPATH
- RESOLV_HOST_CONF
- RES_OPTIONS
- TMPDIR
- TZDIR

31.12.10 safe キーワードおよび unsafe キーワード

実行モードの修飾子を使用する代わりに、`safe` や `unsafe` のキーワードを使用することもできます。たとえば、

```
/example_rule Px,
```

上記は下記のいずれかと同じ意味になります:

```
safe /example_rule px,
safe /example_rule Px,
safe px /example_rule,
safe Px /example_rule,
```

また、

```
/example_rule px,
```

上記は、下記のいずれかと同じ意味になります:

```
unsafe /example_rule px,
unsafe /example_rule Px,
unsafe px /example_rule,
unsafe Px /example_rule,
```

`safe` / `unsafe` のキーワードは相互に排他なキーワードであり、ファイルルールの `owner` の後ろに指定することもできます。書式として記述すると、下記のようになります:

```
[audit] [deny] [owner] [safe|unsafe] ファイルルール
```

31.13 リソース制限制御

AppArmor では、アプリケーションのリソース制限 (rlimits/ulimits としても知られています) を設定したり制御したりすることができます。既定では AppArmor はアプリケーションの rlimits を制限せず、プロファイル内に書かれた制限にのみ従うようになっています。リソース制限に関する詳細については、`setrlimit(2)`, `ulimit(1)`, `ulimit(3)` の各マニュアルページをお読みください。

AppArmor ではシステム側に用意された rlimits を活用しているだけであり、通常発生するような追加の監査を提供するものではありません。また、システム側で設定された rlimits の値を増やすこともできず、AppArmor ではアプリケーションに対して設定されたリソース制限を減らす設定のみを適用することができます。

設定された値は子プロセスに対しても引き継がれ、新しいプロファイルに遷移した場合であっても、アプリケーションの制限がなくなった場合でも、制限が残ります。そのため、アプリケーションが新しいプロファイルに遷移した場合、遷移先のプロファイルではアプリケーションの rlimits をさらに減らすことしかできなくなります。

AppArmor の rlimit ルールでは、アプリケーションのハードリミットの仲介処理も行っています。アプリケーション側では、プロファイル内で指定された値よりもハードリミットを増やすことはできませんが、新しいプロファイル内で制限を増やすように設定しておいて、そのプロファイルに遷移するように設定することは自由に行うことができます。

AppArmor での rlimit の制御においてソフトリミットは、ハードリミットと同じか、より小さい値に設定されているようにするだけで、それ以外の制御は行いません。

AppArmor のハードリミットルールは下記のような書式で指定します:

```
set rlimit リソース <= 値,
```

`リソース` と `値` には、それぞれ下記のものを指定することができます:

`cpu`
: CPU 時間の制限 (秒単位) 。

`fsize` , `data` , `stack` , `core` , `rss` , `as` , `memlock` , `msgqueue`
: バイト単位での数値、もしくは K/KB (キロバイト), M/MB (メガバイト), G/GB (ギガバイト) の接尾辞付きの数値を指定することができます。例:

  ```
  rlimit data <= 100M,
  ```

`fsize` , `nofile` , `locks` , `sigpending` , `nproc` , `rtprio`
: 0 以上の数値を指定することができます。

`nice`
: -20 から 19 までの数値を指定することができます。



注記

nproc のリソース制限は、その他のリソース制限とは異なる動作になっています。標準的なプロセス制限ではなく、任意の時点におけるプロファイル内の最大プロセス数を意味します。新しいプロセスを作成する際、プロファイルで指定された制限に引っかかると、現在実行中のプロセス数が減らない限り、処理が失敗することになります。



注記

現時点では、ツールを利用してプロファイル内に rlimit のルールを追加することはできません。rlimit の制御を追加する唯一の方法は、手作業でテキストファイルを利用してプロファイルを編集する方法です。なお、rlimit を含むプロファイルをツールで編集しても、rlimit の項目を削除することはありませんので安全です。

31.14 監査ルール

AppArmor では、特定のルールに監査を設定して、監査ログ内にメッセージを出力するようにすることができます。指定したルールに対して監査メッセージを出力するように設定するには、ルールの前に `audit` キーワードを指定します:

```
audit /etc/foo/*        rw,
```

特定の処理時にのみ監査メッセージを出力させたい場合は、ルールを 2 つに分割することをお勧めします。下記の例では、ファイルを書き込み用に開いた場合は監査メッセージを出力するものの、読み込み用に開いた場合は出力しません:

```
audit /etc/foo/*  w,
/etc/foo/*        r,
```



注記

監査メッセージはそれぞれの読み込みや書き込み処理で生成されるのではなく、ファイルが読み込み用や書き込み用に開いた際に出力されます。

監査ルールには `owner` / `other` の条件付きファイルルールを追加することもできます。これにより、ユーザが所有するファイルである場合や、ユーザが所有していないファイルである場合にのみ、監査メッセージを出力することができるようになります:

```
audit owner /home/*/.ssh/**       rw,
audit other /home/*/.ssh/**       r,
```

32 AppArmor のプロファイルリポジトリ

改訂履歴

2023-06-06

AppArmor には既定で有効化されているプロファイルセットが添付されています。これらは AppArmor の開発者が作成したもので、`/etc/apparmor.d` 内に配置されます。これらのプロファイルに加えて、openSUSE Leap ではアプリケーションごとの個別のプロファイルも用意されています。これらのプロファイルは既定では有効化されませんし、標準の AppArmor プロファイルとは異なる `/usr/share/apparmor/extra-profiles` というディレクトリ内に配置されます。

AppArmor のツール (YaST, `aa-genprof`, `aa-logprof`) では、ローカルリポジトリの使用にも対応しています。何もない状態から新しいプロファイルを作成しようとしていて、かつローカルのリポジトリ (`/usr/share/apparmor/extra-profiles`) 内に無効化されたプロファイルが存在した場合、そのディレクトリ内にあるプロファイルを使用するかどうかと、それをベースにして設定するかどうかを尋ねるようになっています。これらのプロファイルを使用した場合、まず既定で有効化されるプロファイルのディレクトリ (`/etc/apparmor.d`) にコピーが行われ、AppArmor の起動時に読み込まれるように設定されます。その後の調整は `/etc/apparmor.d` ディレクトリ内のファイルに対して行われるようになります。

33 YaST を利用したプロファイルの構築と管理

改訂履歴

2024-01-10

YaST には AppArmor® のプロファイルを構築したり管理したりするための基本的な仕組みが 2 種類用意されています。一方はグラフィカルなもの、他方はテキストベースのものです。テキストベースの場合、メモリなどのリソースもネットワーク帯域も比較的少なくなるようになっていますので、リモートからの管理を行うような場合や、ローカルのグラフィック環境が貧弱な場合に最適です。グラフィカルなものとテキストベースのものは、外観が異なるものの、同じ機能を同じ手順で利用できるようになっています。また、AppArmor のコマンドも用意されていて、このコマンドは端末ウインドウから AppArmor の制御を行ったりすることができます。なお、コマンドラインツールについては、第34章「コマンドラインからのプロファイル構築」で詳しく説明しています。

メインメニューから YaST を起動して、必要であれば `root` のパスワードを入力します。また、端末ウインドウを開いてその中で `root` になれば、`yast2` と入力して実行することでグラフィカルモードで、`yast` と入力して実行することでテキストモードで YaST を起動することができます。

［セキュリティとユーザ］のセクション内に［AppArmor の設定］というアイコンがありますので、これを押してください。すると YaST の AppArmor モジュールを起動することができます。

33.1 プロファイルの手動追加

AppArmor では、プロファイル内に項目を手作業で追加していくことによって、AppArmor のプロファイルを作成する機能を提供します。まずはプロファイルを作成するアプリケーションを選択してから、項目を追加する流れを行います。

1. YaST を起動して［AppArmor の設定］を選び、表示されたメインウインドウで［手作業でプロファイルを追加］を選択します。
2. プロファイルを作成するアプリケーションを、システム内で選択します。
3. アプリケーションが見つかったら、それを選択して［開く］を押します。すると、［AppArmor プロファイルダイアログ］内に、最低限の設定のみが書かれたプロファイルが現れます。
4. ［AppArmor プロファイルダイアログ］では、それぞれ対応するボタンを押すことによって、項目の追加や編集、削除を行うことができます。詳しくは 33.2.1項「項目の追加」, 33.2.2項「項目の編集」, 33.2.3項「項目の削除」をお読みください。
5. 完了したら［完了］を押します。

33.2 プロファイルの編集

ヒント

YaST には、AppArmor プロファイルの作成や編集など、基本的な仕組みが用意されています。ですが、`vi` などのテキストエディタでプロファイルを編集するのが最も直感的で分かりやすい仕組みでもあります:

```
> sudo vi /etc/apparmor.d/usr.sbin.httpd2-prefork
```

ヒント

`vi` エディタには、文法ハイライト機能や文法エラーの表示機能などが用意されています。これにより、AppArmor のプロファイルを編集しながら、その文法を同時にチェックすることができるようになっています。

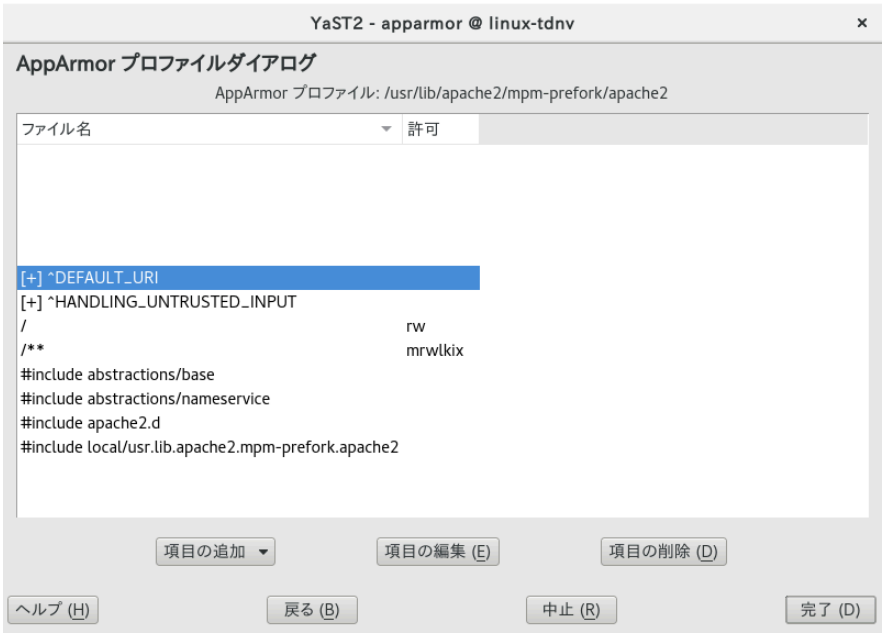
AppArmor では、項目を追加／編集／削除することによって、プロファイルの編集を行うことができます。プロファイルを編集するには、下記の手順を実施します:

1. YaST を起動して［AppArmor の設定］を選び、表示されたメインウインドウで［既存のプロファイルを管理］を選択します。



2. プロファイルの一覧から、編集したいプロファイルを選択します。

3. ［編集］を押します。すると、［AppArmor プロファイルダイアログ］ウインドウが現れ、プロファイルの内容が表示されます。



4. ［AppArmor プロファイルダイアログ］では、それぞれ対応するボタンを押すことによって、項目の追加や編集、削除を行うことができます。詳しくは 33.2.1項「項目の追加」, 33.2.2項「項目の編集」, 33.2.3項「項目の削除」をお読みください。

5. 完了したら［完了］を押します。

6. ポップアップが表示されたら［はい］を押し、AppArmor のプロファイルを保存してプロファイルセットを再読み込みします。



ヒント: AppArmor 内の文法チェックについて

AppArmor には文法チェック機能が用意されているため、YaST の AppArmor ツールで処理を行う際、プロファイル内に文法エラーがあればその旨の表示が現れます。エラーが発生した場合は、`root` になって手作業でプロファイルを編集し、`systemctl reload apparmor` コマンドでプロファイルセットを再読み込みしてください。

33.2.1 項目の追加

［AppArmor プロファイルウインドウ］内にある［項目の追加］ボタンを押すことで、AppArmor プロファイル内に様々な種類の項目を追加することができます。

追加できる項目の種類は下記のとおりです:

ファイル

ポップアップウインドウが表示されますので、そこでファイルの絶対パスと許可するアクセスの種類を指定します。完了したら［OK］を押します。
必要であればグロブ表記を使用することもできます。グロブに関する詳細は 31.6項「プロファイル名／フラグ／パス／グロブ」をお読みください。また、アクセス許可に関する詳細については、31.7項「ファイルアクセス許可のアクセスモード」をお読みください。



ディレクトリ

ポップアップウインドウが表示されますので、そこでディレクトリの絶対パスと許可するアクセスの種類を指定します。必要であればグロブを使用することもできます。完了したら［OK］を押します。
グロブに関する詳細については 31.6項「プロファイル名／フラグ／パス／グロブ」をお読みください。また、アクセス許可に関する詳細については、31.7項「ファイルアクセス許可のアクセスモード」をお読みください。



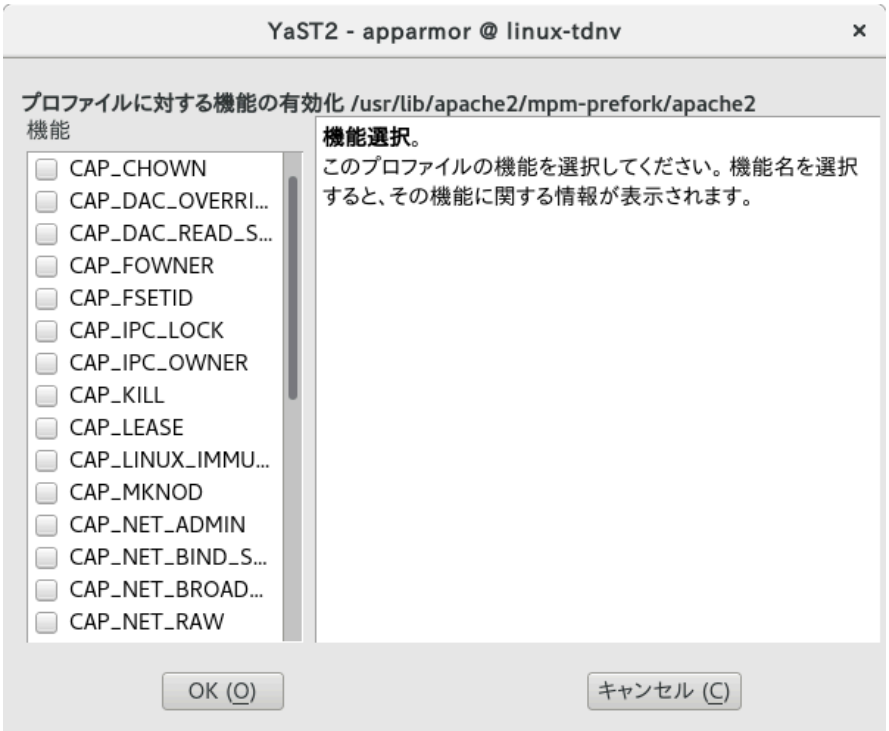
ネットワークルール

ポップアップウインドウが表示されますので、そこでネットワークファミリとソケットの種類を指定します。詳しくは 31.5項「ネットワークアクセス制御」をお読みください。



機能 (ケーパビリティ)

ポップアップウインドウが表示されますので、そこで有効化するケーパビリティを選択します。ケーパビリティは POSIX.1e で規定されたケーパビリティを意味しています。ケーパビリティに関する詳細については、31.4項「ケーパビリティ項目 (POSIX.1e)」をお読みください。完了したら［OK］を押します。



インクルード

ポップアップウインドウが表示されますので、そこで取り込むべきファイルを指定します。インクルードとは、プロファイルを単純化するために、他の AppArmor プロファイル内のコンポーネントを取り込むことを意味します。詳しくは 31.3項「include ステートメント」をお読みください。

ハット

ポップアップウインドウが表示されますので、そこで現在のプロファイルに追加すべきサブプロファイル (ハット) の名前を指定して、［ハットの作成］を押します。詳しくは 第35章「チェンジハット機能による Web アプリケーションのプロファイル作成」をお読みください。



33.2.2 項目の編集

［項目の編集］を押すと、ポップアップウインドウが表示されます。ここから項目の編集を行います。

ポップアップウインドウ内では、必要に応じて編集を行います。必要であれば、グロブを使用することもできます。完了したら［OK］を押します。

グロブに関する詳細については、31.6項「プロファイル名／フラグ／パス／グロブ」をお読みください。また、アクセス許可に関する詳細については、31.7項「ファイルアクセス許可のアクセスモード」をお読みください。

33.2.3 項目の削除

プロファイル内の項目を削除するには、項目を選択してから［項目の削除］を押します。これで AppArmor は選択した項目を削除することができます。

33.3 プロファイルの削除

AppArmor では、プロファイルを手作業で削除する機能も用意されています。単純に一覧からプロファイルを選んで削除するだけですが、下記のようにして行います:

1. YaST を起動して［AppArmor の設定］を選び、表示されたメインウインドウで［既存のプロファイルを管理］を選択します。
2. 削除するプロファイルを選びます。
3. ［削除］を押します。
4. ポップアップが表示されたら［はい］を押し、AppArmor のプロファイルを削除してプロファイルセットを再読み込みします。

33.4 AppArmor の管理

AppArmor の状態を有効もしくは無効にすることができます。AppArmor を有効化すれば、お使いのシステムを潜在的な脅威から保護することができます。AppArmor を無効化すると、プロファイルの設定にかかわらず、システムを保護しなくなります。AppArmor の状態を変更するには、［AppArmor の設定］を選んで、表示されたメインメニューから［設定］を選びます。



AppArmor 全体の状態を変更する場合は、33.4.1項「AppArmor の状態変更」の手順に従って進めます。プロファイルごとにモードを切り替えたい場合は、33.4.2項「個別のプロファイルに対するモード変更」の手順に従って進めます。

33.4.1 AppArmor の状態変更

AppArmor の状態は、有効と無効のいずれかを設定することができます。AppArmor を有効化すると、AppArmor のセキュリティプロファイルをインストールして動作させ、強制することができるようになります。

1. YaST を起動して［AppArmor の設定］を選び、表示されたメインウインドウで［設定］を選択します。
2. AppArmor を有効化するには［AppArmor を有効にする］にチェックを入れます。無効化する場合は、チェックを外します。
3. あとは［AppArmor の設定］ウインドウで［完了］を押すだけです。



ヒント

プロファイルを適用するには、動作中のプログラムを再起動する必要があります。

33.4.2 個別のプロファイルに対するモード変更

AppArmor では、プロファイルごとに 2 種類のモードのうちのいずれかを設定することができます。不平 (complain) モードでは、プロファイル側で許可されていないファイルへのアクセスなど、AppArmor プロファイルへの違反は検出されるものの、禁止されることはなく記録だけが行われます。このモードはプロファイルの作成時や編集時に有用な仕組みで、AppArmor のツールでもプロファイルの生成時にこれを使用しています。もう一方の 強制 (enforce) モードでは、プロファイル内に書かれたポリシーを強制し、ポリシーへの違反を `rsyslogd` (もしくは `auditd` や `journalctl` など、システムの設定によって異なります) に記録します。

［プロファイルモードの設定］ダイアログでは、現在読み込まれているプロファイルのモードを表示したり、変更したりすることができます。この機能はプロファイルの作成時や編集時に、状態を確認したい場合に便利です。一括プロファイル作成 (詳しくは 34.7.2項「一括プロファイル作成」をお読みください) の場合は、このツールを使用することで、学習対象としているプロファイルの範囲を調整したり監視したりすることができます。

アプリケーションのプロファイルモードを編集するには、下記の手順を実施します:

1. YaST を起動して［AppArmor の設定］を選び、表示されたメインウインドウで［設定］を選択します。
2. ［プロファイルモードの設定］の横にある［設定］を押します。
3. モードを変更したいプロファイルを選択します。
4. 不平 (complain) モードと 強制 (enforce) モードを切り替えたい場合は、［モード切替］ボタンを押します。
5. 設定を適用して終了するには、［完了］を押します。

全てのプロファイルのモードを変更したい場合は、［すべてを強制モードに設定］か［すべてを不平モードに設定］のいずれかを押します。



ヒント: 利用可能なプロファイルの一覧表示

既定では、有効なプロファイル (対応するアプリケーションがシステム内にインストールされているプロファイル) のみが表示されます。対応するアプリケーションをインストールする前にプロファイルを設定しておきたい場合は、［すべてのプロファイルを表示］を押して、表じれたプロファイルの中から必要な設定を行ってください。

34 コマンドラインからのプロファイル構築

改訂履歴

2024-05-10

AppArmor® には、システムのセキュリティを管理するにあたって、グラフィカルなインターフェイスだけでなく、コマンドラインインターフェイスも用意されています。AppArmor のコマンドラインツールでは、AppArmor の状態確認のほか、プロファイルの作成や削除、修正などを行うことができます。



ヒント: 背景となる情報について

AppArmor のコマンドラインツールでプロファイルを管理し始める前に、まずは 第30章「プログラムに対する予防接種」と 第31章「プロファイルのコンポーネントと文法」にある AppArmor の一般的な情報をお読みください。

34.1 AppArmor の状態確認

AppArmor は下記に示す 3 種類の状態のいずれかになります:

未読み込み (Unloaded)
: AppArmor がカーネル内に読み込まれていない状態です。

動作中 (Running)
: AppArmor がカーネル内に読み込まれていて、AppArmor がプログラムポリシーを強制している状態です。

停止済み (Stopped)
: AppArmor がカーネル内に読み込まれていますが、何もポリシーを適用していない状態です。

`/sys/kernel/security/apparmor/profiles` ファイルに書かれた内容を読むことで、AppArmor の状態を検知することができます。`cat /sys/kernel/security/apparmor/profiles` を実行すると、プロファイルの一覧と AppArmor の動作状態が表示されます。中身が空であった場合、AppArmor は停止していることになります。また、ファイルが存在していない場合は、未読み込みの状態であることを示しています。

AppArmor は `systemctl` コマンドで管理することができます。下記のとおり入力して実行することで、それぞれの処理を実行することができます:

`sudo systemctl start apparmor`

その時点での AppArmor の状態に応じて動作が変わります。未読み込みの状態であった場合、`start` コマンドを実行すると AppArmor を読み込んで起動し、動作中の状態にします。停止済みの状態であった場合は、`/etc/apparmor.d` ディレクトリ内にある AppArmor プロファイルを読み込み直して、動作中の状態にします。既に AppArmor が動作中であった場合は、警告メッセージを表示して何もしません。



注記: 既に動作中のプロセスについて

既に動作中のプロセスに対して AppArmor のプロファイルを適用するには、それらを再起動する必要があります。

`sudo systemctl stop apparmor`

AppArmor が動作中であった場合は、すべてのプロファイルをカーネルのメモリから削除して、すべてのアクセス制御を無効化したあと、停止状態に移行します。AppArmor が既に停止していた場合も同様の処理を行おうとしますが、特に何も起こりません。

`sudo systemctl reload apparmor`

既に動作中で制限を設定しているプロセスに影響を与えることなく、AppArmor のモジュールに対してプロファイルの再スキャンを求めます。これにより、`/etc/apparmor.d` ディレクトリ内にある新しいプロファイルが適用されるほか、削除されたプロファイルがあればメモリ内からも削除されます。

34.2 AppArmor プロファイルの構築

AppArmor のプロファイル定義は `/etc/apparmor.d` ディレクトリ内に存在していて、プロファイル自身はテキストファイルで記述されています。これらのファイルの書式について、詳しくは 第31章「プロファイルのコンポーネントと文法」をお読みください。

`/etc/apparmor.d` ディレクトリ内にあるすべてのファイルはプロファイルとして扱われ、読み込まれます。ディレクトリ内でのファイル名の変更は、プロファイルを削除するにあたっては不適切です。特定のプロファイルを読み込まれないように設定したい場合は、このディレクトリからファイルを削除するか、プロファイルに対して `aa-disable` を実行してください。これにより、`/etc/apparmor.d/disabled/` 内にシンボリックリンクが作成されます。

プロファイルを読み書きしたい場合は、`vi` のようなテキストエディタをお使いください。下記の章には、プロファイルを作成するための説明が書かれています:

AppArmor プロファイルの追加と作成
: 34.3項「AppArmor プロファイルの追加と作成」をお読みください

AppArmor プロファイルの編集
: 34.4項「AppArmor プロファイルの編集」をお読みください

AppArmor プロファイルの削除
: 34.6項「AppArmor プロファイルの削除」をお読みください

34.3 AppArmor プロファイルの追加と作成

特定のアプリケーションに対して AppArmor のプロファイルを追加もしくは作成するにあたっては、一括プロファイル作成と単独プロファイル作成のいずれかの方式をとることができます。この 2 種類の方式について、詳しい説明は 34.7項「2 種類のプロファイル作成方法」をお読みください。

34.4 AppArmor プロファイルの編集

AppArmor のプロファイルを編集するには、下記の手順を実施します:

1. `root` でログインしていない場合は、端末ウインドウ内で `su` と入力して実行します。
2. `root` のパスワード入力を求められた場合は、そのパスワードを入力します。
3. `cd /etc/apparmor.d/` と入力して実行し、プロファイルのディレクトリに移動します。
4. `ls` と入力して実行すると、現在インストールされているすべてのプロファイルの一覧を表示することができます。
5. 編集したいプロファイルをテキストエディタ (例: vim) で開きます。
6. 必要な変更を行ったあと、ファイルを保存して終了します。
7. 端末ウインドウ内で `systemctl reload apparmor` と入力して実行し、AppArmor を再起動します。

34.5 未知の AppArmor プロファイルの読み込み解除



警告: 読み込み解除の危険性について

`aa-remove-unknown` を実行すると、`/etc/apparmor.d` ディレクトリ内に存在しないすべてのプロファイル (たとえば自動生成された LXD プロファイルなど) の読み込みが解除されます。これにより、システムのセキュリティが危険にさらされる場合があります。読み込みを解除する前に `-n` パラメータを付けて実行し、どのプロファイルの読み込みが解除されるのかを調べてから実行しておくことをお勧めします。

`/etc/apparmor.d/` ディレクトリ内に存在していないすべての AppArmor プロファイルについて、それらの読み込みを解除したい場合は、下記のように入力して実行します:

```
> sudo aa-remove-unknown
```

読み込みが解除されるプロファイルの一覧を表示したい場合は、下記のようにします:

```
> sudo aa-remove-unknown -n
```

34.6 AppArmor プロファイルの削除

AppArmor のプロファイルを削除するには、下記の手順を実施します:

1. まずはカーネルから AppArmor の定義を削除します:

   ```
   > sudo apparmor_parser -R /etc/apparmor.d/プロファイルのファイル名
   ```

2. プロファイルのファイルそのものを削除します:

   ```
   > sudo rm /etc/apparmor.d/プロファイルのファイル名
      > sudo rm /var/lib/apparmor/cache/プロファイルのファイル名
   ```

34.7 2 種類のプロファイル作成方法

第31章「プロファイルのコンポーネントと文法」に AppArmor のプロファイル文法に関する説明があるとおり、AppArmor ツールを使用せずにプロファイルを作成することもできます。しかしながら、何もない状態からプロファイルを作成するのは手間がかかります。そのため、AppArmor の各種ツールを利用して、プロファイルを自動作成したり、自動的に調整したりする機能を利用することをお勧めします。

AppArmor のプロファイルを作成するにあたっては、下記に示す 2 種類の方法があります。いずれの方法であっても、ツールを利用して行うことができます。

単独プロファイル作成

電子メールクライアントなど、アプリケーション自体が比較的小規模で、かつ必要に応じて起動したり終了したりするようなものにお勧めです。詳しくは 34.7.1項「単独プロファイル作成」をお読みください。

一括プロファイル作成

多数のプログラムに対してプロファイルを一括作成するような場合に適切であるほか、Web サーバやメールサーバなどのように、常に動作し続け、システムを再起動した後にも、すぐに動作させる必要があるもののような場合にも適切です。詳しくは 34.7.2項「一括プロファイル作成」をお読みください。

プロファイルを自動生成することで、AppArmor のツールによる管理がよりやりやすくなります:

1. まずは要件にあったプロファイル方法を選びます。
2. 静的な分析を行います。選択した方法にあわせて、`aa-genprof` もしくは `aa-autodep` のいずれかを実行します。
3. 動的な学習を有効化します。プロファイルを作成したすべてのプログラムに対して、学習モードを有効化します。

34.7.1 単独プロファイル作成

単独でプロファイルを作成して改善していく場合は、`aa-genprof` と呼ばれるプログラムを利用します。この方法では、`aa-genprof` が様々な作業を代行してもらえるため、作業が簡単になる一方、プログラムを起動してから終了するまでの間、`aa-genprof` を実行し続けなければならない、という制限が発生します (つまり、プロファイルの作成中はマシンの再起動を行うこともできなくなります) 。

単独プロファイル作成で `aa-genprof` を使用する場合は、34.7.3.8項「aa-genprof: プロファイルの生成」をお読みください。

34.7.2 一括プロファイル作成

この方法は、`aa-genprof` で行う単独 (もしくは少数) のプロファイル作成とは異なり、システム内にある複数のプロファイルを一括で更新することから、一括プロファイル作成 と呼ばれています。一括プロファイル作成ではプロファイルの構築と改善を自動化することができず、手作業が発生しますが、より柔軟に対応することができます。この方法は、システムが再起動しても動作し続ける長期稼働型のアプリケーションや、多数のプログラムに対して一括でプロファイルを作成したい場合に便利です。

一括で AppArmor のプロファイルを作成するには、下記の手順で行います:

1. まずはお使いのアプリケーションの各プログラムに対して、プロファイルを作成します。
   この方式を使用する際には、あらかじめプログラムに対する AppArmor のプロファイルを作成しておかなければなりません。これは、AppArmor がプロファイルのあるプログラムのみを監視するためです。これに対応させるために、`aa-autodep` を利用して、プログラムに対する近似プロファイルを作成してください。詳しくは 34.7.3.1項「aa-autodep: 近似プロファイルの作成」をお読みください。

2. 対応するプロファイルを学習 (不平) モードに設定します。
   プロファイルの存在するすべてのプログラムに対して、プロファイルを学習 (不平) モードに切り替えたい場合は、端末内で `root` になり、下記のように実行します:

   ```
   > sudo aa-complain /etc/apparmor.d/*
   ```

   この機能は、YaST のプロファイルモジュールでも設定することができます。詳しくは 33.4.2項「個別のプロファイルに対するモード変更」をお読みください。
   学習モードの場合、プロファイル側で明示的に禁止されている場合であっても、実際にアクセス制限が発動されることはありません。これにより、ステップ 3 で示しているとおり、様々なテストを実施して、プログラムにとって必要なアクセス権を正しく検出することができるようになります。また、この情報を利用することで、プロファイル内でどこまでの許可を与えればよいのかがわかるようにもなります。
   学習 (不平) モードへの切り替えについて、詳しくは 34.7.3.2項「aa-complain: 不平モード (学習モード) への突入」をお読みください。

3. 対象のアプリケーションを様々にテストします。
   アプリケーション内にある様々な機能を動作させて試します。どれだけプログラムを試せばよいのかはプログラム次第ですが、少なくともプログラム側からアクセスする必要のあるファイルに対して、一通りアクセスするようにしてください。なお、処理は `aa-genprof` の監視下で動作しているわけではありませんので、何日も何週間も動作させたままでかまいませんし、システムを再起動してもかまいません。

4. ログを分析します。

一括プロファイル作成では、単独プロファイル時のように `aa-genprof` から `aa-logprof` を実行するのではなく、`aa-logprof` を直接実行します。`aa-logprof` は下記のように入力して実行します:

```
> sudo aa-logprof [ -d プロファイルのパス ] [ -f /ログファイルのパス ]
```

`aa-logprof` の使用方法について、詳しくは 34.7.3.9項「aa-logprof: システムログのスキャン」をお読みください。

5. ステップ 3 と ステップ 4 を繰り返し実施します。
   これにより、最適なプロファイルを作成することができるようになります。このように繰り返し実施することで、分析対象のログファイルを小さくすることができますので、より素早くプロファイルに反映できるようになります。また、繰り返すたびにログの出力も減るはずですので、処理も高速化されます。

6. プロファイルを編集します。
   生成されたプロファイルの内容を確認します。具体的には、テキストエディタなどで `/etc/apparmor.d/` ディレクトリ内のファイルを開いて、必要であれば編集を行います。

7. 強制モードに戻します。
   強制モードに戻すことで、プロファイルのルールに対して、その違反を記録するだけでなく、違反を明示的に禁止することができるようになります。この作業は、プロファイルをテキストエディタなどで開いて、`flags=(complain)` と書かれた箇所を削除して保存するか、もしくは `aa-enforce` コマンドを使用することで行うことができます (`aa-enforce` コマンドは、`aa-complain` と同じように使用することができるもので、強制モードに移行させることができます) 。また、この作業は YaST のプロファイルモジュールでも実施することができます。詳しくは 33.4.2項「個別のプロファイルに対するモード変更」をお読みください。
   すべてのプロファイルを不平モードから強制モードに移行させたい場合は、`aa-enforce /etc/apparmor.d/*` と入力して実行します。

8. 再度すべてのプロファイルをスキャンします。
   AppArmor に対してすべてのプロファイルを再スキャンさせ、カーネル内を強制モードに変更するには、`systemctl reload apparmor` と入力して実行します。

34.7.3 プロファイル作成ツールの概要

AppArmor におけるプロファイルユーティリティは、`apparmor-utils` パッケージ内にすべて含まれていて、`/usr/sbin` 内にインストールされます。それぞれのツールの概要は下記のとおりです。

34.7.3.1 aa-autodep: 近似プロファイルの作成

このプログラムは、選択したプログラムやアプリケーションに対応する、近似プロファイルを作成するツールです。バイナリ実行ファイルやインタプリタ型のスクリプトプログラムに対応しています。生成されたプロファイルは、AppArmor で適切に制限を設定するにあたって、完全なものではないことから、「近似」プロファイルと呼ばれます。`aa-autodep` で生成される近似プロファイルには、ほとんどのプログラムで必要な項目を列挙した、基本的な include ディレクティブのみが含まれる最小限のプロファイルとなります。また、特定の種類のプログラムであれば、`aa-autodep` はさらに詳しいプロファイルを生成します。コマンドラインで指定したプログラムに対して `ldd(1)` を再帰的に呼び出し、プロファイルを生成しようとします。

近似プロファイルを生成するには `aa-autodep` プログラムを使用します。パラメータとしてプログラム名を単純に指定した場合は、`aa-autodep` がシェルの PATH 環境変数を展開してプログラムを探します。フルパスを指定してもかまいません。また、プログラムそれ自身は任意のものでかまいません (ELF バイナリであっても、シェルや Perl のスクリプトであってもかまいません) 。`aa-autodep` は、後続の動的プロファイル作成で改善を行うための、ベースとなる近似プロファイルを生成します。

生成された近似プロファイルは、`/etc/apparmor.d` ディレクトリ内に、AppArmor のプロファイル命名規約に従って出力されます。具体的には、スラッシュ (`/`) をピリオド (`.`) に置き換えたファイル名になります。`aa-autodep` のコマンドの書式は下記のとおりです:

```
> sudo aa-autodep [ -d プロファイルのパス ] [プログラム_1 プログラム_2...]
```

プログラム名を入力しない場合は、入力を求められます。また、`プロファイルのパス` を指定した場合は、既定のプロファイル保存先である `/etc/apparmor.d` を変更し、既定の場所以外にプロファイルを保存することができます。

プロファイル作成を開始するにあたっては、まずお使いのアプリケーションを構成するメインの実行ファイル (プロファイルが存在しておらず、他のプログラムから呼び出されることのない実行ファイル) に対して、プロファイルを作成しなければなりません。このような条件に該当するプログラムを、お使いのアプリケーション内ですべて見つけてから、プロファイルを作成してください。このようなプログラムを見つける方法には、下記のような方法があります:

ディレクトリ
: プロファイルを作成すべきプログラムが特定の 1 つのディレクトリ内にのみ存在し、それ以外には存在しない場合であれば、`aa-autodep プログラムのパス/*` のように実行することで、そのディレクトリ内にあるすべてのプログラムに対して、近似プロファイルを作成することができます。

pstree -p
: アプリケーションを起動したあと、標準的な Linux コマンドである `pstree` を使用することで、動作中のすべてのプロセスを取得することができます。あとはプログラムがどこに配置されているのかを突き止めて、それぞれに対して `aa-autodep` を実行します。なお、プログラムがお使いのパ

ス内に存在している状態であれば、`aa-autodep` を実行するだけで、実際のコマンドパスを自動的に検索させることができます。パス内に存在していない場合は、標準的な Linux コマンドである `find` を使用して、プログラムパスを探してください。たとえば `find / -name '` アプリケーション名 `' -print` のように実行することで、アプリケーションのパスを取得することができます。また、必要であればワイルドカードを使用してもかまいません。

34.7.3.2 aa-complain: 不平モード (学習モード) への突入

不平モード (学習モード) ツール (`aa-complain`) は、指定した AppArmor のプロファイルルールのモードを変更するツールです。不平モードに設定したプロファイルに対しては、ルールへの違反が許容されるものの、ログに記録されるようになります。これはプロファイルを改善していくための仕組みで、いったん不平モードを設定した後、さまざまなプログラム処理を実施して、プログラムのアクセス要件をログに記録させて、そのログを元にプロファイルを改善する流れを取ります。

コマンドラインから不平モードを手作業で有効化すると、プロファイルの冒頭にフラグを設定し、`/bin/foo` のような項目が `/bin/foo flags=(complain)` のようになります。不平モードを使用するには、端末ウインドウを開いて、`root` で下記のように入力して実行します:

- プログラム (プログラム_1) がパス内に存在する場合は、下記のように実行します:

```
> sudo aa-complain [プログラム_1 プログラム_2 ...]
```

- プログラムがパス内に存在しない場合は、下記のようにフルパスで指定します:

```
> sudo aa-complain /sbin/プログラム_1
```

- また、プロファイルが `/etc/apparmor.d` 内に存在しない場合は、下記のように配置場所を指定します:

```
> sudo aa-complain プロファイルのパスプログラム_1
```

- `/sbin/program1` のプロファイルを指定したい場合は、下記のように入力して実行します:

```
> sudo aa-complain /etc/apparmor.d/sbin.プログラム_1
```

上述のコマンドはいずれも、指定したプロファイルやプログラムを不平モードに切り替える処理を行います。プログラムの指定の際、これをフルパスで指定しないと、`aa-complain` は `$PATH` 内を検索して、プログラムの場所を判断します。たとえば `aa-complain /usr/sbin/*` のように入力して実行すると、`/usr/sbin` 内に存在するすべてのプログラムに対して、対応するプロファイルを検索し、それらすべてを不平モードに設定します。また、`aa-complain /etc/apparmor.d/*` のように入力して実行すると、`/etc/apparmor.d` 内にあるすべてのプロファイルを不平モードに切り替えます。



ヒント: YaST を利用したプロファイルの切り替えについて

YaST には、不平モードと強制モードを切り替えることのできるグラフィカルなフロントエンドが用意されています。詳しくは 33.4.2項「個別のプロファイルに対するモード変更」をお読みください。

34.7.3.3 aa-decode: AppArmor ログファイル内にある 16 進数にエンコードされた文字列のデコード

`aa-decode` は、AppArmor のログ出力内に存在する 16 進数で書かれた文字列をデコードすることができます。標準入力から監査ログを入力させることもできますし、AppArmor のログ内にある 16 進数の文字列をデコードして、標準出力に書き出すこともできます。

34.7.3.4 aa-disable: AppArmor セキュリティプロファイルの無効化

`aa-disable` コマンドは、1 つもしくは複数の AppArmor プロファイルの強制モードを無効化するために使用します。このコマンドを実行すると、カーネルからプロファイルの読み込みを解除して開放し、AppArmor の起動時にもプロファイルが読み込まれないようにします。この動作を変更したい場合は、`aa-enforce` もしくは `aa-complain` の各ツールをお使いください。

34.7.3.5 aa-easyprof: 簡易プロファイル生成

`aa-easyprof` は AppArmor のプロファイル生成に対して、簡易的なインターフェイスを提供する仕組みです。`aa-easyprof` ではテンプレート (雛形) やグループ化の仕組みを使用して、アプリケーションのプロファイルを素早く作成することができます。`aa-easyprof` はプロファイル生成の支援を行いますが、これは使用するテンプレートの品質とプロファイルのグループ、および抽象に依存しています。また、このツールは手作業でプロファイルを作成する場合や、`aa-genprof` と `aa-logprof` を利用してプロファイルを作成する場合に比べると、制限の緩いプロファイルを作成することになります。

詳しくは `aa-easyprof` (8) のマニュアルページをお読みください。

34.7.3.6 aa-enforce: 強制モードへの突入

強制モードでは、プロファイルで許可されていないファイルへのアクセスが禁止されるなど、AppArmor のプロファイルルールへの違反を検知することができます。このモードでは、違反はログに記録されるだけでなく、許可もされません。そのため、普段使用する場合は強制モードの適用を推奨します。違反を記録するのみで禁止したくない場合は、不平モードに設定してください。

コマンドラインから強制モードを手作業で設定すると、プロファイルの冒頭にあるフラグを削除し、`/bin/foo flags=(complain)` のような項目が `/bin/foo` のようになります。強制モードを使用するには、端末ウインドウを開いて、`root` で下記のいずれかのように入力して実行します:

- プログラム (`プログラム_1`) がパス内に存在する場合は、下記のように実行します:

```
> sudo aa-enforce [プログラム_1 プログラム_2 ...]
```

- プログラムがパス内に存在しない場合は、下記のようにフルパスで指定します:

```
> sudo aa-enforce /sbin/プログラム_1
```

- また、プロファイルが `/etc/apparmor.d` 内に存在しない場合は、下記のように配置場所を指定します:

```
> sudo aa-enforce -d プロファイルのパス      プログラム_1
```

- `/sbin/program1` のプロファイルを指定したい場合は、下記のように入力して実行します:

```
> sudo aa-enforce /etc/apparmor.d/sbin.プログラム_1
```

上述のコマンドはいずれも、指定したプロファイルやプログラムを強制モードに切り替える処理を行います。

プログラム名やプロファイル名を入力しない場合は、入力を求められます。また、`プロファイルのパス` を指定した場合は、既定のプロファイル保存先である `/etc/apparmor.d` を変更し、既定の場所以外にあるプロファイルを使用することができます。

パラメータには、複数のプログラムやプロファイルを指定することができます。また、プログラムの指定の際、これをフルパスで指定しないと、`aa-complain` は `$PATH` 内を検索して、プログラムの場所を判断します。



ヒント: YaST を利用したプロファイルの切り替えについて

YaST には、不平モードと強制モードを切り替えることのできるグラフィカルなフロントエンドが用意されています。詳しくは 33.4.2項「個別のプロファイルに対するモード変更」をお読みください。

34.7.3.7 aa-exec: 指定したプロファイルによるプログラムの制限

`aa-exec` は指定したプロファイルやプロファイルネームスペースで、特定のプログラムを起動するためのツールです。プロファイルとネームスペースの両方を指定した場合、プログラムは新しいネームスペース内のプロファイルで制限を受けることになります。ネームスペースのみを指定した場合は、現在の制限内にあるプロファイル名を使用します。いずれも指定しない場合は、コマンドは標準的なプロファイル適用をそのまま使用します (つまり、`aa-exec` コマンドを使用せずに起動した場合と同じ動作になります) 。

子マントのオプションについて、詳しくは `man 8 aa-exec` にあるマニュアルページをお読みください。

34.7.3.8 aa-genprof: プロファイルの生成

`aa-genprof` は AppArmor におけるプロファイル生成ユーティリティです。プロファイルが存在しない場合は、指定したプログラムに対して `aa-autodep` を実行して近似プロファイルを作成したあと、プロファイルを不平モードに設定して AppArmor を再読み込みさせ、ログに印を付けます。その後、ユーザに対してプログラムを実行し、その機能を一通り試すように依頼します。コマンドの書式は下記のとおりです:

```
> sudo aa-genprof [ -d プロファイルのパス ]  プログラム
```

Apache Web サーバである httpd2-prefork に対してプロファイルを作成したい場合は、`root` で下記のように実行します:

1. `systemctl stop apache2` と入力して実行します。
2. 続けて `aa-genprof httpd2-prefork` と入力して実行します。
   すると、`aa-genprof` は下記のような処理を行います:
   1. お使いのシェルの PATH 環境変数を利用して httpd2-prefork のフルパスを取得します。もちろん最初からフルパスで指定してもかまいません。openSUSE Leap では、このプログラムのフルパスは `/usr/sbin/httpd2-prefork` であるはずです。
   2. まずは httpd-prefork に対して、既存のプロファイルが存在していないかどうかを確認します。プロファイルが存在した場合はそれを更新しようとします。存在しない場合は、34.7.3項「プロファイル作成ツールの概要」で説明している `aa-autodep` を使用して、プロファイルを作成します。

3. 指定したプログラムに対するプロファイルを学習 (不平) モードに設定し、プロファイルへの違反を記録するだけで、禁止を行わないようにします。違反の記録は、たとえば下記のようになります (`/var/log/audit/audit.log` 内に記録されます):

```
type=APPARMOR_ALLOWED msg=audit(1189682639.184:20816): \
apparmor="DENIED" operation="file_mmap" parent=2692 \
profile="/usr/sbin/httpd2-prefork//HANDLING_UNTRUSTED_INPUT" \
name="/var/log/apache2/access_log-20140116" pid=28730 comm="httpd2-prefork" \
requested_mask="::r" denied_mask="::r" fsuid=30 ouid=0
```

監査デーモンを動作させていない場合、AppArmor のイベントは `systemd` ジャーナルに直接記録されます (詳しくは『リファレンス』、第11章「`journalctl` : systemd ジャーナルへの問い合わせコマンド」をお読みください):

```
Sep 13 13:20:30 K23 kernel: audit(1189682430.672:20810): \
apparmor="DENIED" operation="file_mmap" parent=2692 \
profile="/usr/sbin/httpd2-prefork//HANDLING_UNTRUSTED_INPUT" \
name="/var/log/apache2/access_log-20140116" pid=28730 comm="httpd2-prefork" \
requested_mask="::r" denied_mask="::r" fsuid=30 ouid=0
```

`dmesg` コマンドを利用して表示することもできます:

```
audit(1189682430.672:20810): apparmor="DENIED" \
operation="file_mmap" parent=2692 \
profile="/usr/sbin/httpd2-prefork//HANDLING_UNTRUSTED_INPUT" \
name="/var/log/apache2/access_log-20140116" pid=28730 comm="httpd2-prefork" \
requested_mask="::r" denied_mask="::r" fsuid=30 ouid=0
```

4. ログ内に印を付けて、作業の開始位置判断とします。たとえば下記のように出力されます:

```
Sep 13 17:48:52 figwit root: GenProf: e2ff78636296f16d0b5301209a04430d
```

3. ツール側から問い合わせメッセージが表示されたら、もう 1 つの端末ウインドウを開いてプロファイルを作成するアプリケーションを起動し、可能な限り様々な処理を実施させます。これにより、学習モード下でプログラムが必要とする様々なファイルやディレクトリなどにアクセスすることになります。この例では、たとえば新しい端末ウインドウを開いて、`systemctl start apache2` と入力し実行します。

4. 様々なプログラムの機能を利用したあとは、`aa-genprof` の端末ウインドウ内で下記のような入力を行うことができます:

   - `S` と入力すると、`aa-genprof` は先ほど印を付けた箇所以降のシステムログを検索する処理を実施します。ログ内に何らかのシステムイベントが見つかった場合、AppArmor はそれらを読み込んで判断します。これにより、セキュリティプロファイルを生成するにあたっての様々な質問を生成します。

   - `F` と入力すると、ツールを終了することができます。

   注記

   ハット (hat) を追加するように求められた場合は、第35章「チェンジハット機能による Web アプリケーションのプロファイル作成」をお読みください。

5. あとは下記に示す 2 種類の質問に答えていきます:

   - プロファイル対象のプログラムから特定のリソースへのアクセスが要求されたものの、プロファイル内には許可が書かれていないもの (詳しくは 例34.1「学習モードの例外: 特定のリソースへのアクセス制御」をお読みください) 。

   - プロファイル対象のプログラムからプログラムが実行されたものの、セキュリティドメイン遷移が定義されていないもの (詳しくは 例34.2「学習モードの例外: 項目に対する許可の設定」をお読みください) 。

   いずれの分類の質問であっても、プロファイル内にリソースやプログラムを追加するために回答しなければならない質問になります。例34.1「学習モードの例外: 特定のリソースへのアクセス制御」や 例34.2「学習モードの例外: 項目に対する許可の設定」には、それぞれ質問の例が示されています。続く手順では、それぞれの質問に対する回答方法を示しています。

   - 実行権限の取り扱いは複雑です。それぞれ表示された内容に対して、どのような実行権限を与えるのかを決定しなければなりません:

     例 34.1: 学習モードの例外: 特定のリソースへのアクセス制御

     ```
     Reading log entries from /var/log/audit/audit.log.
     Updating AppArmor profiles in /etc/apparmor.d.

     Profile:  /usr/sbin/cupsd
     Program:  cupsd
     Execute:  /usr/lib/cups/daemon/cups-lpd
     Severity: unknown
     ```

```
(I)nherit / (P)rofile / (C)hild / (N)ame / (U)nconfined / (X)ix / (D)eny /
 Abo(r)t / (F)inish
```

Inherit (ix)

子プロセスに対して親と同じプロファイルを適用し、親と同じアクセス権限を継承するように指定します。このモードは、制限を受けているプログラムから他のプログラムを呼び出す際、新しい許可を設定したくない場合や、既にある許可を失いたくない場合に指定します。このモードは、たとえば `/usr/bin/mail` のメールクライアントが `less` をページャとして使用する場合など、ヘルパーアプリケーション のようなプログラムを呼び出す際にしばしば用いられます。

Profile (px/Px)

子プロセスに対して、カーネル内に読み込まなければならない独自のプロファイルを適用するように指定します。プロファイルが存在しない場合、子プロセスの実行はアクセス拒否として失敗します。これは、親プログラムから DNS の参照やお使いのシステムの MTA を介したメール送信など、グローバルサービスを実行するような場合に有用です。

［profile with clean exec］(クリーン実行付きプロファイル) (Px) を選択すると、子プロセスに環境変数を渡す際、その実行に影響があるような環境変数を削除して実行することができます。

Child (cx/Cx)

サブプロファイルへの遷移を設定します。px/Px での遷移と同様ですが、子プロファイルへの遷移という点が異なります。

［profile with clean exec］(クリーン実行付きプロファイル) (Cx) を選択すると、子プロセスに環境変数を渡す際、その実行に影響があるような環境変数を削除して実行することができます。

Unconfined (ux/Ux)

子プロセスに対して AppArmor のプロファイルを適用せず、リソースに対する制限を行わないようにします。

［unconfined with clean exec］(クリーン実行付き無制限) (Ux) を選択すると、子プロセスに環境変数を渡す際、その実行に影響があるような環境変数を削除して実行することができます。ただし、制限を取り外してしまうことはセキュリティ面での脆弱性を生む行為であり、危険であることに注意してください。そのため、それ以外の選択肢ではどうしても解決できない場合にのみ、指定してください。

mmap (m)

この許可を設定すると、そのプロファイルが適用されるプログラムに対して、mmap システムコールの際に `PROT_EXEC` フラグを設定できるようになります。これにより、データとして割り当てたメモリを実行することができるようになります。プロファイル生成時に求められた場合にのみ、この許可を設定してください。

Deny

プロファイルに対して `deny` ルールを追加して、指定したディレクトリパス項目に対するアクセスを明示的に禁止するようにします。AppArmor はそのまま次のイベントの処理に移ります。

Abort

`aa-logprof` を中止し、ここまでに入力したルール変更を破棄して終了します。プロファイルの修正は行いません。

Finish

`aa-logprof` を終了し、ここまでに入力したルール変更を保存し、プロファイルの修正を行います。

- 例34.2「学習モードの例外: 項目に対する許可の設定」には、AppArmor がグロブパターン `/var/run/nscd/*` を利用して読み込みアクセスを許可し、一般的な Apache 関連のアクセスルールをカバーするため、抽象を使用しようとしている様子を示しています。

例 34.2: 学習モードの例外: 項目に対する許可の設定

```
Profile:  /usr/sbin/httpd2-prefork
Path:     /var/run/nscd/dbSz9CTr
Mode:     r
Severity: 3

  1 - /var/run/nscd/dbSz9CTr
 [2 - /var/run/nscd/*]

(A)llow / [(D)eny] / (G)lob / Glob w/(E)xt / (N)ew / Abo(r)t / (F)inish /
 (O)pts
Adding /var/run/nscd/* r to profile.

Profile:  /usr/sbin/httpd2-prefork
Path:     /proc/11769/attr/current
Mode:     w
Severity: 9

 [1 - #include <abstractions/apache2-common>]
  2 - /proc/11769/attr/current
  3 - /proc/*/attr/current

(A)llow / [(D)eny] / (G)lob / Glob w/(E)xt / (N)ew / Abo(r)t / (F)inish /
 (O)pts
Adding #include <abstractions/apache2-common> to profile.
```

AppArmor には様々なパスやインクルードが用意されています。選択肢の番号を入力することで必要なオプションを選択し、次のステップに進むことができます。



注記

AppArmor のメニューでは、一部の選択肢のみが表示される場合もあります。

`#include`

これは AppArmor プロファイル内の記述で、他のファイルを参照するように設定するためのディレクティブです。これにより、他のファイル内に書かれているアクセス権を振り込むことができるようになります。また、include を使用することで、複数のプログラムでアクセス許可の設定を共有することもできます。この仕組みにより、プロファイルのサイズを小さくすることにも繋がります。include を使用するように提案された場合は、提案を受け入れるのがよいでしょう。

グロブ版

次の手順で説明しているように、［Glob］を選択することでアクセスすることができます。グロブの書式に関する詳細は、31.6項「プロファイル名／フラグ／パス／グロブ」をお読みください。

実際のパス

プログラムが必要とするパスを直接指定して、正しく動作するように設定します。

パスまたはインクルードの設定を行った後は、［Allow］(許可) または［Deny］(拒否) を選択して、対応する項目を AppArmor のプロファイルに追加します。表示されたディレクトリパスの項目で問題があるとお考えの場合は、［Glob］(グロブ) を選択することもできます。

学習モードで生成された項目を処理し、プロファイルにする方法には、下記のものがあります:

単に `Enter` を押す

選択したディレクトリパスに対して、アクセスを許可します。

Allow

選択したディレクトリパスの項目に対して、アクセスを許可します。AppArmor では、アクセスモードの提案も行います。詳しくは 31.7項「ファイルアクセス許可のアクセスモード」をお読みください。

Deny

指定したディレクトリパスの項目に対して、プログラムからのアクセスを拒否します。AppArmor はそのまま次のイベントの処理に移ります。

New

このイベントに対応するルールを独自に作成します。ここでは、正規表現による指定にも対応しています。表現の指定がイベントの要件と合致しない場合、AppArmor は確認メッセージを表示して再入力を求めます。

Glob

ワイルドカードを利用して幅広くマッチするルールを作成します。提示されたパスのうちのいずれかを選択するには、パスの前に示されている数字を入力して進めます。グロブの書式に関する詳細は、31.6項「プロファイル名／フラグ／パス／グロブ」をお読みください。

Glob w/Ext

元のディレクトリパス情報を変更して、ファイル名の拡張子だけをマッチングルールとして残すグロブを生成します。たとえば `/etc/apache2/file.ext` というファイルへのアクセスであった場合、ファイル名の部分だけをワイルドカードにして `/etc/apache2/*.ext` のような形式に変更します。これにより、特定のディレクトリ内にある同じ拡張子 (`.ext`) のファイルすべてに対して、許可するルールを作成することができます。

Abort

`aa-logprof` を中止し、ここまでに入力したルール変更を破棄して終了します。プロファイルの修正は行いません。

Finish

`aa-logprof` を終了し、ここまでに入力したルール変更を保存し、プロファイルの修正を行います。

6. `vi` を利用してプロファイルの表示や編集を行いたい場合は、端末ウインドウ内で `vi /etc/apparmor.d/` プロファイル名 のように入力して実行してください。vim 内で AppArmor のプロファイルを編集するにあたり、文法ハイライト機能を有効化したい場合は、`:syntax on` と入力してから `:set syntax=apparmor` と入力します。vim および文法ハイライト機能に関する詳細については、34.7.3.14項「apparmor.vim」をお読みください。

7. `systemctl reload apparmor` コマンドを実行し、AppArmor を再起動して、プロファイルセットを再読み込みします。これにより、新しく作成したプロファイルが読み込まれるようになります。

グラフィカルなフロントエンドを利用して AppArmor のプロファイルを作成する場合と同様に、YaST のプロファイル追加ウイザードや `aa-genprof` でも、`/usr/share/apparmor/extra-profiles` 内にあるローカルプロファイルを使用することができます。

ローカルリポジトリにあるプロファイルを使用するには、下記のようにして行います:

1. 上述の手順で `aa-genprof` を起動します。
`aa-genprof` が無効化されているローカルプロファイルを検出すると、端末ウインドウ内に下記のような内容が表示されます:

```
Profile: /usr/bin/opera

 [1 - Inactive local profile for /usr/bin/opera]

[(V)iew Profile] / (U)se Profile / (C)reate New Profile / Abo(r)t / (F)inish
```

2. 提示されたプロファイルを使用するには `U` (［Use Profile］) と入力し、上述のプロファイル生成手順に従って続けます。
   有効化する前にプロファイルの内容を確認したい場合は、`V` (［View Profile］) と入力します。
   既存のプロファイルを無視したい場合は、`C` (［Create New Profile］) と入力して、上述のプロファイル生成手順で、何もない状態からプロファイルを作成していきます。
3. すべての作業が終わったら、`F` (［Finish］) と入力すると、変更点を保存して終了することができます。

34.7.3.9 aa-logprof: システムログのスキャン

`aa-logprof` は不平モードや強制モードで記録され、`/var/log/audit/audit.log` 内もしくは `systemd` ジャーナル (詳しくは『リファレンス』、第11章「`journalctl` : systemd ジャーナルへの問い合わせコマンド」を参照) 内に直接保存されているログを処理して、AppArmor のセキュリティプロファイル内の新しい項目を生成することができる対話型ツールです。

`aa-logprof` を実行すると、まずは不平モードや強制モードで記録されたログファイルを読み込んで処理し、既存のプロファイルセット内でカバーされていない新しいセキュリティイベントが見つかった場合に、既存のプロファイルを変更するための確認メッセージを表示します。`aa-logprof` では、ログファイル内の情報からプログラムの挙動を確認します。

制限を受けているプログラムが fork() したり他のプログラムを起動したりした場合、`aa-logprof` はそれを検出すると、fork() したプログラムや他のプログラムに対して適用すべき実行モードを尋ねます。実行モードは ix , px , Px , ux , Ux , cx , Cx のいずれか、もしくは名前付きプロファイルで、指定した実行モードで子プロセスを開始することになります。子プロセス向けのプロファイルが個別に存在する場合は、既定の選択肢は Px になります。逆に、プロファイルが存在しない場合は ix になります。個別のプロファイルが存在する子プロセスに対しては、`aa-autodep` が実行され、動作中であれば AppArmor 内に読み込まれるようになります。

`aa-logprof` が終了すると、プロファイルは変更した内容に応じて更新されます。AppArmor が有効化されていれば、更新されたプロファイルが読み込まれ。セキュリティイベントを生成したプロセスが現在も null-XXXX プロファイル (不平モードで作成される一時的なプロファイル) で動作していれば、対応する適切なプロファイルが適用されるようになります。

`aa-logprof` を実行するには、`root` でログインしている状態で、端末ウインドウ内で `aa-logprof` と入力します。また、`aa-logprof` では下記のようなオプションを指定することができます:

`aa-logprof -d` プロファイルディレクトリのパス
: プロファイルが標準ディレクトリ `/etc/apparmor.d/` 以外の場所にある場合は、上記のようにしてフルパスを指定します。

`aa-logprof -f` ログファイルのパス

ログファイルが標準ディレクトリパス `/var/log/audit/audit.log` 以外の場所にある場合は、上記のようにしてログファイルのフルパスを指定します。

`aa-logprof -m "ログファイル内に記録するマーカー文字列"`

システムログ内に、`aa-logprof` が後から参照できるように、開始点の印を付けます。`aa-logprof` は、指定した印以前のイベントはすべて無視します。なお、印内にスペースが含まれている場合は、正しく引用符で括らなければなりません。たとえば下記のようになります:

```
# aa-logprof -m "17:04:21"
```

もしくは

```
# aa-logprof -m e2ff78636296f16d0b5301209a04430d
```

`aa-logprof` はログを読み込んでから、それぞれのイベントに対して処理方法を尋ねます。それぞれの質問には番号付きのリストが示され、プロファイル内にどれを追加すべきなのかを尋ねます。

既定では、`aa-logprof` は `/etc/apparmor.d/` 内にプロファイルがあるものとして検索します。また、`aa-logprof` でプロファイルを更新する場合、`root` での実行であれば十分です。ただし、長期にわたるログを読み込ませる場合など、過去の分も含めた複数のログファイルを処理させたい場合は、`zcat -f `ls -1tr` ログファイルのパス/* `` | aa-logprof -f - のように実行してください。

34.7.3.10 aa-logprof 例 1

下記の例では、httpd2-prefork が `/etc/group` ファイルへのアクセスを行う際、`aa-logprof` 側でどのような対応を行うのかについて説明しています。なお、`[]` は既定値を表しています。

この例では、`/etc/group` へのアクセスは httpd2-prefork による名前サービスの処理時に発生します。適切な回答は `1` で、これによってあらかじめ定義された AppArmor ルールを取り込むことができるようになります。`1` を選択すると、`#include` によって名前サービス向けのルール一式が取り込まれ、以後に発生する DNS 関連のイベントすべてを解決することができるようになります。また、DNS 関連の設定が変わったような場合でも、このファイルを修正すれば済むことになりますので、関連する多数のプロファイルを編集せずに済むことになります。

```
Profile:  /usr/sbin/httpd2-prefork
Path:     /etc/group
New Mode: r

[1 - #include <abstractions/nameservice>]
 2 - /etc/group
[(A)llow] / (D)eny / (N)ew / (G)lob / Glob w/(E)xt / Abo(r)t / (F)inish
```

下記のいずれかの回答をすることができます:

単に `Enter` を押す
: 既定の動作を実行します。この例では、表示されているディレクトリパスに対して、アクセスを許可します。

Allow
: 表示されているディレクトリパスに対して、アクセスを許可します。詳しくは 31.7項「ファイルアクセス許可のアクセスモード」をお読みください。

Deny
: 表示されているディレクトリパスに対して、恒久的にアクセスを拒否します。AppArmor はそのまま次のイベントの処理に移ります。

New
: このイベントに対応するルールを独自に作成します。ここでは、正規表現による指定にも対応しています。表現の指定がイベントの要件と合致しない場合、AppArmor は確認メッセージを表示して再入力を求めます。

Glob
: ワイルドカードを利用して幅広くマッチするルールを作成します。提示されたパスのうちのいずれかを選択するには、パスの前に示されている数字を入力して進めます。
グロブの書式に関する詳細は、31.6項「プロファイル名／フラグ／パス／グロブ」をお読みください。

Glob w/Ext
: 元のディレクトリパス情報を変更して、ファイル名の拡張子だけをマッチングルールとして残すグロブを生成します。たとえば `/etc/apache2/file.ext` というファイルへのアクセスであった場合、ファイル名の部分だけをワイルドカードにして `/etc/apache2/*.ext` のような形式に変更します。これにより、特定のディレクトリ内にある同じ拡張子 (`.ext`) のファイルすべてに対して、許可するルールを作成することができます。

Abort
: `aa-logprof` を中止し、ここまでに入力したルール変更を破棄して終了します。プロファイルの修正は行いません。

Finish
: `aa-logprof` を終了し、ここまでに入力したルール変更を保存し、プロファイルの修正を行います。

34.7.3.11 aa-logprof 例 2

この例では、vsftpd に対するプロファイル作成時の質問を示しています:

```
Profile:  /usr/sbin/vsftpd
Path:     /y2k.jpg

New Mode: r

[1 - /y2k.jpg]

(A)llow / [(D)eny] / (N)ew / (G)lob / Glob w/(E)xt / Abo(r)t / (F)inish
```

この質問には、いくつかの興味深い点があります。まず vsftpd がルートディレクトリに対するアクセス許可を求めています。ところが、openSUSE Leap では、vsftpd は既定で `/srv/ftp` 内のコンテンツを提供しているはずです。これは、vsftpd が chroot 環境 (jail) で動作していることによるもので、プログラム上のコードではルートディレクトリにアクセスしているつもりであるため、AppArmor のログファイル内でも元のパスではなく、ルートディレクトリであるかのように表示されてしまいます。

次に興味深い点はファイル名です。FTP クライアントから、ディレクトリ内にあるすべての JPEG ファイルに対するアクセスを許可したい場合は、[Glob w/Ext] を選んで `/*.jpg` に変更して処理を進めます。これにより、個別の `.jpg` ファイルに対していちいち許可を設定する必要がなくなるほか、将来的に `.jpg` ファイルが増えたような場合でも、問題なく対応できることになります。

最後に興味深い点は、FTP ファイル全体に対する点です。[Glob] を選んで、`aa-logprof` からの提案を `/y2k.jpg` から `/*` に変更します。それ以外にも、ディレクトリツリー全体に対してアクセスを許可してしまう方法もあります。この場合は [New] を選んで `/**.jpg` のように入力します。これにより、ディレクトリツリー全体に対して、すべての `.jpg` ファイルへのアクセスを許可します。それ以外にも、`/**` と指定する方法もあります。この場合は、ディレクトリツリー全体のすべてのファイルに対して、アクセスを許可することになります。

ここまでは、読み込みアクセスに関する話題を扱ってきました。書き込みアクセスについても同様に対応することができますが、書き込みアクセスを付与するにあたっては、より保守的に (制限を厳しくする方向に) 設定しておくことをお勧めします。また、実行アクセスについてはさらに複雑です。詳しい例については 例34.1「学習モードの例外: 特定のリソースへのアクセス制御」をお読みください。

下記の例では、`/usr/bin/mail` に対してプロファイルを作成していて、`/usr/bin/mail` が `/usr/bin/less` をヘルパーアプリケーションとして起動した際の質問の例です。`/usr/bin/less` は、長いメールメッセージを「ページ分割」するために使用しています:

```
/usr/bin/nail -> /usr/bin/less
(I)nherit / (P)rofile / (C)hild / (N)ame / (U)nconfined / (X)ix / (D)eny
```



注記

`/usr/bin/mail` の実際の実行ファイルが `/usr/bin/nail` となっていますが、これはスペルミスなどではなく、実際に存在するプログラム名です。

プログラム `/usr/bin/less` は非常にシンプルなプログラムで、1 画面以上にわたる長いテキストをスクロールしながら読むことができるプログラムです。このような便利な仕組みであることから、`/usr/bin/mail` でも使用されているわけです。なお、`less` はシンプルな仕組みでありながらも、同時に巨大でパワフルなプログラムでもあり、`tar` や `rpm` などのプログラムをさらに呼び出すこともできます。



ヒント

tar ファイルや RPM ファイルに対して `less` を実行すると、その中に含まれている内容を表示することができます。

ですが、メールメッセージを読む際には、`rpm` コマンドを自動的に実行してしまうのは危険です。なぜなら、RPM プログラムにはシステムにプログラムをインストールしたり、プログラムを修正したりする機能が含まれるため、Microsoft* Outlook でよく見つかるようなウイルス感染の経路になってしまうためです。そのため、ここで選ぶべき選択肢は［Inherit］(継承) になります。継承を指定すると、`/usr/bin/mail` から実行される状況下で less プログラムを実行する場合、`/usr/bin/mail` のプロファイルを適用し続けることになります。これには下記の 2 つの意味が含まれます:

- `/usr/bin/less` に対する基本的なファイルアクセス設定を、`/usr/bin/mail` のプロファイル内に追加する必要があります。
- `tar` や `rpm` などのヘルパーアプリケーションを `/usr/bin/mail` に登録することを避けることで、`/usr/bin/mail` が `/usr/bin/less` を実行しても、AppArmor が存在すれば非常に安全に動作することになります。もう 1 つの選択肢としては Cx 実行モードがあります。実行モードに関する詳細は、31.12項「実行モード」をお読みください。

その他の状況下では、［Profile］オプションを選択する必要があるかもしれません。この場合は、`aa-logprof` に対して下記のような効果があります:

- 現在のプロファイル内に px/Px モードのルールが書き込まれ、子プロセスに対する独自のプロファイルに遷移するようになります。
- `aa-logprof` が子プロセスに対するプロファイルを作成し、構築を開始します。これは親プロセスのプロファイルと同じ方法で行われ、親プロセスのイベントは親のプロファイルに、子プロセスのイベントは子のプロファイルにそれぞれ保存されるようになります。`aa-logprof` での質問は、両方をあわせて表示します。このプロファイルは、単独のプログラムを子として実行した場合にも当てはまります。

制限を受けているプログラムが fork() したり他のプログラムを起動したりした場合、`aa-logprof` はそれを検出すると、fork() したプログラムや他のプログラムに対して適用すべき実行モードを尋ねます。実行モードは継承 (ix) , プロファイル (px , Px) , 無制限 (ux , Ux) , 子プロセス (cx , Cx) , 名前付きプロファイルのほか、実行拒否を設定することもできます。

子プロセス向けのプロファイルが個別に存在する場合は、既定の選択肢はプロファイル (Px) になります。逆に、プロファイルが存在しない場合は継承 (ix) になります。許可の詳細については、31.7項「ファイルアクセス許可のアクセスモード」をお読みください。

プロファイル (px , Px) を選択した場合、子プロセス側のプログラムでは、独自のプロファイルを使用して実行します。この場合、親プロセスから子プロセスに対して環境変数を継承するにあたって、危険性のある変数の内容を削除するかどうかを尋ねます。削除するよう選択した場合は、プロファイル内に `Px` が書き込まれ、削除しないように選択した場合は、プロファイル内に `px` が書き込まれます。なお、プロファイル実行モードを選択した場合の既定値は `Px` です。

無制限実行モードは非推奨であり、そのプログラムに対するプロファイルをどうやっても作成することができず、それ以外の選択肢が存在しない場合にのみ選択すべきものです。無制限実行モードを選択すると、警告メッセージが表示されて確認を求められますので、［Yes］を入力して進めてください。また、危険性のある変数の内容を削除するかどうかを尋ねます。削除するよう選択した場合は、プロファイル内に `Ux` が書き込まれ、削除しないように選択した場合は、プロファイル内に `ux` が書き込まれます。なお、無制限実行モードを選択した場合の既定値は `Ux` です。

重要: 無制限での実行について

`ux` や `Ux` の使用は非常に危険です。子プロセスの実行に際して、セキュリティ面でのポリシー強制が一切働かなくなります。

34.7.3.12 aa-unconfined: 保護されていないプロセスの識別

`aa-unconfined` コマンドは、お使いのシステムで開いているネットワークポートを調べ、システム内に読み込まれているプロファイルセットと比較を行い、AppArmor のプロファイルのないネットワークサービスを報告します。このコマンドの実行にあたっては `root` の権限が必要となるほか、AppArmor のプロファイルで制限を受けていない状況下で実行する必要があります。

`aa-unconfined` は `/proc` ファイルシステムから実行ファイルの情報を取得する仕組みであるため、`root` で実行しなければなりません。また、下記のような競合状態が発生する可能性があります:

- コマンドの実行時に削除された実行ファイルがある場合、その扱いを誤ることがあります
- `netstat(8)` を実行してから後続のチェックを行うまでに、プロセスが終了したような場合、その扱いを誤ることがあります



注記

このプログラムは TCP と UDP を使用するプロセスのみを一覧表示します。その他のプロトコルを利用する通信については無視されます。そのため、フォレンジック用途にはお使いいただけません。検証環境内で、ネットワークにアクセスするプロセスに対してプロファイルを作成する用途にのみお使いください。

34.7.3.13 aa-notify

`aa-notify` は、お使いのデスクトップ環境内に AppArmor の通知を表示することのできる、便利なユーティリティです。AppArmor のログファイルを直接調査したくなく、代わりにポリシーに違反した通知のみをデスクトップに表示したい場合に有用です。AppArmor のデスクトップ通知機能を有効化するには、下記のように入力して `aa-notify` を実行します:

```
> sudo aa-notify -p -u ユーザ名 --display ディスプレイ番号
```

ここで、*ユーザ名* には現在ログインしているユーザのユーザ名を、*ディスプレイ番号* には現在使用しているディスプレイのディスプレイ番号 (例: `:0`) をそれぞれ指定します。プログラムは裏で動作する仕組みで、拒否イベントが発生するたびに通知が表示されます。



ヒント

なお、現在使用している X Window のディスプレイ番号は、`$DISPLAY` という環境変数に保存されていますので、`--display $DISPLAY` と入力することで、現在のディスプレイをそのまま使用することができます。

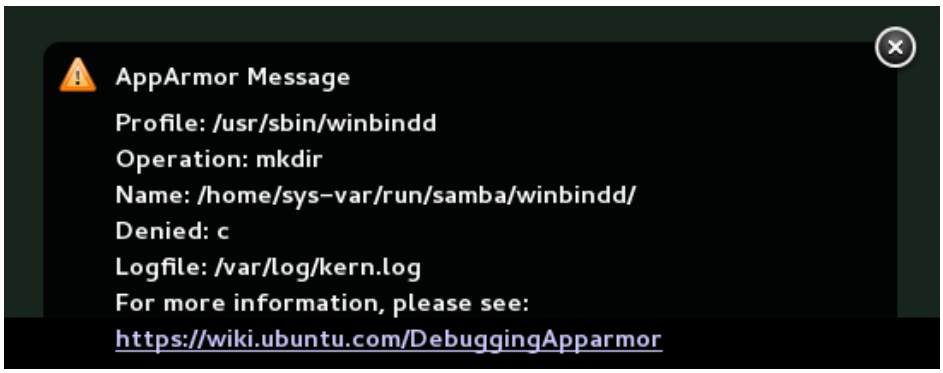


図 34.1: GNOME 内での aa-notify メッセージ

なお、`-s 日数` というオプションを指定すると、指定した過去日数分の通知概要を表示することができます。`aa-notify` の詳細については、`man 8 aa-notify` で表示されるマニュアルページをお読みください。

34.7.3.14 apparmor.vim

vim テキストエディタ向けの文法ハイライト機能を利用することで、AppArmor のプロファイル内にある様々な機能を色付きで表示することができます。vim と vim 向けの AppArmor 文法モードでは、プロファイルの意味を色で識別することができますので、端末ウインドウ内で入力を行いながら内容を確認することができます。

vim で AppArmor のプロファイルを編集している際、文法ハイライト機能を有効にするには、`:syntax on` を実行してから `:set syntax=apparmor` を実行します。また、vim 側でファイルを正しく AppArmor のプロファイルとして認識しているかどうかを調べるには、下記の内容をプロファイルの末尾に入力します:

```
# vim:ft=apparmor
```

"¥n ¥n"



ヒント

なお、`vim` を利用した場合、`/etc/apparmor.d/` ディレクトリ内のファイルを編集すると、自動的に文法ハイライト機能が有効化されます。

この機能を利用した場合は、vim ではプロファイルの内容を下記のように表示します:

青

コメント

白
: 通常の読み込みアクセスの行

茶
: ケーパビリティの項目および不平モードのフラグ

黄
: 書き込み権のあるアクセスの行

緑
: 実行許可 (ix もしくは px) を与えている行

赤
: 無制限アクセス (ux) を設定している行

赤い背景
: 正しく AppArmor モジュールに読み込むことのできない、文法エラーのある行

vim での文法ハイライト機能について、詳しくは `apparmor.vim` および `vim` の各マニュアルページ、もしくは vim エディタ内から `:help syntax` と入力して実行することで表示される、ヘルプをお読みください。また、文法ハイライトに関わる設定は、`/usr/share/vim/current/syntax/apparmor.vim.` 内にあります。

34.8 主なファイル名とディレクトリ

下記の一覧には、AppArmor フレームワークで使用される最も重要なファイルとディレクトリが列挙されています。プロファイルを手作業で管理もしくはトラブルシューティングしたい場合は、下記のファイルやディレクトリについて知っておく必要があります:

`/sys/kernel/security/apparmor/profiles`
: 現在読み込まれているプロファイルセットを表示することのできる、仮想ファイルです。

`/etc/apparmor/`
: AppArmor の設定ファイルの配置場所です。

`/usr/share/apparmor/extra-profiles`
: AppArmor に同梱されているプロファイルのローカルリポジトリです。既定では有効化されていません。

`/etc/apparmor.d/`

プロファイルを保存するためのディレクトリです。なお、パス内の `/` を `.` に置き換えたファイル名 (ただしルートディレクトリを表す冒頭の `/` は除きます) で作成するのが慣習となっています。これにより、プロファイルのファイル名を分かりやすくしています。たとえば `/usr/sbin/smbd` というプログラム向けのプロファイルは、`usr.sbin.smbd` というファイル名になっています。

`/etc/apparmor.d/abstractions/`

抽象を保存しておくためのディレクトリです。

`/etc/apparmor.d/program-chunks/`

プログラムチャンクを保存しておくためのディレクトリです。

`/proc/*/attr/current`

このファイルの内容を出力することで、そのプロセスに対する制限状況と、適用されているプロファイルを表示することができます。また、`ps auxZ` のように入力して実行すると、この情報を自動的に取得することができます。

35 チェンジハット機能による Web アプリケーションのプロファイル作成

改訂履歴

2025-03-17

AppArmor® のプロファイルは本来、個別のプログラムやプロセスに対するセキュリティポリシーを表すものです。つまり、実行可能なプログラムに対してプロファイルを割り当てることになります。ところが、プログラムの一部では他の部分とは異なるアクセス許可が必要となることもあります。このような場合は、「チェンジハット」の機能を利用して、メインのプログラムに対して適用されているものとは異なるセキュリティコンテキストを適用するように設定します。このような「異なるセキュリティコンテキスト」のことを、ハット や サブプロファイル などと呼び、これを変更する行為を チェンジハット と呼びます。

チェンジハット機能は、AppArmor のプロファイルの中からプログラムに対して ハット を変更する機能を提供します。これにより、プロセスごとの制限よりもずっと精密なセキュリティを設定することができます。また、この機能はそれぞれのアプリケーションが「チェンジハット対応」として作られている必要があります。これはアプリケーションの実行時、特定のタイミングで AppArmor のモジュールに要求を送信して、ドメインを切り替える処理のことを指します。チェンジハット対応のアプリケーションの例としては、Apache Web サーバなどがあります。

1 つのプロファイルに対しては、任意の数のサブプロファイルを作成することができますが、2 段階までしか対応していないことに注意する必要があります。つまり、サブプロファイルから更に子となるプロファイルを作成することはできません。また、サブプロファイルは個別のプロファイルとして作成する必要があります。サブプロファイルの名前は、元々のプロファイル名の末尾に `^` を付け、その後ろにサブプロファイルそのものの名前を指定します。

サブプロファイルは親のプロファイル内に含まれていてもかまいませんし、個別のファイルとして存在していてもかまいません。特に多数のハットを必要とするようなサイトであれば、後者のほうがお勧めです。これにより、ハット単位でポリシーキャッシュ機能が働くため、サブプロファイル側の変更を行った場合に処理を減らすことができるためです。逆にすべてのハットが親のプロファイル内に含まれている場合、親プロファイル全体を再コンパイルしなければならなくなります。

ハットとして使用する予定の外部サブプロファイルがある場合、そのサブプロファイルの冒頭には `hat` もしくは `^` 文字で始まらなければなりません。

下記の 2 つのサブプロファイルは、いずれもハットとして使用 できない ものです:

```
/foo//bar { }
```

"¥n ¥n"

```
profile /foo//bar { }
```

逆に、下記の 2 つはいずれもハットとして扱われます:

```
^/foo//bar { }
```

"¥n ¥n"

```
hat /foo//bar { } # (vim では、この書式はハイライト表示されません)
```

なお、ハットを使用する場合のセキュリティは、完全なプロファイルを使用する場合に比べると、比較的弱くなることに注意してください。プログラム内に特定の種類のバグが存在した場合、攻撃者はプロファイル内に含まれるハットに退避することができてしまいます。これは、ハットのセキュリティがプロセス側で保持される機密鍵のみによって成り立っているためで、ハット側内のコードからは鍵に対するアクセスを許可してはならないことによるものです。そのためチェンジハット機能は、プロセス内に含まれるメモリに対して直接的にアクセスする手段のない、インタプリタ型の言語 (Perl, PHP, Java など) で作成されたアプリケーションサーバに対して、最も便利な仕組みとなっています。

本章の以下の項目では、Apache でのチェンジハット機能の使用方法について説明しています。この Apache では `mod_perl` および `mod_php` を利用して、Web サーバコンポーネントを動作させているものとします。同様のアプローチで、Apache における `mod_apparmor` と同様 (詳しくは 35.1.2項「Location ディレクティブと Directory ディレクティブ」で説明しています) に、その他のアプリケーションサーバでもチェンジハット機能を利用することができます。



ヒント: さらなる情報

更に詳しい情報をご希望の場合は、`change_hat` のマニュアルページをお読みください。

35.1 mod_apparmor を利用するための Apache の設定

AppArmor には、Apache Web サーバ 向けの `mod_apparmor` (`apache2-mod-apparmor` パッケージ) が提供されています。このモジュールを使用することで、Apache Web サーバをチェンジハット対応にすることができます。Apache Web サーバと共にインストールしてください。

Apache がチェンジハット対応になることで、受信したそれぞれの URI に対して、下記のような AppArmor セキュリティプロファイルが存在していないかどうかを調べます。

- URI 固有のハット。例: `^www_app_name/templates/classic/images/bar_left.gif`
- `DEFAULT_URI`
- `HANDLING_UNTRUSTED_INPUT`

注記: Apache の設定について

`apache2-mod-apparmor` をインストールしたあとは、モジュールを有効化してから、下記のように実行して Apache を再起動します:

```
> a2enmod apparmor && sudo systemctl reload apache2
```

Apache は純粋なテキスト形式の設定ファイルにディレクティブを配置することで、設定を行うことができます。メインとなる設定ファイルは `/etc/apache2/httpd.conf` です。Apache を独自にコンパイルして使用している場合は、コンパイル時の設定にあわせてファイルを選択してください。ディレクティブは Apache の動作を変更するためのもので、設定ファイル内の任意の場所に配置することができます。メインの設定ファイルを変更したあとは、`sudo systemctl reload apache2` を実行することで、変更点を反映させることができます。

35.1.1 仮想ホストのディレクティブ

<VirtualHost> と </VirtualHost> のディレクティブを利用して、特定の仮想ホストにのみ適用されるディレクティブの範囲を指定します。Apache における仮想ホストの設定方法について、詳しくは https://httpd.apache.org/docs/2.4/mod/core.html#virtualhost をお読みください。

チェンジハット機能固有の設定キーワードは `AADefaultHatName` です。これは `AAHatName` に似た設定で、`AADefaultHatName My_Funky_Default_Hat` のように指定します。

これを設定することで、仮想ホストやその他の Apache サーバディレクティブに対して、使用すべき既定のハットを設定することができます。もちろん仮想ホストごとに異なる既定のハットを設定することもできます。この設定は `AAHatName` で上書きすることができます。逆に言えば、該当する `AAHatName` や URI でのハットに当てはまるものが存在しない場合の、既定のハットとなります。`AADefaultHatName` でのハット指定が存在しない場合は、`DEFAULT_URI` ハット (存在すれば) を使用するようになります。

全くどのハットも設定されていない場合は、「親」プロセスである Apache のハットが適用されます。

35.1.2 Location ディレクティブと Directory ディレクティブ

設定ファイル内の Location および Directory のディレクティブの中でハット名を指定することで、Apache が URI やディレクトリを処理する際、指定したセキュリティプロファイルを適用するようになります。Apache における Location および Directory ディレクティブの使い方について、詳しくは https://httpd.apache.org/docs/2.4/sections.html をお読みください。

下記に示す Location ディレクティブでは、指定した場所に対して、`mod_apparmor` が特定のハットを使用するように指定しています:

```
<Location /foo/>
  AAHatName MY_HAT_NAME
</Location>
```

上記のように設定することで、`/foo/` で始まる URI (`/foo/` , `/foo/bar` , `/foo/cgi/path/blah_blah/blah` など) にアクセスが行われた際、`MY_HAT_NAME` というハットを使用するようになります。

Directory ディレクティブも上記と同様ですが、こちらのディレクティブはファイルシステム内のディレクトリを指定する点が異なることに注意してください:

```
<Directory "/srv/www/www.example.org/docs">
  # 末尾にスラッシュが無いことに注意
  AAHatName example.org
</Directory>
```

35.2 チェンジハット対応のアプリケーションの管理

前の章では `mod_apparmor` に関する情報と、特定の Web アプリケーションのセキュリティを向上させる方法について学んできました。この章では、Web アプリケーションに対してハットを作成するための現実的な例と、AppArmor のチェンジハット機能の使用方法について説明しています。なお、本章では AppArmor のコマンドラインツールを利用して説明を行っていますが、これは YaST の AppArmor モジュールではすべての機能を提供しているわけではないためのものです。

35.2.1 AppArmor のコマンドラインツールによる方法

本章では、例として Adminer (https://www.adminer.org/ja/) と呼ばれる Web アプリケーションのセキュリティを向上させます。これは PHP で書かれた完全機能の SQL データベース管理ツールで、たった 1 つの PHP ファイルだけで作られています。Adminer を動作させるには、まず Apache Web サーバと PHP のほか、PHP の Apache モジュールと PHP に対応したデータベースドライバを用意する必要があります。今回は MariaDB を利用することにします。必要なパッケージをインストールするには、下記のように入力して実行します:

```
zypper in apache2 apache2-mod_apparmor apache2-mod_php5 php5 php5-mysql
```

Adminer を動作させるための Web 環境の構築は、下記のようにして行います:

手順 35.1: WEB サーバ環境の設定

1. まずは Apache 向けの `apparmor` と `php5` のモジュールを有効化します。有効化を行うには、下記のように実行します:

   ```
   > a2enmod apparmor php5
   ```

   あとは Apache を再起動します:

   ```
   > sudo systemctl restart apache2
   ```

2. MariaDB が動作していることを確認します。動作しているかどうかが分からない場合は、下記を実行して再起動します:

   ```
   > sudo systemctl restart mariadb
   ```

3. https://www.adminer.org から Adminer をダウンロードして、`/srv/www/htdocs/adminer/` にコピーしたあと、ファイル名を `adminer.php` に変更します。スクリプトのフルパスは `/srv/www/htdocs/adminer/adminer.php` であることになります。

4. Web ブラウザを起動して URI 欄に `http://localhost/adminer/adminer.php` と入力し、Adminer の動作確認を行います。Adminer をリモートのサーバにインストールしている場合は、`localhost` の部分をサーバのホスト名に置き換えてください。



図 35.1: ADMINER のログインページ



ヒント

Adminer のログインページの表示時に何らかの問題が発生した場合は、まず `/var/log/apache2/error.log` にある Apache のエラーログファイルをご覧になることをお勧めします。Web ページにアクセスできない理由として考えられる理由としては、既に Apache が AppArmor の制御下にあり、適用されているプロファイルが Adminer を動作させるには厳しすぎることも考えられます。この場合は、`aa-status` を実行して確認を行い、必要であれば下記を実行し、Apache を一時的に不平モードに設定してください:

```
# sudo aa-complain usr.sbin.httpd2-prefork
```

Adminer を動作させるための Web 環境が整ったら、次に Apache の `mod_apparmor` を設定します。これにより、AppArmor が Adminer に対するアクセスを検知して、固有の「ハット」に対する変更を行うことができるようになります。

手順 35.2: mod_apparmor の設定

1. Apache には、`/etc/apache2/` と `/etc/apache2/conf.d/` という設定ファイルのディレクトリが存在しています。どちらのディレクトリ内でもかまいませんので、テキストエディタを利用してファイルを作成してください。この例では、`vim` エディタを利用して設定ファイル `/etc/apache2/conf.d/apparmor.conf` を作成することにします。

   ```
   > sudo vim /etc/apache2/conf.d/apparmor.conf
   ```

2. テキストエディタ内で、下記のとおり入力します。

   ```
   <Directory /srv/www/htdocs/adminer>
     AAHatName adminer
   </Directory>
   ```

   上記の設定を追加することで、Apache に対して `/adminer` のディレクトリ (およびそれ以下のサブディレクトリやファイル) 宛のアクセスがあった際、AppArmor 側でチェンジハットを行うことを指示します。このディレクトリ内には、既に `adminer.php` があるはずです。

3. ファイルを保存して、テキストエディタを終了します。あとは下記のように実行して Apache を再起動します:

   ```
   > sudo systemctl restart apache2
   ```

これで Apache に対する Adminer の設定が完了し、Adminer の実行時には「ハット」を変更することができるようになります。あとは AppArmor の設定で、Adminer 向けのハットを作成することになります。この時点までに AppArmor のプロファイルを作成していない場合は、以下の手順を実施する前に作成しておいてください。Apache のメインバイナリは `/usr/sbin/httpd2-prefork` ですから、対応するプロファイルは `/etc/apparmor.d/usr.sbin.httpd2-prefork` というファイル名で存在しているはずです。

手順 35.3: ADMINER 向けのハットの作成

1. テキストエディタを利用して、`/etc/apparmor.d/usr.sbin.httpd2-prefork` というファイル名のファイルを開いて (存在していない場合は作成して) ください。ファイルの内容は下記のようになっているはずです:

```
#include <tunables/global>

/usr/sbin/httpd2-prefork {
  #include <abstractions/apache2-common>
  #include <abstractions/base>
  #include <abstractions/php5>

  capability kill,
  capability setgid,
  capability setuid,

  /etc/apache2/** r,
  /run/httpd.pid rw,
  /usr/lib{,32,64}/apache2*/** mr,
  /var/log/apache2/** rw,

  ^DEFAULT_URI {
    #include <abstractions/apache2-common>
    /var/log/apache2/** rw,
  }

  ^HANDLING_UNTRUSTED_INPUT {
    #include <abstractions/apache2-common>
    /var/log/apache2/** w,
  }
}
```

2. 末尾の閉じ中括弧 (`}`) の前に、下記のような行を挿入します:

```
^adminer flags=(complain) {
}
```

ここでは、ハット名の後ろに `(complain)` を指定していることに注意してください。これにより、AppArmor での `adminer` ハットを不平モードに設定することになります。これは、この時点では Adminer のアクセス要件がわかっていないため、それを調べるための設定です。

3. ファイルを保存して AppArmor を再起動し、Apache を再起動します。

```
> sudo systemctl reload apparmor apache2
```

4. 続いて `adminer` ハットが正しく不平モードになっていることを確認します。

```
> sudo aa-status
apparmor module is loaded.
39 profiles are loaded.
37 profiles are in enforce mode.
[...]
   /usr/sbin/httpd2-prefork
   /usr/sbin/httpd2-prefork//DEFAULT_URI
   /usr/sbin/httpd2-prefork//HANDLING_UNTRUSTED_INPUT
[...]
2 profiles are in complain mode.
   /usr/bin/getopt
   /usr/sbin/httpd2-prefork//adminer
[...]
```

上記のとおり、`httpd2-prefork//adminer` が不平モードになっていることがわかります。

あとは `adminer` ハットに対して、必要なアクセス権限を見つけ出していく作業になります。この時点では `adminer` ハットは不平モードに設定されているため、ログ内にポリシーへの違反が記録されているものの、実際の拒否は発動されていません。そのため、Web ブラウザから `adminer.php` の機能を一通り試して、必要な権限を収集していってください。収集が終わったら、`aa-logprof` を実行することで、ハットのプロファイルの作成支援を行います。

手順 35.4: `adminer` ハット向けのルールの生成

1. Adminer を Web ブラウザで開きます。ローカルにインストールした場合は、URI は `http://localhost/adminer/adminer.php` になるはずです。
2. まずは使用するデータベースエンジン (ここでは MariaDB) を選択し、既存のデータベースにユーザ名とパスワードを入力してログインします。なお、データベース名については、ログイン後に指定することもできますので、指定する必要はありません。あとは Adminer の機能を一通り試してください。新しいデータベースを作成したり、新しいテーブルを作成したり、ユーザ権限を設定したりなどです。
3. ひととおり Adminer のユーザインターフェイスをテストしたら、コンソールに戻って収集したログの調査を行います。

```
> sudo aa-logprof
Reading log entries from /var/log/audit/audit.log.
Updating AppArmor profiles in /etc/apparmor.d.
Complain-mode changes:

Profile:  /usr/sbin/httpd2-prefork^adminer
Path:     /dev/urandom
Mode:     r
Severity: 3

  1 - #include <abstractions/apache2-common>
[...]
 [8 - /dev/urandom]

[(A)llow] / (D)eny / (G)lob / Glob w/(E)xt / (N)ew / Abo(r)t / (F)inish / (O)pts
```

`aa-logprof` のメッセージによると、下記のとおり `adminer` のハットが正しく認識されていることが分かります:

```
Profile:  /usr/sbin/httpd2-prefork^adminer
```

`aa-logprof` は、検出されたそれぞれの AppArmor イベントから、適切なルールを選択するよう依頼します。使用したいルールを指定したあと［Allow］を選択します。`aa-genprof` と `aa-logprof` の使用方法について、詳しくは 34.7.3.8項「aa-genprof: プロファイルの生成」をお読みください。



ヒント

`aa-logprof` では、特定のイベントに対して複数のルール候補を提示して選択を求めます。場合によっては 抽象 と呼ばれる、特定の目的向けのルールセットをまとめた仕組みを提案することもあります。このような場合は、直接のルールではなく抽象を選択したほうが便利な場合があります:

```
 1 - #include <abstractions/php5>
[2 - /var/lib/php5/sess_3jdmii9cacj1e3jnahbtopajl7p064ai242]
```

上記の例では、［1］を押して［A］を押すことで、抽象の取り込みを設定することができます。

4. 最後に、下記のとおり保存を行うかどうかを尋ねられます。

```
The following local profiles were changed. Would you like to save them?
 [1 - /usr/sbin/httpd2-prefork]
```

```
 (S)ave Changes / [(V)iew Changes] / Abo(r)t
```

［S］を押して変更を保存してください。

5. 変更が終わったら、`aa-enforce` を実行してプロファイルを強制モードに設定します:

```
> sudo aa-enforce usr.sbin.httpd2-prefork
```

`aa-status` で状態を確認します:

```
> sudo aa-status
apparmor module is loaded.
39 profiles are loaded.
38 profiles are in enforce mode.
[...]
   /usr/sbin/httpd2-prefork
   /usr/sbin/httpd2-prefork//DEFAULT_URI
   /usr/sbin/httpd2-prefork//HANDLING_UNTRUSTED_INPUT
   /usr/sbin/httpd2-prefork//adminer
[...]
```

上記のとおり、`//adminer` のハットが complain (不平) から enforce (強制) モードに切り替えられています。

6. あとは再度 Adminer を Web ブラウザで開いて動作を確認してください。何らかの問題を見つけた場合は、不平モードに戻して手順を繰り返し、`aa-logprof` でプロファイルを更新してください。

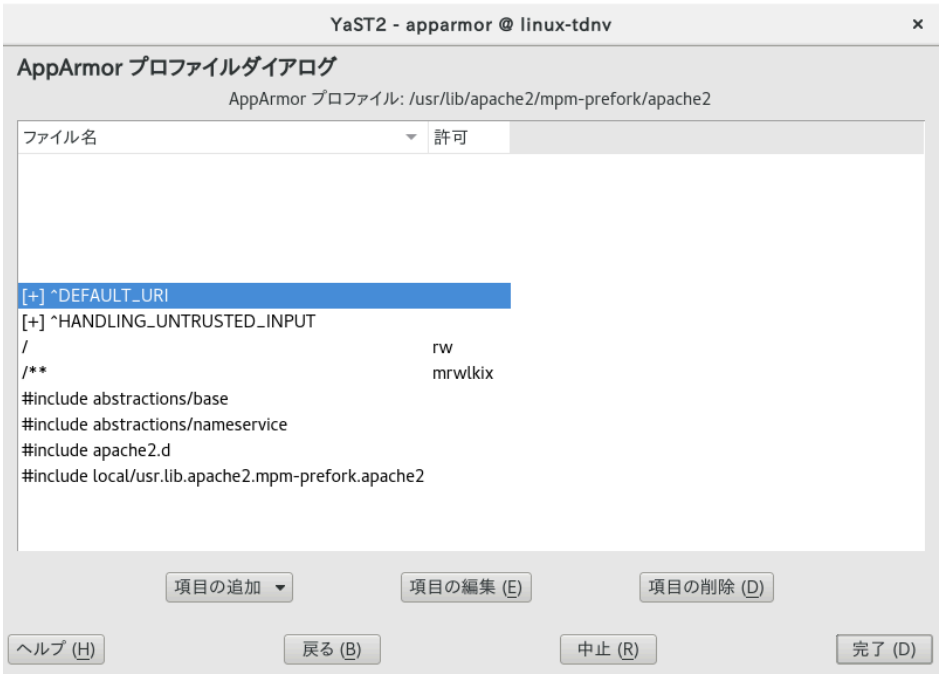


注記: ハットと親プロファイルの関係性について

`^adminer` プロファイルは、親プロファイルである `usr.sbin.httpd2-prefork` の中で動作するプロセスから見た場合にのみ利用することができます。

35.2.2 YaST によるハットと項目の追加方法

［既存のプロファイルを管理］(手順に関する詳細は 33.2項「プロファイルの編集」を参照) を使用するか、もしくは［手作業でプロファイルを追加］(手順に関する詳細は 33.1項「プロファイルの手動追加」を参照) を使用することで、AppArmor のプロファイル内にハット (サブプロファイル) を追加することができます。［AppArmor プロファイルダイアログ］ウインドウ内でチェンジハットのサブプロファイルを追加するには、下記のようにして行います。



1. ［AppArmor プロファイルダイアログ］のウインドウ内で、［項目の追加］を押し、［ハット］を選択します。［追加するハット名］のダイアログが表示されます:



2. AppArmor プロファイルに追加したいハットの名前を入力します。名前はアクセス時の URI を指定します。
3. ［ハットの作成］を押します。すると［AppArmor プロファイルダイアログ］画面に戻ります。
4. 新しいハットを追加したら、［完了］を押して閉じます。

36 pam_apparmor によるユーザの制限

改訂履歴

2022-08-29

AppArmor のプロファイルは実行ファイルに対して適用されるもので、特定のプログラム内の機能が、ほかの箇所とは異なるアクセス許可を必要とする場合は、チェンジハットの機能を利用して、異なる役割であるハット (サブプロファイルとも呼ばれます) に切り替える処理を行います。`pam_apparmor` という PAM モジュールでは、グループ名やユーザ名を基準にしたり、既定のプロファイルを適用したりして、認証済みのユーザを制限する機能を提供します。この機能を利用するには、`pam_apparmor` を PAM のセッションモジュールとして登録する必要があります。

なお、`pam_apparmor` は既定ではインストールされません。インストールを行いたい場合は、YaST もしくは `zypper` をお使いください。`pam_apparmor` の設定方法については、パッケージをインストールした際に配置される `/usr/share/doc/packages/pam_apparmor/README` をお読みください。PAM に関する詳細は、第2章「PAM を利用した認証」をお読みください。

37 プロファイルを作成したアプリケーションの管理

改訂履歴

2024-01-10

AppArmor® でアプリケーションのプロファイルを作成して予防接種を実施し、プロファイルをメンテナンス (ログファイルの分析からプロファイルの更新、プロファイルのバックアップや最新状態への維持) することで、openSUSE® Leap は非常に効率的にシステムを堅牢にすることができます。また、電子メールへのイベント通知を設定し、aa-logprof ツールを利用してログを調査し、プロファイルを更新することで、様々な問題が顕在化する前に対応を取ることができます。

37.1 アクセス拒否イベントへの対応

何らかのアクセス拒否イベントが見つかった場合、まずはその内容がセキュリティ上の脅威によるものなのか、もしくは通常のアプリケーションの動作として発生したものなのかを判断します。この判断にあたっては、アプリケーション固有の知識が必要となります。もしもアクセス拒否イベントが通常のアプリケーション動作として発生したものであった場合は、コマンドラインで `aa-logprof` を実行してください。

アクセス拒否イベントが通常のアプリケーション動作では無いものである場合は、AppArmor の仕組みで未然に防止することができたものの、それは悪意のある侵入である可能性があります。このような状況が発生した場合は、組織内のセキュリティ責任者に連絡を取って、対応を検討することをお勧めします。

37.2 セキュリティプロファイルの管理

本番環境においては、配置済みのすべてのアプリケーションに対して、プロファイルをメンテナンスするよう計画すべきです。セキュリティポリシーの策定と、それに伴うプロファイルの作成を、配置作業の一環として実施すべきです。また、セキュリティポリシーファイルのバックアップと復元のほか、お使いの環境におけるソフトウエア仕様変更の計画や、それに伴うセキュリティポリシーの修正についても、検討を行うべきです。

37.2.1 セキュリティプロファイルのバックアップ

プロファイルをバックアップしておくことで、ディスク障害が発生してもプログラムのプロファイルを作り直す必要が無くなります。また、プロファイルを変更した場合でも、バックアップから復元することで、容易に以前の状態に戻ることができます。

ポリシーのバックアップ作業は、特定のディレクトリ内にプロファイルをコピーするだけです。

1. まずはプロファイルを 1 つのファイルにまとめます。これを行うには、端末ウインドウを開いて、`root` になって下記を実行します:

   ```
   > sudo tar zclpf profiles.tgz /etc/apparmor.d
   ```

   また、システムを定期的にバックアップしている場合は、バックアップ対象のディレクトリに `/etc/apparmor.d` を追加しておくことで、お使いのバックアップ内にプロファイルを入れておくことができます。

2. `scp` や Nautilus のようなファイルマネージャを使用することで、他のストレージメディアやネットワーク、その他のコンピュータなどにコピーすることもできます。

37.2.2 セキュリティプロファイルの変更

セキュリティプロファイルのメンテナンス作業には、アプリケーションに対するセキュリティを修正してシステムを適切に保護する作業が含まれます。AppArmor 内でプロファイルを変更する方法について、詳しくは 33.2項「プロファイルの編集」をお読みください。

37.2.3 お使いの環境に対する新規ソフトウエアの導入

システム内にインストールされているアプリケーションのバージョンを上げた場合や、修正を適用した場合、お使いの要件にあわせてプロファイルを更新する必要があります。更新方法にはいくつかのものがありますが、組織内の配置戦略にあわせて決定してください。また、テスト環境や本番環境に対して、修正やアップグレードを配置することもできます。以下では、これらそれぞれの方法について、どのようにすべきなのかについて説明しています。

テスト環境に修正やアップグレードを適用しようとしている場合は、端末を開いて `root` になり、`aa-logprof` を実行してプロファイルを更新するのが最適な方法です。詳しい手順については 34.7.3.9項「aa-logprof: システムログのスキャン」をお読みください。

本番環境に直接修正やアップグレードを適用しようとしている場合は、`aa-logprof` を利用してシステムを頻繁に監視し、プロファイルに更新すべき内容が無いかどうかを細かく調べて、必要であれば追加するのが最適な方法です。詳しい手順については 34.7.3.9項「aa-logprof: システムログのスキャン」をお読みください。

38 サポート

改訂履歴

2023-12-22

本章では、メンテナンスに関連する作業を説明しています。AppArmor® の更新方法のほか、AppArmor が提供するコマンドラインツールを使用するための基礎情報となる、マニュアルページの一覧取得方法などを説明しています。また、トラブルシューティングの章では、AppArmor で一般的に直面する様々な問題や、その解決方法を説明しています。このほか、AppArmor に対する欠陥報告や機能拡張のリクエスト方法などについても、説明しています。

38.1 AppArmor のオンライン更新

AppArmor 自身の更新は、openSUSE Leap での他のパッケージの更新と同様の形態で提供されます。openSUSE Leap に同梱されているその他のパッケージと同様の手段で取得し、適用を行ってください。

38.2 マニュアルページの使用

様々なコマンドに対してマニュアルページが用意されています。AppArmor のマニュアルページをお読みになりたい場合は、端末を開いて `man apparmor` と入力し、実行してください。マニュアルページはセクション 1 から 8 までに分類され、セクションはマニュアルページの分類として機能しています:

表 38.1: マニュアルページ: セクションと分類

セクション	分類
1	ユーザコマンド
2	システムコール
3	ライブラリ関数
4	デバイスドライバの情報
5	設定ファイルの書式
6	ゲーム
7	高レベルな考え方の説明

セクション	分類
8	管理者向けコマンド

セクション番号は、同じ名前のマニュアルページを識別する方法として使用されています。たとえば `exit(2)` であれば exit という名前のシステムコールに関する説明を、`exit(3)` であれば exit という名前の C 言語ライブラリ関数に関する説明が書かれています。

AppArmor のマニュアルページは下記のとおりです:

- `aa-audit(8)`
- `aa-autodep(8)`
- `aa-complain(8)`
- `aa-decode(8)`
- `aa-disable(8)`
- `aa-easyprof(8)`
- `aa-enforce(8)`
- `aa-enxec(8)`
- `aa-genprof(8)`
- `aa-logprof(8)`
- `aa-notify(8)`
- `aa-status(8)`
- `aa-unconfined(8)`
- `aa_change_hat(8)`
- `logprof.conf(5)`
- `apparmor.d(5)`
- `apparmor.vim(5)`
- `apparmor(7)`
- `apparmor_parser(8)`
- `apparmor_status(8)`

38.3 さらなる情報

AppArmor に関して更に詳しい情報を見つけたい場合は、https://wiki.apparmor.net にある AppArmor の Wiki (英語) をご覧になることをお勧めします。また、AppArmor に関する製品ドキュメンテーションをお読みになりたい場合は、パッケージをインストールしたあと `/usr/share/doc/manual` ディレクトリ内をお探しください。

このほか、ユーザが投稿できる AppArmor 向けメーリングリストや、開発者と対話するためのメーリングリストも用意されています。詳しくは https://lists.ubuntu.com/mailman/listinfo/apparmor (英語) をお読みください。

38.4 トラブルシューティング

本章では、AppArmor を利用していてよく発生する問題やエラーメッセージについて説明しています:

38.4.1 おかしなアプリケーション動作に対する対応

不審なアプリケーションの動作が見つかった場合や、何らかの問題が見つかった場合、まずはログファイル内を調査して、AppArmor がアプリケーションの動作を阻害していないかどうかを調べてください。お使いのアプリケーションでの正常な動作で、AppArmor のプロファイルが厳しく制限しすぎていることが判明した場合は、アプリケーションの動作にあわせてプロファイルを更新し、適切に動作できるように設定してください。この作業は `aa-logprof` (34.7.3.9項「aa-logprof: システムログのスキャン」) で実施することができます。

AppArmor の保護を外してアプリケーションやサービスを動作させたい場合は、`/etc/apparmor.d` ディレクトリ内にあるアプリケーションのプロファイルを削除するか、別の場所に移動させてください。

38.4.2 プロファイルが期待通りに動作しない場合の対応

以前のバージョンの AppArmor を使用していて、プロファイルを維持したままシステムを更新すると、以前は完璧に動作していたにもかかわらず、正しく動作しなくなってしまったり、全く動作しなくなったりすることがあります。

本バージョンの AppArmor では、プロファイルの書式に影響する新しい機能と、古いバージョンの AppArmor プロファイルでは問題が発生する可能性のある AppArmor ツールが含まれています。具体的には下記のとおりです:

- ファイルロック (施錠) 関連
- ネットワークアクセス制御

- SYS_PTRACE のケーパビリティ
- ディレクトリパスへのアクセス

現在のバージョンの AppArmor ではファイルロックの権限についても制限を行うことができるようになり、対応する許可モードとして (k) が新設されています。ファイルロックを要求するアプリケーションがある場合、古いプロファイルをそのまま実行してしまうと、明示的な許可が無いことから、正しく動作しなかったり、エラーが発生したりすることがあります。このような現象に直面した場合は、まず /var/log/audit/audit.log をご覧ください。この問題であれば、下記のような項目が出力されているはずです:

```
type=AVC msg=audit(1389862802.727:13939): apparmor="DENIED" \
operation="file_lock" parent=2692 profile="/usr/bin/opera" \
name="/home/tux/.qt/.qtrc.lock" pid=28730 comm="httpd2-prefork" \
requested_mask="::k" denied_mask="::k" fsuid=30 ouid=0
```

この場合は、上述の手順で aa-logprof を使用し、プロファイルを更新してください。

また、新しいネットワークアクセス制御の書式では、ネットワークファミリと種類の指定を行うようになっています (詳しくは 31.5項「ネットワークアクセス制御」をお読みください) 。この機能追加により、アプリケーションの動作が阻害されたり、全く動作しなくなったりすることがあります。ネットワーク関連のアプリケーションの動作が不審である場合は、まず /var/log/audit/audit.log をご確認ください。この問題であれば、下記のような項目が出力されているはずです:

```
type=AVC msg=audit(1389864332.233:13947): apparmor="DENIED" \
operation="socket_create" family="inet" parent=29985 profile="/bin/ping" \
sock_type="raw" pid=30251 comm="ping"
```

上記のログ項目は /bin/ping での出力例ですが、このログによると AppArmor のネットワーク接続機能の許可獲得に失敗しています。アプリケーションがネットワークアクセスを行う場合、この許可は明示的に設定しなければなりません。新しい書式にあわせてプロファイルを更新するには、上述のとおり aa-logprof を利用して更新してください。

現在のカーネルでは、/proc/PID/fd/* 内のファイルにアクセスする場合、SYS_PTRACE のケーパビリティを必要とするようになっています。新しいプロファイルではファイルとケーパビリティの両方を設定しますが、古いプロファイルではファイルの設定しか行っていません。たとえば下記のようになっているはずです:

```
/proc/*/fd/**  rw,
```

新しいカーネルを使用する場合は、下記のように修正してください:

```
capability SYS_PTRACE,
```

```
/proc/*/fd/**  rw,
```

新しい書式にあわせてプロファイルを更新するには、YaST のプロファイル更新ウイザードを使用するか、上述の `aa-logprof` コマンドをお使いください。

このバージョンの AppArmor では、プロファイル内のルール書式に対していくつかの変更が加えられています。これにより、ディレクトリとファイルの各アクセスを区別しやすくなっています。そのため、以前のバージョンではファイルとディレクトリの両方にマッチするルールであっても、現在のバージョンではファイルにのみマッチするようになっていることがあります。このような理由により、AppArmor が必要なディレクトリへのアクセスを拒否してしまい、アプリケーションが正しく動作しなくなってしまうことがあります。下記の例では、パスの書式に関する変更点の概要を説明しています。

古い書式では、下記のルールは `/proc/net` 以下のファイルとディレクトリの両方に対してアクセスを許可していました。またディレクトリに対しては、ディレクトリ内のファイルの読み込みアクセスのみが許可され、その中のファイルやディレクトリへのアクセスは与えられていませんでした。たとえば `/proc/net/dir/foo` はアスタリスク (*) でマッチしますが、`foo` は `dir` 以下のファイルもしくはディレクトリであるため、アクセスを与えられていないことになります。

```
/proc/net/*  r,
```

新しい書式で同じような許可を設定したい場合は、2 つの項目に分けて設定する必要があります。1 つめのルールで `/proc/net` 以下のファイルに対するアクセス許可を設定し、2 つめのルールで `/proc/net` 以下のディレクトリに対してアクセス許可を設定します。ディレクトリへのアクセスは、その中にある項目の一覧を取得するために使用するもので、ディレクトリ内にあるファイルやディレクトリに対する許可にはなりません。

```
/proc/net/*  r,
/proc/net/*/  r,
```

なお、下記のルールは古い書式でも新しい書式でも同じように動作するルールで、`/proc/net` 以下のファイルとディレクトリの両方に対してアクセス許可を設定しています (ただし、`/proc/net/` 自身の内容一覧は許可していません):

```
/proc/net/**  r,
```

上述の新しい書式でファイルとディレクトリのアクセスを区別したい場合は、下記のように 2 つのルールに分割してください。1 つめのルールは `/proc/net` 以下のディレクトリに対するアクセス許可を再帰的に設定していて、2 つめのルールはファイルに対するアクセス許可を再帰的に設定しています。

```
/proc/net/**/  r,
/proc/net/**[^/]  r,
```

また、下記の例は古い書式でも新しい書式でも同じように動作するルールで、`/proc/net` 以下の `foo` で始まるファイルやディレクトリに対して、許可を設定しています:

```
/proc/net/foo**  r,
```

新しい書式でファイルアクセスとディレクトリアクセスを区別し、`**` というグロブパターンを使用したい場合は、下記のように 2 つのルールに分割してください。1 つめのルールでは古い書式でファイルとディレクトリの両方にマッチしますが、新しい書式では末尾にスラッシュがないことから、ファイルにのみマッチします。2 つめのルールは、古い書式ではファイルにもディレクトリにもマッチしませんが、新しい書式ではディレクトリにのみマッチします:

```
/proc/net/**foo  r,
/proc/net/**foo/  r,
```

また、下記のルールでは `?` のグロブパターンを使用していますが、こちらの使用方法についても変更がされています。古い書式では、1 つめのルールはファイルとディレクトリの両方にマッチします (4 文字で末尾がスラッシュ以外のもの) 。ですが新しい書式では、ファイルにのみマッチします (末尾にスラッシュがないため) 。2 つめのルールは古い書式では何にもマッチしませんが、新しい書式ではディレクトリにのみマッチします。最後のルールは `/proc/net/foo?` 以下の `bar` というファイルにマッチしますが、古い書式ではファイルだけでなく、ディレクトリにもマッチしてしまいます:

```
/proc/net/foo?  r,
/proc/net/foo?/  r,
/proc/net/foo?/bar  r,
```

書式の変更に伴う問題を検出および解決するには、システムの更新後にそれぞれのアプリケーションに対して下記の手順でプロファイルの確認を行います:

1. まずはアプリケーションのプロファイルを不平モードに切り替えます:

   ```
   > sudo aa-complain アプリケーションのパス
   ```

   このように設定することで、プロファイルへの違反は記録されるものの、プロファイルが強制されることはなく、アプリケーションの動作も阻害されなくなります。

2. アプリケーションの動作を一通り試して、必要なリソースに一通りアクセスさせます。

3. あとはアプリケーションの実行時に生成されたログ項目にあわせて、プロファイルを更新します:

   ```
   > sudo aa-logprof アプリケーションのパス
   ```

4. 最後にプロファイルを強制モードに戻します:

```
> sudo aa-enforce アプリケーションのパス
```

38.4.3 Apache での問題解決

Apache に対して新しいモジュール (例: `apache2-mod_apparmor`) を追加したり、設定を変更したりした場合は、Apache のプロファイルを必要に応じて更新する必要があるかもしれません。Apache のプロファイルを更新しないと、正しく起動しなくなってしまったり、正しく Web ページを提供できなくなったりすることもあります。

38.4.4 使用するプロファイルの一覧から特定のプロファイルを除外する方法

`プログラム名` に対するプロファイルを無効化したい場合は、`aa-disable プログラム名` と入力して実行してください。このコマンドは、プロファイルに対するシンボリックリンクを `/etc/apparmor.d/disable/` 内に作成します。プロファイルを再度有効に戻したい場合は、シンボリックリンクを削除して `systemctl reload apparmor` を実行してください。

38.4.5 システムにインストールされていないアプリケーションのプロファイル管理方法

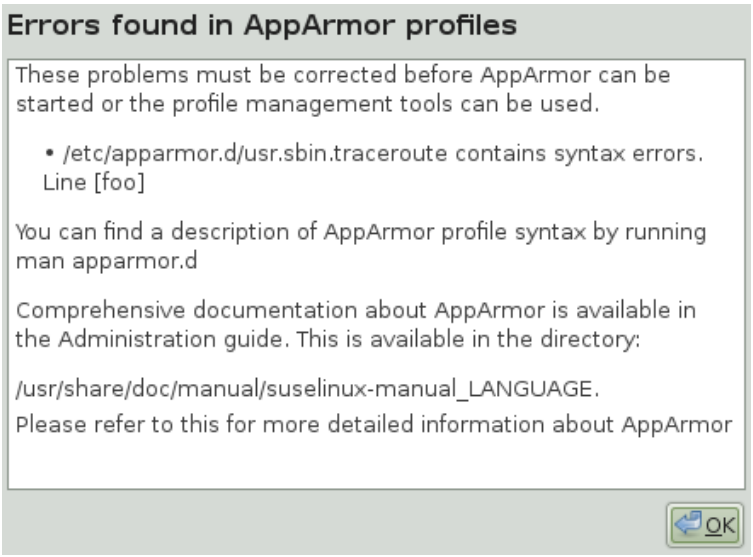
AppArmor でプロファイルを管理するにあたっては、お使いのシステム内で動作していた際のログファイルにアクセスする必要があります。そのため、実際にアプリケーションの動作しているマシンからログを採取できる環境であれば、プロファイル構築用のマシンを動作させる必要はありません。このほかにも、一方のシステムでアプリケーションを動作させておいて、ログ (`audit` がインストールされていれば `/var/log/audit.log`、インストールされていなければ `journalctl | grep -i apparmor > ログファイルのパス`) をプロファイル構築用のホストに転送して、`aa-logprof -f ログファイルのパス` のように実行する方法もあります。

38.4.6 AppArmor の文法エラーの検出および修正方法

AppArmor のプロファイルを手作業で編集していると、書式のミスを起こす可能性があります。AppArmor のプロファイル内に文法エラーがあると、AppArmor の起動時や再起動時にエラーが発生します。下記の例では、全体的なパーサーエラーが発生している様子を示しています:

```
#  systemctl start apparmor.service
Loading AppArmor profiles AppArmor parser error in /etc/apparmor.d/usr.sbin.squid \
 at line 410: syntax error, unexpected TOK_ID, expecting TOK_MODE
Profile /etc/apparmor.d/usr.sbin.squid failed to load
```

AppArmor の YaST ツールを使用することで、プロファイル内に含まれているエラーの詳細と、その修正を促すメッセージをグラフィカルに表示することができます。



文法エラーを修正するには、端末ウインドウで `root` になり、プロファイルを開いてエラーを修正します。修正が終わったら、`systemctl reload apparmor` と入力して実行することで、プロファイルセットを再読み込みさせることができます。



ヒント: `vi` による AppArmor の文法ハイライト機能について

openSUSE Leap で `vi` エディタを使用すると、AppArmor のプロファイル編集時に文法ハイライト機能を使用することができます。この文法ハイライト機能では、書式エラーは赤い背景色で示されます。

38.5 AppArmor のバグ報告

AppArmor の開発者は、この製品を最高の品質に仕上げたいと考えています。フィードバックやバグ報告などがあれば、この品質を更に高めることになります。AppArmor で何らかのバグを見つけた場合は、この製品に対してバグ報告を行ってください 。なお、やり取りはすべて英語にてお願いいたします:

1. Web ブラウザを開いて https://bugzilla.opensuse.org/ に移動し、［Log In］を押します。
2. SUSE アカウントのデータを入力して［Login］を押します。SUSE のアカウントをお持ちでない場合は、［Create Account］を押して、必要な情報を入力してください。
3. 既に同じ問題が報告されている場合は、そのバグレポートを開いて、補足できる情報があれば情報提供を行ってください。
4. これまでに同じ問題が報告されていない場合は、上部のナビゲーションバーから［New］を押して、［Enter Bug］のページに移動してください。
5. バグ報告の対象となる製品を選択します。
6. 製品のバージョン、コンポーネント (この場合は AppArmor) 、ハードウエアプラットフォーム、重要度をそれぞれ指定します。
7. 次に問題点の概要を［Summary］欄に、詳細や再現手順などを［Description］欄にそれぞれ入力します。ログファイルの内容などを貼り付けてもかまいません。また、スクリーンショットやログファイル、テストケースなどの添付を行うこともできます。
8. 必要な項目を入力したら、［Submit］を押すと、レポート内容が開発者に送信されます。

39 AppArmor 用語集

改訂履歴

2023-12-22

抽象

下記の プロファイルファウンデーションクラス をお読みください。

Apache

Apache は無償で利用することのできる Unix ベースの Web サーバです。現時点では、インターネット内で最も使用されている Web サーバです。Apache に関する詳細については、Apache の Web サイト https://www.apache.org をご覧ください。

アプリケーションファイアウオール

AppArmor はアプリケーションを制限し、使用することのできる動作を制限するための仕組みです。不正なプログラムからの攻撃を防ぐため、たとえそれが信頼されているアプリケーションであっても、保護されているコンピュータ内で人手を介することなく権限を制限することができます。

攻撃シグネーチャ

ウイルスやクラッカーからの攻撃を表す、システムやネットワーク動作のパターンのことを指します。侵入検知システムでは、攻撃シグネーチャを利用して、正当な動作と不正な動作を識別し、不正な動作のみを防止するように作られています。
AppArmor では、このような攻撃シグネーチャに依存した仕組みにはなっていません。攻撃シグネーチャのような攻撃に対する "反応" として防御を発動するのではなく、"先回り型" の防御を行っています。攻撃シグネーチャの場合、製品に対する攻撃方法が様々に考えられるため、様々な場合を想定しなければなりませんが、AppArmor では脆弱性が存在する可能性も含めた "先回り" を行うため、よりよいセキュリティを実現することができます。

GUI

グラフィカルユーザインターフェイス (Graphical User Interface) の略です。コンピュータの利用者とアプリケーションとの間を取り持つ、直感的で使いやすいインターフェイスを意味します。ウインドウやアイコン、ボタンやカーソル、スクロールバーなどの要素が含まれます。

グロブ

ファイル名の代用として使用するもので、具体的なファイル名やパスを指定する代わりに、`*` (`/` や `?` などの特殊文字を除く、任意の長さかつ任意の文字の代用) や `?` (任意の 1 文字) を使用して、複数のファイルやディレクトリをまとめて指定する方法です。なお、`**` は特殊なグロブで、指定したディレクトリ以下にある任意の階層内かつ任意のファイルやディレクトリの代用として使用します。

HIP

ホスト侵入防止 (Host Intrusion Prevention) の略です。オペレーティングシステムのカーネルと共に動作し、不正なアプリケーション動作を未知の攻撃手法として判断して、防止するための仕組みです。また、ネットワークレベルでは、ホスト宛の不正なパケットを防いで、目的のアプリケーションが「傷つく」前にブロックすることもできます。

強制アクセス制御

ユーザやファイル、その他のオブジェクトなどに対して割り当てられたセキュリティ属性に従って、オブジェクトに対するアクセスを制限する手法を指します。この制限は、ユーザやプログラムなどから変更することができないように作られていることから、これを「強制」と表しています。

プロファイル

AppArmor のプロファイルは、個別のアプリケーションがどのシステムリソースにアクセスし、どのような権限が必要となるのかを完全に定義するためのものです。

プロファイルファウンデーションクラス

DNS 参照やユーザ認証など、一般的なアプリケーション動作に必要な、プロファイルの構築ブロックを意味する用語です。

RPM

RPM パッケージマネージャ (RPM Package Manager) の略です。誰にでも使用することのできる、オープンなパッケージシステムです。Red Hat Linux や openSUSE Leap のほか、様々な Linux および Unix システムで使用されています。コンピュータ内のソフトウエアパッケージのインストールやアンインストール、検証や問い合わせ、更新などを行うことができます。詳しくは https://rpm.org/ をお読みください。

SSH

Secure SHell の略です。リモートのコンピュータからテキストコマンドを受け付けることのできるサービスで、その通信を暗号化するのが特長です。

直感的なアクセス制御

AppArmor では、ネットワークサービスに対する直感的なアクセス制御の仕組みが提供されています。この仕組みでは、それぞれのプログラムが読み込んだり書き込んだり、実行したりすることを許すファイルのほか、どの種類のネットワークに接続できるのかを制御します。このような仕組みにより、それぞれのプログラムが期待通りの動作をするように強制することができます。

URI

統一資源識別子 (Universal Resource Identifier) の略です。World Wide Web の世界において、特定の資源を表すための名前や場所を指し示すための一般的な用語です。なお、URL は URI の一部として存在しています。

URL

統一資源位置指定子 (Uniform Resource Locator) の略です。Web 上での文書やその他の資源を指し示す位置情報です。
位置情報の最初のパートは使用すべきプロトコルで、2 番目にはリソースの存在する IP アドレスやドメイン名などが書かれます。
たとえば `http://www.opensuse.org` であれば、URL の冒頭にあるとおり、使用するプロトコルは http であることがわかるようになっています。

脆弱性

攻撃を受けやすい状態のままになっているシステムやネットワークの要素を示す用語です。コンピュータシステムでは、それが正しく動作し続け、許可を与えていないユーザに対して制御を許さないようにする必要がありますが、設計面や管理面、実装面の弱さもしくは欠陥により、ハードウエアやファームウエア、ソフトウエアに対する不正な制御が許されてしまうことがあります。脆弱な箇所に対して攻撃が成功してしまうと、許可を与えていないユーザに対して制御を許し、情報を漏洩させたり重要な処理が中断されてしまったりすることがあります。

V SELinux

40 SELinux の設定 426

40 SELinux の設定

改訂履歴

2024-07-12

本章では、openSUSE Leap 上で SELinux を使用する際の、設定と管理の方法について説明しています。下記のトピックをカバーしています:

- なぜ SELinux を使用するのか
- SELinux とは
- SELinux の設定方法
- SELinux の管理

40.1 なぜ SELinux を使用するのか

SELinux は Linux カーネル内でセキュリティフレームワークを使用する追加のセキュリティソリューションとして開発されてきました。従来型の標準ファイルパーミッション (所有者、グループ、全体に対して、それぞれ読み込み、書き込み、実行を許可する仕組み) という随意アクセス制御 (DAC) の範疇を超えた、きめ細かいセキュリティポリシーを構築できる仕組みを目指しています。

下記の例では、なぜ SELinux (もしくはその同等品である AppArmor) が必要となるのかについて、説明しています:

「ある朝、私はサーバがクラックされていることに気がついた。サーバは openSUSE Leap ですべての修正 (パッチ) が適用され動作していた。ファイアウオールについても正しく設定が行われ、そのサーバでは不要なサービスは全く提供していなかった。より詳しく調査を行うと、クラッカーはサーバ内の Apache 仮想ホスト内の 1 つに設置された、脆弱性のある PHP スクリプトを通して侵入を行っていた。クラッカーはその後、Apache Web サーバで使用されている `wwwrun` を利用して、シェルへのアクセスを取得していた。この `wwwrun` での侵入後、`/var/tmp` および `/tmp` の各ディレクトリ内に複数のスクリプトを作成し、これを様々なサーバへの分散型サービス拒否攻撃のボットネットとして動作させていた。」

ここで注目すべき点は、このクラッキングでは、サーバ側の設定を誤っているわけではない、という点です。すべてのパーミッションの設定が正しいにも関わらず、侵入者はシステム内に入り込むことができてしまっています。上記の例は、場合によっては追加のセキュリティ機構が必要となることを示しているわけです。

SELinux ではオブジェクト (ファイルやネットワークソケット) に対してラベルを設定し、このラベルを権限の判断基準として使用します。

40.1.1 サポート状態

SELinux フレームワークは openSUSE Leap 15.7 でもサポート対象となっています。つまり openSUSE Leap では、お使いのサーバ内で SELinux を使用するのに必要な、すべてのバイナリやライブラリが提供されます。

ただし、ポリシーは添付されていませんので、要件に合わせて構築していかなければなりません。サードパーティ製のポリシーを使用してもかまいません。40.3項「SELinux パッケージのインストール」や40.4項「SELinux ポリシーのインストール」では、テスト用に openSUSE のポリシーをインストールするための情報が示されています。

40.1.2 SELinux のコンポーネント

SELinux の設定を始める前に、まずは SELinux の構成について知っておく必要があります。SELinux では、下記の 3 種類から構成されています:

- Linux カーネル内にあるセキュリティフレームワーク
- SELinux のライブラリとバイナリ
- SELinux のポリシー

openSUSE Leap の既定のカーネルであれば SELinux に対応していますので、ツールを使用するだけで管理を行うことができます。そのため、管理者にとって最も重要な作業は、SELinux のポリシー管理ということになります。

SELinux のポリシーでは、Linux サーバ内の様々なオブジェクトに対して、セキュリティラベルを設定します。これらのオブジェクトには、ユーザやポート、プロセスやファイルなどが含まれます。このようなセキュリティラベルを使用することで、サーバ内での許可を判断することになります。ただし、既定の SELinux ではすべてのものを拒否する仕組みであることから、どうしても必要なアクセスのみを許可する適切なルール構築が重要となります。そのため、ルールはシステム内で使用するすべてのプログラムに対して存在していなければならないことになります。

それ以外にも、システム内の一部 (特定のポートやプログラム、ユーザやファイル、ディレクトリなど) を無制限モードで動作させることもできます。無制限モードは SELinux での保護を外すための仕組みで、システム内の重要なサービスにのみ SELinux の保護を働かせるような用途で便利な仕組みです。もちろんこの仕組みでは保護が不完全になりますので、可能な限り SELinux をシステム全体に適用しておいたほうが適切です。

お使いのシステムに対して適切な保護を実現するには、SELinux のポリシーを設定する必要があります。これはすべてのファイルに対してラベルを提供するよう仕立て上げられたポリシーでなければならず、すべてのサービスやユーザに対してセキュリティラベルを設定し、サーバ内でどのファイルやディレクトリにアクセスを許可するのかを設定するものでなければなりません。このようなポリシーの作成には、途方もない労力が必要となります。

なお、無償で提供されている様々な SELinux ポリシーは、お使いのサーバでも問題なく動作するとは思いますが、独自にポリシーを構築するのに比べると、保護レベルが落ちることがほとんどであることに注意してください。また、SUSE ではサードパーティ製のポリシーへのサポートは行っておりません。

40.2 SELinux ポリシーの概要

ポリシーとは SELinux における主要なコンポーネントです。SELinux のポリシーは、特定のオブジェクトがシステム内のどのファイルやディレクトリ、ポートやプロセスなどにアクセスできるのかを示すためのもので、これら全てに対してセキュリティコンテキストを設定することで、この仕組みを実現しています。ポリシーが適用され、ファイルシステムにラベルが設定されている SELinux システムであれば、`ls -Z` コマンドを実行することで、ディレクトリ内のセキュリティコンテキストを表示することができます。例 40.1:「`ls -Z` を利用したセキュリティコンテキストの表示」では、SELinux によってラベルが設定された openSUSE Leap システムにおいて、`/` ディレクトリ内のセキュリティコンテキストを表示した場合の例を示しています。

例 40.1: `ls -Z` を利用したセキュリティコンテキストの表示

```
> ls -Z /
system_u:object_r:bin_t bin
system_u:object_r:boot_t boot
system_u:object_r:device_t dev
system_u:object_r:etc_t etc
system_u:object_r:home_root_t home
system_u:object_r:lib_t lib
system_u:object_r:lib_t lib64
system_u:object_r:lost_found_t lost+found
system_u:object_r:mnt_t media
system_u:object_r:mnt_t mnt
system_u:object_r:usr_t opt
system_u:object_r:proc_t proc
system_u:object_r:default_t root
system_u:object_r:bin_t sbin
system_u:object_r:security_t selinux
system_u:object_r:var_t srv
system_u:object_r:sysfs_t sys
system_u:object_r:tmp_t tmp
system_u:object_r:usr_t usr
system_u:object_r:var_t var
```

セキュリティコンテキストで最も重要な箇所は、コンテキストタイプ (種類) です。これはしばしば `_t` で終わるセキュリティコンテキストとして示され、SELinux が対象のオブジェクトに対して、どのような種類のアクセスを許可するのかを示しています。ポリシー側では、どのような種類のユーザまたはロールに対して、どの種類のコンテキストにアクセスを許可するのかを指定します。たとえば下記のようなルールがあります:

```
allow user_t bin_t:file {read execute gettattr};
```

上記のルールでは、`user_t` というコンテキストタイプのユーザ (これをソースオブジェクトと呼びます) に対して、`bin_t` というコンテキストタイプが設定された "ファイル" (これをターゲットと呼びます) へのアクセスを許可しています。ただし、許可されるアクセスは `read` (読み込み) , `execute` (実行) , `getattr` (属性取得) の 3 種類だけです。

SELinux のポリシーには数多くのルールが含まれています。管理をやりやすくするには、ポリシーをモジュールとして分割したほうがよい場合があります。これにより、システム内の様々な箇所で、保護の有無を切り替えることができるようになります。

お使いのシステムでポリシーを構築する場合、モジュール型のポリシーを構築するのか、もしくはモノリシックな (1 つの巨大なポリシーでシステム全体を保護する) ポリシーを構築するのかを選択することができますが、通常はモジュール型のポリシーを構築しておくことを強くお勧めします。それは、モジュール型のほうが管理がやりやすいためです。

40.3 SELinux パッケージのインストール

コマンドラインを利用して下記のパッケージをインストールします:

```
> sudo zypper in restorecond policycoreutils setools-console
```

ただし、これではポリシーがインストールされません。テスト用に openSUSE のポリシーをインストールしたい場合は、40.4項「SELinux ポリシーのインストール」を参照してください。

40.4 SELinux ポリシーのインストール

ポリシーは SELinux で必須の部品です。openSUSE Leap 15.7 では、既定のポリシーは 同梱されておりません ので、お使いの環境に合わせてポリシーを構築しなければなりません。これは、SELinux が要件に合わせてカスタマイズしなければならない特性があるためです。支援を求めたい場合は、SUSE 社のコンサルタントサービスにお問い合わせください。なお、SELinux を完全にサポートするコンテナ環境や仮想環境を必要とするお客様やパートナー様には、`slemicro` をお勧めします。こちらであればサポート済みポリシーも提供されます。

テスト用 のポリシーは https://download.opensuse.org/repositories/security:/SELinux_legacy/ で公開されています。ここでは openSUSE Leap に対応したポリシーと追加パッケージが提供されています。

1. お使いの openSUSE Leap バージョンに合ったリポジトリリンクをコピーして、zypper に追加してください:

```
> sudo zypper ar -f \
https://download.opensuse.org/repositories/security:/SELinux/15.7/ \
SELinux-Legacy
```

2. あとは下記のパッケージをインストールします:

```
> sudo zypper in selinux-policy-targeted selinux-policy-devel
```

40.5 SELinux での許容 (permissive) モードへの移行

許容モードでは、SELinux はサーバを保護しませんが、ログファイル内への記録は変わらず行われます。このモードはシステムの動作テストや設定時に使用します。

既定では、SELinux は `/etc/selinux/config` ファイル内で許容モードに設定されています。

なお、お使いのシステムで SELinux の使用を開始したい場合は、GRUB 2 のブートローダの設定を変更します。具体的には、`/etc/default/grub` ファイル内にある `GRUB_CMDLINE_LINUX_DEFAULT=` という行内に、下記 2 つの項目を追加してください:

```
lsm=selinux,bpf selinux=1
```

最初のパラメータは、セキュリティシステムに AppArmor ではなく SELinux を使用する設定で、2 つめのパラメータは SELinux を有効化する設定です。これらのパラメータを追加したあとは、下記を実行して GRUB 2 の設定ファイルを再構築します:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

あとはシステムを再起動するだけです。ここまでの作業で SELinux システムを動作させるための準備が整いましたが、最後にやるべき作業が残っています。ここまでの設定では動作の拒否は行わず、ポリシーへの違反を全て記録するだけであるためです。まずはログファイルの内容を確認して、どのような違反が発生していたのかを確認してください。

40.6 SELinux での強制 (enforcing) モードへの移行

SELinux を許容モードで動作させていると、ユーザやプロセスの処理によってファイルシステム内のオブジェクトが適切にラベル付けされなくなる場合があります。SELinux はファイルシステムの各オブジェクトへのラベル付けによって動作する仕組みであることから、強制モードに移行した際に最も問題となる点でもあります。

このようなことを防ぐため、強制モードに移行する前にセキュリティコンテキスト (拡張属性) のリセットを行っておいてください:

```
> sudo restorecon -R /
```

あとは SELinux を強制モードに移行するだけです。`/etc/selinux/config` ファイルを編集して、`SELINUX=enforcing` を設定してください。

設定が終わったらサーバを再起動し、問題なく起動できるかと、ログインできるかを確認します。

ログイン後は `sestatus -v` を実行して、正しく動作していることを確認してください。問題なく動作していれば、例 40.2:「SELinux が動作していることの確認」のような出力が現れるはずです。

例 40.2: SELINUX が動作していることの確認

```
> sudo sestatus -v
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     requested(insecure)
Max kernel policy version:      33

Process contexts:
Current context:                unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
Init context:                   system_u:system_r:init_t:s0
/usr/sbin/sshd                  system_u:system_r:sshd_t:s0-s0:c0.c1023

File contexts:
Controlling terminal:           unconfined_u:object_r:user_tty_device_t:s0
/etc/passwd                     system_u:object_r:passwd_file_t:s0
/etc/shadow                     system_u:object_r:shadow_t:s0
/bin/bash                       system_u:object_r:shell_exec_t:s0 \
                                -> system_u:object_r:shell_exec_t:s0
/bin/login                      system_u:object_r:login_exec_t:s0
/bin/sh                         system_u:object_r:bin_t:s0 \
                                -> system_u:object_r:shell_exec_t:s0
/sbin/agetty                    system_u:object_r:bin_t:s0 \
                                -> system_u:object_r:getty_exec_t:s0
```

```
/sbin/init                      system_u:object_r:bin_t:s0 -> \
                                system_u:object_r:init_exec_t:s0
/usr/sbin/sshd                  system_u:object_r:sshd_exec_t:s0
```

強制モードでサーバが想定通りに起動できない場合は、許容モードに戻して再設定を行います。まずは `less /var/log/audit/audit.log` ファイルを確認してください。詳しくは 40.9項「トラブルシューティング」をお読みください。

40.7 SELinux の設定

サーバの調整作業を行う場合、まずは SELinux のインストールを確認します。既に `sestatus -v` というコマンドを使用していますので、ここに表示されている現在のモードとプロセス、ファイルコンテキストをそれぞれ確認します。あとは下記のように実行して、利用可能な可否設定を確認します:

```
> sudo semanage boolean -l
```

この可否設定の一覧を表示することで、ポリシーへのアクセスも同時に検証することができます。例 40.3「可否設定の一覧表示とポリシーアクセスの確認」には、このコマンドの出力例が示されています:

例 40.3: 可否設定の一覧表示とポリシーアクセスの確認

```
> sudo semanage boolean -l
SELinux boolean                              Description
ftp_home_dir                   -> off   ftp_home_dir
mozilla_read_content           -> off   mozilla_read_content
spamassassin_can_network       -> off   spamassassin_can_network
httpd_can_network_relay        -> off   httpd_can_network_relay
openvpn_enable_homedirs        -> off   openvpn_enable_homedirs
gpg_agent_env_file             -> off   gpg_agent_env_file
allow_httpd_awstats_script_anon_write -> off   allow_httpd_awstats_script_anon_write
httpd_can_network_connect_db   -> off   httpd_can_network_connect_db
allow_ftpd_full_access         -> off   allow_ftpd_full_access
samba_domain_controller        -> off   samba_domain_controller
httpd_enable_cgi               -> off   httpd_enable_cgi
virt_use_nfs                   -> off   virt_use_nfs
```

この時点で有用なコマンドがもう 1 つあります。それは下記のとおりです:

```
> sudo semanage fcontext -l
```

上記は、ポリシーによって提供される既定のファイルコンテキスト設定を表示することができます (詳しくは 例 40.4:「ファイルのコンテキスト情報の取得」をご覧ください) 。

例 40.4: ファイルのコンテキスト情報の取得

```
> sudo semanage fcontext -l
/var/run/usb(/.*)?                                all files
 system_u:object_r:hotplug_var_run_t
```

```
/var/run/utmp                                          regular file
 system_u:object_r:initrc_var_run_t
/var/run/vbe.*                                         regular file
 system_u:object_r:hald_var_run_t
/var/run/vmnat.*                                       socket
 system_u:object_r:vmware_var_run_t
/var/run/vmware.*                                      all files
 system_u:object_r:vmware_var_run_t
/var/run/watchdog\.pid                                 regular file
 system_u:object_r:watchdog_var_run_t
/var/run/winbindd(/.*)?                                all files
 system_u:object_r:winbind_var_run_t
/var/run/wnn-unix(/.*)                                 all files
 system_u:object_r:canna_var_run_t
/var/run/wpa_supplicant(/.*)?                          all files
 system_u:object_r:NetworkManager_var_run_t
/var/run/wpa_supplicant-global                         socket
 system_u:object_r:NetworkManager_var_run_t
/var/run/xdmctl(/.*)?                                  all files
 system_u:object_r:xdm_var_run_t
/var/run/yiff-[0-9]+\.pid                              regular file
 system_u:object_r:soundd_var_run_t
```

40.8 SELinux の管理

ここまでの作業で SELinux を動作させることができましたので、次は設定作業に移ります。SELinux では、プロセスやユーザがどのファイルやディレクトリ、ポートなどにアクセスできるのかを、正確に定義するための追加のルールセットを使用します。ルールセットの設定を行うにあたって、SELinux はそれぞれのファイルやディレクトリ、プロセスやポートに対してコンテキストを設定し、それをセキュリティラベルとして使用します。セキュリティラベルは、それらをどのように扱うべきかを示しているもので、SELinux のポリシー側から指定が行われます。既定では、ポリシーは全てのアクセスを拒否するように設定されていますので、お使いのシステムが何らかの動作を必要とする場合、管理者はルールを作成してポリシー内に配置する必要があります。

40.8.1 セキュリティコンテキストの表示

上述のとおり、ファイルやディレクトリ、ポートなどに対してラベルを設定することができます。ラベル内では様々なコンテキストを使用します。管理者が日々の作業を実施できるようにするためには、コンテキストタイプについて最も知識を持っておく必要があります。様々なコマンドに用意された `-Z` オプションを使用することで、現在のコンテキスト設定を表示することができるようになります。例 40.5:「ルートディレクトリ内でのディレクトリ向け既定コンテキスト」には、ルートディレクトリ内でのコンテキスト設定の例を示しています。

例 40.5: ルートディレクトリ内でのディレクトリ向け既定コンテキスト

```
> sudo ls -Z
dr-xr-xr-x. root root system_u:object_r:bin_t:s0       bin
dr-xr-xr-x. root root system_u:object_r:boot_t:s0      boot
drwxr-xr-x. root root system_u:object_r:cgroup_t:s0    cgroup
drwxr-xr-x+ root root unconfined_u:object_r:default_t:s0 data
drwxr-xr-x. root root system_u:object_r:device_t:s0    dev
drwxr-xr-x. root root system_u:object_r:etc_t:s0       etc
drwxr-xr-x. root root system_u:object_r:home_root_t:s0 home
dr-xr-xr-x. root root system_u:object_r:lib_t:s0       lib
dr-xr-xr-x. root root system_u:object_r:lib_t:s0       lib64
drwx------. root root system_u:object_r:lost_found_t:s0 lost+found
drwxr-xr-x. root root system_u:object_r:mnt_t:s0       media
drwxr-xr-x. root root system_u:object_r:autofs_t:s0    misc
drwxr-xr-x. root root system_u:object_r:mnt_t:s0       mnt
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 mnt2
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 mounts
drwxr-xr-x. root root system_u:object_r:autofs_t:s0    net
drwxr-xr-x. root root system_u:object_r:usr_t:s0       opt
dr-xr-xr-x. root root system_u:object_r:proc_t:s0      proc
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 repo
dr-xr-x---. root root system_u:object_r:admin_home_t:s0 root
dr-xr-xr-x. root root system_u:object_r:bin_t:s0       sbin
drwxr-xr-x. root root system_u:object_r:security_t:s0  selinux
drwxr-xr-x. root root system_u:object_r:var_t:s0       srv
-rw-r--r--. root root unconfined_u:object_r:swapfile_t:s0 swapfile
drwxr-xr-x. root root system_u:object_r:sysfs_t:s0     sys
drwxrwxrwt. root root system_u:object_r:tmp_t:s0       tmp
-rw-r--r--. root root unconfined_u:object_r:etc_runtime_t:s0 tmp2.tar
-rw-r--r--. root root unconfined_u:object_r:etc_runtime_t:s0 tmp.tar
drwxr-xr-x. root root system_u:object_r:usr_t:s0       usr
drwxr-xr-x. root root system_u:object_r:var_t:s0       var
```

上述の一覧に書かれているとおり、全てのディレクトリに対してコンテキストが設定されていることがわかります。ここにはユーザと役割、タイプがそれぞれ示されています。残る s0 はマルチレベルセキュリティ環境でのセキュリティレベルを示すもので、ここでは説明していません。このような環境では s0 が設定されていることを確認してください。コンテキストの種類は、そのディレクトリ内でどのような種類の動作を許可するのかを表しています。たとえば `/root` ディレクトリには `admin_home_t` というコンテキストタイプが割り当てられ、`/home` ディレクトリには `admin_home_t` というコンテキストタイプが割り当てられています。SELinux のポリシーでは、これらのコンテキストタイプに対して様々な種類のアクセスが定義されています。

セキュリティラベルはファイルに対して割り当てられるだけでなく、ポートやプロセスなどにも割り当てられます。例 40.6:「`ps Zaux` によるプロセス向け SELinux 設定の表示」では、サーバ内のプロセスに対するコンテキスト設定の表示例を示しています。

例 40.6: `ps Zaux` によるプロセス向け SELINUX 設定の表示

```
> sudo ps Zaux
LABEL                             USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START
  TIME COMMAND
system_u:system_r:init_t          root         1  0.0  0.0  10640   808 ?        Ss   05:31
  0:00 init [5]
system_u:system_r:kernel_t        root         2  0.0  0.0      0     0 ?        S    05:31
  0:00 [kthreadd]
system_u:system_r:kernel_t        root         3  0.0  0.0      0     0 ?        S    05:31
  0:00 [ksoftirqd/0]
system_u:system_r:kernel_t        root         6  0.0  0.0      0     0 ?        S    05:31
  0:00 [migration/0]
system_u:system_r:kernel_t        root         7  0.0  0.0      0     0 ?        S    05:31
  0:00 [watchdog/0]
system_u:system_r:sysadm_t        root      2344  0.0  0.0  27640   852 ?        Ss   05:32
  0:00 /usr/sbin/mcelog --daemon --config-file /etc/mcelog/mcelog.conf
system_u:system_r:sshd_t          root      3245  0.0  0.0  69300  1492 ?        Ss   05:32
  0:00 /usr/sbin/sshd -o PidFile=/var/run/sshd.init.pid
system_u:system_r:cupsd_t         root      3265  0.0  0.0  68176  2852 ?        Ss   05:32
  0:00 /usr/sbin/cupsd
system_u:system_r:nscd_t          root      3267  0.0  0.0 772876  1380 ?        Ssl  05:32
  0:00 /usr/sbin/nscd
system_u:system_r:postfix_master_t root     3334  0.0  0.0  38320  2424 ?        Ss   05:32
  0:00 /usr/lib/postfix/master
system_u:system_r:postfix_qmgr_t postfix    3358  0.0  0.0  40216  2252 ?        S    05:32
  0:00 qmgr -l -t fifo -u
system_u:system_r:crond_t         root      3415  0.0  0.0  14900   800 ?        Ss   05:32
  0:00 /usr/sbin/cron
system_u:system_r:fsdaemon_t      root      3437  0.0  0.0  16468  1040 ?        S    05:32
  0:00 /usr/sbin/smartd
system_u:system_r:sysadm_t        root      3441  0.0  0.0  66916  2152 ?        Ss   05:32
  0:00 login -- root
system_u:system_r:sysadm_t        root      3442  0.0  0.0   4596   800 tty2     Ss+  05:32
  0:00 /sbin/mingetty tty2
```

40.8.2 SELinux モードの選択

SELinux では、3 種類のモードを使用します:

強制 (Enforcing):
: こちらが既定のモードです。SELinux では、ポリシー内のルールに従ってサーバを保護します。合わせて SELinux では、監査ログ内に動作状況を記録します。

許容 (Permissive):
: このモードはトラブルシューティングで有用な仕組みです。許容モードでは、SELinux はサーバを保護しませんが、ログファイル内への記録は変わらず行われます。

無効 (Disabled):

このモードでは、SELinux は完全に無効化され、ログへの記録も行われません。ファイルシステム内に設定されたラベルについては、ファイルシステムから削除されることはありません。

SELinux のモードの設定方法については前述のとおり、システム起動時の GRUB 2 でパラメータを指定して行います。

40.8.3 SELinux のコンテキストタイプの変更

管理者の作業として重要なものとして、SELinux を正しく動作させるため、ファイルにコンテキストタイプを設定する作業があります。

ファイルが特定のディレクトリ内に作成されると、既定では親のディレクトリのコンテキストタイプを継承します。ですが、ファイルが一方のディレクトリから他方のディレクトリに移動されると、以前の場所でのコンテキストタイプをそのまま適用し続けます。

ファイルに対してコンテキストタイプを設定するには、`semanage fcontext` コマンドを使用します。このコマンドでは、新しいコンテキストタイプをポリシー内に書き込むまでは行うものの、実際のコンテキストタイプを即時に反映することは行いません。ポリシー内のコンテキストタイプを実際に反映させるには、設定後に `restorecon` コマンドを実行します。

`semanage fcontext` を利用してみる前に、まずは実際にどのようなコンテキストが必要なのかを確認します。具体的には下記のように実行します:

```
> sudo semanage fcontext -l
```

上記のように実行することで、ポリシー内に設定されている全てのコンテキストを表示することができます。ですが、出力は比較的長いものであるため、実際に必要なコンテキストを知るには少し面倒ではあります (例 40.7:「既定のファイルコンテキストの表示」をご覧ください) 。

例 40.7: 既定のファイルコンテキストの表示

```
> sudo semanage fcontext -l | less
SELinux fcontext                                    type               Context

/                                                   directory
 system_u:object_r:root_t:s0
/.*                                                 all files
 system_u:object_r:default_t:s0
/[^/]+                                              regular file
 system_u:object_r:etc_runtime_t:s0
/\.autofsck                                         regular file
 system_u:object_r:etc_runtime_t:s0
/\.autorelabel                                      regular file
 system_u:object_r:etc_runtime_t:s0
```

```
/\.journal                                              all files          X:>>None>>
/\.suspended                                            regular file
 system_u:object_r:etc_runtime_t:s0
/a?quota\.(user|group)                                  regular file
 system_u:object_r:quota_db_t:s0
/afs                                                    directory
 system_u:object_r:mnt_t:s0
/bin                                                    directory
 system_u:object_r:bin_t:s0
/bin/.*                                                 all files
 system_u:object_r:bin_t:s0
```

お使いのサービスでどのようなコンテキスト設定が利用できるのかを確認する方法には、下記の 3 つの方法があります:

- サービスをインストールして、使用されている既定のコンテキストを判別する方法。この方法が最も簡単であり、推奨される方法です。
- 特定のサービスに対してマニュアルページを確認します。サービスによっては `_selinux` で終わる名前のマニュアルページを用意しているものがあり、必要なコンテキスト設定の情報を取得できるものがあります。
  正しいコンテキスト設定を見つけることができたら、`semanage fcontext` を利用して設定を適用します。このコマンドでは `-t` のコンテキストタイプの指定を最初のパラメータとして指定し、その後ろに適用したい先のディレクトリやファイルを指定します。コンテキストを適用したいディレクトリ内に存在する全てに対して、コンテキストを適用したい場合は、ディレクトリ名の箇所に `(/.*)?` を指定します。これは正規表現で、スラッシュに続いて任意の文字が続くものを表しています。
  このほかにも、`semanage` のマニュアルページ内の examples セクションには、`semanage` を使用する際の様々な便利な例が示されています。正規表現に関する詳しい情報については、https://ja.wikipedia.org/wiki/%E6%AD%A3%E8%A6%8F%E8%A1%A8%E7%8F%BE をお読みください。
- お使いのシステムで利用可能な全てのコンテキストタイプを一覧表示する方法:

  ```
  > sudo seinfo -t
  ```

  コマンドそれ自身は非常に多くの情報を出力します。そのため、`grep` などのコマンドを併用して、必要な情報だけを取得するようにしてください。

40.8.4 ファイルコンテキストの適用

SELinux のコンテキストを正しく適用するため、下記の手順では `semanage fcontext` と `restorecon` を使用して、コンテキストを設定する方法を示しています。ここまでの説明でご存じのとおり、既定のドキュメントルート以外を使用した場合、Web サーバが動作しません。SELinux の設定を変更して、正しく動作するようにするまでの手順を示します:

1. `/web` ディレクトリを作成し、そのディレクトリに移動します:

```
> sudo mkdir /web  && cd /web
```

2. テキストエディタを利用して、`/web/index.html` ファイルを作成し、どんなものでもかまいませんのでコンテンツを作成します。

3. `/etc/apache2/default-server.conf` ファイルをエディタで開いて、DocumentRoot の行を `DocumentRoot /web` のように変更します。

4. Apache Web サーバを再起動します:

```
> sudo systemctl start apache2
```

5. ローカルの Web サーバが表示できるかどうか、テストを行います:

```
> w3m localhost
```

   Connection refused (接続が拒否されました) のようなメッセージが表示されるかと思います。`Enter` を押したあと、`q` を押して w3m を終了します。

6. Apache Web サーバ既定の **ドキュメントルート** に対して、コンテキストタイプを表示します。このディレクトリは `/srv/www/htdocs` になっています。このディレクトリに対しては、`httpd_sys_content_t` が設定されているはずです:

```
> sudo ls -Z /srv/www
```

7. ポリシー内に新しいコンテキストを設定します。下記のとおり入力して `Enter` を押します:

```
> sudo semanage fcontext -a -f "" -t httpd_sys_content_t '/web(/.*) ?'
```

8. 新しいコンテキストタイプを適用します:

```
> sudo restorecon /web
```

9. `/web` ディレクトリ内のファイルのコンテキスト情報を表示します。`/web` ディレクトリには正しく設定されているものの、そのディレクトリ内のファイルには設定されていないはずです:

```
> sudo ls -Z /web
```

10. 今度は新しいコンテキストを、`/web` ディレクトリ以下の全てに対して再帰的に適用します。これでファイルにも新しいコンテキストが適用されるようになります:

```
> sudo restorecon -R /web
```

11. Web サーバを再起動します:

```
> sudo systemctl restart apache2
```

これで `/web` ディレクトリ内のコンテンツについても、問題なくアクセスできるようになっているはずです。

40.8.5 SELinux ポリシーの設定

ポリシーの動作を変更する際、最も簡単なものは可否設定 (boolean) です。可否設定はオンとオフを切り替えることのできる項目で、ポリシー内で変更することのできるものです。利用可能な設定の一覧を取得するには、下記のように入力して実行します:

```
> sudo semanage boolean -l
```

上記を実行すると、可否設定の長い一覧と共に、それぞれの可否設定に対する短い説明文が表示されます。必要な可否設定を見つけることができたら、あとは `setsebool -P` を利用して、設定作業を行います。このコマンドの後ろには可否設定の名前を指定します。このとき、`setsebool` に `-P` というオプションを指定していることに注目してください。このオプションは、ディスク上にあるポリシーファイル内に設定を書き込むためのもので、システムを再起動しても設定を残すように設定するための唯一の方法でもあります。

下記の手順では、可否設定の変更例を示しています:

1. まずは FTP サーバに関連する可否設定を一覧表示します:

```
> sudo semanage boolean -l | grep ftp
```

2. 可否設定をオフにします:

```
> sudo setsebool allow_ftpd_anon_write off
```

なお、変更を書き込むのにそれほど時間はかかりません。設定を書き込んだら、本当にオフになっているのかを確認します:

```
> sudo semanage boolean -l|grep ftpd_anon
```

3. あとはサーバを再起動します。
4. 再起動後、`allow_ftpd_anon_write` の設定がオンに戻っていることを確認します。これは、ポリシーがファイルには書き込まれなかったためです。
5. 再度可否設定を切り替えて、今度はポリシーファイルにも書き込みます:

```
> sudo setsebool -P allow_ftpd_anon_write
```

40.8.6 SELinux モジュールの使用

既定では、SELinux はモジュール型のポリシーを使用します。これは、SELinux の機能が単一の巨大なポリシーとして作られているのではなく、多数の小さいモジュールとして構成されていることを意味しています。それぞれのモジュールには、それぞれ対応する SELinux の設定パートがあります。このような SELinux の仕組みにより、サードパーティのベンダがサービスを提供する場合でも、自身のサービスを SELinux に対応できるようにすることができます。SELinux のモジュールの一覧を取得するには、`semodule -l` コマンドを実行します。このコマンドは、SELinux で使用されている全てのモジュールと、そのバージョン番号を表示します。

管理者の権限を使用すると、モジュールを個別にオンもしくはオフにすることができます。これは SELinux の一部の機能のみを無効化したい場合や、全てのサービスを SELinux で保護する必要がない場合に便利な仕組みです。特に openSUSE Leap では、SELinux ポリシーを完全にサポートしているわけではないので、全てのモジュールをいったんオフに設定して、必要なサービスにのみ SELinux の保護を適用したほうが都合がよいことがあります。SELinux のモジュールをオフにしたい場合は、下記のように入力して実行します:

```
> sudo semodule -d モジュール名
```

オンに戻したい場合は、下記のように入力して実行します:

```
> sudo semodule -e モジュール名
```

40.9 トラブルシューティング

既定では、SELinux が何らかの理由で動作しない場合、ログメッセージが `/var/log/audit/audit.log` ファイル内に記録されます。ただし、このファイルに書き込みを行うには、auditd サービスを動作させて置かなければなりません。`/var/log/audit` ディレクトリ内に何もファイルが存在しない場合は、下記のように入力して実行し、auditd サービスを開始してください:

```
> sudo systemctl start auditd
```

また、システムの起動時に開始するように設定したい場合は、下記のように入力して実行します:

```
> sudo systemctl enable auditd
```

下記に `/var/log/audit/audit.log` の内容の出力例を示します:

例 40.8: `/etc/audit/audit.log` の出力例

```
type=DAEMON_START msg=audit(1348173810.874:6248): auditd start, ver=1.7.7 format=raw
 kernel=6.4.0-150600.9-default auid=0 pid=4235 subj=system_u:system_r:auditd_t res=success
type=AVC msg=audit(1348173901.081:292): avc:  denied  { write } for
  pid=3426 comm="smartd" name="smartmontools" dev=sda6 ino=581743
 scontext=system_u:system_r:fsdaemon_t tcontext=system_u:object_r:var_lib_t tclass=dir
type=AVC msg=audit(1348173901.081:293): avc:  denied  { remove_name } for  pid=3426
 comm="smartd" name="smartd.WDC_WD2500BEKT_75PVMT0-WD_WXC1A21E0454.ata.state~" dev=sda6
 ino=582390 scontext=system_u:system_r:fsdaemon_t tcontext=system_u:object_r:var_lib_t
 tclass=dir
type=AVC msg=audit(1348173901.081:294): avc:  denied  { unlink } for  pid=3426
 comm="smartd" name="smartd.WDC_WD2500BEKT_75PVMT0-WD_WXC1A21E0454.ata.state~" dev=sda6
 ino=582390 scontext=system_u:system_r:fsdaemon_t tcontext=system_u:object_r:var_lib_t
 tclass=file
type=AVC msg=audit(1348173901.081:295): avc:  denied  { rename } for  pid=3426
 comm="smartd" name="smartd.WDC_WD2500BEKT_75PVMT0-WD_WXC1A21E0454.ata.state" dev=sda6
 ino=582373 scontext=system_u:system_r:fsdaemon_t tcontext=system_u:object_r:var_lib_t
 tclass=file
type=AVC msg=audit(1348173901.081:296): avc:  denied  { add_name } for  pid=3426
 comm="smartd" name="smartd.WDC_WD2500BEKT_75PVMT0-WD_WXC1A21E0454.ata.state~"
 scontext=system_u:system_r:fsdaemon_t tcontext=system_u:object_r:var_lib_t tclass=dir
type=AVC msg=audit(1348173901.081:297): avc:  denied  { create } for  pid=3426
 comm="smartd" name="smartd.WDC_WD2500BEKT_75PVMT0-WD_WXC1A21E0454.ata.state"
 scontext=system_u:system_r:fsdaemon_t tcontext=system_u:object_r:var_lib_t tclass=file
type=AVC msg=audit(1348173901.081:298): avc:  denied  { write open } for  pid=3426
 comm="smartd" name="smartd.WDC_WD2500BEKT_75PVMT0-WD_WXC1A21E0454.ata.state" dev=sda6
 ino=582390 scontext=system_u:system_r:fsdaemon_t tcontext=system_u:object_r:var_lib_t
 tclass=file
type=AVC msg=audit(1348173901.081:299): avc:  denied  { getattr } for  pid=3426
 comm="smartd" path="/var/lib/smartmontools/smartd.WDC_WD2500BEKT_75PVMT0-
WD_WXC1A21E0454.ata.state" dev=sda6 ino=582390 scontext=system_u:system_r:fsdaemon_t
 tcontext=system_u:object_r:var_lib_t tclass=file
type=AVC msg=audit(1348173901.309:300): avc:  denied  { append } for  pid=1316
```

はじめて `audit.log` をお読みになる方にとっては、少しわかりにくいと感じることでしょう。しかしながら、読み方がわかればそれほど難しくはありません。各行はそれぞれセクションに分割することができます。たとえば末尾の行の意味は下記のようになります:

`type=AVC` :
: それぞれの SELinux 関連の監査ログ行には、`type=AVC` という種類判別用の記述が書き込まれます。

`msg=audit(1348173901.309:300)` :
: これはタイムスタンプを意味しています。ただし、エポック秒と呼ばれる、1970 年 1 月 1 日からの経過秒数で書かれています。整数部分については、`date -d` コマンドで日時に変換することができます:

```
> date -d @1348173901
2012年  9月 21日 金曜日 05:45:01 JST
```

`avc: denied { append }` :
: 拒否された処理の内容を示しています。この場合、システムはデータをファイルに追記する処理が拒否されたことを表しています。監査ログファイル内では、ファイルを開いたり属性を取得したりなど、様々な処理が記録されることになります。

`for pid=1316` :
: その処理を実施したコマンドもしくはプロセスのプロセス ID です。

`comm="rsyslogd"` :
: その PID に結びつけられたコマンドを表しています。

`name="smartmontools"` :
: 処理を実行しようとしたアプリケーションの名前を表しています。

`dev=sda6 ino=582296` :
: 対象となるファイルが存在するブロックデバイス名と inode 番号です。

`scontext=system_u:system_r:syslogd_t` :
: ソースコンテキストと呼ばれ、処理を実行しようとした側のコンテキストを表しています。

`tclass=file` :
: 処理内容の分類名を表しています。

audit.log 内の各イベントの意味を自分自身で解釈する以外の方法もあります。それは、`audit2allow` というコマンドです。このコマンドは、わかりにくい `/var/log/audit/audit.log` 内のメッセージを分析するための支援を行うソフトウエアです。下記に示す 3 種類の形態が用意され

ています。1 つめは `audit2allow -w -a` のような実行形態で、より読みやすい形式で監査情報を表示することができます。`audit2allow -w -a` は、既定では audit.log ファイルを読み込みます。audit.log ファイル内の特定のメッセージのみを分析したい場合は、一時ファイルにコピーして下記のように入力して実行してください:

```
> sudo audit2allow -w -i ファイル名
```

例 40.9: 監査メッセージの分析

```
> sudo audit2allow -w -i testfile
type=AVC msg=audit(1348173901.309:300): avc:  denied  { append } for  pid=1316
comm="rsyslogd" name="acpid" dev=sda6 ino=582296
scontext=system_u:system_r:syslogd_t tcontext=system_u:object_r:apmd_log_t tclass=file
```

これは下記の理由によって発生しています:

タイプ強制 (Type Enforcement; TE) の許可ルールが存在しないためです。
このアクセスを許可するような読み込みモジュールを生成するには、下記のように入力して実行します:

```
> sudo audit2allow
```

アクセスを拒否したルールを見つけたい場合は、`audit2allow -a` と入力して実行し、`audit.log` ファイル内に記録された全てのメッセージから、対応する強制ルールを表示することができるほか、`audit2allow -i ファイル名` のように入力して実行することで、指定したファイル内に保存されているメッセージに対する強制ルールを表示することもできます。

例 40.10: アクセス拒否内容の表示

```
> sudo audit2allow -i testfile
#============= syslogd_t ==============
allow syslogd_t apmd_log_t:file append;
```

`mymodule` という名前の SELinux モジュールを作成し、以前は拒否されていたアクセスを許可するように設定したい場合は、下記のように入力して実行します:

```
> sudo audit2allow -a -R -M mymodule
```

audit.log に記録されている全てのイベントに対して、同様のことを行いたい場合は、パラメータに `-a -M` を追加します。また、特定のファイル内のメッセージに対してのみ実施したい場合は、下記のように `-i -M` を追加します:

例 40.11: 以前に拒否されたアクションを許可するためのポリシー作成

```
> sudo audit2allow -i testfile -M example
```

```
****************** IMPORTANT **********************
To make this policy package active, execute:

semodule -i example.pp
```

`audit2allow` コマンドの出力にもあるとおり、この後に `semodule -i` コマンドの後ろにモジュール名指定して、実行します。このとき、モジュール名は `audit2allow` で指定した名前となります (上記の例では `example.pp` です) 。

VI Linux 監査フレームワーク

41 Linux 監査システムの概要 446

42 Linux 監査フレームワークの設定 479

43 監査ルールセットの紹介 491

44 その他の情報源 502

41 Linux 監査システムの概要

改訂履歴

2026-09-01

本バージョンの openSUSE Leap に同梱されている Linux 監査フレームワークは、CAPP (Controlled Access Protection Profiles) 互換の監査システムとして提供され、セキュリティに関連するイベントを確実に収集します。監査記録からは、セキュリティポリシーへの違反有無と、それが誰によって行われたものであるのかを調べることができます。

監査フレームワークは CC-CAPP/EAL (Common Criteria-Controlled Access Protection Profiles/Evaluation Assurance Level) の認証を受ける際に重要な要件となるものです。また、Common Criteria (CC) for Information Technology Security Information は、独立したセキュリティ評価に対する国際標準です。Common Criteria の仕組みにより、顧客が任意の IT 製品に対するセキュリティレベルを判断し、ミッションクリティカルな環境への導入を支援することができます。

Common Criteria でのセキュリティ評価には、2 種類の評価要件セット (機能要件／保証要件) があります。機能要件は評価対象の製品のセキュリティ属性を記述するもので、Controlled Access Protection Profiles (CAPP) のもとまとめられるものです。一方の保証要件は、Evaluation Assurance Level (EAL) に従ってまとめられるものです。EAL は、評価者がセキュリティ属性を判断するにあたって、それが存在し、効果があって、実装されていることを確認するためのすべての作業を記述するものです。この種類の作業には、たとえば開発者によるセキュリティの脆弱性の検索、修正処理、テストなどの文書化が含まれます。

このガイドでは、監査システムの動作概要と、設定方法について説明しています。Common Criteria に関する一般的な情報については、コモンクライテリア (Wikipedia) (https://ja.wikipedia.org/wiki/%E3%82%B3%E3%83%A2%E3%83%B3%E3%82%AF%E3%83%A9%E3%82%A4%E3%83%86%E3%83%AA%E3%82%A2) などの情報をお読みください。

Linux 監査システムでは、お使いのシステム内で発生している内容を非常に細かく分析することによって、セキュリティをより強固にする支援を行います。ただし、それ自身でセキュリティが強化されるわけではありません。つまり、コードの誤動作や様々な脆弱性からシステムを保護するわけではありません。その代わり、監査システムではそれらの問題を追跡し、AppArmor などのそれらを保護するための様々な仕組みを使用するよう促すためのものです。

監査システムは複数のコンポーネントから構成される仕組みですが、それぞれがフレームワーク全体に重要な機能を提供しています。監査システムのカーネルモジュールは、システムコールを中継し、その際に関連するイベントを記録します。また `auditd` デーモンは、監査レポートをディスクに記録します。このほか、様々なコマンドラインユーティリティを利用することで、監査証跡の表示や問い合わせ、書庫保存などを行うことができます。

監査システムは下記の機能を提供します:

ユーザとプロセスとの関連づけ

監査システムはプロセスとそれを起動したユーザ ID との間の関連づけを行います。これにより、管理者やセキュリティ責任者が、プロセスを所有しているユーザを正確に追跡することができるほか、システム内での不審な動作を検出することができるようになります。



重要: ユーザ ID の名前変更について

監査デーモンは UID の変更を処理しません。そのため、たとえばユーザ名 `tux` の UID を `uid=1001` から `uid=2000` に変更したり、廃止したりすることは避けてください。このような作業を行ってしまった場合は、`auditctl` データ (監査ルール) を変更する必要が発生してしまうほか、古いデータを正しく取得できなくなってしまうことがあります。

監査証跡の確認

Linux 監査システムでは、監査レポートのディスクへの書き込みだけでなく、レポートを人間にとって読みやすい形式に変換する処理も行います。

特定の監査イベントの確認

監査システムでは、監査レポート内の特定のイベントのみを抽出するためのユーティリティを提供しています。下記の検索条件を指定することができます:

- ユーザ
- グループ
- 監査 ID
- リモートホスト名
- リモートホストアドレス
- システムコール
- システムコールのパラメータ
- ファイル

- ファイル操作
- 成功もしくは失敗

選択的監査の適用
: 監査システムにはフィルタリング機能が用意され、注目対象のイベントに対して監査レポートを作成することができるほか、特定のイベントのみを記録するよう監査システムを構成することもできます。このほか、独自のルールセットを作成して、これらの記録のみを実施するよう監査デーモンを構成することもできます。

レポートデータの可用性の保証
: 監査レポートは `root` が所有するもので、`root` のみが削除できます。それ以外のユーザは監査ログを削除できません。

監査データの喪失保護
: カーネルのメモリが不足した場合や監査デーモンのバックログが超過した場合、もしくは監査デーモンの流量制限を超過した場合、監査デーモンはシステムのシャットダウンを実施して、監査システムの監視から外れないように設定することができます。このシャットダウンは、監査システムの中枢部からシステムの即時停止を求めるもので、この場合は直近のログがディスクに書き込まれることはありません。既定の設定では、システムの停止ではなく、syslog への警告記録のみを行うように設定されています。
ログの記録に必要なディスク領域が不足した場合、監査システムに対してクリーンなシャットダウンを求めるように設定することもできます。既定の設定では、ディスク領域が不足した場合、ログを停止するように設定されています。

41.1 Linux 監査システムのコンポーネント紹介

重要: audispd の auditd への併合について

openSUSE Leap 15.4 およびそれ以降のバージョンでは、`audispd` のソースコードが `auditd` に併合されています。また、`audispd` の全ての設定は `/etc/audit/auditd.conf` および `/etc/audit/plugins.d` 内に移動しています。

下記の図には、監査システムでの様々なコンポーネントと、相互作用の関係を示しています:

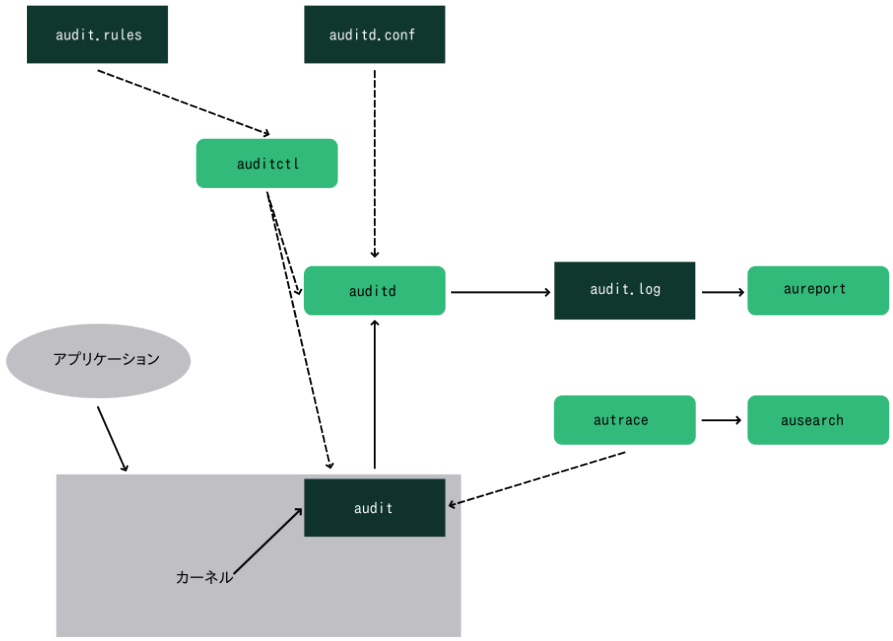


図 41.1: LINUX 監査システムのコンポーネント紹介

実線の矢印はコンポーネント間でのデータの流れを、点線の矢印はコンポーネント間の制御を表しています。

auditd

監査デーモンは、アプリケーションやシステムの動作によって生成され、監査カーネルインターフェイスを介して提供された監査メッセージを、ディスクに書き込むデーモンです。監査デーモンは `systemd` によって起動されます。監査システムの (起動時の) 設定は、`/etc/audit/auditd.conf` で制御します。`auditd` に関する詳細情報と設定方法について、詳しくは 41.2項「監査デーモンの設定」をお読みください。

`auditctl`

`auditctl` ユーティリティは、監査システムを制御するためのユーティリティです。ログ生成に関わるパラメータのほか、監査インターフェイスのカーネル設定や、どのイベントを追跡するかのルールなどを設定することができます。`auditctl` に関する詳細は、41.3項「`auditctl` による監査システムの制御」をお読みください。

監査ルール

`/etc/audit/audit.rules` ファイルには、`auditctl` コマンドのシーケンスが含まれていて、監査デーモンの起動直後のシステム起動時に読み込まれる内容が記述されています。監査ルールに関する詳細は、41.4項「監査システムに対するパラメータ指定」をお読みください。

aureport

`aureport` ユーティリティは、監査イベントログから独自のレポートを作成することができるユーティリティです。このレポート生成は簡単にスクリプトに組み込むことができますし、出力は他のアプリケーションから使用することができる (たとえば結果のグラフ化など) ようになっています。`aureport` に関する詳細は、41.5項「監査ログの仕組みとレポートの生成」をお読みください。

ausearch

`ausearch` ユーティリティは監査ログ内を検索するためのユーティリティで、様々なキーを指定してイベントを検索したり、様々なログ形式の属性を指定して検索したりすることができるようになっています。`ausearch` に関する詳細は、41.6項「`ausearch` による監査デーモンログへの問い合わせ」をお読みください。

autrace

`autrace` は `strace` のように、特定のプロセスを追跡するためのユーティリティです。`autrace` の出力についても、監査ログに記録されるようになっています。`autrace` に関する詳細は、41.7項「`autrace` によるプロセスの解析」をお読みください。

aulast

`last` コマンドのように、ユーザが直近にログインした日時を一覧表示します。`aulast` は過去の監査ログ (もしくは指定した監査ログファイル) を読み込んで、指定した日時範囲におけるユーザのログインおよびログアウトを一覧で表示します。

aulastlog

`lastlog` コマンドのように、マシンに対する最新のログイン日時を一覧表示します。ログイン名のほか、ポートと最終ログイン日時が表示されます。

41.2 監査デーモンの設定

監査ログの生成やそのログの処理を行う前に、まずは監査デーモン自身の設定を行います。設定ファイル `/etc/audit/auditd.conf` では、デーモンの起動時にどのようにして監査システムを動作させるのかを設定します。ほとんどの用途では、openSUSE Leap に同梱されている既定の設定のままでかまいません。CAPP 環境向けに使用する場合は、ほとんどのパラメータに調整が必要です。下記は既定の設定です:

例 41.1: /ETC/AUDIT/AUDITD.CONF の既定値

```
local_events = yes
write_logs = yes
log_file = /var/log/audit/audit.log
log_group = audit
log_format = RAW
```

```
flush = INCREMENTAL_ASYNC
freq = 50
max_log_file = 8
num_logs = 5
priority_boost = 4
name_format = NONE
##name = mydomain
max_log_file_action = ROTATE
space_left = 75
space_left_action = SYSLOG
verify_email = yes
action_mail_acct = root
admin_space_left = 50
admin_space_left_action = SUSPEND
disk_full_action = SUSPEND
disk_error_action = SUSPEND
use_libwrap = yes
##tcp_listen_port = 60
tcp_listen_queue = 5
tcp_max_per_addr = 1
##tcp_client_ports = 1024-65535
tcp_client_max_idle = 0
transport = TCP
distribute_network = no
q_depth = 1200
overflow_action = SYSLOG
max_restarts = 10
plugin_dir = /etc/audit/plugins.d
end_of_event_timeout = 2
```

これらの設定値に関する詳細は、`man 5 auditd.conf` で表示されるマニュアルページをお読みください。

CAPP の要件を満たすよう環境を設定する必要があるかどうかによって異なりますが、必要である場合は監査デーモンの設定をより厳密に行う必要があります。CAPP の要件を満たす場合は、「CAPP 環境について」と書かれた注記に従って設定を行ってください。

`/etc/audit/auditd.conf` でのデーモン設定が完了したら、次は監査デーモンが監査する範囲の制御と、デーモンが適切に動作するために必要な十分なリソースの割り当てと制限を行います。

41.3 `auditctl` による監査システムの制御

`auditctl` は監査デーモンの状態の制御のほか、基本的なシステムパラメータを設定することができます。このほか、システム内で実施される監査範囲の制御を行うこともできます。監査ルールを使用することで、`auditctl` はお使いのシステム内のどのコンポーネントを監査するのか、およびどの範囲までを監査するのかを制御することができます。監査ルールは `auditctl` コマンドを使用してコマンドライ

ンから渡すことができるほか、ルールセットを作成して監査デーモン側で処理させることもできます。既定では、`auditd` は `/etc/audit/audit.rules` にある監査ルールを使用するようになっています。監査ルールに関する詳細は、41.4項「監査システムに対するパラメータ指定」をお読みください。

基本的な監査システムパラメータを制御するための主な `auditctl` コマンドは下記のとおりです:

- `auditctl -e`: 監査の有効化と無効化
- `auditctl -f`: 失敗フラグの制御
- `auditctl -r`: 監査メッセージの流量制限の制御
- `auditctl -b`: バックログ制限の制御
- `auditctl -s`: 監査デーモンに関する現在の状態の表示
- `auditctl -S` では監査対象とするシステムコールを指定します。なお、お使いのシステムで `auditctl -S` を実行する際は、`-F arch=b64` を指定して、アーキテクチャの間違いに関する警告が発生しないようにしてください。

`-e`, `-f`, `-r`, `-b` オプションは `audit.rules` ファイル内でも指定することができます。これにより、監査デーモンの起動時に毎回入力するような手間を省くことができます。

`auditctl -s` で監査デーモンの状態を問い合わせる場合や、`auditctl -eフラグ` で状態フラグを変更する場合は、状態メッセージ (上述のパラメータに関する情報を含む) が出力されます。下記は、一般的な監査状態のメッセージ例を示しています。

例 41.2: `auditctl -s` の出力例

```
enabled 1
failure 1
pid 790
rate_limit 0
backlog_limit 64
lost 0
backlog 0
backlog_wait_time 15000
loginuid_immutable 0 unlocked
```

表 41.1: 監査状態フラグ

フラグ	意味 [取り得る値]	コマンド
`enabled`	有効化フラグを設定します。[0..2] 0=無効, 1=有効, 2=有効にして設定をロックダウンします。なお、この有効化フラグはシステムコールのログにのみ	`auditctl -e [0|1|2]`

フラグ	意味 [取り得る値]	コマンド
	適用されるものであり、その他のイベントについては効果がありません (詳しくは `audit-devel` パッケージ内の `man 3 audit_set_enabled` をお読みください) 。	
`flag`	失敗フラグを設定します。[0..2] 0=何もしない, 1=printk, 2=panic (ディスクへの書き込み待ちのデータを書き込むことなく、即時に停止する)	`auditctl -f [0|1|2]`
`pid`	`auditd` のプロセス ID を表します。	—
`rate_limit`	メッセージの流量制限 (毎秒メッセージ数) を設定します。この値にゼロより大きい値が設定され、その値を超過した場合は、失敗フラグ内で設定した動作を実行します。	`auditctl -r 流量`
`backlog_limit`	未処理の監査バッファの最大サイズを指定します。すべてのバッファが使用中になると、失敗フラグ内で設定した動作を実行します。	`auditctl -b バックログ`
`lost`	現時点までに喪失した監査メッセージの数を表します。	—
`backlog`	現時点での未処理の監査バッファ数を表します。	—

41.4 監査システムに対するパラメータ指定

監査システムに対するコマンドは `auditctl` コマンドを利用してシェルから個別に実行することもできますし、`auditctl -R` のように指定して、ファイルから読み込んで一括処理することもできます。後者のやり方は起動時のスクリプトでも使用していて、監査デーモンの起動後に、`/etc/audit/audit.rules` ファイルからルールを読み込む目的で使用しています。ルールは上から順に実行されます。それぞれのルールは、それぞれが `auditctl` のコマンドに展開されます。ルールファイル内で使用する書式は、`auditctl` コマンドと同じです。

動作中の監査システムを `auditctl` で設定変更しても、その変更点はシステムの再起動で失われてしまいます。恒久的に変更したい場合は、`/etc/audit/audit.rules` ファイルに追加するとともに、現時点で監査デーモン内に読み込まれていない場合は、`systemctl restart auditd` を実行して、監査システムを再起動してください。

例 41.3: 監査ルール例: 監査システムのパラメータ

```
-b 1000❶
-f 1❷
-r 10❸
-e 1❹
```

❶ 未処理の監査バッファの最大数を指定しています。ログの発生頻度にも依存しますが、監査の負荷がシステムにとって重すぎない程度の値に設定してください。

❷ 使用する失敗フラグを指定しています。指定可能な値は 表41.1「監査状態フラグ」 をご覧ください。

❸ カーネル側で発信される最大流量のメッセージ数を指定しています。詳しくは 表41.1「監査状態フラグ」 をご覧ください。

❹ 監査サブシステムの有効化／無効化を設定しています。

監査の仕組みを使用することで、重要なファイルや設定、リソースなどのファイルシステムアクセスを追跡することができるようになります。また、これらに対して監視を設定してキーを割り当てることで、ログ内でより見つけやすくすることができます。

例 41.4: 監査ルール例: ファイルシステムの監査

```
-w /etc/shadow❶
-w /etc -p rx❷
-w /etc/passwd -k fk_passwd -p rwxa❸
```

❶ `-w` オプションは、指定したファイル (この例では `/etc/shadow`) に対する監視を監査システムに設定しています。このファイルに対するアクセス許可要求に関わるすべてのシステムコールが、分析の対象となります。

❷ このルールは `/etc` ディレクトリに対する監視を追加していて、読み込みと実行のアクセスに対するフィルタリングを設定 (`-p rx`) しています。これら 2 種類の許可に関わるシステムコールが分析の対象となります。ディレクトリに対する監視では、新しいファイルの作成と既存のファイルの削除のみが記録の対象となります。特定のディレクトリ内にあるファイルに対して、より詳しいイベントを監視したい場合は、それぞれのファイルに対してルールを作成してください。ただし、監視を設定する前に、あらかじめファイルが存在していなければなりません。なお、ファイル作成時点からの監査には対応していません。

❸ このルールは、`/etc/passwd` ファイルに対する監視を追加しています。このとき、読み込みと書き込み、実行と属性変更を監視しています。`-k` オプションを指定すると、後から特定のイベントを検索する (たとえば `ausearch` など) 目的で、監査ログ内にキーを書き込むことができます。なお、複数のルールに対して同じキーを指定しておくことで、それらをまとめて出力することもできるようになります。逆に、1 つのルールに複数のキーを設定することもできます。

システムコールの監査では、アプリケーションよりも下のレベルで、システムの動作を追跡することができます。これらのルールを設計する際は、多数のシステムコールに対して監査を行うと、システムの負荷が上昇するだけでなく、ディスク領域も不足する危険性があることに注意してください。追跡対象のイベントと、それらのフィルタリング方法については、より具体的に設定するようにしてください。

例 41.5: 監査ルール例: システムコールの監査

```
-a exit,always -S mkdir ❶
-a exit,always -S access -F a1=4 ❷
-a exit,always -S ipc -F a0=2 ❸
-a exit,always -S open -F success!=0 ❹
-a task,always -F auid=0 ❺
-a task,always -F uid=0 -F auid=501 -F gid=wheel ❻
```

❶ このルールは `mkdir` システムコールに対する監査を設定しています。`-a` オプションは、システムコールルールを追加するためのオプションです。この監査は、`mkdir` システムコールが実行されるたびに毎回 (`exit` , `always`) 行われます。また、`-S` オプションでは、このルールの適用先となるシステムコール名を指定しています。

❷ このルールは access システムコールに対する監査を設定していますが、システムコールの 2 番目のパラメータ (`mode`) が `4` (`R_OK`) である場合にのみ監査を行っています。`exit,always` は監査システムに対して、このシステムコールへの突入時に監査コンテキストを追加し、監査発生時にレポートを書き出すように指定しています。

❸ このルールは IPC 多重化システムコールに対して監査コンテキストを設定しています。特定の `ipc` システムコールは最初の syscall パラメータとして渡され、`-F a0=IPC_コール番号` の指定で選択することができます。

❹ このルールは open システムコールが失敗した場合に対する監査を設定しています。

❺ このルールはタスクルール (キーワード: `task`) の例です。これはこれまでのルールとは異なり、fork や clone されるプロセスに適用されます。これらの種類のイベントにフィルタを設定したい場合は、fork の時点で既知の状態にあるフィールドのみを使用することができます。具体的には UID, GID, AUID です。この例では、監査 ID が `0` であるすべてのタスクに対して監査を行っています。

❻ 最後のルールはフィルタを複雑に設定した場合の例です。すべてのフィルタオプションが組み合わせられていて、それらを論理積 (AND) で結んでいます。つまり、このルールは監査 ID が `501` であり、`root` で実行され、かつグループが `wheel` であるものに適用されます。監査 ID はユーザのログイン時にプロセスに対して設定されます。この ID は、そのユーザの初期プロセスから起動されたすべての子プロセスに渡されます。ユーザが自身の識別情報を変更した場合でも、監査 ID は変わらずそのままであり続けるため、元のユーザの動きをそのまま追跡できることになります。



ヒント: システムコールパラメータのフィルタリングについて

システムコールパラメータのフィルタリングについて、詳しくは 43.6項「システムコールのパラメータでのフィルタリング」をお読みください。

監査システムにはルールを追加するだけでなく、ルールを削除することもできます。ルール全体を一括で削除することができるほか、個別のシステムコールルールやファイル／ディレクトリの監視ルールを削除することができます:

例 41.6: 監査ルールとイベントの削除

```
-D ❶
-d exit,always -S mkdir ❷
-W /etc ❸
```

❶ 監査ルールのキューを消去して、既存のルールをすべて削除します。このルールは `/etc/audit/audit.rules` 内の冒頭で使用すべきもので、ルールを適用し直す際に、それまでに存在したルールをいったん削除するために設定します。もちろん `auditctl -D` のように入力して実行し、`autrace` を実行する前に `audit.rules` 内にある既存のトレース規則をいったん削除しておくような使い方もできます。

❷ このルールでは、システムコールのルールを削除しています。ルールキューから削除する必要のあるシステムコールを指定する場合は、その前に `-d` オプションを指定する必要があるほか、`-d` は正確にマッチするものでなければなりません。

❸ このルールでは、監査システムのルールキューから `/etc` ディレクトリに対するルールを破棄するよう指定しています。このルール削除では、パーミッションのフィルタリングやキーのオプション指定にかかわらず、すべての `/etc` を含むディレクトリの監視を削除します。

現時点での監査設定で使用しているルールの概要を知りたい場合は、`auditctl -l` と入力して実行します。このコマンドは、1 ルールあたり 1 行で表示します。

例 41.7: `auditctl -l` によるルール一覧の表示

```
exit,always watch=/etc perm=rx
exit,always watch=/etc/passwd perm=rwxa key=fk_passwd
exit,always watch=/etc/shadow perm=rwxa
exit,always syscall=mkdir
exit,always a1=4 (0x4) syscall=access
exit,always a0=2 (0x2) syscall=ipc
exit,always success!=0 syscall=open
```



注記: フィルタルールの作成について

様々なフィルタオプションを組み合わせることで、より洗練された監査ルールを構築することができます。監査フィルタルールを構築する際に利用可能なオプションのほか、監査ルールに関する詳細については、`auditctl(8)` のマニュアルページをお読みください。

41.5 監査ログの仕組みとレポートの生成

`aureport` ユーティリティが何をするものなのかを知るには、まず監査デーモンが生成するログの構造について知っておく必要があるほか、イベント内にどのような情報が存在するのかを知っておく必要があります。これにより、要件に対応できる最適なレポートタイプを判断することができるようになります。

41.5.1 監査ログの仕組み

下記の例には、監査デーモンが記録した 2 種類の典型的なイベントと、それらの証跡が示されています。監査ログは、ローテーションしたものを含め、`/var/log/audit` 内に保存されます。

ログには 2 種類の情報が記録されます。それぞれレコードタイプとイベントフィールドと呼びます。レコードタイプはそれぞれの記録内の `type=` 以下に書かれています。イベントフィールドはそれ以外の項目で、イコール記号の前に書かれたものを指します。下記の例では `type=SYSCALL` や `type=CWD` がレコードタイプ、`arch=c000003e` や `syscall=2` がイベントフィールド (値付きのもの) となります。

レコードタイプの全ての一覧とそれらの意味について、詳しくは `/usr/include/libaudit.h` ファイル (`audit-devel` パッケージに含まれています) をお読みください。

また、`ausyscall --dump` コマンドを実行することで、システムコールの番号と対応する関数名を出力することができます:

```
> ausyscall --dump
```

```
Using x86_64 syscall table:
0       read
1       write
2       open
3       close
4       stat
5       fstat
[...]
```

最初の例は単純な `less` コマンドを実行した場合の例、2 番目の例は、監査を実施しているマシン内にユーザがリモートからログインしようとした場合、ログ内に PAM の動作に関わる大量のログが出力されている場合の例を示しています。

例 41.8: シンプルな監査イベント例: 監査ログの表示

```
type=SYSCALL msg=audit(1234874638.599:5207): arch=c000003e syscall=2
success=yes exit=4 a0=62fb60 a1=0 a2=31 a3=0 items=1 ppid=25400 pid
=25616 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0
tty=pts1 ses=1164 comm="less" exe="/usr/bin/less" key="doc_log"
type=CWD msg=audit(1234874638.599:5207):  cwd="/root"
type=PATH msg=audit(1234874638.599:5207): item=0 name="/var/log/audit/
audit.log" inode=1219041 dev=08:06 mode=0100644 ouid=0 ogid=0 rdev=00:00
```

上記のイベントは `less /var/log/audit/audit.log` を実行した場合のログで、3 種類のメッセージが記録されています。これらのすべては密接にリンクしているため、その他の情報が無ければ意味を理解するのが難しいものと思います。最初のメッセージには下記の情報が記録されています:

`type`

記録されたイベントの種類を示しています。この場合は `SYSCALL` という種類で、システムコールによる記録であることを示しています。それ以外には `CWD` という種類のイベントもあり、これはシステムコール時点でのカレントディレクトリを示しています。次の `PATH` という種類のイベントは、システムコールに渡されたパスに関する情報が示されています。open システムコールは 1 つのパラメータを取るシステムコールであることから、`PATH` イベントも 1 つだけ生成されます。ここで重要となるのが、`PATH` イベントはパス名の文字列パラメータを報告するだけで、それ以外の解釈については何も行わないものである点です。そのため、相対パスを指定した場合は、`CWD` のイベントで示されたカレントディレクトリの情報とあわせて、手作業で絶対パスに戻さなければならないことになります。

`msg`

括弧内にはメッセージ ID が書かれています。メッセージ ID は 2 種類のパーツから構成されていて、`:` の前は Unix エポックタイムを、後ろは実際のイベント ID を表しています。同じシステムコール内であれば、同じイベント ID が割り当てられます。アプリケーションがさらに別のシステムコールを行うと、イベント ID が新たに割り当てられます。

`arch`
: システムコールに対応する CPU アーキテクチャを示しています。この情報は `ausearch` コマンドでログを検索する際、`-i` オプションを指定するとデコードすることができます。

`syscall`
: このシステムコールに対して `strace` で表示することのできる、システムコールの種類を表しています。このデータは `/usr/include/asm/unistd.h` ファイル内に示されていて、アーキテクチャによって番号体系が異なります。このアーキテクチャでは、`syscall=2` は open システムコール (詳しくは `man open(2)` をお読みください) を表しています。

`success`
: システムコールが成功したか、失敗したかを示しています。

`exit`
: システムコールの返り値を表しています。この例では `open` システムコールに対する返り値ですが、これは作成されたファイルディスクリプタ番号を表しています。返り値の意味はシステムコールによって異なります。

`a0` から `a3`
: 数値形式で表現された、システムコールの最初の 4 つのパラメータです。これらはシステムコールによって異なる解釈になります。この例 (`open` システムコール) では、下記のように解釈します:

```
a0=62fb60 a1=8000 a2=31 a3=0
```

`a0` はパス名の開始アドレスを表しています。`a1` はフラグを表していて、16 進数の `8000` は 8 進数でいうと `100000` にあたることから、`O_LARGEFILE` というフラグを指定していることになります。また `a2` はモード指定で、`O_CREAT` を指定していないことから、使用していないパラメータであることがわかります。`a3` は `open` システムコールでは使用していないパラメータです。それぞれのパラメータの意味については、マニュアルページをお読みください。

`items`
: アプリケーションに渡された文字列の数を表しています。

`ppid`
: このプロセスの親のプロセス ID を表しています。

`pid`
: このプロセスのプロセス ID を表しています。

`auid`

監査 ID を表しています。監査 ID はユーザのログイン時にプロセスに対して設定されます。この ID は、そのユーザの初期プロセスから起動されたすべての子プロセスに渡されます。ユーザが自身の識別情報を変更した場合 (たとえば `root` になった場合) でも、監査 ID は変わらずそのままであり続けるため、元のユーザの動きをそのまま追跡できることになります。

`uid`

プロセスを起動したユーザの実ユーザ ID を表しています。この場合は `0` であることから、`root` であることになります。

`gid`

プロセスを起動したユーザの実グループ ID を表しています。この場合は `0` であることから、`root` であることになります。

`euid` , `suid` , `fsuid`

プロセスを起動したユーザの実効ユーザ ID, Set User ID, ファイルシステムユーザ ID をそれぞれ表しています。

`egid` , `sgid` , `fsgid`

プロセスを起動したユーザの実効グループ ID, Set Group ID, ファイルシステムグループ ID をそれぞれ表しています。

`tty`

アプリケーションを起動した端末を表しています。この場合は、SSH セッション内で疑似端末 (pts) を使用していることを表しています。

`ses`

ログインセッション ID を表しています。このプロセス属性はユーザのログイン時に設定されるほか、どのプロセスも特定のユーザログインに結びつけることができます。

`comm`

タスクリストに表示されるアプリケーション名を表しています。

`exe`

バイナリプログラムに対する解決済みパスを表しています。

`subj`

`auditd` は、プロセスに対して AppArmor などのセキュリティコンテキストが割り当てられている場合、その内容を記録することができます。この場合は `unconstrained` になっていますが、これは AppArmor でプロセスが保護されていないことを表しています。プロセスが何らかの制限を受けた場合、バイナリパスと AppArmor のプロファイルモードも記録されます。

`key`

多数のディレクトリやファイルに対して監査を行っている際、監視対象の判別用にキー文字列を指定することができます。これらのキーを `ausearch` に指定すると、そのキーに合致するもののみを表示することができます。

`less` コマンドが生成した 2 つ目のメッセージには、`less` コマンドを実行した時点でのカレントディレクトリを表しています。

3 つ目のメッセージの意味は下記のとおりです (`type` と `message` のフラグは既に説明しているとおりです):

`item`

この例では、`item` は `a0` パラメータを表しています。ここには元々の `SYSCALL` メッセージ内に対応する、パス名が書かれています。`cp` や `mv` などのように、複数のパスパラメータを指定する場合は、さらに `PATH` イベントが生成されて記録されることになります。

`name`

open システムコールのパラメータとして渡されたパスを表しています。

`inode`

`name` に対応する inode 番号を表しています。

`dev`

ファイルが保存されているデバイスを表しています。この場合、`08:06` と書かれていますので、`/dev/sda1` もしくは「最初の IDE デバイス上の最初のパーティション」であることを示しています。

`mode`

ファイルのアクセス許可を数値で表現したものです。この場合は `root` が読み書きの権限を持ち、グループ (`root`) は読み込みのみ、その他のユーザはファイルへのアクセス権を持たない設定になっています。

`ouid` と `ogid`

inode が指し示すものに対する UID と GID を表しています。

`rdev`

この例では意味を持たないものです。`rdev` はブロックデバイスやキャラクタデバイスの場合にのみ意味を持つもので、ファイルでは意味を持ちません。

例41.9「複雑な監査イベント例: SSH でのログイン」には、SSH 接続を受け付けた際の監査イベントを示しています。ほとんどのメッセージは PAM スタックに関連するもので、SSH PAM の処理における様々なステージを表しています。また、いくつかの監査メッセージは入れ子になっていて、それぞれの PAM の処理段階を示しています。監査システムでは PAM の処理についても記録を行いますが、独自のメッセージタイプが割り当てられます:

例 41.9: 複雑な監査イベント例: SSH でのログイン

```
type=USER_AUTH msg=audit(1234877011.791:7731): user pid=26127 uid=0 ❶
auid=4294967295 ses=4294967295 msg='op=PAM:authentication acct="root" exe="/usr/sbin/sshd"
(hostname=jupiter.example.com, addr=192.168.2.100, terminal=ssh res=success)'
type=USER_ACCT msg=audit(1234877011.795:7732): user pid=26127 uid=0 ❷
auid=4294967295 ses=4294967295 msg='op=PAM:accounting acct="root" exe="/usr/sbin/sshd"
(hostname=jupiter.example.com, addr=192.168.2.100, terminal=ssh res=success)'
type=CRED_ACQ msg=audit(1234877011.799:7733): user pid=26125 uid=0 ❸
auid=4294967295 ses=4294967295 msg='op=PAM:setcred acct="root" exe="/usr/sbin/sshd"
(hostname=jupiter.example.com, addr=192.168.2.100, terminal=/dev/pts/0 res=success)'
type=LOGIN msg=audit(1234877011.799:7734): login pid=26125 uid=0
old auid=4294967295 new auid=0 old ses=4294967295 new ses=1172
type=USER_START msg=audit(1234877011.799:7735): user pid=26125 uid=0 ❹
auid=0 ses=1172 msg='op=PAM:session_open acct="root" exe="/usr/sbin/sshd"
(hostname=jupiter.example.com, addr=192.168.2.100, terminal=/dev/pts/0 res=success)'
type=USER_LOGIN msg=audit(1234877011.823:7736): user pid=26128 uid=0 ❺
auid=0 ses=1172 msg='uid=0: exe="/usr/sbin/sshd"
(hostname=jupiter.example.com, addr=192.168.2.100, terminal=/dev/pts/0 res=success)'
type=CRED_REFR msg=audit(1234877011.828:7737): user pid=26128 uid=0 ❻
auid=0 ses=1172 msg='op=PAM:setcred acct="root" exe="/usr/sbin/sshd"
(hostname=jupiter.example.com, addr=192.168.2.100, terminal=/dev/pts/0 res=success)'
```

❶ リモートのホスト (jupiter.example.com, 192.168.2.100) から `root` に対してユーザ認証が求められ、それが成功したことを PAM が報告しています。ここで使用されている端末は `ssh` になっています。

❷ PAM は、ユーザにログイン権限があることを確認した、と報告しています。

❸ PAM はログインにあたって適切な資格情報が必要である旨を報告するとともに、端末が通常の端末 (`/dev/pts0`) に変更されたことを報告しています。

❹ PAM は、`root` に対してセッションを開くことができたことを報告しています。

❺ ユーザのログインが成功したことを示しています。このイベントは `aureport -l` でユーザログイン情報を報告させる際に、使用されるイベントです。

❻ PAM は資格情報を再取得したことを報告しています。

41.5.2 独自の監査レポートの生成

`/var/log/audit` ディレクトリ内にある未加工の監査レポートは、かさばって理解しにくいものになってしまっています。必要なメッセージを簡単に取得できるようにするには、`aureport` ユーティリティを利用して、独自にレポートを作成してください。

下記の利用例では、`aureport` で生成することのできる、いくつかのレポートタイプについて説明しています:

他のファイルからの監査ログ読み込み

監査ログを他のマシンに移動させた場合や、それぞれのマシンに接続することなく、ローカルのマシン内で複数のマシンの監査ログを分析したりしたい場合は、対象となるログファイルをローカルに移動したあと、`aureport` を利用してローカルで分析してください:

```
> sudo aureport -if myfile

Summary Report
======================
Range of time in logs: 2009年02月03日 14:13:38.225 - 2009年02月17日 14:52:27.971
Selected time for report: 2009年02月03日 14:13:38 - 2009年02月17日 14:52:27.971
Number of changes in configuration: 13
Number of changes to accounts, groups, or roles: 0
Number of logins: 6
Number of failed logins: 13
Number of authentications: 7
Number of failed authentications: 573
Number of users: 1
Number of terminals: 9
Number of host names: 4
Number of executables: 17
Number of files: 279
Number of AVC's: 0
Number of MAC events: 0
Number of failed syscalls: 994
Number of anomaly events: 0
Number of responses to anomaly events: 0
Number of crypto events: 0
Number of keys: 2
Number of process IDs: 1211
Number of events: 5320
```

上記のコマンドでは `aureport` に何もパラメータを指定しておらず、`myfile` 内に含まれているログ情報から、標準的で一般的な概要レポートのみを生成するように指定しています。より詳しいレポートを作成したい場合は、`-if` オプションに加えて、必要なオプションを指定してください。たとえば特定の時間帯のログインレポートのみを生成したい場合は、下記のように入力して実行します:

```
> sudo aureport -l -ts 14:00 -te 15:00 -if myfile

Login Report
============================================
# date time auid host term exe success event
============================================
1. 2017年02月09日 14:21:09 root: 192.168.2.100 sshd /usr/sbin/sshd no 7718
2. 2017年02月09日 14:21:15 0 jupiter /dev/pts/3 /usr/sbin/sshd yes 7724
```

数値表現の文字列変換

ユーザ ID などの情報は数値で出力されます。これらを読みやすいテキスト形式に変換するには、`aureport` コマンドに `-i` オプションを追加してください。

大まかな概要レポートの作成

現在の監査統計情報 (イベント数, ログイン数, プロセス数など) のみを知りたい場合は、`aureport` に何もパラメータを指定せずに実行してください。

失敗イベントの概要レポートの作成

純粋な `aureport` の概要統計情報を失敗イベントのみの統計情報に変更したい場合は、`aureport --failed` のように入力して実行します:

```
> sudo aureport --failed

Failed Summary Report
======================
Range of time in logs: 2009年02月03日 14:13:38.225 - 2009年02月17日 14:57:35.183
Selected time for report: 2009年02月03日 14:13:38 - 2009年02月17日 14:57:35.183
Number of changes in configuration: 0
Number of changes to accounts, groups, or roles: 0
Number of logins: 0
Number of failed logins: 13
Number of authentications: 0
Number of failed authentications: 574
Number of users: 1
Number of terminals: 5
Number of host names: 4
Number of executables: 11
Number of files: 77
Number of AVC's: 0
Number of MAC events: 0
Number of failed syscalls: 994
Number of anomaly events: 0
Number of responses to anomaly events: 0
Number of crypto events: 0
Number of keys: 2
Number of process IDs: 708
Number of events: 1583
```

成功イベントの概要レポートの作成

純粋な `aureport` の概要統計情報を成功イベントのみの統計情報に変更したい場合は、`aureport --success` のように入力して実行します:

```
> sudo aureport --success

Success Summary Report
======================
Range of time in logs: 2009年02月03日 14:13:38.225 - 2009年02月17日 15:00:01.535
Selected time for report: 2009年02月03日 14:13:38 - 2009年02月17日 15:00:01.535
Number of changes in configuration: 13
Number of changes to accounts, groups, or roles: 0
Number of logins: 6
Number of failed logins: 0
Number of authentications: 7
Number of failed authentications: 0
Number of users: 1
```

```
Number of terminals: 7
Number of host names: 3
Number of executables: 16
Number of files: 215
Number of AVC's: 0
Number of MAC events: 0
Number of failed syscalls: 0
Number of anomaly events: 0
Number of responses to anomaly events: 0
Number of crypto events: 0
Number of keys: 2
Number of process IDs: 558
Number of events: 3739
```

概要レポートの作成

専用の概要レポート (全体概要／失敗概要／成功概要) に加えて、`--summary` オプションと他のオプションを併用することで、指定した分野のみの概要レポートを作成することもできます。ただし、すべてのオプションがレポートに対応しているわけではありません。この例では、ユーザのログインイベントに関する概要レポートを作成しています:

```
> sudo aureport -u -i --summary

User Summary Report
===========================
total  auid
===========================
5640  root
13  tux
3  wilber
```

イベント一覧レポートの作成

監査デーモンが記録したイベントの概要を取得するには、`aureport -e` コマンドを使用します。このコマンドは順序番号付きのリストで、すべてのイベントに対する日付と時刻、イベント番号と種類、そして監査 ID を出力します。

```
> sudo aureport -e -ts 14:00 -te 14:21

Event Report
===================================
# date time event type auid success
===================================
1. 2009年02月17日 14:20:27 7462 DAEMON_START 0 yes
2. 2009年02月17日 14:20:27 7715 CONFIG_CHANGE 0 yes
3. 2009年02月17日 14:20:57 7716 USER_END 0 yes
4. 2009年02月17日 14:20:57 7717 CRED_DISP 0 yes
5. 2009年02月17日 14:21:09 7718 USER_LOGIN -1 no
6. 2009年02月17日 14:21:15 7719 USER_AUTH -1 yes
7. 2009年02月17日 14:21:15 7720 USER_ACCT -1 yes
```

```
8. 2009年02月17日 14:21:15 7721 CRED_ACQ -1 yes
9. 2009年02月17日 14:21:15 7722 LOGIN 0 yes
10. 2009年02月17日 14:21:15 7723 USER_START 0 yes
11. 2009年02月17日 14:21:15 7724 USER_LOGIN 0 yes
12. 2009年02月17日 14:21:15 7725 CRED_REFR 0 yes
```

すべてのプロセスイベントからのレポート作成

プロセスの観点からログを分析したい場合は、`aureport -p` コマンドを使用します。このコマンドは順序番号付きのリストで、プロセス関連のイベントを列挙します。これには日付と時刻、プロセス ID と実行ファイルの名前、システムコールと監査 ID 、イベント番号が含まれます。

```
aureport -p

Process ID Report
=======================================
# date time pid exe syscall auid event
=======================================
1. 2009年02月13日 15:30:01 32742 /usr/sbin/cron 0 0 35
2. 2009年02月13日 15:30:01 32742 /usr/sbin/cron 0 0 36
3. 2009年02月13日 15:38:34 32734 /usr/lib/gdm/gdm-session-worker 0 -1 37
```

すべてのシステムコールイベントからのレポート作成

システムコールの観点からログを分析したい場合は、`aureport -s` コマンドを使用します。このコマンドは順序番号付きのリストで、システムコール関連のイベントを列挙します。これには日付と時刻、システムコールの回数とそのコールを使用したコマンドの名前、監査 ID とイベント番号が含まれます。

```
> sudo aureport -s

Syscall Report
=======================================
# date time syscall pid comm auid event
=======================================
1. 2009年02月16日 17:45:01 2 20343 cron -1 2279
2. 2009年02月16日 17:45:02 83 20350 mktemp 0 2284
3. 2009年02月16日 17:45:02 83 20351 mkdir 0 2285
```

すべての実行ファイルイベントからのレポート作成

実行ファイルの観点からログを分析したい場合は、`aureport -x` コマンドを使用します。このコマンドは順序番号付きのリストで、実行ファイル関連のイベントを列挙します。これには日付と時刻、実行ファイルの名前と実行していた端末、実行していたホストと監査 ID 、イベント番号が含まれます。

```
aureport -x

Executable Report
```

```
====================================
# date time exe term host auid event
====================================
1. 2009年02月13日 15:08:26 /usr/sbin/sshd sshd 192.168.2.100 -1 12
2. 2009年02月13日 15:08:28 /usr/lib/gdm/gdm-session-worker :0 ? -1 13
3. 2009年02月13日 15:08:28 /usr/sbin/sshd ssh 192.168.2.100 -1 14
```

ファイルに関するレポート作成

ファイルアクセスの観点からログを分析したい場合は、`aureport -f` コマンドを使用します。このコマンドは順序番号付きのリストで、アクセスしているファイル、アクセスに使用したシステムコールの数と成功可否、実行ファイルと監査 ID 、イベント番号が含まれます。

```
> sudo aureport -f

File Report
===============================================
# date time file syscall success exe auid event
===============================================
1. 2009年02月16日 17:45:01 /etc/shadow 2 yes /usr/sbin/cron -1 2279
2. 2009年02月16日 17:45:02 /tmp/ 83 yes /bin/mktemp 0 2284
3. 2009年02月16日 17:45:02 /var 83 no /bin/mkdir 0 2285
```

ユーザに関するレポートの作成

システム内で、どのユーザが何の実行ファイルを実行させたのかを監査ログから調べ、レポートとしてまとめたい場合は、`aureport -u` コマンドを使用します。このコマンドは順序番号付きのリストで、日付や時刻、監査 ID や使用した端末とホスト、実行ファイルの名前とイベント ID がそれぞれ示されます。

```
aureport -u

User ID Report
====================================
# date time auid term host exe event
====================================
1. 2009年02月13日 15:08:26 -1 sshd 192.168.2.100 /usr/sbin/sshd 12
2. 2009年02月13日 15:08:28 -1 :0 ? /usr/lib/gdm/gdm-session-worker 13
3. 2009年02月13日 08:25:39 -1 ssh 192.168.2.101 /usr/sbin/sshd 14
```

ログインに関するレポートの作成

マシンに対するログイン試行に着目してレポートを作成するには、`aureport -l` コマンドを使用します。このコマンドは順序番号付きのリストで、日付や時刻、監査 ID や使用したホストおよび端末、実行ファイルの名前と試行の成功可否、イベント ID がそれぞれ含まれる、ログイン関連のレポートを生成します。

```
> sudo aureport -l -i
```

```
Login Report
============================================
# date time auid host term exe success event
============================================
1. 2009年02月13日 15:08:31 tux: 192.168.2.100 sshd /usr/sbin/sshd no 19
2. 2009年02月16日 12:39:05 root: 192.168.2.101 sshd /usr/sbin/sshd no 2108
3. 2009年02月17日 15:29:07 geeko: ? tty3 /bin/login yes 7809
```

特定の時刻範囲に限定したレポート作成

特定の時刻範囲に限定してログを分析したい場合、たとえば 2009 年 2 月 16 日の日中時間帯のみを調べたい場合は、まず対象のデータが現在の `audit.log` 内に含まれているか、それとも既にローテーションされてしまったものかどうかを確認します。これは `aureport -t` を実行することで判別することができます。

```
aureport -t

Log Time Range Report
=====================
/var/log/audit/audit.log: 2009年02月03日 14:13:38.225 - 2009年02月17日 15:30:01.636
```

現在の `audit.log` ファイル内に必要な時間帯のデータがすべて含まれていれば、通常通りにレポートを作成します。そうでない場合は、`aureport` コマンドに `-if` オプションを指定し、必要なデータを含むログファイルを指定します。

あとは開始日時と終了日時を指定して、作成するレポートの種類を選ぶオプションを加えます。下記の例では、ログイン試行に着目してレポートを作成しています:

```
> sudo aureport -ts 02/16/09 8:00 -te 02/16/09 18:00 -l

Login Report
============================================
# date time auid host term exe success event
============================================
1. 2009年02月16日 12:39:05 root: 192.168.2.100 sshd /usr/sbin/sshd no 2108
2. 2009年02月16日 12:39:12 0 192.168.2.100 /dev/pts/1 /usr/sbin/sshd yes 2114
3. 2009年02月16日 13:09:28 root: 192.168.2.100 sshd /usr/sbin/sshd no 2131
4. 2009年02月16日 13:09:32 root: 192.168.2.100 sshd /usr/sbin/sshd no 2133
5. 2009年02月16日 13:09:37 0 192.168.2.100 /dev/pts/2 /usr/sbin/sshd yes 2139
```

開始日時は `-ts` で指定します。指定した日時と同じか、それより後のイベントがレポート内に出力されます。日付部分を省略した場合、`aureport` は 今日 を指定したものとして扱います。時刻部分を省略した場合は、日付の切り替わった深夜を開始時刻として指定したものとして扱います。なお、日付は 月 / 日 / 年 の形式で指定することに注意してください。

終了日時は `-te` で指定します。指定した日時と同じか、それより前のイベントがレポート内に出力されます。日付を省略した場合、`aureport` は 今日 を指定したものとして扱います。また、時刻部分を省略した場合、`aureport` は 現在時刻 を指定したものとして扱います。指定可能な書式は `-ts` と同じです。

概要を除き、レポートは表形式で標準出力に出力されます。つまり、出力された情報を他のコマンドに取り込むことが容易であることになります。たとえば 41.8項「監査データの可視化」で紹介されている可視化スクリプトを使用すると、監査デーモンが生成したデータをさらに詳しく処理することができます。

41.6 ausearch による監査デーモンログへの問い合わせ

`aureport` では、システム内で何が起こっているのかを概要レベルで表示することができますが、特定のイベントの詳細までは追うことができません。特定のイベントを追跡したい場合は、`ausearch` ツールを使用します。

`ausearch` コマンドは `/var/log/audit/audit.log` にある監査ログ内を検索するためのツールで、キーや語句などを条件に指定して検索することができます。ただし、すべてのレコードタイプに同じ項目が含まれているわけではないことに注意してください。たとえば `PATH` レコードの場合、`hostname` や `uid` の項目はありません。

検索を行う場合、必要なレコードをすべて抽出できるような適切な検索条件を指定してください。その一方、特定のレコードタイプを抽出したり、それに付随するその他の関連レコードを抽出したりすることもできます。これは、カーネル内の様々な箇所で、イベントに関連する複数のレコードを出力する仕組みであるためです。たとえば `open` システムコールの場合、`SYSCALL` レコードだけでなく `PATH` レコードも同時に出力されます。



ヒント: 複数の検索オプションの使用について

複数のコマンドラインオプションを指定すると、それらは論理積 (AND) 演算子で結びつけられ、検索を絞り込む用途として使用することができます。

他のファイルからの監査ログ読み込み

監査ログを他のマシンに移動させた場合や、それぞれのマシンに接続することなく、ローカルのマシン内で複数のマシンの監査ログを分析したりしたい場合は、対象となるログファイルをローカルに移動したあと、`ausearch` を利用してローカルで分析してください:

```
> sudo ausearch - オプション -if ファイル名
```

数値表現の文字列変換

ユーザ ID などの情報は数値で出力されます。これらを読みやすいテキスト形式に変換するには、`ausearch` コマンドに `-i` オプションを追加してください。

監査イベント ID での検索

既に監査レポートを作成している場合や、`autrace` を実行している場合は、ログ内の特定のイベントに対して、証跡を分析する必要があります。41.5項「監査ログの仕組みとレポートの生成」で説明しているほとんどの種類のレポートには、出力内に監査イベント ID が含まれます。監査イベント ID は監査メッセージ ID の第 2 パートの部分ですが、ここには Unix エポックのタイムスタンプと監査イベント ID がコロン区切りで書かれています。また、同じシステムコール内であれば、同じイベント ID が割り当てられます。`ausearch` でこのイベント ID を使用することで、ログ内のイベント証跡を取得することができます。
下記のようにしてコマンドを使用してください:

```
> sudo ausearch -a 5207
----
time->Tue Feb 17 13:43:58 2009
type=PATH msg=audit(1234874638.599:5207): item=0 name="/var/log/audit/audit.log"
 inode=1219041 dev=08:06 mode=0100644 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1234874638.599:5207):  cwd="/root"
type=SYSCALL msg=audit(1234874638.599:5207): arch=c000003e syscall=2 success=yes
 exit=4 a0=62fb60 a1=0 a2=31 a3=0 items=1 ppid=25400 pid=25616 auid=0 uid=0 gid=0
 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1 ses=1164 comm="less" exe="/usr/
bin/less" key="doc_log"
```

`ausearch -a` コマンドを使用すると、指定したイベント ID に関連するログ内のすべてのレコードを検索して、表示します。このオプションは、他のオプションと組み合わせて指定することもできます。

メッセージタイプによる検索

特定のメッセージタイプの監査レコードを検索するには、`ausearch -m メッセージタイプ` コマンドを実行します。指定できるメッセージタイプには、`PATH`，`SYSCALL`，`USER_LOGIN` などがあります。メッセージタイプを指定せずに `ausearch -m` を実行すると、すべてのメッセージタイプの一覧を表示することができます。

ログイン ID による検索

特定のログインユーザ ID に関連するレコードを表示するには、`ausearch -ul` コマンドを使用します。このコマンドは、指定したログインユーザ ID に関連し、ログインに成功したすべてのレコードを取得します。

ユーザ ID による検索

特定の実ユーザ ID および実効ユーザ ID の両方に関連するレコードを表示するには、`ausearch -ua` コマンドを使用します。特定の実ユーザ ID に関連するレコードを表示するには、`ausearch -ui UID` のように入力して実行します。特定の実効ユーザ ID に関連するレ

コードを表示するには、`ausearch -ue EUID` のように入力して実行します。実ユーザ ID での検索はプロセスを作成したユーザの ID を、実効ユーザ ID での検索は、そのプロセスを実行するのに必要なユーザ ID と権限の ID を意味しています。

グループ ID による検索
: 特定の実グループ ID および実効グループ ID の両方に関連するレコードを表示するには、`ausearch -ga` コマンドを使用します。特定の実グループ ID に関連するレコードを表示するには、`ausearch -gi GID` のように入力して実行します。特定の実効グループ ID に関連するレコードを表示するには、`ausearch -ge EGID` のように入力して実行します。

コマンドライン名による検索
: 特定のコマンドに関連するレコードを表示するには、`ausearch -c コマンド名` コマンドを使用します。たとえば `ausearch -c less` のように入力して実行すると、`less` コマンドに関連するすべてのレコードを表示することができます。

実行ファイル名による検索
: 特定の実行ファイルに関連するレコードを表示するには、`ausearch -x 実行ファイルのパス` コマンドを使用します。たとえば `ausearch -x /usr/bin/less` のように入力して実行すると、`/usr/bin/less` の実行ファイルに関連するすべてのレコードを表示することができます。

システムコール名による検索
: 特定のシステムコールに関連するレコードを表示するには、`ausearch -sc システムコール名` コマンドを使用します。たとえば `ausearch -sc open` のように入力して実行すると、`open` システムコールに関連するすべてのレコードを表示することができます。

プロセス ID による検索
: 特定のプロセス ID に関連するレコードを表示するには、`ausearch -p プロセス_ID` コマンドを使用します。たとえば `ausearch -p 13368` のように入力して実行すると、指定したプロセス ID に関連するすべてのレコードを表示することができます。

イベントもしくはシステムコールの成功可否による検索
: システムコールの成功可否を指定してレコードを表示するには、`ausearch -sv 成功可否` コマンドを使用します。たとえば `ausearch -sv yes` のように入力して実行すると、成功したシステムコールに関連するすべてのレコードを表示することができます。

ファイル名による検索
: ファイル名を指定してレコードを表示するには、`ausearch -f ファイル名` コマンドを使用します。たとえば `ausearch -f /foo/bar` のように入力して実行すると、`/foo/bar` ファイルに関連するすべてのレコードを表示することができます。なお、ファイル名だけの指定でも問題なく動作しますが、相対パスでの指定はできません。

端末による検索

特定の端末に関連するレコードを表示するには、`ausearch -tm 端末名` コマンドを使用します。たとえば `ausearch -tm ssh` のように入力して実行すると、SSH 端末に関連するすべてのレコードを表示することができますし、`ausearch -tm tty` のように入力して実行すると、コンソールに関連するすべてのレコードを表示することができます。

ホスト名による検索

特定のホストに関連するレコードを表示するには、`ausearch -hn ホスト名` コマンドを使用します。たとえば `ausearch -hn jupiter.example.com` のように入力して実行します。ホスト名や完全修飾ホスト名、ネットワークアドレスのいずれかを指定することができます。

キー項目による検索

あらかじめ特定の種類のイベントを識別するために設定して記録しておいたキー項目を指定してレコードを表示するには、`ausearch -k キー項目` コマンドを使用します。たとえば `ausearch -k CFG_etc` のように入力して実行すると、`CFG_etc` というキーを含むレコードのみを表示します。

単語による検索

監査ルールセット内に含まれる、さまざまなイベントの情報に対して、文字列での検索を行うこともできます。文字列はファイル名やホスト名のほか、端末名に対しても検索することができます。具体的には `ausearch -w 文字列` のように入力して実行してください。

特定の時刻範囲に限定した検索

`-ts` と `-te` の各オプションを指定することで、特定の時間帯範囲のものを検索することができるようになります。`-ts` オプションでは開始日時を、`-te` オプションでは終了日時をそれぞれ指定します。これらのオプションは、上述の任意のオプションと組み合わせて使用することができます。また、使用方法は `aureport` と同じです。

41.7 `autrace` によるプロセスの解析

監査システムでは、設定したルールを利用してシステムを監視するだけでなく、`autrace` コマンドを使用することで、特定のプロセス専用の監査を実施することもできます。`autrace` は `strace` コマンドのように動作するコマンドですが、少し異なる情報を採取します。また、`autrace` の出力は `/var/log/audit/audit.log` に書き込まれ、標準的な監査ログ項目と同様の書式で出力されます。

プロセスに対して `autrace` を実行する前に、まずはキュー内から監査ルールすべてを削除して、`autrace` 自身の監査と衝突しないようにしておいてください。監査ルールの削除は `auditctl -D` で行います。これにより、すべての通常監査が停止します。

```
> sudo auditctl -D

No rules

autrace /usr/bin/less

Waiting to execute: /usr/bin/less
Cleaning up...
No rules
Trace complete. You can locate the records with 'ausearch -i -p 7642'
```

`autrace` で実行ファイルを指定する際には、必ずフルパスで指定してください。追跡が完了すると、`autrace` はトレースのイベント ID を通知しますので、証跡を `ausearch` で検索できるようになります。作業後は監査ルールを元に戻すため、`systemctl restart auditd` と入力して実行してください。

41.8 監査データの可視化

`/var/log/audit/audit.log` にある監査証跡も、41.5.2項「独自の監査レポートの生成」で説明している様々な種類の `aureport` のレポートも、ユーザが読む場合にはある程度の経験が必要となります。`aureport` の出力は表形式であるため、sed, Perl, awk などのスクリプトを使用することで、容易に監査データを可視化することができます。

可視化に対応するスクリプト (詳しくは 42.6項「ログの可視化の設定」をお読みください) は、openSUSE Leap やその他の Linux ディストリビューションで、読みやすい監査出力を作成するための標準的な Linux ツールの使い方の例となっているものです。下記の例では、純粋なテキスト形式の監査レポートを、グラフに変換する作業を行っています。

最初の例では、プログラムとシステムコールの対応付けを可視化しているものです。この種類のデータを取得するには、最終的なグラフを生成するための元となるデータを `aureport` で作成する必要があります:

```
> sudo aureport -s -i

Syscall Report
=======================================
# date time syscall pid comm auid event
=======================================
1. 2009年02月16日 17:45:01 open 20343 cron unset 2279
2. 2009年02月16日 17:45:02 mkdir 20350 mktemp root 2284
3. 2009年02月16日 17:45:02 mkdir 20351 mkdir root 2285
...
```

このレポートに対して可視化スクリプトが行うべき最初の処理は、必要な列の情報のみを抽出することです。この例では、`syscall` と `comm` の列になります。出力は並べ替えられたあと重複を排除し、グラフ作成プログラム自身に渡すことになります:

```
LC_ALL=C aureport -s -i | awk '/^[0-9]/ { print $6" "$4 }' | sort | uniq | mkgraph
```

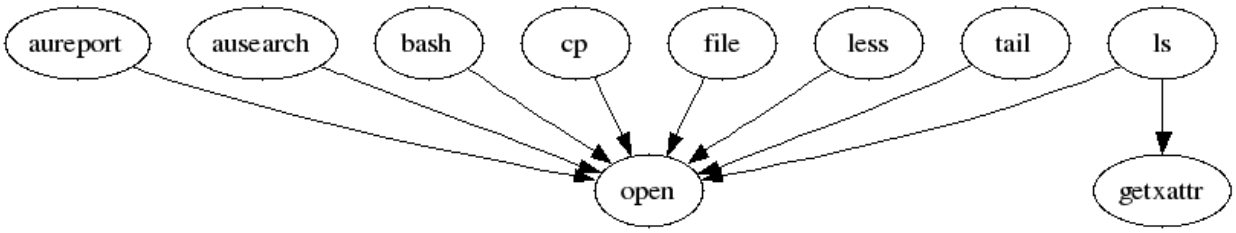


図 41.2: フローグラフ: プログラムとシステムコールの関係性

2 つ目の例では、様々な種類のイベントとそのログ記録数をまとめたグラフを示しています。この種類のグラフを作成するにあたって、`aureport` で対応するコマンドは `aureport -e` です:

```
> sudo aureport -e -i --summary

Event Summary Report
======================
total  type
======================
2434  SYSCALL
816  USER_START
816  USER_ACCT
814  CRED_ACQ
810  LOGIN
806  CRED_DISP
779  USER_END
99  CONFIG_CHANGE
52  USER_LOGIN
```

この種類のレポートは既に 2 列のみの出力になっていることから、単純にグラフ作成プログラムに渡して棒グラフにするだけです。

```
> sudo aureport -e -i --summary  | mkbar events
```

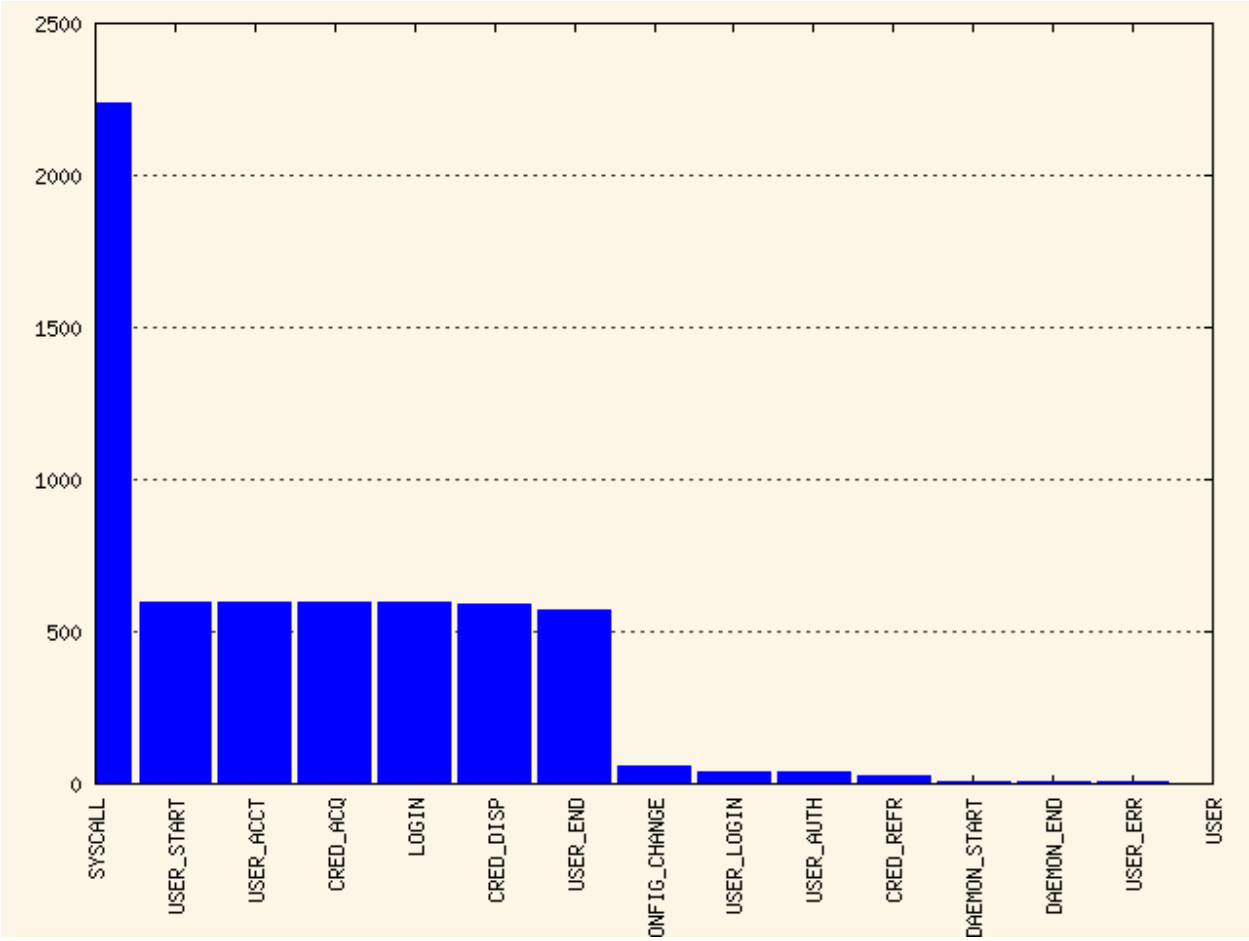


図 41.3: 棒グラフ: 一般的なイベントの種類

監査データの可視化の背景となる情報について、詳しくは https://people.redhat.com/sgrubb/audit/visualize/index.html (英語) をお読みください。

41.9 監査イベント通知の中継

監査システムでは、リアルタイムに外部プログラムを呼び出して、`auditd` にアクセスしたり、使用したりすることができます。この機能は 監査ディスパッチャ と呼ばれる仕組みで、たとえば侵入検知システムなどが `auditd` を使用し、より詳しい検知情報を取得したりすることができます。

`auditd` の設定ファイルは `/etc/audit/auditd.conf` に保存されています。このファイルには下記のようなオプションを記述することができます:

`q_depth`

監査ディスパッチャの内部キューのサイズを指定します。syslog 内に監査イベントの喪失を表すメッセージが現れた場合は、この値を増やしてください。既定値は 250 です。

`overflow_action`

監査ディスパッチャの内部キューが溢れてしまった場合、監査デーモンがどのように対応するのかを指定します。指定可能な値は `ignore` (何もしない), `syslog` (syslog に警告を発する), `suspend` (イベントの処理を停止させる), `single` (コンピュータシステムをシングルユーザモードに移行させる), `halt` (システムをシャットダウンする) のいずれかです。

`priority_boost`

監査イベントディスパッチャに対する優先度設定を指定します (監査デーモン自身の優先度からの相対指定です) 。既定値は 4 で、優先度を変更しません。

`name_format`

監査イベント内へ挿入するコンピュータノード名の設定方法を指定します。指定可能な値は `none` (コンピュータ名を挿入しない), `hostname` (`gethostname` システムコールで取得した名前), `fqd` (マシンの完全修飾ホスト名), `numeric` (マシンの IP アドレス), `user` (`name` オプションで指定する任意の値) のいずれかです。既定値は `none` です。

`name`

マシンを識別するためのユーザ定義の文字列です。このオプションを動作させるには、`name_format` オプションが `user` でなければなりません。それ以外の場合、このオプションは無視されます。

`max_restarts`

非負の整数を入れる項目で、監査イベントディスパッチャに対して、何回までクラッシュしたプラグインを再起動しようとするかを指定します。既定値は 10 です。

例 41.10: /ETC/AUDIT/AUDITD.CONF の例

```
q_depth = 250
overflow_action = SYSLOG
priority_boost = 4
name_format = HOSTNAME
#name = mydomain
```

プラグインプログラムでは、自身の設定ファイルを専用のディレクトリである `/etc/audit/plugins.d` にインストールします。プラグインの設定ファイルには、下記のようなオプションがあります:

`active`

プログラムが `auditd` を使用するかどうかを指定します。指定可能な値は `yes` (はい), `no` (いいえ) のいずれかです。

`direction`

プラグインと監査システムの通信方法を指定します。これはイベントディスパッチャに対して、イベントの流れる方向を示すことになります。指定可能な値は `in` (入力), `out` (出力) のいずれかです。

`path`

プラグインの実行ファイルに対する絶対パスを指定します。内部プラグインの場合、ここにはプラグイン名を指定します。

`type`

プラグインの動作方法を指定します。指定可能な値は `builtin`, `always` のいずれかです。`builtin` は内部プラグイン (`af_unix` および `syslog`) であることを示すもので、`always` はほとんどすべて (ただし全てではありません) のプラグイン向けの値です。既定値は `always` です。

`args`

プラグインプログラムに渡すパラメータを指定します。通常、プラグインプログラムは自身の設定ファイルからパラメータを読み込む仕組みであるため、何もパラメータを指定する必要はありません。また、2 個までのパラメータに対応しています。

`format`

監査ディスパッチャがプラグインプログラムに渡すデータの形式を指定します。指定可能な値は `binary`, `string` のいずれかです。`binary` はイベントディスパッチャが監査デーモンから受信したデータをそのまま渡す指定で、`string` はディスパッチャに対して、イベントデータを監査処理ライブラリで処理できる文字列に変換する指定です。既定値は `string` です。

例 41.11: /ETC/AUDIT/PLUGINS.D/SYSLOG.CONF の例

```
active = no
direction = out
path = /sbin/audisp-syslog
type = always
args = LOG_INFO
format = string
```

42 Linux 監査フレームワークの設定

改訂履歴

2024-05-10

本章では、シンプルな監査環境を構築するまでの説明を行っています。各手順には設定や監査の有効化に関する詳細な説明が含まれています。下記の内容を学習したあとは、第43章「監査ルールセットの紹介」に示されている実際の例に読み進めてください。

openSUSE Leap で監査を設定するには、下記の手順を踏む必要があります:

手順 42.1: LINUX 監査フレームワークの設定

1. まずは `audit` パッケージをインストールします。また、42.6項「ログの可視化の設定」で説明しているログの可視化機能が必要である場合は、`gnuplot` と `graphviz` の各パッケージもインストールしてください。
2. 監査対象のコンポーネントを決定します。詳しくは 42.1項「監査対象のコンポーネントの決定」をお読みください。
3. 基本的な監査デーモンの設定を確認し、必要であれば修正します。詳しくは 42.2項「監査デーモンの設定」をお読みください。
4. システムコールに対する監査を有効化します。詳しくは 42.3項「システムコールに対する監査の有効化」をお読みください。
5. お使いの目的にあわせて監査ルールを構築します。詳しくは 42.4項「監査ルールの設定」をお読みください。
6. ログを生成し、目的にあったレポートを設定します。詳しくは 42.5項「監査レポートの設定」をお読みください。
7. 必要であれば、ログの可視化を行います。詳しくは 42.6項「ログの可視化の設定」をお読みください。

重要: 監査デーモンの制御について

監査システムのコンポーネントの設定を行う場合は、あらかじめ `root` で `systemctl status auditd` を実行し、監査デーモンが動作していないことを確認してください。既定の openSUSE Leap では監査システムがシステムの起動時に動作するように設定されてしまっていますので、`systemctl stop auditd` を実行して停止させる必要があります。設定が完了したら、`systemctl start auditd` を実行して監査デーモンを開始してください。

42.1 監査対象のコンポーネントの決定

独自の監査設定を作成する前に、まずはどの程度監査するのかを決定する必要があります。下記の一般的な判断基準を利用して、用途や環境に合わせたコンポーネントを選択してください:

- CAPP/EAL 認証を取るために完全なセキュリティ監査を行う必要がある場合は、システムコールに対して完全な監査を行い、様々なファイルやディレクトリに対して監視を設定してください。ルールセットは 第43章「監査ルールセットの紹介」 に示されているものをベースにすることができます。
- 監査ルールをベースにしたプロセス追跡を行う必要がある場合は、`autrace` を使用します。
- 重要なデータや機密データを含むファイルやディレクトリに対するアクセスを追跡し、監視したい場合は、これらの要件に沿ったルールセットを作成する必要があります。42.3項「システムコールに対する監査の有効化」 に示されている手順で監査を有効化し、その後 42.4項「監査ルールの設定」 に進んでください。

42.2 監査デーモンの設定

監査デーモンの基本的な設定は、`/etc/audit/auditd.conf` ファイルの編集で行うことができます。[YaST] › [セキュリティとユーザ] › [Linux 監査フレームワーク (LAF)] を選択することで、YaST から基本設定を行うこともできます。YaST から設定を行う場合は、[ログファイル] と [ディスク容量] の各タブを使用してください。

```
log_file = /var/log/audit/audit.log
log_format = RAW
log_group = root
priority_boost = 4
flush = INCREMENTAL
freq = 20
num_logs = 5
disp_qos = lossy
dispatcher = /sbin/audispd
name_format = NONE
##name = mydomain
max_log_file = 6
max_log_file_action = ROTATE
space_left = 75
space_left_action = SYSLOG
action_mail_acct = root
admin_space_left = 50
admin_space_left_action = SUSPEND
disk_full_action = SUSPEND
disk_error_action = SUSPEND
##tcp_listen_port =
tcp_listen_queue = 5
tcp_max_per_addr = 1
##tcp_client_ports = 1024-65535
tcp_client_max_idle = 0
cp_client_max_idle = 0
```

多くの環境では既定の設定で問題なく動作します。ただし、`num_logs`，`max_log_file`，`space_left`，`admin_space_left` などの各値については、お使いの環境のサイズに合わせて調整してください。ディスク領域が少ない場合は、ローテーション時に保持しておく古いログファイルの数を減らすとともに、ディスク領域が残り少なくなった場合により早く警告を得られるよう設定を行ってください。CAPP 準拠の設定を行う場合は、41.2項「監査デーモンの設定」で説明しているとおりに `log_file`，`flush`，`max_log_file`，`max_log_file_action`，`space_left`，`space_left_action`，`admin_space_left`，`admin_space_left_action`，`disk_full_action`，`disk_error_action` の設定を変更してください。CAPP 準拠の設定は、たとえば下記のようになります:

```
log_file = 監査ログ記録専用のパーティションのパス/audit.log
log_format = RAW
priority_boost = 4
flush = SYNC                        ### DATA でもかまいません
freq = 20
num_logs = 4
dispatcher = /sbin/audispd
disp_qos = lossy
max_log_file = 5
max_log_file_action = KEEP_LOGS
space_left = 75
```

```
space_left_action = EMAIL
action_mail_acct = root
admin_space_left = 50
admin_space_left_action = SINGLE   ### HALT でもかまいません
disk_full_action = SUSPEND         ### HALT でもかまいません
disk_error_action = SUSPEND        ### HALT でもかまいません
```

`###` 以降はコメントで、その他の選択肢を説明しているものです。実際の設定ファイル内には、このようなコメントは追加しないでください。



ヒント: さらなる情報

`auditd.conf` の設定パラメータについて、背景となる詳細情報を読みたい場合は、41.2項「監査デーモンの設定」を参照してください。

42.3 システムコールに対する監査の有効化

監査フレームワークがインストールされていない場合は、まず `audit` をインストールしてください。標準的な openSUSE Leap システムでは、auditd が既定で 動作しています。システムの起動時に開始するよう設定されていない場合は、下記のようにして有効化してください:

```
> sudo systemctl enable auditd
```

利用可能な監査機能には、下記のものがあります:

基本的なログ機能

openSUSE Leap の標準状態 (特に何も設定を行っていない状態) では、auditd は `/var/log/audit/audit.log` に対する設定変更のイベントのみを記録します。また、カーネルの監査コンポーネントは、`auditctl` で制御を行わない限り、ファイルアクセスやシステムコールなどのイベントも生成しません。しかしながら、その他のカーネルコンポーネントやモジュールが、`auditctl` の制御範囲外で監査イベントを生成し、監査ログ内に現れることがあります。既定では、AppArmor モジュールのみが監査イベントを生成します。

システムコールを監査する高度なログ機能

システムコールを監視し、意味のあるファイル監視を実施したい場合は、システムコールに対する監査コンテキストを有効化する必要があります。

通常のファイルやディレクトリを監視するよう設定している場合で、システムコールの監査機能も必要とする場合は、システムコールに対する監査コンテキストを有効化する必要があります。システムを再起動するまでの間にのみ監査コンテキストを有効化したい場合は、`root` で `auditctl -e 1` と入力して実行します。また、この機能を無効化したい場合は、同様に `root` で `auditctl -e 0` と入力して実行します。

監査コンテキストは既定で有効化されています。この機能を一時的に無効化したい場合は、`auditctl -e 0` と入力して実行してください。

42.4 監査ルールの設定

監査ルールを使用することで、監査システムがシステムのどの部分を分析するのかを決定することになります。通常は重要なデータベースやセキュリティに関連する設定ファイルを監視の対象とします。お使いのシステムに幅広い分析が必要である場合は、様々なシステムコールを分析することもあります。CAPP 準拠の環境下で必要となるほとんどのルールを網羅する、詳細な設定例は 第43章「監査ルールセットの紹介」に示されています。

監査ルールはコマンドラインツールである `auditctl` を利用することで、監査デーモンに渡すことができるほか、`/etc/audit/audit.rules` 内にルールセットを記述することでも構成することができます (この場合、監査デーモンの起動時に読み込まれます) 。`/etc/audit/audit.rules` ファイルはエディタで編集することができるほか、YaST でカスタマイズすることもできます。YaST でカスタマイズする場合は、[セキュリティとユーザ] › [Linux 監査フレームワーク (LAF)] › [「auditctl」のルール] から行ってください。なお、コマンドラインで渡されたルールはファイルに保存されたりすることがありませんので、監査デーモンを再起動するたびに再入力を行う必要があります。

いくつかの主要なファイルとディレクトリに対して、基本的に設定のみを行ったシンプルなルールセットは下記のとおりです:

```
# 基本的な監査システムのパラメータ
-D
-b 8192
-f 1
-e 1

# ファイルやディレクトリの監視（キー機能付き）
-w /var/log/audit/ -k LOG_audit
-w /etc/audit/auditd.conf -k CFG_audit_conf -p rxwa
-w /etc/audit/audit.rules -k CFG_audit_rules -p rxwa

-w /etc/passwd -k CFG_passwd -p rwxa
-w /etc/sysconfig/ -k CFG_sysconfig

# システムコールルールの例
-a entry,always -S umask

### 以下に独自のルールを追加してください
```

基本的な監査システムのパラメータ (たとえば `-b` で指定することのできるバックログパラメータなど) を変更する場合は、あらかじめ実際の監査ルールセットで問題がないかどうかをテストしておいてください。たとえば `-b` であれば、現在の監査ルールセットで生成されるログの量に対して、バックログの値が

適切なレベルかどうかを確認する必要があります。選択したバックログの値が小さすぎる場合は、お使いのシステムが監査の負荷に耐えられないことがありますので、バックログ制限を超過することで失敗フラグ (`-f`) が設定されてしまうことがあります。

重要: 失敗フラグの選択について

失敗フラグを設定する場合、`-f 2` はお使いの監査システムに対する制限を超過した場合、システムを即時にシャットダウンさせる設定であり、ディスクに未書き込みのデータが存在していても、それを書き込まずにシャットダウンさせてしまうことに注意してください。この方式によるシャットダウンは正常終了ではありませんので、セキュリティを最重視するシステムにのみ使用するものとし、通常は `-f 1` を設定して、警告を発して監査システムを停止させるだけの処理に留めてください。これにより、データの損失や破壊などを防ぐことができます。

ディレクトリに対する監視は、そのディレクトリ内にあるファイルに対する監視に比べると、冗長性に欠ける出力になります。たとえば `/etc/sysconfig` ディレクトリ内にあるシステム設定ファイルに対して、詳細なログを採取したい場合は、各ファイルに対して監視を設定してください。ただし、監査ルールではグロブ (ワイルドカード) を使用することができないことに注意してください。たとえば `/etc` ディレクトリ内のファイルやディレクトリを監視する目的で、`-w /etc/*` のようなルールを作成しても、正しく動作しません。

ログファイル内での識別を行う目的で、ファイルやディレクトリに対する監視にキーを設定することができます。キーを使用することで、特定のルールに関連するイベントを容易に取り出すことができるようになります。また、キーを設定する場合は、単なるログファイルの監視と設定ファイルの監視を区別して設定することをお勧めします。この場合、`LOG` で始まるキーをログファイル監視用に、`CFG` で始まるキーを設定ファイル監視用に使用しています。それ以外にも、キーの一部をファイル名と同じに設定しておくと、ログファイル内で必要な項目を簡単に見つけられるようになります。

また、ファイルやディレクトリの作成に対する監視に対してもう 1 つ注意しておくべき点として、ルールの作成時に存在していないファイルに対しては、監視を設定することができないという点です。監査システムが動作している間にシステムに追加されたファイルは、そのファイルに対して後からルールセットに追加しない限り、監視することができません。

なお、独自のルールの作成方法について、詳しくは 41.4項「監査システムに対するパラメータ指定」をお読みください。

重要: 監査ルールの変更について

監査ルールを変更した後は、変更点を読み込ませるため、`systemctl restart auditd` と入力して実行し、監査デーモンを再起動してください。

42.5 監査レポートの設定

未加工の監査ログを読んでお使いのシステムの動作を確認するような面倒な事態を避けるため、時間範囲を指定して独自の監査レポートを作成し、手間を省くことをお勧めします。独自の監査レポートを読むことで、注目したい範囲を絞り込むことができるほか、お使いのシステムの特徴を統計から取得し、更に細かいイベント監視の基礎資料とすることができるようになります。さらに細かく個別のイベントの詳細を追跡したい場合は、ausearch ツールをお使いください。

監査レポート機能を設定する前に、まずは下記を考慮しておくことをお勧めします:

- どのような種類のイベントに対して定期的なレポートを生成し、監視を行うのか?... 41.5.2項「独自の監査レポートの生成」で説明している aureport の使用方法を参照して、適切なコマンドラインを作成してください。
- 監査レポートで何をするのか? ... 蓄積されたデータからグラフを作成するのか、それとも表計算やデータベースシステムなどに取り込んで利用するのか、などです。レポートを可視化したい場合は、42.6項「ログの可視化の設定」での例などを参考にして、aureport のコマンドラインを設定し、処理を進めてください。
- レポートの作成間隔とタイミングをどうするのか? ... cron 等を利用して適切な自動化を構成してください。

この例では、監査システムや PAM 、システムの設定へのアクセス試行に対して監査を行う想定でレポートを作成しています。お使いのシステム内のファイルイベントに関するレポートを作成するには、下記の手順で行います:

1. 全てのイベントに対して完全な概要レポートを生成していて、何らかの異常が存在していないかどうかを調べたい場合は、「failed syscalls」の箇所をご確認ください。この失敗は、許可のないファイルに対するアクセスのほか、存在していないファイルに対するアクセスなどが含まれるためです:

```
> sudo aureport

Summary Report
======================
Range of time in logs: 2009年02月03日 14:13:38.225 - 2009年02月17日 16:30:10.352
Selected time for report: 2009年02月03日 14:13:38 - 2009年02月17日 16:30:10.352
Number of changes in configuration: 24
Number of changes to accounts, groups, or roles: 0
Number of logins: 9
Number of failed logins: 15
Number of authentications: 19
Number of failed authentications: 578
Number of users: 3
Number of terminals: 15
Number of host names: 4
Number of executables: 20
Number of files: 279
Number of AVC's: 0
Number of MAC events: 0
Number of failed syscalls: 994
Number of anomaly events: 0
Number of responses to anomaly events: 0
Number of crypto events: 0
Number of keys: 2
Number of process IDs: 1238
Number of events: 5435
```

2. さらに失敗イベントに対する概要レポートを作成する (`--failed`) と、「files」内にファイルアクセスの失敗数が表示されるようになります:

```
> sudo aureport --failed

Failed Summary Report
======================
Range of time in logs: 2009年02月03日 14:13:38.225 - 2009年02月17日 16:30:10.352
Selected time for report: 2009年02月03日 14:13:38 - 2009年02月17日 16:30:10.352
Number of changes in configuration: 0
Number of changes to accounts, groups, or roles: 0
Number of logins: 0
```

```
Number of failed logins: 15
Number of authentications: 0
Number of failed authentications: 578
Number of users: 1
Number of terminals: 7
Number of host names: 4
Number of executables: 12
Number of files: 77
Number of AVC's: 0
Number of MAC events: 0
Number of failed syscalls: 994
Number of anomaly events: 0
Number of responses to anomaly events: 0
Number of crypto events: 0
Number of keys: 2
Number of process IDs: 713
Number of events: 1589
```

3. アクセスのできなかったファイルを一覧表示したい場合は、さらにファイルイベントに限定してレポートを作成します:

```
> sudo aureport -f -i --failed --summary

Failed File Summary Report
===========================
total  file
===========================
80  /var
80  spool
80  cron
80  lastrun
46  /usr/lib/locale/en_GB.UTF-8/LC_CTYPE
45  /usr/lib/locale/locale-archive
38  /usr/lib/locale/en_GB.UTF-8/LC_IDENTIFICATION
38  /usr/lib/locale/en_GB.UTF-8/LC_MEASUREMENT
38  /usr/lib/locale/en_GB.UTF-8/LC_TELEPHONE
38  /usr/lib/locale/en_GB.UTF-8/LC_ADDRESS
38  /usr/lib/locale/en_GB.UTF-8/LC_NAME
38  /usr/lib/locale/en_GB.UTF-8/LC_PAPER
38  /usr/lib/locale/en_GB.UTF-8/LC_MESSAGES
38  /usr/lib/locale/en_GB.UTF-8/LC_MONETARY
38  /usr/lib/locale/en_GB.UTF-8/LC_COLLATE
38  /usr/lib/locale/en_GB.UTF-8/LC_TIME
38  /usr/lib/locale/en_GB.UTF-8/LC_NUMERIC
8  /etc/magic.mgc
...
```

`/etc/audit/auditd.conf`，`/etc/pam.d`，`/etc/sysconfig` など、特定のファイルやディレクトリに対してレポートを作成したい場合は、下記のようなコマンドを実行します:

```
> sudo aureport -f -i --failed --summary |grep -e "/etc/audit/auditd.conf" -e "/etc/
pam.d/" -e "/etc/sysconfig"

1  /etc/sysconfig/displaymanager
```

4. 上記の概要レポートを読むことで、ようやくログファイルの中から調査対象を絞り込むことができました。あとはさらに詳しく調べるため、イベント ID を取得します:

```
> sudo aureport -f -i --failed |grep -e "/etc/audit/auditd.conf" -e "/etc/pam.d/" -e
 "/etc/sysconfig"

993. 2009年02月17日 16:47:34 /etc/sysconfig/displaymanager readlink no /bin/vim-
normal root 7887
994. 2009年02月17日 16:48:23 /etc/sysconfig/displaymanager getxattr no /bin/vim-
normal root 7889
```

5. あとはイベント ID を指定して詳細情報を取得します:

```
> sudo ausearch -a 7887 -i
----
time->Tue Feb 17 16:48:23 2009
type=PATH msg=audit(1234885703.090:7889): item=0 name="/etc/sysconfig/
displaymanager" inode=369282 dev=08:06 mode=0100644 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1234885703.090:7889):  cwd="/root"
type=SYSCALL msg=audit(1234885703.090:7889): arch=c000003e syscall=191 success=no
 exit=-61 a0=7e1e20 a1=7f90e4cf9187 a2=7fffed5b57d0 a3=84 items=1 ppid=25548
 pid=23045 auid=0 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2
 ses=1166 comm="vim" exe="/bin/vim-normal" key=(null)
```



ヒント: 特定の時間帯のみに着目したい場合について

特定の日時に発生したイベントに着目したい場合は、`aureport` コマンドに開始と終了の日時を指定（`-ts` および `-te`）して、範囲を狭めることができます。詳しくは 41.5.2項「独自の監査レポートの生成」をお読みください。

最後のものを除いて、全ての手順はスクリプトから自動実行することができますので、cron などのジョブとして実行することができます。たとえば `--failed --summary` オプションを指定したものについては、ファイルとアクセス失敗数を元にグラフ化することも簡単にできます。監査レポートデータの可視化について、詳しくは 42.6項「ログの可視化の設定」をお読みください。

42.6 ログの可視化の設定

`mkbar` や `mkgraph` のスクリプトを使用することで、グラフの形で監査の統計情報を可視化することができます。`aureport` コマンドと同様に、グラフ化のコマンドもスクリプトとして実行することができますので、グラフ化までの部分を簡単に cron ジョブにすることができます。

`mkbar` や `mkgraph` のスクリプトは、Red Hat 社の Steve Grubb 氏が作成したものです。スクリプト本体は https://people.redhat.com/sgrubb/audit/visualize/ からダウンロードすることができます。なお、openSUSE Leap の現在の監査システムには、これらのスクリプトが含まれていませんので、下記の手順に従って作業を行ってください:



警告: ダウンロードコンテンツの危険性について

`mkbar` や `mkgraph` はインターネット上に公開されているソフトウエアであり、openSUSE Leap の管理外にあるものです。使用にあたっては、ご自身の責任の下でお願いいたします。インターネットからのソフトウエアのダウンロードは、特にそれを `root` 権限で動作させるような場合は、基本的に危険を伴うものであることに留意してください。

1. スクリプトを `root` のホームディレクトリ以下の `~/bin` にダウンロードします:

```
> sudo wget http://people.redhat.com/sgrubb/audit/visualize/mkbar -O ~/bin/mkbar
> sudo wget http://people.redhat.com/sgrubb/audit/visualize/mkgraph -O ~/bin/mkgraph
```

2. `root` に対して、読み込みと書き込み、実行のファイルパーミッションを設定します:

```
> sudo chmod 744 ~/bin/mk{bar,graph}
```

42.5項「監査レポートの設定」などで作成した概要レポートのグラフを作成するには、`mkbar` スクリプトを使用します。たとえば下記のように実行します:

イベントの概要グラフの作成

```
> sudo aureport -e -i --summary | mkbar events
```

ファイルイベントの概要グラフの作成

```
> sudo aureport -f -i --summary | mkbar files
```

ログインイベントの概要グラフの作成

```
> sudo aureport -l -i --summary | mkbar login
```

ユーザイベントの概要グラフの作成

```
> sudo aureport -u -i --summary | mkbar users
```

システムコールイベントの概要グラフの作成

```
> sudo aureport -s -i --summary | mkbar syscalls
```

上述のイベントタイプに対する失敗イベントのみの概要グラフを作成したい場合は、それぞれの `aureport` コマンド内に `--failed` オプションを追加してください。また、特定の時間帯のみに絞りたい場合は、それぞれの `aureport` コマンド内に `-ts` および `-te` オプションを追加してください。また、これらのコマンドラインに `grep` や `egrep` を追加して、正規表現で出力をフィルタすることで、さらに細かく調整を行うことができます。詳しくは `mkbar` スクリプト内のコメントなどをお読みください。また、上記のコマンドラインはいずれも、PNG ファイルの書かれた棒グラフを生成します。

ユーザとシステムコールなど、様々な監査オブジェクト同士の関係性を可視化したい場合は、`mkgraph` などのスクリプトをお使いください。たとえば下記のように実行します:

ユーザと実行ファイルの関係性グラフ

```
> sudo LC_ALL=C aureport -u -i | awk '/^[0-9]/ { print $4" "$7 }' | sort | uniq |
 mkgraph users_vs_exec
```

ユーザとファイルの関係性グラフ

```
> sudo LC_ALL=C aureport -f -i | awk '/^[0-9]/ { print $8" "$4 }' | sort | uniq |
 mkgraph users_vs_files
```

システムコールとコマンドの関係性グラフ

```
> sudo LC_ALL=C aureport -s -i | awk '/^[0-9]/ { print $4" "$6 }' | sort | uniq |
 mkgraph syscall_vs_com
```

システムコールとファイルの関係性グラフ

```
> sudo LC_ALL=C aureport -s -i | awk '/^[0-9]/ { print $5" "$4 }' | sort | uniq |
 mkgraph | syscall_vs_file
```

グラフを組み合わせることで、複雑な関係性を示すこともできます。詳しい情報と使用例については、`mkgraph` スクリプト内のコメントをお読みください。また、このスクリプトで生成されるグラフは既定では PostScript 形式ですが、スクリプト内の `EXT` 変数の内容を `ps` から `png` や `jpg` に変更することで、形式を変更することもできます。

43 監査ルールセットの紹介

改訂履歴

2024-05-10

下記の設定例では、どのようにして監査システムからお使いのシステムを監視することができるのかについて説明しています。また、Controlled Access Protection Profile (CAPP) で指定されている監査対象のイベントをカバーするのに必要な項目について、主なものを紹介しています。

ルールセットの例は下記のようなパーツから構成されています:

- 基本的な監査設定 (43.1項「基本的な監査設定パラメータの追加」)
- 監査ログファイルと監査設定ファイルに対する監視 (43.2項「監査ログファイルと設定ファイルに対する監視の追加」)
- ファイルシステムオブジェクトに対する操作の監視 (43.3項「ファイルシステムオブジェクトの監視」)
- セキュリティデータベースの監視 (43.4項「セキュリティ設定ファイルとデータベースの監視」)
- その他のシステムコールの監視 (43.5項「その他のシステムコールに対する監視」)
- システムコールのパラメータのフィルタリング (43.6項「システムコールのパラメータでのフィルタリング」)

下記に示す設定例を、実際に使用する設定ファイルに変換するには、下記の手順を実施してください:

1. まずはお使いの環境に合った設定を選んで、必要に応じて調整を行ってください。
2. 下記の設定例から、`/etc/audit/audit.rules` ファイルに対してルールを追加していってください。このとき、必要に応じてルールを変更してもかまいません。



注記: 監査ログレベルの調整について

お使いの環境の要件にあわせることなく、設定例をそのまま使用して監査を行うことは避けてください。まずは何を監査するのか、どの範囲まで監査するのかを決定してから実施してください。

`audit.rules` ファイルは `auditctl` コマンドの集合体であると考えられます。このファイルの各行は、それぞれ `auditctl` のコマンドラインとして解釈されます。そのため、このルールファイル内で使用する文法は、`auditctl` のコマンドラインと同じです。

43.1 基本的な監査設定パラメータの追加

```
-D❶
-b 8192❷
-f 2❸
```

❶ 新しくルールセットを定義するため、既存のルールを全て削除しています。

❷ 監査メッセージを受け付けるためのバッファ数を指定しています。お使いのシステムにおける監査ログの量に応じて、増やしたり減らしたりしてください。

❸ カーネルが致命的なエラーを処理する際に使用する、失敗フラグを指定しています。指定可能な値は `0` (何もしない), `1` (printk (失敗メッセージを表示する)), `2` (パニックモード、システムを停止する) のいずれかです。

`-D` オプションを指定してルールキューをいったん空にすることで、それまでに設定されていたルールによる影響を受けることなく、このファイル内に書かれているルールを正しく適用することができるようになります。また、バッファ数の指定 (`-b`) は、監査システムの負荷を上昇させすぎてシステムに障害が発生したりしないようにするために重要な設定です。このほか、失敗フラグ `-f 2` の選択は、お使いのシステムに対して致命的なエラーが発生した際、監査レコードを保全するために必要な設定です。致命的なエラーの際にシステムをシャットダウンすることで、監査から外れたプロセスが現れないようにすることができるためです。このような要件が無い場合は、1 (`printk`) を選択してください。

重要: 失敗フラグの選択について

本番環境にお使いの監査ルールを適用するにあたっては、あらかじめテスト環境で 想定される最大の負荷 をかけて、監査が正しく動作することをご確認ください。また、 `-f 2` フラグを指定しておくことも重要です。この設定は、何らかの閾値を超過した場合に、カーネルをパニック状態 (書き込み待ちのデータを書き込むことなくシステムを即時に停止させる) にすることができる設定です。ただし、 `-f 2` の設定は、セキュリティを最重要視する環境でのみお使いください。

43.2 監査ログファイルと設定ファイルに対する監視の追加

ご利用の監査設定ファイルとログファイルに対して監視を設定して、設定ファイルやログファイルに対して改ざんを試みた形跡や、不正に読み取ろうとした形跡を記録するようにしておいてください。

注記: ディレクトリとファイルの作成に対する監視について

ディレクトリに対する監視は、ファイルアクセスに対するイベントだけが必要な環境では、特に設定を行う必要はありません。ディレクトリアクセスのイベントは、メタデータの変更を伴うディレクトリの inode 変更の際にのみ発行されるためです。ファイルアクセスに対してイベントを発生させたい場合は、各ファイルを監視対象にしてください。

```
-w /var/log/audit/ ❶
-w /var/log/audit/audit.log

-w /var/log/audit/audit_log.1
-w /var/log/audit/audit_log.2
-w /var/log/audit/audit_log.3
-w /var/log/audit/audit_log.4

-w /etc/audit/auditd.conf -p wa❷
-w /etc/audit/audit.rules -p wa
-w /etc/libaudit.conf -p wa
```

❶ 監査ログの存在するディレクトリに対して監視を設定しています。このディレクトリに対してアクセスが試みられると、どのような種類のアクセスであってもイベントが発生します。ログのローテーションを使用している場合は、その際にもイベントが発生します。

❷ 監査の設定ファイルに対して監視を設定しています。このファイルに対する書き込みと、属性変更が記録されるようになります。

43.3 ファイルシステムオブジェクトの監視

システムコールに対する監査を行うことで、アプリケーションレベルよりも細かく、お使いのシステムにおける動作を追跡することができるようになります。ファイルシステム関連のシステムコールを追跡することで、お使いのアプリケーションがどのようなシステムコールを使用しているのかを判断することができるほか、その使用が適切であるかどうかを確認することもできます。また、マウントやマウント解除の操作を追跡することで、外部リソース (リムーバブルメディアやリモートのファイルシステムなど) の使用についても追跡を行うことができます。

重要: システムコールの監査について

システムコールの監査を行うと、ログを頻繁に書き込むことになります。そのため、カーネルに対しても重い負荷を与えることになります。監査を行った結果、通常よりも応答が遅くなったと感じた場合は、おそらくシステムのバックログ設定と流量制限の設定を超過したものと思われます。

この場合は、監査ルール内にどのシステムコールを含めるのかを慎重に確認し、ログ設定についても調整を行ってください。41.2項「監査デーモンの設定」では、関連する設定を調整する方法について説明しています。

```
-a entry,always -S chmod -S fchmod -S chown -S chown32 -S fchown -S fchown32 -S lchown -S
 lchown32 ❶

-a entry,always -S creat -S open -S truncate -S truncate64 -S ftruncate -S ftruncate64 ❷

-a entry,always -S mkdir -S rmdir ❸

-a entry,always -S unlink -S rename -S link -S symlink ❹

-a entry,always -S setxattr ❺
-a entry,always -S lsetxattr
-a entry,always -S fsetxattr
-a entry,always -S removexattr
-a entry,always -S lremovexattr
-a entry,always -S fremovexattr

-a entry,always -S mknod ❻

-a entry,always -S mount -S umount -S umount2 ❼
```

❶ ファイルの所有権とパーミッションを変更する操作に関連するシステムコールに対して、監査コンテキストを有効化しています。お使いのハードウエアアーキテクチャによっても異なりますが、`*32` のルールが必要となる環境と不要な環境があります。AMD64/Intel 64 のような 64 ビットシステムでは、`*32` のルールは削除する必要があります。

❷ ファイルの内容変更に関連するシステムコールに対して、監査コンテキストを有効化しています。お使いのハードウエアアーキテクチャによっても異なりますが、`*64` のルールが必要となる環境と不要な環境があります。AMD64/Intel 64 のような 64 ビットシステムでは、`*64` のルールは削除する必要があります。

❸ ディレクトリの作成や削除など、ディレクトリ操作に対する監査コンテキストを有効化しています。

❹ シンボリックリンクの作成やハードリンクの作成、削除や名前変更などのリンク操作に対して、監査コンテキストを有効化しています。

❺ ファイルシステムの属性操作に関する監査コンテキストを有効化しています。

❻ デバイスファイルを作成するための `mknod` システムコールに対して、監査コンテキストを有効化しています。

❼ マウントやマウント解除の操作に対して、監査コンテキストを有効化しています。x86 アーキテクチャの場合は `umount` のルールを、Intel 64 アーキテクチャの場合は `umount2` のルールをそれぞれ無効化してください。

43.4 セキュリティ設定ファイルとデータベースの監視

お使いのシステムに対して望ましくない行為をさせないようにするため、`cron` や `at` の設定ファイルのほか、ジョブスケジュールの変更を行おうとする行為を追跡しておくことをお勧めします。このほか、ユーザやグループ、パスワードやログインデータベースやアクセスログなどへの書き込みアクセスについても追跡することで、お使いのシステムのユーザデータベースを改ざんしようとする行為を検出できるようになります。

また、お使いのシステムにおける設定 (カーネル, サービス, 時刻など) に対する変更を追跡しておくことで、お使いのシステムにおける重要な機能の改ざんを防ぐことができます。特に PAM の設定ファイルの変更は、認証スタックの設定変更は管理者以外から行われるべきではないことから、監視対象としておくべきです。また、アプリケーション側からの PAM の使用状況や用途についても、ログに記録を行っておくべきです。このほか、機密を確保しておくべき認証や通信にかかわる設定ファイルについても、同様に監視対象としておくのがよいでしょう。

```
❶
-w /var/spool/atspool
-w /etc/at.allow
-w /etc/at.deny

-w /etc/cron.allow -p wa
-w /etc/cron.deny -p wa
-w /etc/cron.d/ -p wa
-w /etc/cron.daily/ -p wa
-w /etc/cron.hourly/ -p wa
-w /etc/cron.monthly/ -p wa
-w /etc/cron.weekly/ -p wa
-w /etc/crontab -p wa
-w /var/spool/cron/root

❷
-w /etc/group -p wa
-w /etc/passwd -p wa
-w /etc/shadow

-w /etc/login.defs -p wa
-w /etc/securetty
-w /var/log/lastlog

❸
-w /etc/hosts -p wa
-w /etc/sysconfig/
w /etc/init.d/
w /etc/ld.so.conf -p wa
w /etc/localtime -p wa
w /etc/sysctl.conf -p wa
w /etc/modprobe.d/
```

```
w /etc/modprobe.conf.local -p wa
w /etc/modprobe.conf -p wa
❹
w /etc/pam.d/
❺
-w /etc/aliases -p wa
-w /etc/postfix/ -p wa

❻
-w /etc/ssh/sshd_config

-w /etc/stunnel/stunnel.conf
-w /etc/stunnel/stunnel.pem

-w /etc/vsftpd.ftpusers
-w /etc/vsftpd.conf

❼
-a exit,always -S sethostname
-w /etc/issue -p wa
-w /etc/issue.net -p wa
```

❶ `at` や `cron` の設定ファイルと、ジョブのスケジュール設定、そしてそれらのイベントに対するラベルの割り当てに対して、それぞれ監視を設定しています。

❷ ユーザ, グループ, パスワード, ログインデータベースに対して監視を設定しているほか、たとえばログインの失敗など、ログイン関連のイベントを記録し、ラベルでわかりやすく識別するように設定しています。

❸ まずはホスト名の設定を行っている `/etc/hosts` に対して、監視を設定しラベルを指定しています。また、`/etc/sysconfig` というシステム設定ディレクトリに対しても、変更を追跡しています。それぞれのファイルイベントを追跡したい場合は、ファイルごとの監視を設定してください。さらに、`/etc/init.d` ディレクトリ内にある起動設定の変更に対しても、監視を設定しラベルを指定しています。こちらについても同様に、それぞれのファイルイベントを追跡したい場合は、ファイルごとの監視を設定してください。また、`/etc/ld.so.conf` というリンカの設定ファイル、`/etc/localtime` というローカル時刻の設定ファイルについても、監視とラベルを設定しています。このほか、`/etc/sysctl.conf` , `/etc/modprobe.d/` , `/etc/modprobe.conf.local` , `/etc/modprobe.conf` にあるカーネルの設定ファイルに対しても、監視とラベルを設定しています。

❹ PAM の設定ファイルのディレクトリに対して監視を設定しています。このディレクトリ内にあるファイルを監視したい場合は、それぞれのファイルに対して監視を設定してください。

❺ postfix の設定ファイルに対して監視を設定し、書き込みや属性変更を行おうとした場合に、後から検索できるようラベルを指定して記録を行っています。

❻ SSH, `stunnel` , `vsftpd` の各設定ファイルに対して、後から検索できるようラベルを指定して監視を設定しています。

❼ `sethostname` システムコールに対して監査を設定するほか、`/etc/issue` と `/etc/issue.net` にあるシステムの識別情報の設定ファイルに対して、後から検索できるようラベルを指定して監視を設定しています。

43.5 その他のシステムコールに対する監視

43.3項「ファイルシステムオブジェクトの監視」で説明しているファイルシステム関連のシステムコールの監査だけでなく、その他のシステムコールに対しても追跡を行うことができます。たとえばタスク (プロセス) の作成を追跡することで、アプリケーションの振る舞いを理解する手助けになったりすることがあります。また、`umask` システムコールを監査することで、プロセスがどのような作成マスクを使用しているのかを調べたりすることもできます。また、システム時刻の変更に関わるシステムコールを追跡することで、誰かがシステムの時刻を改ざんしようとしていないかどうかを確認することもできます。

```
❶
-a entry,always -S clone -S fork -S vfork

❷
-a entry,always -S umask

❸
-a entry,always -S adjtimex -S settimeofday
```

❶ タスク (プロセス) の作成を追跡する設定です。

❷ umask システムコールに対して監査コンテキストを追加しています。

❸ システム時刻の変更に関わるシステムコールの追跡です。`adjtimex` はシステムの時刻調節の設定を行うためのシステムコールで、`settimeofday` は絶対時刻を設定するためのシステムコールです。

43.6 システムコールのパラメータでのフィルタリング

43.3項「ファイルシステムオブジェクトの監視」や 43.5項「その他のシステムコールに対する監視」で説明しているシステムコールの監査に加えて、さらに高度にアプリケーションの動作を追跡することができます。具体的にはフィルタを設定することで、監査の範囲を絞り込むことができるようになります。本章では、多重化を伴わないシステムコールのほか、socketcall や ipc などのように、多重化を伴うシステムコールのパラメータに対して、フィルタリングを設定する方法を紹介しています。なお、AMD64/Intel 64 など、64 ビット環境では、socketcall も ipc も多重化されません。

重要: システムコールの監査について

システムコールの監査を行うと、ログを頻繁に書き込むことになります。そのため、カーネルに対しても重い負荷を与えることになります。監査を行った結果、通常よりも応答が遅くなったと感じた場合は、おそらくシステムのバックログ設定と流量制限の設定を超過したものと思われます。この場合は、監査ルール内にどのシステムコールを含めるのかを慎重に確認し、ログ設定についても調整を行ってください。41.2項「監査デーモンの設定」では、関連する設定を調整する方法について説明しています。

access システムコールは、ファイルやファイルシステムのオブジェクトに対して、読み込みや書き込み、存在のチェックの機能を提供するシステムコールです。`-F` フィルタフラグを設定することで、特定の条件に該当するシステムコールのみを記録することができるようになります。具体的な書式は `-F a1=アクセスモード` です。access システムコールにおけるパラメータについて、詳しくは `/usr/include/fcntl.h` ファイルをお読みください。

```
-a entry,always -S access -F a1=4❶
-a entry,always -S access -F a1=6❷
-a entry,always -S access -F a1=7❸
```

❶ access システムコールに対して監査を設定していますが、システムコールの 2 番目のパラメータ (`モード`) が `4` (`R_OK`)の場合にのみ、監査を行う指定です。このルールフィルタでは、ユーザやプロセスがアクセスしようとしているファイルやファイルシステムのオブジェクトが、読み込みのための権限を持っているかどうかをテストする際に、監査を行う指定です。

❷ access システムコールに対して監査を設定していますが、システムコールの 2 番目のパラメータ (`モード`) が `6` 、つまり `4` (`R_OK`) および `2` (`W_OK`) の場合にのみ、監査を行う指定です。このルールフィルタでは、読み込みおよび書き込みのための権限を持っているかどうかをテストする際に、監査を行う指定です。

❸ access システムコールに対して監査を設定していますが、システムコールの 2 番目のパラメータ (`モード`) が `7` 、つまり `4` (`R_OK`) および `2` (`W_OK`) および `1` (`X_OK`) の場合にのみ、監査を行う指定です。このルールフィルタでは、読み込みおよび書き込み、そして実行のための権限を持っているかどうかをテストする際に、監査を行う指定です。

socketcall システムコールは多重化型のシステムコールです。多重化とは複数のシステムコールを 1 つのシステムコールで賄う仕組みで、libc 側で実際のシステムコールを最初のパラメータ (`a0`) として設定する仕組みです。socketcall で利用できるシステムコールについては、マニュアルページをお読みください。また、パラメータの値とシステムコールの名前の一覧については、`/usr/src/linux/include/linux/net.h` ファイルをお読みください。監査システムでは、`-F a0=システムコール番号` の形式で、システムコールを指定することができます。

```
-a entry,always -S socketcall -F a0=1 -F a1=10❶
## x86_64, ia64 アーキテクチャの場合は下記をお使いください
#-a entry,always -S socket -F a0=10

-a entry,always -S socketcall -F a0=5❷
## x86_64, ia64 アーキテクチャの場合は下記をお使いください
#-a entry, always -S accept
```

❶ socket(PF_INET6) システムコールを監査しています。`-F a0=1` は socket システムコールを指し、`-F a1=10` は IPv6 プロトコルファミリ (PF_INET6) を指定した場合を意味するパラメータになります。最初のパラメータ (`a0`) に対応する値は `/usr/include/linux/net.h` ファイルを、2 つ目のパラメータ (`a1`) に対応する値は `/usr/src/linux/include/linux/socket.h` ファイルをそれぞれお読みください。AMD64/Intel 64 などの 64 ビットプラットフォームでは、socketcall システムコールによる多重化は行っていませんので、これらの環境ではシステムコール名を直接指定してください。

❷ 上記と同様に socketcall システムコールを監査していますが、`a0=5` は accept システムコールを意味しています (詳しくは `/usr/include/linux/net.h` をお読みください) 。AMD64/Intel 64 などの 64 ビットプラットフォームでは、socketcall システムコールによる多重化は行っていませんので、これらの環境ではシステムコール名を直接指定してください。

ipc システムコールは多重化されるシステムコールのもう 1 つの例となります。実際のシステムコールは ipc システムコールの 1 つ目のパラメータとして指定されます。このようにしてフィルタを設定することで、特定の ipc システムコールのみを監査することができます。値の意味について、詳しくは `/usr/include/linux/ipc.h` をお読みください。

```
❶
## msgctl
-a entry,always -S ipc -F a0=14
## msgget
-a entry,always -S ipc -F a0=13
## x86_64, ia64 アーキテクチャの場合は下記をお使いください
#-a entry,always -S msgctl
#-a entry,always -S msgget

❷
## semctl
-a entry,always -S ipc -F a0=3
## semget
-a entry,always -S ipc -F a0=2
## semop
-a entry,always -S ipc -F a0=1
## semtimedop
-a entry,always -S ipc -F a0=4
## x86_64, ia64 アーキテクチャの場合は下記をお使いください
#-a entry,always -S semctl
```

```
#-a entry,always -S semget
#-a entry,always -S semop
#-a entry,always -S semtimedop

❸
## shmctl
-a entry,always -S ipc -F a0=24
## shmget
-a entry,always -S ipc -F a0=23
## x86_64, ia64 アーキテクチャの場合は下記をお使いください
#-a entry,always -S shmctl
#-a entry,always -S shmget
```

❶ IPC SYSV メッセージキューに関連するシステムコールを監査しています。この場合、`a0` の値は msgctl (`14`) と msgget (`13`) のシステムコールを指しています。AMD64/Intel 64 などの 64 ビットプラットフォームでは、ipc システムコールによる多重化は行っていませんので、これらの環境ではシステムコール名を直接指定してください。

❷ IPC SYSV メッセージセマフォに関連するシステムコールを監査しています。この場合、`a0` の値は semctl (`3`), semget (`2`), semop (`1`), semtimedop (`4`) の各システムコールを指しています。上記と同様に、AMD64/Intel 64 などの 64 ビットプラットフォームでは、ipc システムコールによる多重化は行っていませんので、これらの環境ではシステムコール名を直接指定してください。

❸ IPC SYSV 共有メモリに関連するシステムコールを監査しています。この場合、`a0` の値は shmctl (`24`), shmget (`23`) の各システムコールを指しています。これまでと同様に、AMD64/Intel 64 などの 64 ビットプラットフォームでは、ipc システムコールによる多重化は行っていませんので、これらの環境ではシステムコール名を直接指定してください。

43.7 キーを利用した監査イベントレコードの管理

イベントを生成するためのルールを複数設定し、ログに出力するようにした場合、ログには一般に多くの項目が出力されることになり、必要なものを識別するだけでも手間がかかってしまいます。`ausearch` コマンドを使用することで、様々な条件を指定して検索を行うことができます。たとえば `ausearch -m メッセージの種類` のように指定すると、メッセージの種類を指定して検索を行うことができます。しかしながら、同じようなメッセージを出力するようなルールが複数あって、それらを区別して扱いたい場合、`/etc/audit/audit.rules` ファイルのルール内に、あらかじめ何らかの検索用のキーを指定しておくことができます。このキーはその条件に合致してログを出力する際、必ず書き込まれる仕組みになっています。そのため、ログを記録したあとから `ausearch -k キー名` のようにして検索することで、特定のキーが設定されたイベントのみを抽出することができるようになります。

例として、下記のようなルールをルールセットに追加したものします:

```
-w /etc/audit/audit.rules -p wa
```

キーを指定しない場合、まずは `SYSCALL` や `PATH` のイベントのみを抽出してから、さらに grep などのツールを利用して必要な項目を抜き出すような処理が必要になります。そのため、下記のように `-k` オプションを指定して、キーを書き込むように設定します:

```
-w /etc/audit/audit.rules -p wa -k CFG_audit.rules
```

キーには任意の文字列を指定することができます。また、上記の例のように、`CFG` は設定ファイル用、`LOG` はログファイル用等のように設定し、その後ろに実際のファイル名を続けるようにして、わかりやすくするとよいでしょう。上記のようなキーを設定した場合、検索は下記のようになります:

```
ausearch -k CFG_audit.rules
----
time->Thu Feb 19 09:09:54 2009
type=PATH msg=audit(1235030994.032:8649): item=3 name="audit.rules~" inode=370603
 dev=08:06 mode=0100640 ouid=0 ogid=0 rdev=00:00
type=PATH msg=audit(1235030994.032:8649): item=2 name="audit.rules" inode=370603 dev=08:06
 mode=0100640 ouid=0 ogid=0 rdev=00:00
type=PATH msg=audit(1235030994.032:8649): item=1  name="/etc/audit" inode=368599 dev=08:06
 mode=040750 ouid=0 ogid=0 rdev=00:00
type=PATH msg=audit(1235030994.032:8649): item=0  name="/etc/audit" inode=368599 dev=08:06
 mode=040750 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1235030994.032:8649):  cwd="/etc/audit"
type=SYSCALL msg=audit(1235030994.032:8649): arch=c000003e syscall=82 success=yes exit=0
 a0=7deeb0 a1=883b30 a2=2 a3=ffffffffffffffff items=4 ppid=25400 pid=32619 auid=0 uid=0
 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1 ses=1164 comm="vim" exe="/bin/
vim-normal" key="CFG_audit.rules"
```

44 その他の情報源

改訂履歴

2023-12-22

Linux 監査フレームワークに関しては、ここまでに説明したもの以外にも様々な情報源が用意されています:

audit のマニュアルページ

監査ツールをインストールすることで、様々な情報を含むいくつかのマニュアルページを利用することができます:

`auditd(8)`

Linux 監査デーモン

`auditd.conf(5)`

Linux 監査デーモンの設定ファイル

`auditctl(8)`

カーネルの監査システムを制御するためのユーティリティ

`autrace(8)`

`strace` に似たプログラム

`ausearch(8)`

監査デーモンのログを素早く検索するためのツール

`aureport(8)`

監査デーモンのログから概要レポートを生成するためのツール

https://people.redhat.com/sgrubb/audit/index.html

Linux 監査プロジェクトの Web ページ (英語のみ) です。このサイトには、Linux 監査システムに関連する各種の仕様のほか、短い FAQ も用意されています。

`/usr/share/doc/packages/audit`

audit パッケージ自身にも README ファイルが含まれていて、ここには基本的な設計情報 (英語) が書かれているほか、様々な用途に対応した `.rules` ファイルの例も添付されています:

`capp.rules` : Controlled Access Protection Profile (CAPP) 対応の設定
`lspp.rules` : Labeled Security Protection Profile (LSPP) 対応の設定
`nispom.rules` : National Industrial Security Program Operating Manual Chapter 8(NISPOM) 対応の設定

`stig.rules` : Secure Technical Implementation Guide (STIG) 対応の設定

https://www.commoncriteriaportal.org/

コモンクライテリアプロジェクトの公式 Web サイトです。コモンクライテリアによるセキュリティ認証への取り組みに関する情報のほか、このフレームワーク内で監査が果たす役割についても説明が書かれています (英語のみ) 。

A GNU ライセンス

本付録には、GNU Free Documentation License バージョン 1.2 とその日本語訳 (八田真行氏 (mhatta@gnu.org) による翻訳) を収録しています。

GNU Free Documentation License

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

GNU フリー文書利用許諾契約書

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. この利用許諾契約書を、一字一句そのままに複製し頒布することは許可する。しかし変更は認めない。

0. はじめに

この利用許諾契約書の目的は、この契約書が適用されるマニュアルや教科書、その他機能本位で実用的な文書を(無料ではなく)自由という意味で「フリー」とすること、すなわち、改変の有無あるいは目的の営利非営利を問わず、文書を複製し再頒布する自由をすべての人々に効果的に保証することです。加えてこの契約書により、著者や出版者が自分たちの著作物に対して相応の敬意と賞賛を得る手段も保護されます。また、他人が行った改変に対して責任を負わずに済むようになります。

この利用許諾契約書は「コピーレフト」的なライセンスの一つであり、この契約書が適用された文書から派生した著作物は、それ自身もまた原本と同じ意味でフリーでなければなりません。この契約書は、フリーソフトウェアのために設計されたコピーレフトなライセンスであるGNU一般公衆使用許諾契約書を補足するものです。

この利用許諾契約書は、フリーソフトウェア用のマニュアルに適用することを目的として書かれました。フリーソフトウェアはフリーな文書を必要としており、フリーなプログラムはそのソフトウェアが保証するのと同じ自由を提供するマニュアルと共に頒布されるべきだからです。しかし、この契約書の適用範囲はソフトウェアのマニュアルに留まりません。対象となる著作物において扱われる主題が何であれ、あるいはそれが印刷された書籍として出版されるか否かに関わらず、この契約書は文字で書かれたいかなる著作物にも適用することが可能です。私たちとしては、主にこの契約書を解説や参照を目的とする著作物に適用することをお勧めします。

1. この利用許諾契約書の適用範囲と用語の定義

著作物がこの利用許諾契約書の定める条件の下で頒布される旨の告知を、著作権者がその中に書いたすべてのマニュアルあるいはその他の著作物は、いかなる媒体上にあってもこの契約書の適用対象となる。そのような告知を置くことで、全世界において、著作権使用料を必要とせず、許可の存続期間を限定されること無く、この契約書の中で述べられている条件の下で当該著作物を利用できるという許可を与えることとする。以下において、「『文書』(Document)」とはそのような告知が記載されたマニュアルないし著作物すべてを指す。公衆の一員ならば誰でも契約の当事者となることができ、この契約書中では「あなた」と表現される。あなたは、著作権法の下で許可を必要とするような方法で著作物を複製や改変、あるいは頒布することにより、この契約書を受諾することになる。

『文書』の「改変版 (Modified Version)」とは、一字一句忠実に複製したか、あるいは改変や他言語への翻訳を行ったかどうかに関わらず、その『文書』の全体あるいは一部分を含む著作物すべてを意味する。

「補遺部分 (Secondary Section)」とは、『文書』中でその旨指定された補遺ないし本文に先だって前付けとして置かれる一部分であり、『文書』の出版者あるいは著者と、『文書』全体の主題(あるいはそれに関連する事柄)との関係のみを論じ、全体としての主題の範疇に直接属する内容を全く含まないものである(たとえば、『文書』の一部が数学の教科書だった場合、補遺部分では数学について何も解説してはならない)。補遺部分で扱われる関係は、その主題あるいは関連する事柄との歴史的なつながりのことかも知れないし、それらに関する法的、商業的、哲学的、倫理的、あるいは政治的立場についてかも知れない。

「変更不可部分 (Invariant Sections)」とは補遺部分の一種で、それらが変更不可部分であることが、『文書』をこの利用許諾契約書の下で発表する旨述べた告知中においてその部分の題名と共に明示されているものである。ある部分が上記のような「補遺」性の定義にそぐわない場合は、その部分を「変更不可」として指定することは認められない。『文書』は、変更不可部分を全く含まなくても良い。『文書』において変更不可部分が全く指定されていなければ、その『文書』に変更不可部分は存在しないということである。

「カバーテキスト(Cover Texts)」とは、『文書』がこの利用許諾契約書の指定する条件の下で発表される旨述べた告知において、「表カバーテキスト」あるいは「裏カバーテキスト」として列挙された短い文章のことを指す。表カバーテキストは最大で5語、裏カバーテキストは最大で25語までとする。

『文書』の「透過的」複製物とは、機械による読み取りが可能な『文書』の複製物のことを指す。透過的な複製物の文書形式は、その仕様が一般の人々に入手可能で、『文書』の内容を一般的なテキストエディタ、または(画素で構成される画像ならば)一般的なペイントプログラム、あるいは(図面ならば)いくつかの広く入手可能な製図エディタで簡単に改訂するのに適しており、なおかつテキストフォーマッタへの入力に適する(あるいはテキストフォーマッタへの入力に適する諸形式への自動的な変換に適する)ものでなければならない。透過的なファイル形式への複製であっても、マークアップ、あるいはマークアップの不在が読者によるそれ以降の改変をわざと邪魔し阻害するように仕組まれたものは透過的であるとは見做されない。ある画像形式が、相当量のテキスト文章を表現するために使われた場合、それは透過的ではない。透過的ではない複製は「非透過的」複製と呼ばれる。

透過的複製に適した形式の例としては、マークアップを含まないプレーンなASCII形式、Texinfo入力形式、LaTeX入力形式、一般に入手可能なDTDを用いたSGMLあるいはXML、または人間による改変を想定して設計された、標準に準拠したシンプルなHTMLやPostScript、PDFなどが挙げられる。透過的な画像形式の例には、PNGやXCF、JPGが含まれる。非透過な形式としては、独占的なワードプロセッサでのみ閲覧編集できる独占的なファイル形式、普通には入手できないDTDまたは処理系を使ったSGMLやXML、ある種のワードプロセッサが生成する、出力のみを目的とした機械生成のHTMLやPostScript、PDFなどが含まれる。

「題扉 (Title Page)」とは、印刷された書籍に於いては、実際の表紙自身のみならず、この利用許諾契約書が表紙に掲載することを義務づける文章や図などを、読みやすい形で載せるのに必要なだけの、表紙に引き続く数ページをも意味する。表紙に類するものが無い形式で発表される著作物においては、「題扉」とは本文の始まりに先だって、その著作物の題名が最も目立つ形で現れる場所の近くに置かれる文章のことを指す。

「XYZと題された (Entitled XYZ)」部分とは、『文書』において「XYZ」と名付けられた一部分であり、その題名は正確に「XYZ」であるか、「XYZ」を他の言語に翻訳した上でその後ろに「XYZ」をそのまま括弧で括ったものを含む記述のどちらかである(ここでの「XYZ」とは、この利用許諾契約書において以下で言 及される特定の部分名を意味している。例えば「謝辞 (Acknowledgements)」、「献辞 (Dedications)」、「推薦の辞 (Endorsements)」、「履歴 (History)」)。あなたが『文書』を改変する場合、そのような部分の「題名を保存する (Preserve the Title)」とは、「XYZと題された」部分として、ここでの定義に従い題名を残すということである。

『文書』は、「保証否認警告 (Warranty Disclaimers)」を、この利用許諾契約書が『文書』に適用されると述べた告知の次に含んでも良い。この種の保証否認警告は、この契約書からの言及という形で利用条件に含まれるものと解されるが、保証の否認に関することについてのみ有効とする。こういった保証否認警告で示しうるその他のいかなる含意も無効であり、この契約書の効能には何ら影響を持たない。

2. 逐語的に忠実な複製

この利用許諾契約書、著作権表示、この契約書が『文書』に適用される旨述べた告知の三つがすべての複製物に複製され、かつあなたがこの契約書で指定されている以外のいかなる条件も追加しない限り、あなたはこの『文書』を、商用であるか否かを問わずいかなる形でも複製頒布することができる。あなたは、あなたが作成あるいは頒布する複製物に対して、閲覧や再複製を技術的な手法によって妨害、規制してはならない。しかしながら、複製と引き換えに代価を得てもかまわない。あなたが相当量の複製物を頒布する際には、本契約書第3項で指定される条件にも従わなければならない。

またあなたは、上記と同じ条件の下で、複製物を貸与したり複製物を公に開示することができる。

3. 大量の複製

もしあなたが、『文書』の印刷された(あるいは通常は印刷された表紙を持つ媒体における)複製物を100部を超えて出版し、また『文書』の利用許諾告知がカバーテキストの掲載を要求している場合には、指定されたすべてのカバーテキストを、表カバーテキストは表表紙に、裏カバーテキストは裏表紙に、はっきりと読みやすい形で載せた表紙の中に複製物本体を綴じ込まなければならない。また、両方の表紙において、それらの複製物の出版者としてのあなたをはっきりとかつ読みやすい形で確認できなければならない。表表紙では『文書』の完全な題名を、題名を構成するすべての語が等しく目立つようにして、視認可能な形で示さなければならない。それらの情報に加えて、表紙に他の文章や図などを加えることは許可される。表紙のみを変更した複製物は、それが『文書』の題名を保存し上記の条件を満たす限り、ほかの点では逐語的に忠実な複製物 として扱われる。

もしどちらかの表紙に要求されるカバーテキストの量が多すぎて読みやすく収めることが不可能ならば、あなたはテキスト先頭の一文(あるいは適切に収まるだけ)を実際の表紙に載せ、続きは隣接したページに載せるべきである。

あなたが『文書』の「非透過的」複製物を100部を超えて出版あるいは頒布する場合、それぞれの非透過な複製物と一緒に機械で読み取り可能な透過的複製物を添付するか、それぞれの非透過な複製物(あるいはそれに付属する文書)中で、公にアクセス可能なコンピュータネットワーク上の所在地を記述しなければならない。その場所には、非透過な複製物と内容的に寸分違わず、余計なものが追加されていない完全な『文書』の透過的複製物が置かれ、またそこから、ネットワークを利用する一般公衆が、一般に標準的と考えられるネットワークプロトコルを使ってダウンロードすることができなければならない。もしあなたが後者の選択肢を選ぶならば、その版の非透過な複製物を公衆に(直接、あるいはあなたの代理人ないし小売業者が)最後に頒布してから最低1年間は、その透過的複製物が指定の場所でアクセス可能であり続けることを保証するよう、非透過な複製物の大量頒布を始める際に十分に慎重な手順を踏まなければならない。

これは要望であり必要条件ではないが、『文書』の著者に、『文書』の更新された版をあなたに提供する機会を与えるため、透過非透過を問わず大量の複製物を再頒布し始める前には彼らにきちんと連絡しておいてほしい。

4. 改変

『文書』の改変版を、この利用許諾契約書と細部まで同一の契約の下で発表する限り、すなわち原本の役割を改変版で置き換えた形での頒布と改変を、その複製物を所有するすべての人々に許可する限り、あなたは改変版を上記第2項および第3項が指定する条件の下で複製および頒布することができる。さらに、あなたは改変版において以下のことを行わなければならない。

A. 題扉に(もしあればその他の表紙にも)、『文書』お よび『文書』のそれ以前の版と見分けがつく題名を載せること(もし以前の 版があれば、『文書』の「履歴 (History)」の部分に列記されているはずである)。もし元の版の出版者から許可を得たならば、以前の版と同じ題名を 使っても良い。

B. 題扉に、改変版における改変を行った1人以上の人物か団体名を列記すること。あわせて元の『文書』の著者として、最低5人(もし5人以下ならばすべて)の主要著者を列記すること。ただし元の著者たちがこの条件を免除した場合は除く。

C. 題扉に、改変版の出版者名を出版者として記載すること。

D. 『文書』にあるすべての著作権表示を残すこと。

E. 他の著作権表示の近くに、あなたの改変に対する適当な著作権表示を追加すること。

F. 著作権表示のすぐ後に、改変版をこの契約書の条件の下で利用することを公衆に対して許可する告知を含めること。その形式はこの契約書の末尾にある付記で示されている。

G. 元の『文書』の利用許諾告知に書かれた、変更不可部分の完全な一覧と、要求されるカバーテキストとを、改変版の利用許諾告 知でもそのまま残すこと。

H. この契約書の、変更されていない複製物を含めること。

I. 「履歴 (History)」と題された部分とその題名を保存し、そこに改変版の、少なくとも題名、出版年、新しく変更した部分の著 者名、出版者名を、題扉に掲載するのと同じように記載した一項を加えること。もし『文書』中に「履歴」と題された部分が存在しない場合には、『文書』の題名、出版年、著者、出版者を題扉に掲載するのと同じように記載した部分を用意し、上記で述べたような、改変版を説明する一項を加えること。

J. 『文書』中に、『文書』の透過的複製物への公共的アクセスのために指定されたネットワーク的所在地が記載されていたならば、それを保存すること。同様に、その『文書』の元になった以前の版で指定されていたネットワーク的所在地も載っていたならば、それも保存すること。これらの情報は「履歴(History)」の部分に置いても良い。ただし、それが『文書』自身より少なくとも4年前に出版された著作物の情報であったり、あるいは改変版が参考にしている版の元々の出版者から許可を得たならば、その情報を削除してもかまわない。

K. 「謝辞 (Acknowledgement)」あるいは「献辞 (Dedication)」等と題されたいかなる部分も、その部分の題名を保存し、その部分の内容(各貢献者への謝意あるいは献呈の意)と語調を保存すること。

L. 『文書』の変更不可部分を、その本文および題名を変更せずに保存すること。章番号やそれに相当するものは部分の題名の一部とは見做さない。

M. 「推薦の辞 (Endorsement)」というような章名が題された部分はすべて削除すること。そのような部分を改変版に含めてはならない。

N. すでに存在する部分を「推薦の辞 (Endorsement)」と題されるように改名したり、題名の点で変更不可部分のどれかと衝突するように改名してはならない。

O. 保証否認警告を保存すること。

もし改変版に、補遺部分としての条件を満たし、かつ『文書』から複製物された文章や図などをいっさい含んでいない、前書き的な章あるいは付録が新しく含まれるならば、あなたは希望によりそれらの部分の一部あるいはすべてを変更不可と宣言することができる。変更不可を宣言するためには、それらの部分の題名を改変版の利用許諾告知中の変更不可部分一覧に追加すれば良い。これらの題名は他の章名とは全く別のものでなければならない。

含まれる内容が、さまざまな集団によるあなたの改変版に対する推薦の辞のみである限り、あなたは、「推薦の辞 (Endorsement)」と題された章を追加することができる。推薦の辞の例としては、ピアレビューの陳述、あるいは文書がある標準の権威ある定義としてその団体に承認されたという声明などがある。

あなたは、5語までの一文を表カバーテキストとして、25語までの文を裏表紙テキストとして、改変版のカバーテキスト一覧の末尾に加えることができる。一個人ないし一団体が直接(あるいは団体内で結ばれた協定によって)加えることができるのは、表カバーテキストおよび裏カバーテキストとしてそれぞれ一文ずつのみである。もし以前すでにその文書において、表裏いずれかの表紙にあなたの(またはあなたが代表する同じ団体内で為された協定に基づく)カバーテキストが含まれていたならば、あなたが新たに追加することはできない。し かしあなたは、その古い文を加えた以前の出版者から明示的な許可を得たならば、古い文を置き換えることができる。

『文書』の著者あるいは出版者は、この利用許諾契約書によって、彼らの名前を利用することを許可しているわけではない。彼らの名前を改変版の宣伝に使ったり、改変版への明示的あるいは黙示的な保証のために使うことを許可するものではない。

5. 文書の結合

あなたは、上記第4項において改変版に関して定義された条件の下で、この利用許諾契約書の下で発表された複数の文書を一つにまとめることができる。その際、原本となる文書にある変更不可部分を全て、改変せずに結合後の著作物中に含め、それらをあなたが統合した著作物の変更不可部分としてその利用許諾告知において列記し、かつ原本にある全ての保証否認警告を保存しなければならない。

結合後の著作物についてはこの契約書の複製物を一つ含んでいればよく、同一内容の変更不可部分が複数ある場合には一つで代用してよい。もし同じ題名だが内容の異なる変更不可部分が複数あるならば、そのような部分のそれぞれの題名の最後に、(もし分かっているならば)その部分の原著者あるいは出版者の名前で、あるいは他と重ならないような番号を括弧で括って記載することで、それぞれ見分けが付くようにしなければならない。結合後の著作物の利用許諾告知における変更不可部分の一覧においても、章の題名に同様の調整をすること。

結合後の著作物においては、あなたはそれぞれの原本の「履歴 (History)」と 題されたあらゆる部分をまとめて、「履歴 (History)」と題された一章にしなければならない。同様に、「謝辞 (Acknowledgements)」あるいは「献辞 (Dedications)」と題されたあらゆる部分もまとめなければならない。あなたは「推薦の辞 (Endowsements)」と題されたあらゆる部分も削除しなければならない。

6. 文書の収集

あなたは、この利用許諾契約書の下で発表された複数の文書で構成される収集著作物を作ることができる。その場合、それぞれの文書が逐語的に忠実に複製されることを保障するために他のすべての点でこの契約書の定める条件に従う限り、さまざまな文書中のこの契約書の個々の複製物を、収集著作物中に複製物を一つ含めることで代用することができる。

あなたは、このような収集著作物から文書を一つ取り出し、それをこの契約書の下で頒布することができる。ただしその際には、この契約書の複製物を抽出された文書に挿入し、またその他すべての点でこの文書の逐語的に忠実な複製に関してこの契約書が定める条件に従わなければならない。

7. 独立した著作物の集積

『文書』あるいはその派生物を、他の別の独立した文書あるいは著作物と一緒にし、一巻の記憶装置あるいは頒布媒体に収めた編集著作物は、編集に起因する著作権が編集著作物に含まれる個々の著作物がその利用者に許可した法的権利を制限するよう行使されない限り、「集積」著作物と呼ばれる。『文書』が集積著作物に含まれる場合、この契約書は、『文書』と共にまとめられた他の独立した著作物には、それら自身が『文書』の派生物で無い限り適用されることにはならない。

このような『文書』の複製物において、この利用許諾契約書の第3項によりカバーテキストの掲載が要求されている場合、『文書』の量が集積著作物全体の2分の1以下であれば、『文書』のカバーテキストは集積著作物中で『文書』そのものの周りを囲む中表紙、あるいは『文書』が電子的形式である場合には表紙の電子的等価物にのみ配置するだけでよい。その場合以外は、カバーテキストは集積著作物全体を取り巻く印刷された表紙に掲載されなければならない。

8. 翻訳

翻訳は改変の一種と見做すので、あなたは『文書』の翻訳をこの利用許諾契約書の第4項の定める条件の下で頒布することができる。変更不可部分を翻訳によって置き換えるには著作権者の特別許可を必要とするが、元の変更不可部分に追加する形で変更不可部分の全てないし一部の翻訳を含めることはかまわない。この契約書や『文書』中の利用許諾告知、保証否認警告すべての英語原本も含める限り、あなたはこの契約書、告知、警告の翻訳を含めることができる。契約書や告知、警告に関して翻訳と英語原本との間に食い違いが生じた場合、英語原本が優先される。

典型的な例として、『文書』のある部分が原文で「Acknowledgements」、「Dedications」、あるいは「History」と題されていた場合、実際の題名を変更するには、題名を保存する(この契約書の第1項)ための条件(同第4項)を満たすことが必要となる。

9. 契約の終了

この利用許諾契約書の下で明確に提示されている場合を除き、あなたは『文書』を複製、改変、サブライセンス、あるいは頒布してはならない。このライセンスで指定されている以外の、『文書』の複製、改変、サブライセンス、頒布に関するすべての企ては無効であり、この契約書によって保証されるあなたの権利を自動的に終結させることとなる。しかし、この契約書の下であなたから複製物ないし諸権利を得た個人や団体に関しては、そういった人々がこの契約書に完全に従ったままである限り、彼らに与えられた許諾は終結しない。

10. 将来における本利用許諾契約書の改訂

フリーソフトウェア財団は、時によってGNUフリー文書利用許諾契約書の新しい改訂版を出版することができる。そのような新版は現在の版と理念においては似たものになるであろうが、新たに生じた問題や懸念を解決するため細部においては違ったものになるだろう。詳しくは https://www.gnu.org/copyleft/ を参照せよ。

GNUフリー文書利用許諾契約書のそれぞれの版には、新旧の区別が付くようなバージョン番号が振られている。もし『文書』において、この契約書のある特定の版か「それ以降のどの版でも」適用して良いと指定されている場合、あなたはフリーソフトウェア財団から発行された(草稿として発表されたものを除く)指定の版かそれ以降の版のうちどれか一つを選び、その条項や条件に従うことができる。もし『文書』がこの契約書のバージョン番号を指定していない場合には、あなたはフリーソフトウェア財団から今までに出版された(草稿として発表されたものを除く)版のうちからどれか一つを選ぶことができる。

ADDENDUM: この利用許諾契約書をあなたの文書に適用するには

この利用許諾契約書をあなたが書いた文書に適用するには、この契約書の複製物一つを文書中に含め、以下に示す著作権表示と利用許諾告知を題扉のすぐ後に置いて下さい:

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

(訳: Copyright (C) 西暦年 あなたの名前. この文書を、フリーソフトウェア財団発行のGNU フリー文書利用許諾契約書(バージョン1.2かそれ以降から一つを選択)が定める条件の下で複製、頒布、あるいは改変することを許可する。変更不可部分、表カバーテキスト、裏カバーテキストは存在しない。この利用許諾契約書の複製物は「GNU フリー文書利用許諾契約書」という章に含まれている。)

もし変更不可部分や表カバーテキスト、裏カバーテキストがあれば、「変更不可部分…は存在しない。」というところを以下で置き換えてください:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

(訳: (章の題名を列記)は変更不可部分であり、(表カバーテキストを列記)は表カバーテキスト、(裏カバーテキストを列記)は裏カバーテキストである。)

変更不可部分はあるがカバーテキストは存在しないなど、その他の三者の組み合せに関しては、状況に合わせて上記二つの選択肢を混ぜてください。

あなたの文書に、他に類を見ない独自のプログラムコードのサンプルが含まれる場合、フリーソフトウェアにおいてそのコードを利用することを許可するために、そういったサンプルに関してはこの利用許諾契約書と同時にGNU一般公衆許諾契約書のようなフリーソフトウェア向けライセンスのうちどれか一つを選択して適用してもよい、というような条件の下で発表することを推奨します。