Oracle® Solaris 11.3 での Kerberos およびその他の認証サービスの管理

Part No: E62754
2017 年 3 月

Part No: E62754

ドキュメントのアクセシビリティについて

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWeb サイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info) か、聴覚に障害のあるお客様は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

目次

このドキュメントの使用法 17

1 プラグイン可能認証モジュールの使用 19
PAM について 19
PAM フレームワークの概要 19
PAM を使用する利点 21
サイト固有の PAM 構成の計画 21
ユーザーごとの PAM ポリシーの割り当て 22
PAM の構成 23
▼ コンソールにログインできるユーザーを制限する方法 24
▼ サイト固有の PAM 構成ファイルを作成する方法 25
▼ PAM モジュールを追加する方法 27
▼ 変更された PAM ポリシーを割り当てる方法 29
▼ PAM のエラーレポートを記録する方法 32
▼ PAM 構成のエラーをトラブルシューティングする方法 33
PAM 構成のリファレンス 34
PAM 構成ファイル 34
PAM 構成の検索順序 35
PAM 構成ファイルの構文 35
PAM スタック 36
PAM スタックの例 40
PAM サービスモジュール 41

2 Kerberos サービスについて 45
Kerberos サービスとは 45
Kerberos サービスの動作 46
初期認証: チケット認可チケット (TGT) 47
後続の Kerberos 認証 49
バッチジョブの Kerberos 認証 50

Kerberos、DNS、およびネームサービス 51
Kerberos のコンポーネント 51
Kerberos ネットワークプログラム 51
Kerberos 主体 52
Kerberos レルム 53
Kerberos サーバー 53
Kerberos ユーティリティー 54
Kerberos セキュリティーサービス 55
Kerberos 暗号化タイプ 56
FIPS 140-2 アルゴリズムと Kerberos 暗号化タイプ 58
Kerberos 資格によってサービスへのアクセスが提供されるしくみ 58
チケット認可サービスに対する資格の取得 59
Kerberos サーバーに対する資格の取得 60
特定の Kerberos サービスへのアクセスの取得 61
Oracle Solaris Kerberos と MIT Kerberos の大きな違い 62

3 Kerberos サービスの計画 63
Kerberos 配備の計画 63
Kerberos レルムの計画 63
Kerberos レルム名 64
Kerberos レルムの数 64
Kerberos レルムの階層 64
ホスト名の Kerberos レルムへのマッピング 65
Kerberos クライアント名とサービス主体名 65
Kerberos レルム内でのクロック同期 66
Kerberos でサポートされる暗号化タイプ 66
KDC の計画 67
KDC と管理サービス用のポート 67
スレーブ KDC の数 67
Kerberos データベースの伝播 68
KDC の構成オプション 68
Kerberos クライアントの計画 69
Kerberos クライアントの自動インストールの計画 69
Kerberos クライアントの構成オプション 70
Kerberos クライアントログインのセキュリティー 70
Kerberos での信頼できる委任されたサービス 71
Kerberos での UNIX 名と UNIX 資格の使用の計画 71
GSS 資格の UNIX 資格へのマッピング 71

gsscred テーブル 72
Kerberos レルムへのユーザーの自動的な移行 72

4 Kerberos サービスの構成 73
Kerberos サービスの構成 73
追加の Kerberos サービスの構成 74
KDC サーバーの構成 75
▼ KDC パッケージをインストールする方法 76
▼ FIPS 140-2 モードで実行するように Kerberos を構成する方法 77
▼ kdcmgr を使用してマスター KDC を構成する方法 78
▼ kdcmgr を使用してスレーブ KDC を構成する方法 80
▼ マスター KDC を手動で構成する方法 81
▼ スレーブ KDC を手動で構成する方法 86
▼ 2 番目のマスター KDC を手動で構成する方法 90
マスターサーバー上のチケット認可サービス鍵の置き換え 92
LDAP ディレクトリサーバー上での KDC サーバーの構成 93
OpenLDAP ディレクトリサーバー上でのマスター KDC の構成 94
Oracle Unified Directory サーバー上でのマスター KDC の構成 98
Oracle DSEE LDAP ディレクトリサーバー上でのマスター KDC の構成 103
▼ Oracle DSEE サーバー上で Kerberos 以外のオブジェクトクラス型に Kerberos 主体の属性を混在させる方法 110
▼ LDAP ディレクトリサーバー上で Kerberos レルムを破棄する方法 111
Kerberos クライアントの構成 111
▼ Kerberos クライアントのインストールプロファイルの作成方法 112
▼ Kerberos クライアントを自動的に構成する方法 113
▼ Kerberos クライアントを対話的に構成する方法 114
▼ Kerberos クライアントを Active Directory サーバーに参加させる方法 117
▼ Kerberos クライアントを手動で構成する方法 118
チケット認可チケットの確認の無効化 123
▼ Kerberos によって保護された NFS ファイルシステムに root ユーザーとしてアクセスする方法 124
▼ Kerberos レルム内のユーザーを自動的に移行するように構成する方法 124
すべてのチケット認可チケットの自動的な更新 129
Kerberos ネットワークアプリケーションサーバーの構成 130

▼ Kerberos ネットワークアプリケーションサーバーを構成する方法 130
▼ FTP の実行時に Generic Security Service を Kerberos とともに使用する方法 131
Kerberos NFS サーバーの構成 132
▼ Kerberos NFS サーバーを構成する方法 133
▼ 資格テーブルを作成および変更する方法 134
▼ レルム間の資格マッピングを提供する方法 135
▼ 複数の Kerberos セキュリティーモードで安全な NFS 環境を設定する方法 136
Kerberos サービスへのアクセスのための遅延実行の構成 138
▼ Kerberos サービスにアクセスするための cron ホストを構成する方法 139
レルム間認証の構成 140
▼ 階層関係のレルム間認証を設定する方法 140
▼ 直接接続のレルム間認証を確立する方法 141
KDC と Kerberos クライアントのクロックの同期化 142
マスター KDC とスレーブ KDC の入れ替え 144
▼ 入れ替え可能なスレーブ KDC を構成する方法 144
▼ マスター KDC とスレーブ KDC を入れ替えする方法 145
Kerberos データベースの管理 148
Kerberos データベースのバックアップと伝播 148
▼ Kerberos データベースのバックアップを復元する方法 151
▼ サーバーのアップグレード後に Kerberos データベースを変換する方法 151
▼ マスター KDC を再構成して増分伝播を使用する方法 152
▼ スレーブ KDC を再構成して増分伝播を使用する方法 154
▼ KDC サーバーが同期しているかを検査する方法 154
Kerberos データベースのスレーブ KDC への手動での伝播 156
Kerberos のための並列伝播の設定 157
並列伝播を設定するための構成手順 157
Kerberos データベースの stash ファイルの管理 158
▼ Kerberos データベースの新しいマスター鍵を作成、使用、および格納する方法 159
Kerberos サーバー上のセキュリティーの強化 161
KDC サーバーへのアクセスの制限 161
辞書ファイルを使用したパスワードセキュリティーの強化 161

5 Kerberos 主体とポリシーの管理 163

Kerberos 主体とポリシーの管理方法 163
新しい Kerberos 主体の自動作成 164
Kerberos 主体の管理 164
Kerberos 主体とその属性の表示 165
新しい Kerberos 主体の作成 165
Kerberos 主体の変更 166
Kerberos 主体の削除 166
主体の Kerberos 管理権限の変更 167
Kerberos ポリシーの管理 168
keytab ファイルの管理 170
keytab ファイルへの Kerberos サービス主体の追加 171
キータブファイルからのサービス主体の削除 172
keytab ファイル内の主体の表示 172
ホスト上の Kerberos サービスの一時的な無効化 173
▼ ホスト上の Kerberos サービスの認証を一時的に無効にする方法 173

6 Kerberos アプリケーションの使用 177
Kerberos チケットの管理 177
Kerberos チケットの作成 178
Kerberos チケットの表示 178
Kerberos チケットの破棄 180
Kerberos パスワードの管理 180
パスワードの変更 180
Kerberos でのリモートログイン 181
Kerberos のユーザーコマンド 182

7 Kerberos サービスのリファレンス 183
Kerberos ファイル 183
Kerberos コマンド 184
Kerberos デーモン 186
Kerberos の用語 186
Kerberos 固有の用語 187
認証固有の用語 187
チケットの種類 188

8 Kerberos エラーメッセージとトラブルシューティング 195
Kerberos のエラーメッセージ 195
Kerberos 共通エラーメッセージ (A - M) 195

Kerberos 共通エラーメッセージ (N - Z) 204
Kerberos のトラブルシューティング 207
鍵バージョン番号に関する問題 207
krb5.conf ファイルの形式に関する問題 208
Kerberos データベースの伝播の問題 208
Kerberos NFS ファイルシステムのマウントの問題 208
root ユーザーの認証の問題 209
GSS 資格の UNIX 資格へのマッピングの監視 209
Kerberos サービスでの DTrace の使用 210

9 簡易認証セキュリティー層の使用 217
SASL について 217
SASL のリファレンス 217
SASL プラグイン 218
SASL の環境変数 218
SASL のオプション 219

10 Oracle Solaris でのマルチファクタ認証へのスマートカードの使用 221
ツーファクタ認証とスマートカード 221
ツーファクタ認証について 221
Oracle Solaris でのツーファクタ認証の実装 225
スマートカードのログインのための Oracle Solaris システムの構成 230
主なスマートカード構成タスク 231
スマートカードパッケージのインストール 232
スマートカードへの pcsclite の使用 232
スマートカードリーダー用の libccid の構成 233
スマートカードを持つユーザーのデスクトップの構成 234
スマートカード用の OCSP 証明書の構成 235
スマートカード用の PAM の構成 239
Secure Shell クライアントのスマートカード用の構成 246
スマートカードログインのための Oracle Solaris システムの有効化 247
▼ スマートカード認証を有効にする方法 248
Web ブラウザおよび電子メールでのスマートカード使用の有効化 248
▼ Web および電子メールの使用のためにスマートカード証明書をダウンロードする方法 249
▼ 認証にスマートカードを使用するように Firefox を構成する方法 249
▼ 電子メールの署名および暗号化にスマートカードを使用するように Thunderbird を構成する方法 250

スマートカードの使用 251
▼ スマートカード認証を使用してローカルコンソールからログインする方法 252
▼ スマートカード認証を使用して役割としてログインする方法 253
▼ スマートカード認証で ssh を使用してリモートでログインする方法 254
▼ スマートカードを使用してリモートの GNOME デスクトップに ssh を実行する方法 255
▼ スマートカードを使用してローカルの GNOME デスクトップにログインする方法 255
▼ スクリーンセーバーでスマートカードを使用して認証する方法 259

11 Oracle Solaris でのマルチファクタ認証へのワンタイムパスワードの使用 263
Oracle Solaris での OTP について 263
Oracle Solaris での OTP 管理 264
Oracle Solaris での OTP の構成および使用 264
▼ OTP の構成方法 266
▼ OTP の秘密鍵を構成および確認する方法 266
▼ OTP ユーザー用に秘密鍵を設定する方法 268
▼ Oracle Solaris システムへのログインに UNIX パスワードおよび OTP を要求する方法 269

12 ネットワークサービスの認証の構成 273
Secure RPC について 273
NFS サービスと Secure RPC 273
Kerberos 認証 274
Secure NFS での DES 暗号化 274
Diffie-Hellman 認証と Secure RPC 274
Secure RPC による認証の管理 275
▼ Secure RPC キーサーバーを再起動する方法 275
▼ NIS ホストに Diffie-Hellman 鍵を設定する方法 276
▼ NIS ユーザーに Diffie-Hellman 鍵を設定する方法 277
▼ Diffie-Hellman 認証で NFS ファイルを共有する方法 278

A Kerberos 用の DTrace プローブ 279
Kerberos での DTrace プローブ 279
Kerberos DTrace プローブの定義 280
Kerberos での DTrace 引数構造体 281

DTrace での Kerberos メッセージ情報 281
DTrace での Kerberos 接続情報 281
DTrace での Kerberos オーセンティケータ情報 282

用語集 285

索引 295

表目次

表 1 PAM のタスクマップ 23
表 2 タスクマップ: Kerberos サービスの構成 74
表 3 タスクマップ: 追加の Kerberos サービスの構成 74
表 4 タスクマップ: KDC サーバーの構成 75
表 5 タスクマップ: LDAP を使用するための Kerberos の構成 93
表 6 タスクマップ: Kerberos クライアントの構成 111
表 7 タスクマップ: Kerberos NFS サーバーの構成 133
表 8 タスクマップ: Oracle Solaris での OTP の使用 265
表 9 タスクマップ: Secure RPC による認証の管理 275

例目次

例 1 変更された PAM スタックを使用した暗号化されたホームディレクトリの作成 26
例 2 ログイン時のユーザーへのエラーメッセージ表示の回避 27
例 3 ユーザーごとの PAM ポリシーファイルへの新しいモジュールの追加 28
例 4 権利プロファイルを使用したユーザーごとの PAM ポリシーの設定 29
例 5 `ktelnet` PAM スタックの選択されたユーザーへの制限 31
例 6 引数なしでの `kdcmgr` コマンドの実行 79
例 7 インストールプロファイルを使用した Kerberos クライアントの構成 114
例 8 `kclient` スクリプトの実行例 116
例 9 マルチマスター KDC と連携するための Oracle Solaris クライアントの構成 122
例 10 Oracle Solaris 以外の KDC のための Kerberos クライアントの構成 122
例 11 ホストおよびドメイン名の Kerberos レルムへのマッピングのための DNS TXT レコード 122
例 12 Kerberos サーバーの場所を記録する DNS SRV レコード 123
例 13 すべてのユーザーへの TGT 有効期限切れメッセージの構成 129
例 14 ユーザーへの TGT 有効期限メッセージの構成 129
例 15 異なるドメイン内の主体の Kerberos 資格テーブルへの追加 135
例 16 1 つの Kerberos セキュリティーモードでファイルシステムを共有する 138
例 17 複数の Kerberos セキュリティーモードでファイルシステムを共有する 138
例 18 Kerberos データベースの手動でのバックアップ 150
例 19 Kerberos データベースの復元 151
例 20 KDC サーバーが同期されていることの確認 155
例 21 Kerberos での並列伝播の設定 158
例 22 Kerberos 主体の表示 165

例 23 Kerberos 主体の属性の表示 165
例 24 新しい Kerberos 主体の作成 166
例 25 Kerberos 主体のパスワード再試行の最大数の変更 166
例 26 Kerberos 主体のパスワードの変更 166
例 27 Kerberos 主体の権限の変更 168
例 28 Kerberos ポリシーのリストの表示 168
例 29 Kerberos ポリシーの属性の表示 168
例 30 新しい Kerberos パスワードポリシーの作成 169
例 31 Kerberos アカウントのロックアウトポリシーの処理 169
例 32 Kerberos ポリシーの変更 169
例 33 Kerberos ポリシーの削除 169
例 34 サービス主体のキータブファイルへの追加 171
例 35 キータブファイルからのサービス主体の削除 172
例 36 キータブファイル内のキー一覧 (主体) の表示 173
例 37 Kerberos ホストの一時的な無効化 174
例 38 Kerberos チケットの作成 178
例 39 Kerberos チケットの表示 179
例 40 DTrace を使用した Kerberos メッセージの追跡 210
例 41 DTrace を使用した Kerberos 事前認証タイプの表示 211
例 42 DTrace を使用した Kerberos のエラーメッセージのダンプ 213
例 43 DTrace を使用した SSH サーバーのサービスチケットの表示 213
例 44 DTrace を使用した、初期 TGT のリクエスト時に使用できない KDC のアドレスとポートの表示 213
例 45 DTrace を使用した Kerberos 主体からのリクエストの表示 214
例 46 ユーザーによる長い OTP および強力なアルゴリズムへの変更 267
例 47 16 進数の秘密鍵の設定および表示 268
例 48 長い OTP および強力なアルゴリズムへの変更の強制 270
例 49 OTP 認証にタイマーではなくカウンタを使用 271
例 50 NIS クライアント上で `root` の新しい鍵を設定する 277
例 51 NIS で新しいユーザー鍵を設定して暗号化する 277

このドキュメントの使用法

- **概要** – 1 つ以上の Oracle Solaris システム上でセキュア認証を管理する方法について説明します。このガイドでは、プラグイン可能認証モジュール (PAM)、Kerberos、簡易認証セキュリティー層 (SASL)、スマートカードとワンタイムパスワード (OTP) を使用したツーファクタ認証 (2FA)、および Secure RPC for NFS および NIS について説明します。付録では、Kerberos 用の DTrace プローブを、その使用法の例とともに説明します。
- **対象読者** – システム、セキュリティー、およびネットワークセキュリティー管理者。
- **前提知識** – アクセス要件およびネットワークセキュリティー要件。

製品ドキュメントライブラリ

この製品および関連製品のドキュメントとリソースは http://www.oracle.com/pls/topic/lookup?ctx=E62101-01 で入手可能です。

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

♦ ♦ ♦ 第 1 章

1 プラグイン可能認証モジュールの使用

この章では、プラグイン可能認証モジュール (PAM) について説明します。PAM は、Oracle Solaris OS 上でアプリケーションのユーザーに対するチェックをプラグインするためのフレームワークを提供します。PAM は、ユーザーの認証、パスワード変更の管理、ユーザーのセッションの終了および開始、アカウント制限 (時間など) の追跡をはじめとする、アプリケーションの使用を管理するための集中管理フレームワークを提供します。PAM はサードパーティーのアプリケーションまで拡張できるため、システム上のサービスへのアクセスのシームレスな管理を提供できます。

この章で扱う内容は、次のとおりです。

- 19 ページの「PAM について」
- 23 ページの「PAM の構成」
- 34 ページの「PAM 構成のリファレンス」

PAM について

PAM は、アプリケーションがさまざまな認証機能を実行するためのフレームワークを提供します。アプリケーション (システムプログラムを含む) のユーザーに対する集中管理認証、セッション管理、パスワード管理、およびアカウント制限を提供します。PAM を使用すると、これらのプログラム (`login`、`su`、`ssh` など) は、ユーザー管理の詳細が変更された場合でも変更する必要がありません。サイトアプリケーションは、PAM を使用して、独自のアカウント、資格、セッション、パスワードなどの要件を管理できます。PAM は、これらのアプリケーションに「プラグイン」されます。

PAM フレームワークの概要

PAM フレームワークは、次の 4 つの部分で構成されます。

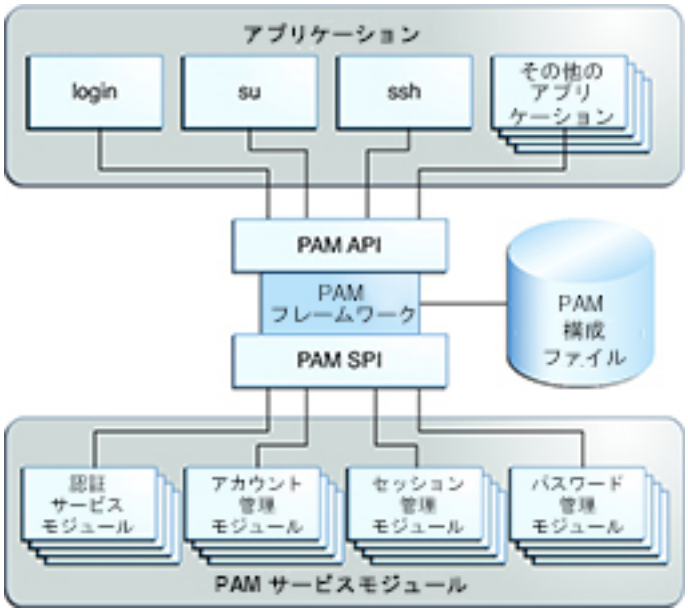
- PAM を使用するアプリケーション
- PAM フレームワーク

- PAM サービスモジュール
- PAM 構成 (モジュールとユーザー割り当ての選択を含む)

このフレームワークは、認証関連のアクティビティーを実行するための統一された方法を提供します。このアプローチを使用すると、アプリケーション開発者は、認証ポリシーの意味を知らなくても PAM サービスを使用できます。PAM を使えば、管理者は、アプリケーションを変更しないで、特定システムのニーズに合わせて認証プロセスを調整できるようになります。代わりに、管理者は PAM 構成を調整します。

次の図は、PAM のアーキテクチャーを示したものです。

図 1 PAM のアーキテクチャー



このアーキテクチャーは、次のように機能します。

- アプリケーションは、PAM アプリケーションプログラミングインタフェース (API) 経由で PAM フレームワークと通信します。

  API の使用については、pam(3PAM) のマニュアルページおよび『Oracle Solaris 11 セキュリティーサービス開発ガイド』 の 第 3 章, 「PAM アプリケーションおよび PAM サービスの記述」を参照してください。
- PAM サービスモジュールは、PAM サービスプロバイダインタフェース (SPI) 経由で PAM フレームワークと通信します。詳細は、pam_sm(3PAM) のマニュアルページを参照してください。

選択されたサービスモジュールの簡単な説明については、41 ページの「PAM サービスモジュール」のほか、pam.conf(4) および pam_user_policy(5) のマニュアルページを参照してください。

管理者は、サイト要件を管理するために 1 つ以上の一連のモジュールを構成できます。この一連のモジュールは、PAM スタックと呼ばれます。このスタックは、順番に評価されます。アプリケーションに複数の PAM スタックが必要な場合、アプリケーション開発者は、複数のサービス名を作成する必要があります。たとえば、sshd デーモンは、PAM 用の複数のサービス名を提供または要求します。sshd デーモンの PAM サービス名のリストについては、sshd(1M) のマニュアルページで PAM という単語を探してください。PAM スタックの詳細は、36 ページの「PAM スタック」を参照してください。40 ページの「PAM スタックの例」では、PAM 認証スタックを示しています。

PAM を使用する利点

PAM フレームワークを使用すると、ユーザーがアプリケーションを使用するために満たす必要のある要件を構成できます。PAM によって提供される利点のいくつかを次に示します。

- 柔軟な PAM 構成ポリシー
  - サービスごとの名前認証ポリシー
  - サイト全体にわたる PAM ポリシーとユーザーごとの PAM ポリシー
  - デフォルトの認証ポリシーの管理上の選択
  - セキュリティーの高いシステム上での複数のユーザー要件の適用
- 一般ユーザーにも使いやすい
  - 認証サービスが異なっても同じパスワードの再入力は必要なし
  - ユーザーに複数のコマンドの入力を求めるのではなく、複数の認証サービスからユーザーに入力を要求
- 管理者にとっての構成のしやすさ
  - PAM サービスモジュールにオプションを渡すことが可能
  - アプリケーションを変更しなくても、サイト固有のセキュリティーポリシーを実装できる

サイト固有の PAM 構成の計画

配布時に、PAM 構成は、認証を必要とするシステムサービス (login や ssh など) を対象とする標準のセキュリティーポリシーを実装します。一部のシステムサービスのために異なるセキュリティーポリシーを実装するか、またはサードパーティーのアプ

リケーションのためのポリシーを作成する必要がある場合は、次の問題を考慮してください。

- 提供されている構成ファイルによって要件が満たされていないかどうかを判定します。

  デフォルト構成をテストします。`/etc/security/pam_policy` ディレクトリ内のユーザーごとのファイルをテストします。デフォルトのサービス名 `other` によって要件が満たされているかどうかをテストします。40 ページの「PAM スタックの例」では、`other` スタックを示しています。
- スタックに変更が必要なサービス名があるかどうかを識別します。サービス名の PAM スタックの変更の例については、25 ページの「サイト固有の PAM 構成ファイルを作成する方法」を参照してください。
- PAM フレームワークを使用するようにコーディングされているサードパーティーのアプリケーションの場合は、そのアプリケーションが使用する PAM サービス名を判定します。
- 各サービス名について、どの PAM モジュールを使用するかを判定します。

  各 PAM モジュールのマニュアルページのセクション 5 を確認します。これらのマニュアルページでは、各モジュールの機能、使用可能なオプション、およびスタック内のモジュール間の相互作用について説明しています。選択されたモジュールの簡単なサマリーについては、41 ページの「PAM サービスモジュール」を参照してください。PAM モジュールはまた、外部のソースからも使用できます。
- サービス名ごとに、各モジュールを実行する順序を決定します。
- 各モジュールに対する制御フラグを選択します。制御フラグの詳細は、36 ページの「PAM スタック」を参照してください。制御フラグがセキュリティーに影響する場合があることに注意してください。

  図解については、図2および図3を参照してください。
- 各モジュールに必要なオプションを選択します。各モジュールのマニュアルページには、そのモジュールで使用可能なオプションが記載されています。
- PAM 構成でのアプリケーションの使用をテストします。`root` 役割、その他の役割、特権ユーザー、および通常のユーザーとしてテストします。一部のユーザーがそのアプリケーションの使用を許可されていない場合は、それらのユーザーをテストします。

ユーザーごとの PAM ポリシーの割り当て

`pam_user_policy` PAM モジュールを使用すると、システム管理者は、ユーザーごとに特定の PAM 構成を割り当てることができます。このモジュールが PAM スタック内の最初のモジュールであり、かつユーザーの `pam_policy` セキュリティー属性が PAM 構成ファイルを指定している場合は、そのファイルがユーザーの PAM ポリシーを指定します。

詳細については、次を参照してください。

- pam_user_policy(5) man page
- 36 ページの「PAM スタック」
- 25 ページの「サイト固有の PAM 構成ファイルを作成する方法」
- 例4「権利プロファイルを使用したユーザーごとの PAM ポリシーの設定」

PAM の構成

PAM は、そのままで使用できます。このセクションでは、デフォルトで有効になっていない PAM 構成の例を示します。

表 1 PAM のタスクマップ

タスク	説明	参照先
PAM のインストールを計画します。	サイト用に PAM のカスタマイズを計画する方法を対象としています。	21 ページの「サイト固有の PAM 構成の計画」
コンソールログインが制限されていることを確認します。	コンソールログインを指定されたユーザーおよびネットグループに制限します。	24 ページの「コンソールにログインできるユーザーを制限する方法」
ユーザーに新しい PAM ポリシーを割り当てます。	複数のサービスに対するユーザーごとの認証要件をカスタマイズします。	25 ページの「サイト固有の PAM 構成ファイルを作成する方法」
ホームディレクトリが暗号化されているユーザーを作成します。	暗号化されたホームディレクトリを作成できるように PAM スタックを変更します。	例1「変更された PAM スタックを使用した暗号化されたホームディレクトリの作成」
新しい PAM モジュールを追加します。	カスタマイズされた PAM モジュールをインストールしてテストする方法について説明します。	27 ページの「PAM モジュールを追加する方法」
デフォルト以外の PAM ポリシーをユーザーに割り当てます。	Kerberos、LDAP、またはログインの組み合わせを使用するサイトで、ある範囲のユーザーに割り当てる権利プロファイルに PAM ポリシーを追加する方法を示します。	29 ページの「変更された PAM ポリシーを割り当てる方法」
デフォルト以外の PAM ポリシーをユーザーに割り当てます。	カスタマイズされた PAM スタックをすべてのシステムに配布します。	29 ページの「変更された PAM ポリシーを割り当てる方法」
エラーロギングを開始します。	syslog を使用して PAM エラーメッセージをログに記録します。	32 ページの「PAM のエラーレポートを記録する方法」
PAM エラーをトラブルシューティングします。	PAM の構成の誤りを見つけて、解決し、テストするための手順を提供します。	33 ページの「PAM 構成のエラーをトラブルシューティングする方法」

▼ コンソールにログインできるユーザーを制限する方法

このタスクでは、特定のユーザーに対してコンソールへのアクセスを制限します。/etc/pam.d/login 構成ファイルは、コンソールログインを制御します。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **/etc/pam.d/login ファイルを変更します。**

   a. **login ファイルのコピーを保存してから、元のファイルを開きます。**

   ```
   # cd /etc/pam.d
   # cp login login.orig
   # pfedit login
   ```

   b. **次のエントリを追加します。**

   ```
   ## Account management for login(1) incorporates pam_list(5)
   ## Restricts who can log in on the console to the users and netgroups
   ## that are listed in the /etc/users.allow file
   account requisite pam_roles.so.1
   account definitive pam_user_policy.so.1
   account required pam_unix_account.so.1
   account required pam_list.so.1 allow=/etc/users.allow
   account required pam_tsol_account.so.1
   ```

2. **/etc/users.allow ファイルを作成し、保護します。**

   ```
   # cd /etc
   # touch users.allow ; chmod 644 users.allow
   ```

3. **/etc/users.allow ファイルにユーザーを追加します。**

   - **たとえば、jdoe アカウントを追加します。**

     ```
     ## permitted console logins
     jdoe
     ```

   - **たとえば、ネットグループを追加します。**

     ネットグループは、LDAP または NIS で一元的に定義され、ユーザーメンバーを持つグループです。一覧表示されているネットグループのメンバーは、コンソールのこの特定のシステムにログインできるようになります。

     ```
     ## permitted console logins
     jdoe
     @alladmins
     ```

     詳細は、netgroup(4)および pam_list(5) のマニュアルページを参照してください。

▼ サイト固有の PAM 構成ファイルを作成する方法

デフォルト構成では、ssh および telnet エントリサービスが other サービス名の対象になっています。この手順の PAM 構成ファイルによって、ssh および telnet に対する要件が変更されます。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **新しい PAM ポリシー構成ファイルを作成します。**

   pfedit コマンドを使用してファイルを作成します。そのファイルを /opt などのサイトの構成ディレクトリ内に配置します。また、/etc/security/pam_policy ディレクトリ内に配置することもできます。

   注記 - /etc/security/pam_policy ディレクトリ内の既存のファイルを変更しないでください。

   ファイル内に説明のコメントを含めます。

```
# pfedit /opt/local_pam/ssh-telnet-conf
#
# PAM configuration which uses UNIX authentication for console logins,
# (see pam.d/login), and LDAP for SSH keyboard-interactive logins
# This stack explicitly denies telnet logins.
#
sshd-kbdint  auth requisite          pam_authtok_get.so.1
sshd-kbdint  auth binding            pam_unix_auth.so.1 server_policy
sshd-kbdint  auth required           pam_unix_cred.so.1
sshd-kbdint  auth required           pam_ldap.so.1
#
telnet auth     requisite    pam_deny.so.1
telnet account  requisite    pam_deny.so.1
telnet session  requisite    pam_deny.so.1
telnet password requisite    pam_deny.so.1
```

2. **ファイルを保護します。**

   root 所有権と 444 のアクセス権でファイルを保護します。

```
# ls -l /opt/local_pam

total 5
-r--r--r--   1 root         4570 Jun 21 12:08 ssh-telnet-conf
```

3. **ポリシーを割り当てます。**

   29 ページの「変更された PAM ポリシーを割り当てる方法」を参照してください。

例 1 変更された PAM スタックを使用した暗号化されたホームディレクトリの作成

デフォルトでは、zfs_pam_key モジュールは /etc/security/pam_policy/unix ファイルに含まれていません。この例では、管理者はユーザーごとの PAM ポリシーの unix バージョンを作成したあと、その新しいバージョンを使用して、ホームディレクトリが暗号化されているユーザーを作成します。

```
# cp /etc/security/pam_policy/unix /opt/local_pam/unix-encrypt
# pfedit /opt/local_pam/unix-encrypt.conf
...
other   auth required           pam_unix_auth.so.1
other   auth required           pam_unix_cred.so.1
## pam_zfs_key auto-creates an encrypted home directory
##
other auth required          pam_zfs_key.so.1 create
```

管理者は、ユーザーを追加するときにこのポリシーファイルを使用します。暗号化はファイルシステムに追加できないことに注意してください。暗号化を有効にしてファイルシステムを作成する必要があります。詳細は、zfs_encrypt(1M) を参照してください。

管理者はユーザーを作成し、パスワードを割り当てます。

```
# useradd -K pam_policy=/opt/local_pam/unix-encrypt.conf jill
# passwd jill
New Password: xxxxxxxx
Re-enter new Password: xxxxxxxx
passwd: password successfully changed for jill
```

次に、管理者はそのユーザーとしてログインすることによって、暗号化されたホームディレクトリを作成します。

```
# su - jill
Password: xxxxxxxx
Creating home directory with encryption=on.
Your login password will be used as the wrapping key.
Oracle Corporation      SunOS 5.11      11.3    October 2014

# logout
```

ZFS サービスモジュールへのオプションについては、pam_zfs_key(5) のマニュアルページを参照してください。

最後に、管理者は、新しいホームディレクトリが暗号化されたファイルシステムであることを確認します。

```
# mount -p | grep ~jill
rpool/export/home/jill - /export/home/jill zfs - no
rw,devices,setuid,nonbmand,exec,rstchown,xattr,atime
# zfs get encryption,keysource rpool/export/home/jill
NAME                      PROPERTY    VALUE              SOURCE
rpool/export/home/jill  encryption  on                  local
rpool/export/home/jill  keysource   passphrase,prompt  local
```

例 2 ログイン時のユーザーへのエラーメッセージ表示の回避

この例では、管理者が nowarn オプションを使用して、ログインエラーのメッセージが表示されないようにします。

```
# pfedit /opt/local_pam/unix_ldap.conf
...
##
## turn off login error message`
sshd-kbdint  auth required          pam_unix_cred.so.1    nowarn
sshd-kbdint  auth required          pam_ldap.so.1    nowarn
```

▼ PAM モジュールを追加する方法

この手順では、新しい PAM モジュールを追加、保護、およびテストする方法を示します。新しいモジュールは、サイト固有のセキュリティーポリシーや、サードパーティーのアプリケーションのサポートのために必要になる可能性があります。PAM モジュールを作成するには、『Oracle Solaris 11 セキュリティーサービス開発ガイド』の 第 3 章, 「PAM アプリケーションおよび PAM サービスの記述」を参照してください。

注記 - PAM サービスモジュールの 32 ビットバージョンと 64 ビットバージョンをインストールする必要があります。

始める前に 21 ページの「サイト固有の PAM 構成の計画」を完了します。

root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **PAM サービスモジュールの両方のバージョンをディスク上にインストールして保護します。**

   所有権とアクセス権、つまり root 所有権と 444 のアクセス権でモジュールファイルが保護されていることを確認します。

```
# cd /opt/pam_modules
# ls -lR
.:
total 4
-r--r--r--   1 root     root        4570 Nov 27 12:34 pam_app1.so.1
drwxrwxrwx   2 root     root           3 Nov 27 12:38 sparcv9

./64:
total 1
-r--r--r--   1 root     root        4862 Nov 27 12:38 pam_app1.so.1
```

   32 ビットモジュールは /opt/pam_modules ディレクトリ内にあり、64 ビットモジュールは 64 サブディレクトリ内にあります。

2. **モジュールを適切な PAM 構成ファイルに追加します。**

   次の例では、モジュールを新しいアプリケーション app1 用にします。そのサービス名は、アプリケーション名と同じです。/etc/pam.d ディレクトリ内に app1 の service-name ファイルを作成します。このファイル内の最初のエントリにより、app1 サービスを個々のユーザーに割り当てることができます。

```
# cd /etc/pam.d
# pfedit app1
...
# PAM configuration
#
# app1 service
#
auth definitive         pam_user_policy.so.1
auth required           /opt/pam_modules/$ISA/pam.app1.so.1  debug
```

   モジュールパス内の $ISA トークンは、呼び出し側アプリケーションに対応する 32 ビットまたは 64 ビットアーキテクチャーバージョンのサービスモジュールを PAM フレームワークに指示します。32 ビットアプリケーションの場合は */a/b/*$ISA/*module*.so が */a/b/module*.so となり、64 ビットアプリケーションの場合はそれが */a/b/*64/*module*.so となります。この例では、/opt/pam_modules ディレクトリにある 32 ビットの pam.app1.so.1 サービスモジュールと、/opt/pam_modules/64 ディレクトリにある 64 ビットモジュールをインストールしました。

   詳細は、pfedit(1M) および pam.conf(4) のマニュアルページを参照してください。

   app1 の PAM ポリシーを選択されたユーザーに制限するには、例3「ユーザーごとの PAM ポリシーファイルへの新しいモジュールの追加」を参照してください。

3. **新しいサービスをテストします。**

   login または ssh を使用して、直接ログインします。次に、新しいモジュールによって影響を受けるコマンドを実行します。影響を受けるコマンドの使用を許可されるユーザーと拒否されるユーザーをテストします。トラブルシューティングの詳細は、33 ページの「PAM 構成のエラーをトラブルシューティングする方法」を参照してください。

4. **ポリシーを割り当てます。**

   29 ページの「変更された PAM ポリシーを割り当てる方法」を参照してください。

例 3 ユーザーごとの PAM ポリシーファイルへの新しいモジュールの追加

この例では、すべてのユーザーが app1 サービスを使用しているわけではないため、管理者はそのサービスをユーザーごとのポリシーとして追加します。

```
# cd /etc/pam.d
# cp app1 /opt/local_pam/app1-conf
# pfedit /opt/local_pam/app1-conf

## app1 service
##
```

```
app1 auth definitive         pam_user_policy.so.1
app1 auth required           /opt/pam_modules/$ISA/pam_app1.so.1  debug
```

管理者は、pam.d ディレクトリから app1 ファイルを削除します。

```
# rm /etc/pam.d/app1
```

次に、管理者は、システム管理者の PAM ポリシーに app1-conf ポリシーを追加します。

```
# rolemod -K pam_policy=/opt/local_pam/app1-conf sysadmin
```

例 4 権利プロファイルを使用したユーザーごとの PAM ポリシーの設定

この例では、pam_policy セキュリティー属性を使用して、異なるネームサービスからのユーザーを認証できるようにします。any PAM ポリシーファイルは、/etc/security/pam_policy ディレクトリ内に提供されています。このファイル内のコメントがこのポリシーを説明しています。

このディレクトリ内のファイルを変更しないでください。

```
# profiles -p "PAM Per-User Policy of Any" \
'set desc="Profile which sets pam_policy=any";
set pam_policy=any; exit;'
```

この権利プロファイルを割り当てるには、29 ページの「変更された PAM ポリシーを割り当てる方法」を参照してください。

▼ 変更された PAM ポリシーを割り当てる方法

この手順では、すべてのシステムイメージ上にデフォルト以外の PAM ポリシーを構成します。すべてのファイルがコピーされたら、新しい PAM ポリシーまたは変更された PAM ポリシーを個々のユーザーまたはすべてのユーザーに割り当てることができます。

始める前に 新しいポリシーを実装する PAM 構成ファイルが変更およびテストされています。

root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **デフォルト以外の PAM ファイルをすべてのシステムに追加します。**

   すべての新しい PAM モジュールと、新しい PAM 構成ファイルおよび変更された PAM 構成ファイルをすべてのシステムに追加する必要があります。

   a. **最初に、すべての新しい PAM モジュールをすべてのシステムに追加します。**

i. **32 ビットの PAM モジュールをアーキテクチャーに適したディレクトリに追加します。**

ii. **64 ビットの PAM モジュールをアーキテクチャーに適したディレクトリに追加します。**

ディレクトリの設定の例については、27 ページの「PAM モジュールを追加する方法」のステップ 1 を参照してください。

b. **次に、すべての新しい PAM 構成ファイルをすべてのシステムに追加します。**

たとえば、`/opt/local_pam/ssh-telnet-conf` ファイルをすべてのシステムに追加します。

c. **次に、すべての変更された PAM 構成ファイルをすべてのシステムにコピーします。**

たとえば、変更された `/etc/pam.conf` ファイルと、すべての変更された `/etc/pam.d/`*service-name-files* をすべてのシステムにコピーします。

2. **デフォルト以外の PAM ポリシーをすべてのユーザーに割り当てます。**

a. **`policy.conf` ファイルを次のいずれかの方法で変更します。**

- **`policy.conf` ファイル内の `PAM_POLICY` キーワードに PAM 構成ファイルを追加します。**

```
# pfedit /etc/security/policy.conf
...
# PAM_POLICY=
PAM_POLICY=/opt/local_pam/ssh-telnet-conf
...
```

- **`policy.conf` ファイル内の `PROFS_GRANTED` キーワードに権利プロファイルを追加します。**

たとえば、例4「権利プロファイルを使用したユーザーごとの PAM ポリシーの設定」のユーザーごとの PAM ポリシーの任意の権利プロファイルを割り当てます。

```
# pfedit /etc/security/policy.conf
...
AUTHS_GRANTED=
# PROFS_GRANTED=Basic Solaris User
PROFS_GRANTED=PAM Per-User Policy of Any,Basic Solaris User
...
```

b. **変更された `policy.conf` ファイルをすべてのシステムにコピーします。**

3. **デフォルト以外の PAM ポリシーを個々のユーザーに割り当てるには、そのポリシーをユーザーに直接割り当てるか、またはそのポリシーをユーザーに割り当てられている権利プロファイルに追加できます。**

   - **PAM ポリシーを個々のユーザーに直接割り当てます。**

     ```
     # usermod -K pam_policy="/opt/local_pam/ssh-telnet-conf" jill
     ```

   - **権利プロファイル内に PAM ポリシーを含め、そのプロファイルを個々のユーザーに割り当てます。**

     この例では、ldap PAM ポリシーを使用します。

     ```
     # profiles -p "PAM Per-User Policy of LDAP" \
     'set desc="Profile which sets pam_policy=ldap";
     set pam_policy=ldap; exit;'
     ```

     次に、この権利プロファイルをユーザーに割り当てます。

     ```
     # usermod -P +"PAM Per-User Policy of LDAP" jill
     ```

例 5 ktelnet PAM スタックの選択されたユーザーへの制限

管理者が、Kerberos レルムで telnet を使用することを許可するユーザーの数を制限したいとします。そのため、telnet サービスが有効になる前に、管理者はデフォルトの ktelnet 構成ファイルを変更し、そのデフォルトの ktelnet ファイルを pam_policy ディレクトリ内に配置します。

最初に、管理者は、ユーザーごとの ktelnet ファイルを構成します。

```
# cp /etc/pam.d/ktelnet /etc/security/pam_policy/ktelnet-conf
# pfedit /etc/security/pam_policy/ktelnet-conf
...
# Kerberized telnet service
#
ktelnet  auth required          pam_unix_cred.so.1
ktelnet  auth required          pam_krb5.so.1
```

管理者は、そのファイルを 444 のアクセス権で保護します。

```
# chmod 444 /etc/security/pam_policy/ktelnet-conf
# ls -l /etc/security/pam_policy/ktelnet-conf
-r--r--r--   1 root     root         228 Nov 27 15:04 ktelnet-conf
```

次に、管理者は、pam.d ディレクトリ内の ktelnet ファイルを変更します。

- 最初のエントリにより、ユーザーごとの割り当てが有効になります。
- 2 番目のエントリは、管理者によって pam_policy=ktelnet が割り当てられていないかぎり、ktelnet の使用を拒否します。

```
# cp /etc/pam.d/ktelnet /etc/pam.d/ktelnet.orig
                # pfedit /etc/pam.d/ktelnet
...
# Denied Kerberized telnet service
#
```

```
auth definitive         pam_user_policy.so.1
auth required             pam_deny.so.1
```

管理者は、特権ユーザー、通常のユーザー、および root 役割を使用して構成をテストします。構成が合格すると、管理者は telnet サービスを有効にして、Kerberos 管理者にユーザーごとのポリシーを割り当てます。

```
# svcadm enable telnet
# rolemod -S ldap -K pam_policy=ktelnet-conf kerbadmin
```

管理者は、変更されたファイルをすべての Kerberos サーバーにコピーし、それらのサーバー上の telnet を有効にします。

▼ PAM のエラーレポートを記録する方法

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **どの system-log サービスインスタンスがオンラインであるかを判定します。**

```
# svcs system-log
STATE          STIME    FMRI
disabled       13:11:55 svc:/system/system-log:rsyslog
online         13:13:27 svc:/system/system-log:default
```

2. **syslog.conf ファイルを必要なロギングのレベルで構成します。**

ロギングレベルについては、syslog.conf(4) のマニュアルページの説明のセクションを参照してください。ほとんどの PAM エラー報告は、LOG_AUTH 機能を使用して実行されます。

たとえば、デバッグ出力のためのファイルを作成します。

```
# touch /var/adm/pam_debuglog
```

次に、デバッグ出力を送信するための syslog.conf エントリをそのファイルに追加します。

注記 - rsyslog サービスインスタンスがオンラインである場合は、rsyslog.conf ファイルを変更します。

```
# pfedit /etc/syslog.conf
...
*.debug         /var/adm/pam_debuglog
...
```

3. **system-log サービスの構成情報をリフレッシュします。**

```
# svcadm refresh system-log:default
```

注記 - rsyslog サービスがオンラインである場合は、system-log:rsyslog サービスインスタンスをリフレッシュします。

▼ PAM 構成のエラーをトラブルシューティングする方法

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **トラブルシューティングしている PAM エントリごとに、debug オプションを追加します。**

   たとえば、/etc/pam.d/cron ファイル内の次のエントリは、サービスに関するデバッグ出力を作成します。

   ```
   account definitive      pam_user_policy.so.1     debug
   account required        pam_unix_account.so.1    debug
   ```

2. **PAM エラーを適切なレベルでログに記録し、syslog デーモンをリフレッシュします。**

   詳細は、32 ページの「PAM のエラーレポートを記録する方法」を参照してください。

3. **問題が PAM 構成の破損である場合は、次の手順を実行します。**

   a. **1 つの端末ウィンドウからアプリケーションを実行し、別のウィンドウで PAM 構成ファイルを変更します。**

   b. **アプリケーションウィンドウで変更をテストすることによって、エラーが修正されていることを確認します。**

4. **問題が、ログインを妨げる PAM 構成の破損である場合は、シングルユーザーモードにブートしてからファイルを修正し、リブートして、テストします。**

   - **SPARC システムをブートするには、PROM プロンプトで次のコマンドを入力します。**

     ```
     ok > boot -s
     ```

   - **x86 システムをブートするには、GRUB メニューでカーネルオプションの行に -s オプションを追加します。**

詳細は、boot(1M) および grub(5) のマニュアルページを参照してください。

5. **エラーが修正されていることを確認します。**
   login または ssh を使用して、直接ログインします。通常のユーザー、特権ユーザー、および役割が、影響を受けるコマンドを使用できることをテストします。

PAM 構成のリファレンス

このセクションでは、PAM に関する追加の詳細 (PAM スタックを含む) を提供します。

PAM 構成ファイル

PAM フレームワークを使用するシステムアプリケーション (login や ssh など) は、/etc/pam.d ディレクトリ内の PAM 構成ファイルで構成されます。また、/etc/pam.conf ファイルも使用できます。これらのファイルへの変更は、システム上のすべてのユーザーに影響を与えます。

さらに、/etc/security/pam_policy ディレクトリにも PAM 構成ファイルが保持されています。これらのファイルは複数のサービスを対象とし、またユーザーごとの割り当て用に設計されています。このディレクトリ内のファイルを変更してはいけません。

- /etc/pam.d **ディレクトリ** – ワイルドカードファイル other を含む、サービス固有の PAM 構成ファイルが含まれています。アプリケーション用のサービスを追加するには、そのアプリケーションによって使用されているサービス名である 1 つの *service-name* ファイルを追加します。必要に応じて、アプリケーションは、other ファイル内の PAM スタックを使用できます。

  /etc/pam.d ディレクトリ内のサービスファイルは、ほとんどの PAM 実装でのデフォルト構成を提供します。pkg(5) のマニュアルページで説明しているように、これらのファイルは IPS メカニズムを使用して自己アセンブルされます。このデフォルト設定により、ほかのクロスプラットフォームの PAM アプリケーションとの相互運用性が簡略化されます。詳細は、pam.conf(4) のマニュアルページを参照してください。
- /etc/pam.conf **ファイル** – レガシー PAM 構成およびポリシーファイル。このファイルは、空で配信されます。PAM を構成するための推奨されるメカニズムは、/etc/pam.d ディレクトリ内のファイルの使用です。詳細は、pam.conf(4) のマニュアルページを参照してください。
- /etc/security/pam_policy **ディレクトリ** – 複数のサービスのためのポリシーを含む PAM ポリシーファイルが含まれています。これらのファイルは、必要に応じ

て、個人、個人のグループ、またはすべてのユーザーに割り当てることができます。このような割り当ては、pam.conf または /etc/pam.d ディレクトリ内のシステム PAM 構成ファイルよりも優先されます。これらのファイルを変更しないでください。ユーザーごとのファイルを追加するには、25 ページの「サイト固有の PAM 構成ファイルを作成する方法」を参照してください。ユーザーごとのファイルについては、pam_user_policy(5) のマニュアルページを参照してください。

セキュリティー管理者は、すべての PAM 構成ファイルを管理します。エントリの順序が正しくない、つまり PAM スタックが正しくないと、予期しない副作用が発生する場合があります。たとえば、不適切に構成されたファイルによってユーザーがロックアウトされ、修復のためにシングルユーザーモードが必要になることがあります。詳細は、36 ページの「PAM スタック」および33 ページの「PAM 構成のエラーをトラブルシューティングする方法」を参照してください。

PAM 構成の検索順序

アプリケーションが PAM フレームワークを呼び出すと、構成された PAM サービスが次の順序で検索されます。

1. サービス名が /etc/pam.conf ファイル内で検索されます。
2. 特定のサービスが /etc/pam.d/*service-name* ファイル内で使用されています。
3. サービス名 other が /etc/pam.conf ファイル内でチェックされます。
4. /etc/pam.d/other ファイルが使用されます。

この順序によって、既存の /etc/pam.conf ファイルが /etc/pam.d ディレクトリ内のサービスごとの PAM 構成ファイルと連携できるようになります。

PAM 構成ファイルの構文

pam.conf ファイルと PAM ユーザーごとのファイルは、pam.d ディレクトリ内のサービス固有のファイルとは異なる構文を使用します。

- /etc/pam.conf ファイルと /etc/security/pam_policy ファイル内のエントリの形式は、次の 2 つのうちのどちらかです。

  service-name module-type control-flag module-path module-options

  service-name module-type include *path-to-included-PAM-configuration*

- /etc/pam.d ディレクトリ内の *service-name* ファイル内のエントリでは、サービス名が省略されます。各ファイルの名前がサービス名を指定します。

  module-type control-flag module-path module-options

  module-type include *path-to-included-PAM-configuration*

PAM 構成ファイルの構文の項目は次のとおりです。

service-name

サービスの大文字と小文字が区別されない名前 (`login` や `ssh` など)。アプリケーションは、自身が提供するサービスごとに異なるサービス名を使用できます。たとえば、`sshd`(1M) のマニュアルページで `PAM` という単語を検索して、`sshd` デーモンが提供するさまざまなサービスのサービス名を探します。

事前に定義されたサービス名「`other`」は、特定のサービス構成が提供されていない場合のデフォルトのサービス名です。

module-type

サービスのタイプ、つまり `auth`、`account`、`session`、または `password` を示します。

control-flag

サービスの成功または失敗の値の決定におけるモジュールの役割を示します。有効な制御フラグは、36 ページの「PAM スタック」で説明されています。

module-path

そのモジュールタイプを実装するモジュールへのパス。パス名が絶対パスでない場合、そのパス名は、`/usr/lib/security/$ISA/` の相対パスであると見なされます。`$ISA` マクロ (トークン) は、PAM フレームワークに対して、モジュールパスのアーキテクチャー固有のディレクトリ内を探すように指示します。

module-options

サービスモジュールに渡すことができる、`nowarn` や `debug` などのオプション。モジュールのマニュアルページでは、そのモジュールのオプションが説明されています。

path-to-included-PAM-configuration

`/usr/lib/security` ディレクトリを基準にした、PAM 構成ファイルまたはファイル名へのフルパスを指定します。

PAM スタック

アプリケーションが次のいずれかの関数を呼び出すと、PAM フレームワークは PAM 構成ファイルを読み取って、どのモジュールがこのアプリケーションの PAM サービス名を実装するかを決定します。

- `pam_authenticate`(3PAM)
- `pam_acct_mgmt`(3PAM)
- `pam_setcred`(3PAM)

- pam_open_session(3PAM)
- pam_close_session(3PAM)
- pam_chauthtok(3PAM)

構成ファイルに 1 つのモジュールしか含まれていない場合は、そのモジュールの結果によってこの操作の結果が決定されます。たとえば、passwd アプリケーションのデフォルトの認証操作には、/etc/pam.d/passwd ファイル内に 1 つのモジュール pam_passwd_auth.so.1 が含まれています。

```
auth required           pam_passwd_auth.so.1
```

これに対して、サービスが複数のモジュールで実装されている場合は、これらのモジュールをスタックと呼びます。つまり、そのサービス名に対して PAM スタックが存在します。たとえば、サンプルの /etc/pam.d/login サービス内のエントリを考えてみます。

```
auth definitive         pam_user_policy.so.1
auth requisite          pam_authtok_get.so.1
auth required           pam_unix_auth.so.1
auth required           pam_dhkeys.so.1
auth required           pam_unix_cred.so.1
auth required           pam_dial_auth.so.1
```

これらのエントリは、login サービス名の auth スタックを作成します。このスタックの結果を決定するには、個々のモジュールの結果コードに対して「統合プロセス」を実行する必要があります。

統合プロセスでは、各モジュールがファイル内の順序に従って実行されます。それぞれの成功または失敗コードが、モジュールの制御フラグに従って、全体的な結果に統合されます。制御フラグによっては、スタックの早期終了が発生する可能性があります。たとえば、requisite または definitive モジュールの失敗によってスタックが終了します。その前に失敗がない場合は、sufficient、definitive、または binding モジュールの成功によってもスタックが終了します。スタックが処理されたあと、個々の結果が、アプリケーションに配信される 1 つの全体的な結果に結合されます。フローの図解については、図2および図3を参照してください。

制御フラグは、成功または失敗の決定において PAM モジュールが果たす役割を示します。制御フラグとその効果は、次のとおりです。

- **binding** – binding モジュールの要件を満たすことに成功すると、以前の失敗が記録されていない場合は、成功がただちにアプリケーションに返されます。これらの条件が満たされると、後続のモジュールは実行されません。

  失敗すると、required 失敗が記録され、モジュールの処理が継続されます。
- **definitive** – definitive モジュールの要件を満たすことに成功すると、以前の失敗が記録されていない場合は、成功がただちにアプリケーションに返されます。

  以前の失敗が記録されている場合は、その失敗がただちにアプリケーションに返され、モジュールはそれ以上実行されません。失敗するとすぐにエラーが返され、後続のモジュールは実行されません。

- **include** – 別の PAM 構成ファイルから、PAM スタック内のこの時点で使用される行を追加します。このフラグは成功または失敗の動作を制御しません。新しいファイルが読み取られると、PAM の `include` スタックが 1 増やされます。新しいファイル内でのスタックチェックが完了すると、`include` スタックの値が 1 減らされます。ファイルの最後に達したときに、PAM の `include` スタックが `0` である場合は、スタックの処理が終了します。PAM の `include` スタックの最大数は `32` です。
- **optional** – optional モジュールの要件を満たすことに成功することは、サービスを使用するために必要ではありません。

  失敗すると、optional 失敗が記録されます。
- **required** – required モジュールの要件を満たすことに成功することは、スタックが成功するために必要です。スタックの最終的な成功が返されるのは、どの binding または required モジュールも失敗を報告しなかった場合だけです。

  失敗すると、このサービスの残りのモジュールの実行完了後に、エラーが返されます。
- **requisite** – requisite モジュールの要件を満たすことに成功することは、スタックが成功するために必要です。スタックがアプリケーションに成功を返せるようにするには、スタック内のすべての requisite モジュールが成功を返す必要があります。

  失敗するとすぐにエラーが返され、後続のモジュールは実行されません。
- **sufficient** – 以前のどの required の失敗も記録されていない場合は、sufficient モジュールが成功するとただちに成功が返され、モジュールはそれ以上実行されません。

  失敗すると、optional 失敗が記録されます。

次の 2 つの接続図は、統合プロセスで結果がどのように決定されるかを示しています。

- 最初の図は、制御フラグのタイプごとに成功または失敗がどのように記録されるかを示しています。これらの結果は、2 番目の図に示されています。
- 2 番目の図は、統合された値の決定方法を示しています。optional の失敗と required の失敗は失敗を返し、成功は成功を返します。アプリケーションは、これらのリターンコードを処理する方法を決定します。

図 2 PAM スタック: 制御フラグの効果

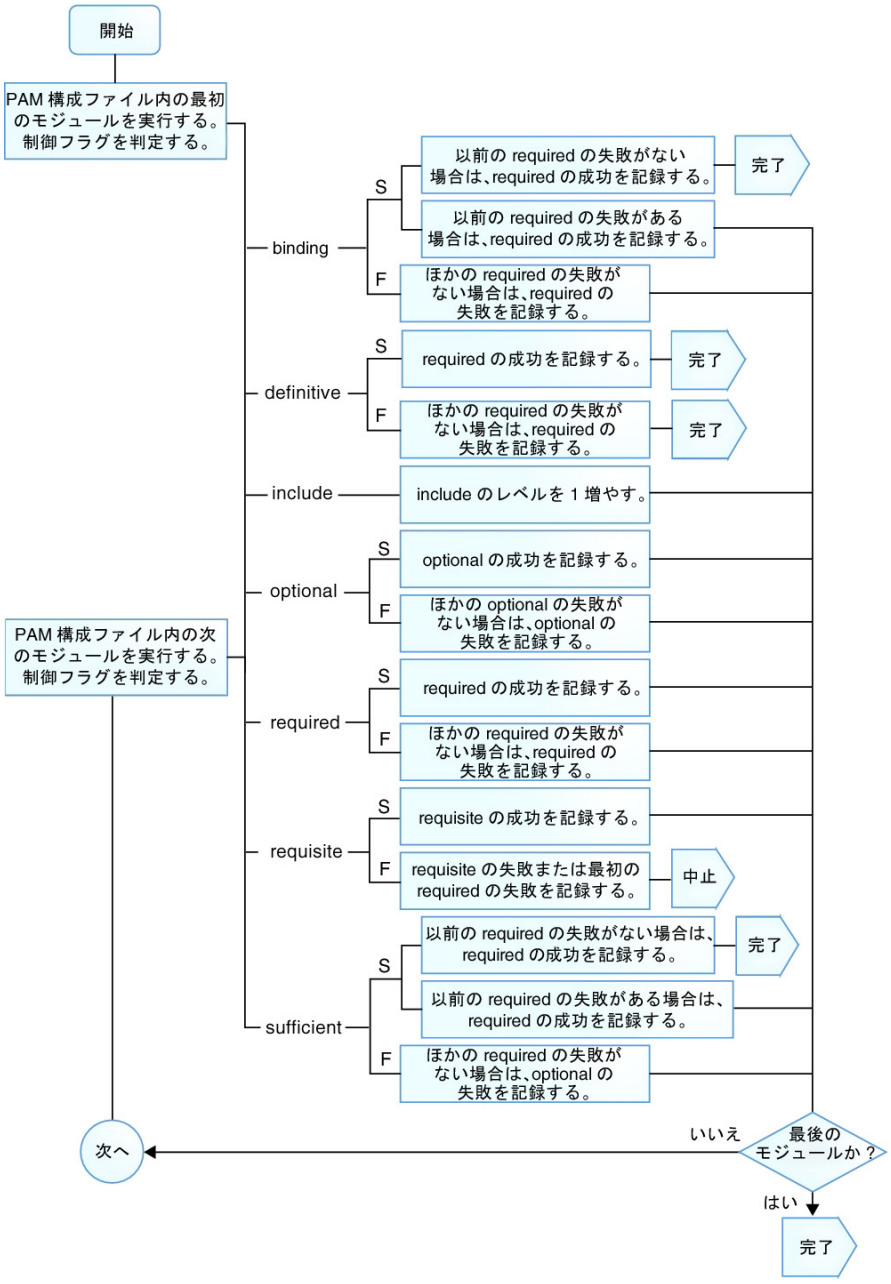
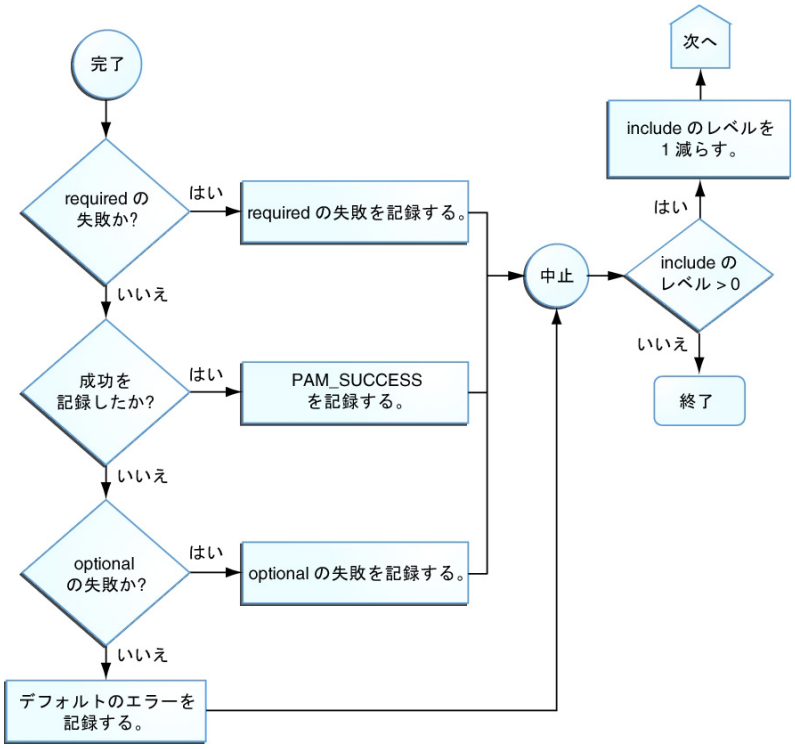


図 3 PAM スタック: 統合された値の決定方法



PAM スタックの例

次の例は、サンプルの `/etc/pam.d/other` ファイル内の認証管理のためのデフォルトの定義を示しています。これらの定義は、サービス固有の認証定義が構成されていない場合に認証に使用されます。

```
##
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
#
auth definitive         pam_user_policy.so.1
auth requisite          pam_authtok_get.so.1
auth required           pam_dhkeys.so.1
auth required           pam_unix_auth.so.1
auth required           pam_unix_cred.so.1
```

最初に、pam_user_policy.so モジュールを使用して、ユーザーの PAM ポリシーがチェックされます。この時点ではほかのどのモジュールもチェックされていないため、definitive 制御フラグは、構成された PAM スタックの評価が成功した場合は成功がアプリケーションに返されることを示します。構成された PAM スタックの評価が失敗した場合は、失敗コードがアプリケーションに返され、それ以上のチェックは実行されません。このユーザーにユーザーごとの PAM ポリシーが割り当てられていない場合は、次のモジュールが実行されます。

このユーザーにユーザーごとの PAM ポリシーが割り当てられていない場合は、pam_authtok_get モジュールが実行されます。このモジュールの制御フラグは、requisite に設定されています。pam_authtok_get が失敗した場合は、認証プロセスが終了し、失敗がアプリケーションに返されます。

pam_authtok_get が失敗しなかった場合は、次の 3 つのモジュールが実行されます。個々の失敗が返されたかどうかには関係なく統合プロセスが続行されるように、これらのモジュールは required 制御フラグで構成されています。pam_unix_cred が実行されたあと、残っているモジュールはありません。この時点で、すべてのモジュールが成功した場合は、成功がアプリケーションに返されます。pam_dhkeys、pam_unix_auth、pam_unix_cred のいずれかが失敗を返している場合は、失敗がアプリケーションに返されます。

PAM サービスモジュール

このセクションでは、選択された PAM サービスモジュールを一覧表示します。モジュールはそれぞれのマニュアルページで一覧表示され、そのあとにいつ、どこで使用されるかに関する簡単な説明を示します。詳細は、マニュアルページを参照してください。

Oracle Solaris が提供するすべての PAM サービスモジュールのリストについては、各マニュアルページのセクション 5 を参照してください。新しいモジュールが定期的に追加されます。たとえば、このリリースでは、Windows システムでの認証のためのモジュールがいくつか追加されています。また、各サイトでサードパーティーの PAM モジュールを追加することもあります。

pam_allow(5)　すべての呼び出しに対して PAM_SUCCESS を返します。pam_deny(5) のマニュアルページも参照してください。

pam_authtok_check(5)　パスワード変更のためにパスワードトークンを検証します。

pam_authtok_get(5)　PAM スタックにパスワード入力を求める機能を提供します。

pam_authtok_store(5)　PAM_USER のパスワードトークンを更新します。

pam_deny(5)	すべての呼び出しに対してモジュールタイプのデフォルトの失敗リターンコードを返します。pam_allow(5) のマニュアルページも参照してください。
pam_dhkeys(5)	Secure RPC 認証と Secure RPC 認証トークン管理の 2 つの PAM サービスに機能を提供します。
pam_krb5(5)	Kerberos ユーザーの識別情報を確認したり、Kerberos 資格キャッシュを管理したりするための機能を提供します。
pam_krb5_migrate(5)	PAM_USER をクライアントのローカル Kerberos レルムに移行するのに役立ちます。
pam_ldap(5)	構成された LDAP ディレクトリサーバーによる PAM 認証およびアカウント管理スタックの機能を提供します。
pam_list(5)	このホスト上のユーザーのアカウントを検証するための機能を提供します。この検証は、ホスト上のユーザーとネットグループのリストに基づいて行われます。
pam_passwd_auth(5)	パスワードスタックに認証機能を提供します。
pam_pkcs11(5)	ユーザーが PKCS#11 トークン内に格納されている X.509 証明書とその専用の非公開鍵を使用してシステムにログインできるようにします。
pam_roles(5)	ユーザーがある役割になることを承認されていることを検証し、役割による直接ログインを防止します。
pam_smb_passwd(5)	ローカルの Oracle Solaris ユーザーのための SMB パスワードの変更または追加をサポートします。smb(4) のマニュアルページも参照してください。
pam_smbfs_login(5)	Oracle Solaris クライアントとその CIFS/SMB サーバーの間でパスワードを同期します。
pam_tsol_account(5)	ラベルに関連した Trusted Extensions のアカウント制限を検証します。
pam_tty_tickets(5)	以前の正常な認証で作成されたチケットをチェックするためのメカニズムを提供します。
pam_unix_account(5)	ユーザーのアカウントがロックされたり、期限切れになったりしていないこと、およびユーザーのパスワードを変更する必要がないことを検証するための機能を提供します。

access_times と access_tz のチェックを含みます。

pam_unix_auth(5) パスワードが PAM_USER の正しいパスワードであることを検証するための機能を提供します。

pam_unix_cred(5) ユーザー資格情報を確立する機能を提供します。認証機能を資格機能とは独立に置き換えることができるようにします。

pam_unix_session(5) セッションを開いて閉じます。また、/var/adm/lastlog ファイルの更新も行います。

pam_user_policy(5) ユーザー固有の PAM 構成を呼び出します。

pam_zfs_key(5) ユーザーの暗号化されたホームディレクトリの ZFS 暗号化パスフレーズをロードして変更するための機能を提供します。

♦ ♦ ♦ 第 2 章

2 Kerberos サービスについて

この章では、Kerberos サービスについて説明します。この章では、次の内容について説明します。

- 45 ページの「Kerberos サービスとは」
- 46 ページの「Kerberos サービスの動作」
- 51 ページの「Kerberos のコンポーネント」
- 58 ページの「FIPS 140-2 アルゴリズムと Kerberos 暗号化タイプ」
- 58 ページの「Kerberos 資格によってサービスへのアクセスが提供されるしくみ」
- 62 ページの「Oracle Solaris Kerberos と MIT Kerberos の大きな違い」

Kerberos サービスとは

Kerberos サービスは、ネットワーク経由のセキュアなトランザクションを提供するクライアントサーバーアーキテクチャーです。Kerberos サービスでは、強力なユーザー認証とともに、整合性とプライバシを提供します。「認証」により、ネットワークトランザクションの送信者と受信者の識別情報が正しいことが保証されます。このサービスはまた、送受信されているデータの有効性を検証したり (整合性)、転送中のデータを暗号化したりすることもできます (プライバシ)。Kerberos サービスを使用して、他のシステムにログインしてコマンドを実行したり、データを交換したりファイルを安全に転送したりできます。さらに、このサービスでは、管理者がサービスやシステムへのアクセスを制限できる承認サービスも提供されます。また、Kerberos ユーザーは、自分のアカウントに他人がアクセスするのを制限できます。

Kerberos サービスはシングルサインオンシステムです。つまり、このサービスから、セッションあたり 1 回しか認証を受ける必要がありません。そのセッション中の以降のトランザクションはすべて、自動的にセキュリティー保護されます。このサービスから認証を受けたあと、`ftp` や `ssh` などの Kerberos に基づくコマンド、または NFS ファイルシステム上のデータにアクセスするためのコマンドを使用するたびに認証を受ける必要はありません。つまり、これらのサービスを使用するたびに、ネットワークを介してパスワードを送り、傍受される危険を冒す必要がありません。

Oracle Solaris の Kerberos サービスは、マサチューセッツ工科大学 (MIT) で開発された Kerberos V5 ネットワーク認証プロトコルに基づいています。そのため、Kerberos V5 製品を使用したことがあれば、Oracle Solaris バージョンにもすぐに慣れるはずです。Kerberos V5 プロトコルはネットワークセキュリティーの事実上の業界標準であるため、Oracle Solaris バージョンによって、異機種混在ネットワーク上でセキュアなトランザクションが可能になります。さらに Kerberos サービスでは、複数のドメイン間でも単一のドメイン内でも認証やセキュリティーの機能を使用できます。

Kerberos サービスは、Oracle Solaris アプリケーションを実行するための柔軟性を備えています。このサービスは、NFS サービスや `ftp` などのネットワークサービスに対する Kerberos に基づくリクエストと Kerberos に基づかないリクエストの両方を有効にするように構成できます。このため、Kerberos サービスが有効になっていないシステムで動作するアプリケーションも正しく動作します。もちろん、Kerberos に基づくネットワークリクエストのみを有効にするように Kerberos サービスを構成することもできます。

Kerberos サービスのセキュリティーメカニズムにより、Generic Security Service Application Programming Interface (GSS-API) を使用するアプリケーションの使用時に、認証、整合性、およびプライバシのために Kerberos を使用できます。ただし、ほかのセキュリティーメカニズムが開発されている場合には、アプリケーションで使用されるセキュリティーメカニズムを Kerberos サービスに限定しておく必要はありません。このサービスは GSS-API にモジュールとして統合されるように設計されているため、GSS-API を使用するアプリケーションは、そのニーズにもっとも適したセキュリティーメカニズムを選択できます。

Kerberos サービスの動作

このセクションでは、Kerberos 認証プロセスの概要について説明します。詳細は、58 ページの「Kerberos 資格によってサービスへのアクセスが提供されるしくみ」を参照してください。

Kerberos セッションが起動されたあとは、ユーザーから見ると Kerberos サービスが意識されることはほとんどありません。`ssh` や `ftp` などのコマンドは、ほぼ同様に動作します。Kerberos セッションの初期化には通常、ログインと Kerberos パスワードの入力しか必要ありません。

Kerberos システムは、チケットの概念を中心に動作します。チケットは、ユーザー、および NFS サービスなどのサービスを特定する一連の電子情報です。運転免許証が運転する人と免許の種類を表すのと同じように、チケットもユーザーとユーザーのネットワークアクセス権を表します。Kerberos に基づくトランザクションを実行すると (NFS マウントしたファイルをリクエストする場合など)、チケットに対するリクエストが鍵配布センター (KDC) に透過的に送信されます。KDC は、データベースにア

クセスしてユーザーの識別情報を認証し、NFS サーバーにアクセスするためのアクセス権を許可するチケットを返します。「透過的に」とは、チケットを明示的にリクエストする必要がないことを示します。このリクエストは、サーバーにアクセスしようとしたときに発生します。特定のサービスのチケットを取得できるのは認証されたクライアントだけであるため、別のクライアントが、引き継がれた識別情報で NFS サーバーにアクセスすることはできません。

チケットには、特定の属性が関連付けられています。たとえば、チケットには、新しい認証処理を行わなくても別のシステムで使用できる転送可能の属性があります。また、指定の日付まで有効にならない「遅延」の属性もあります。チケットをどのように使用できるかは、ポリシーによって設定されます (どのユーザーが、どのタイプのチケットの取得を許可されるかを指定する場合など)。ポリシーは、Kerberos サービスのインストールや管理の際に決定します。

注記 - 資格とチケットという用語は、頻繁に使用されます。広い意味の Kerberos では、これらの用語は同じ意味で使われることがありますが、技術的には資格は、チケットとそのセッションに対する「セッション鍵」からなります。この違いは、58 ページの「Kerberos 資格によってサービスへのアクセスが提供されるしくみ」でさらに詳細に説明されています。

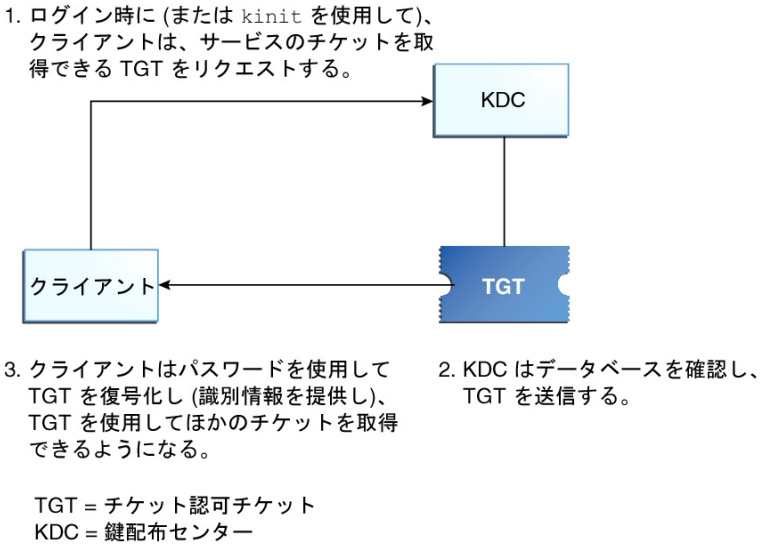
次のセクションでは、Kerberos 認証プロセスについて詳細に説明します。

初期認証: チケット認可チケット (TGT)

Kerberos 認証には、すべての後続の認証を有効にする初期認証と、後続の認証自体の 2 つのフェーズがあります。

次の図では、初期認証の手順を示します。

図 4 Kerberos セッションの初期認証



1. クライアント (ユーザー、または NFS などのサービス) は、KDC に TGT を要求して Kerberos セッションを開始します。ほとんどの場合、この要求はログイン時に自動的に実行されます。

   TGT は、ほかの特定のサービスのチケットを取得するために必要です。TGT は、パスポートに似ています。パスポートと同様に、チケット認可チケットはユーザーを識別し、ユーザーが多数の「ビザ」(チケット) を取得できるようにします。これらのチケットは、外国へのアクセスを許可する代わりに、ユーザーがリモートシステムやネットワークサービスにアクセスできるようにします。パスポートやビザと同様に、TGT などのチケットには有効期限があります。ただし、Kerberos コマンドは、ユーザーがパスポートを所有していることを通知し、ユーザーに代わってビザを取得します。ユーザー自身がトランザクションを実行する必要はありません。

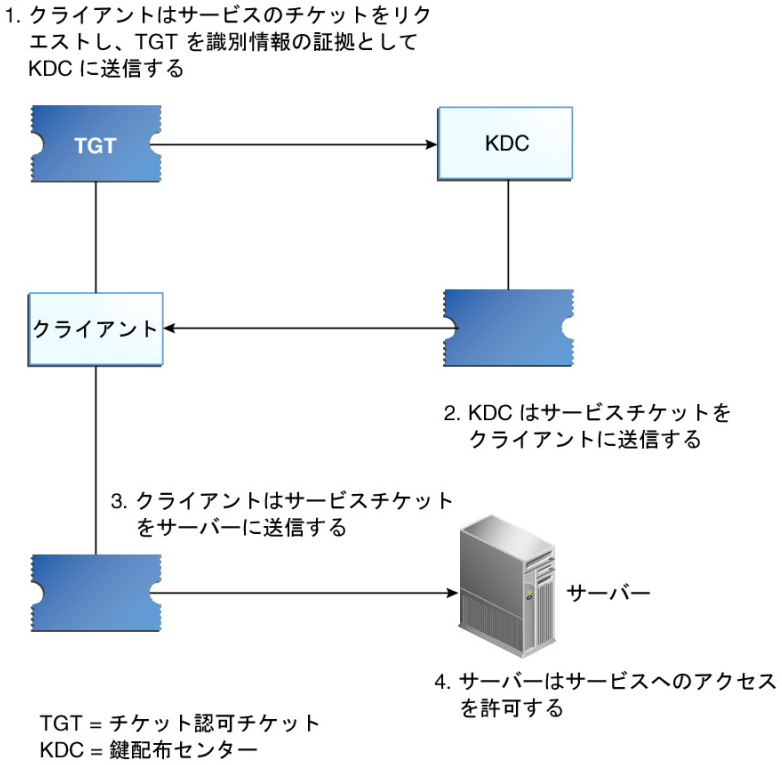
   チケット認可チケットに類似した例として、4 つのスキー場で使える 3 日間のスキーパスを挙げます。ユーザーは、そのパスの期限が切れていないかぎり、行くと決めたどのリゾート地でもこのパスを見せ、そのリゾート地のリフトチケットを受け取ります。リフトチケットを入手したら、そのスキー場で好きなだけスキーをすることができます。次の日に別のリゾート地に行った場合は、またこのパスを見せ、その新しいリゾート地のリフトチケットを追加で受け取ります。違いは、Kerberos に基づくコマンドの場合は週末のスキーパスがあることに気付き、ユーザーに代わってリフトチケットを入手するため、ユーザーがこれらのトランザクションを自分で実行する必要がない点です。

2. KDC は、チケット認可チケットを作成し、それを暗号化された形式でクライアントに戻します。クライアントは、自身のパスワードを使用して TGT を復号化します。
3. これで有効なチケット認可チケットを取得できたため、クライアントはそのチケット認可チケットが有効であるかぎり、すべての種類のネットワーク操作 (`nfs` や `ssh` など) のチケットをリクエストできます。この TGT の有効期限は通常、数時間です。クライアントは一意のネットワーク操作を実行するたびに、TGT は KDC にその操作のチケットを要求します。

後続の Kerberos 認証

クライアントが初期認証を受け取ると、後続の認証はそれぞれ次の図のように実行されます。

図 5 Kerberos 認証を使用してサービスへのアクセスを取得する



1. クライアントは、チケット認可チケットを識別情報の証拠として KDC に送信することによって、特定のサービスのチケットを KDC にリクエストします (別のシステムにリモートでログインする場合など)。
2. KDC は、そのサービスのチケットをクライアントに送信します。

   ユーザー `jdoe` が、`krb5` 認証が必要な状態で共有されてきた NFS ファイルシステムにアクセスするとします。`jdoe` はすでに認証されている (つまり、`jdoe` はすでにチケット認可チケットを持っている) ため、`jdoe` がファイルにアクセスしようとすると、NFS クライアントシステムは自動的かつ透過的に NFS サービスの KDC からチケットを取得します。別の Kerberos サービスを使用するには、手順 1 の場合のように、`jdoe` は別のチケットを取得します。
3. クライアントはサーバーにチケットを送信します。

   NFS サービスを使用している場合、NFS クライアントは自動的および透過的に NFS サービスのチケットを NFS サーバーに送信します。
4. サーバーはクライアントにアクセス権を許可します。

これらの手順ではサーバーが KDC とは通信しないように見えますが、最初のクライアントと同様に、サーバーも KDC に自身を登録します。簡単にするために、そのセクションは省略しています。

バッチジョブの Kerberos 認証

バッチジョブ (cron、at、batch など) は、遅延実行プロセスです。Kerberos 環境では、遅延実行プロセスを含むすべてのプロセスに資格が必要です。ただし、ユーザーの資格の有効期間は比較的短時間です。デフォルトでは、ユーザー資格は 8 時間だけ有効であり、1 週間まで更新できます。これらの時間は、機密の鍵が悪意のあるユーザーに公開される事態を制限するように設計されていますが、任意の時点でジョブを実行することができない場合があります。

Oracle Solaris では、Kerberos サービスにアクセスするバッチジョブは、ユーザーの有効期間の長い鍵を公開することなく実行できます。この解決方法では、Kerberos サービス、ユーザー名、およびクライアントホスト名を含む資格を、セッションごとのユーザー資格キャッシュ内に格納します。このバッチジョブを認証するために、PAM モジュールが使用されます。ホストがどのサービスに関してチケットを取得できるかについての情報は、LDAP ディレクトリサーバー内に集中して格納できます。

詳細は、`pam_krb5_keytab(5)` および `pam_gss_s4u(5)` のマニュアルページのほか、138 ページの「Kerberos サービスへのアクセスのための遅延実行の構成」を参照してください。

Kerberos、DNS、およびネームサービス

Kerberos サービスは、ホスト名の解決に DNS を使用するようにコンパイルされています。ホスト名を解決するとき、`nsswitch` サービスはまったくチェックされません。

Kerberos のコンポーネント

Kerberos には、レルム、ネットワークプログラム、主体、サーバーなどのコンポーネントが含まれています。コマンドとモジュールのリストについては、54 ページの「Kerberos ユーティリティー」を参照してください。

Kerberos ネットワークプログラム

注記 - この Oracle Solaris リリースでは、`ssh` を除くすべてのリモートログインコマンドが非推奨になっています。古いシステムに接続するために次のいずれかの非推奨のコマンドを使用する場合、使用することは可能です。レガシースクリプトで非推奨のコマンドが使用されているためにそのコマンドをこのシステム上で使用する場合は、`svcadm enable login:rlogin` のように、そのコマンドの SMF サービスを有効にする必要があります。あるいは、`ssh` コマンドを使用するようにスクリプトを変更することもできます。同様に、古いシステムから非推奨のコマンドを使用してこのシステムに接続するには、このシステム上でそのコマンドのサービスを有効にする必要があります。

`telnet` などの非推奨のコマンドでは、弱い暗号化鍵が必要になる場合があります。これらの鍵は、`krb5.conf` ファイル内でデフォルトでは許可されません。詳細は、66 ページの「Kerberos でサポートされる暗号化タイプ」および `krb5.conf(4)` のマニュアルページを参照してください。

詳細は、『Oracle Solaris 11.3 でのシステムおよび接続されたデバイスのセキュリティー保護』 の 「システムリソースへのアクセス制御」を参照してください。182 ページの「Kerberos のユーザーコマンド」で一覧表示されているマニュアルページも参照してください。

ユーザーが使用できる Kerberos に基づく (「Kerberos 化された」) コマンドは次のとおりです。

- `ftp`
- `rcp`、`rlogin`、`rsh`
- `ssh`、`scp`、`sftp`

- `telnet`

これらのアプリケーションは、同じ名前の Oracle Solaris アプリケーションと同じです。ただし、トランザクションを認証するときに Kerberos 主体を使用できるようにアプリケーションを拡張することにより、Kerberos に基づくセキュリティーを提供します。主体については、52 ページの「Kerberos 主体」を参照してください。

これらのコマンドについては、182 ページの「Kerberos のユーザーコマンド」で詳しく説明します。

Kerberos 主体

Kerberos サービス内のクライアントは、その「主体」で識別されます。主体は、KDC がチケットを割り当てることができる一意の ID です。主体には、`jdoe` などのユーザーや、`nfs` などのサービスがあります。

慣例上、主体名はプライマリ、インスタンス、レルムの 3 つのコンポーネントに分割されます。標準的な Kerberos 主体は、たとえば `jdoe/admin@CORP.EXAMPLE.COM` のようになります。この例では、次のようになります。

- `jdoe` はプライマリです。プライマリには、この例のようなユーザー名や `nfs` などのサービスを指定します。また、プライマリが `host` という単語である場合もあります。これは、この主体が `ftp`、`scp`、`ssh` などのさまざまなネットワークサービスを提供するように設定されているサービス主体であることを示します。
- `admin` はインスタンスです。インスタンスは、ユーザー主体の場合はオプションですが、サービス主体では必須です。たとえば、ユーザー `jdoe` がシステム管理者の役割を果たす場合もあるときは、主体 `jdoe/admin` によってユーザーと管理者を区別できます。同様に、`jdoe` が 2 つの異なるホスト上にアカウントを持っている場合、これらのアカウントでは、インスタンスが異なる 2 つの主体名 (`jdoe/denver.example.com` と `jdoe/boston.example.com` など) を使用できます。Kerberos サービスでは `jdoe` と `jdoe/admin` が 2 つの完全に異なる主体として処理されることに注意してください。

  サービス主体では、インスタンスは完全指定されたホスト名です。`bighop.corp.example.com` はこのようなインスタンスの例です。この例のプライマリとインスタンスは、`ftp/bighop.corp.example.com` または `host/bighop.corp.example.com` になる可能性があります。
- `CORP.EXAMPLE.COM` は Kerberos レルムです。レルムについては、53 ページの「Kerberos レルム」を参照してください。

次に有効な主体名を示します。

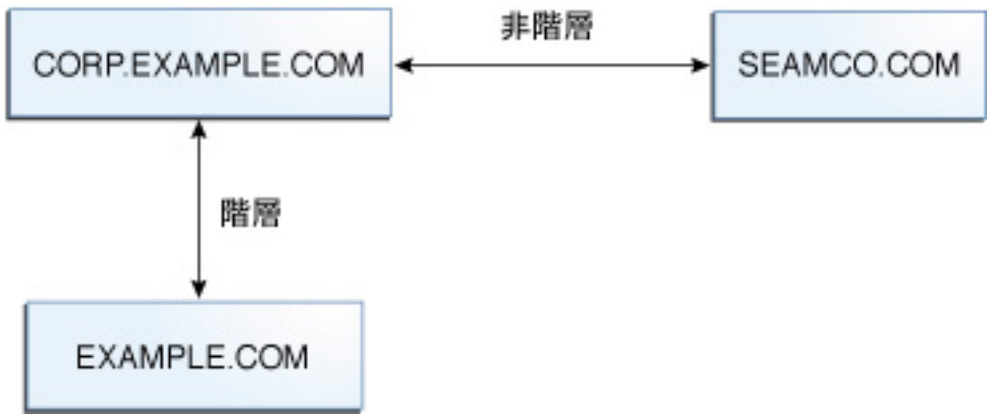
- `jdoe`
- `jdoe/admin`
- `jdoe/admin@CORP.EXAMPLE.COM`

- `nfs/host.corp.example.com@CORP.EXAMPLE.COM`
- `host/corp.example.com@CORP.EXAMPLE.COM`

Kerberos レルム

「レルム」とはドメインのようなもので、同じ「マスター KDC」の下にあるシステムをグループとして定義する論理ネットワークです。図6は、各レルムが相互にどのように関連するかを示しています。階層構造のレルムでは、1 つのレルムがほかのレルムの上位集合になります。階層ではない (直接接続の) レルムでは、2 つのレルム間のマッピングを定義する必要があります。Kerberos のレルム間認証を使用すると、レルムにまたがる認証が可能になります。その場合、各レルムの KDC に、他のレルムの主体エントリだけが必要になります。

図 6 Kerberos レルム



Kerberos サーバー

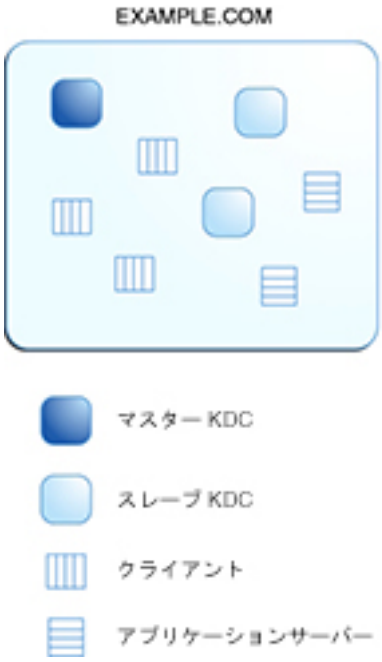
各レルムには、主体データベースのマスターコピーを保守するサーバーが含まれる必要があります。このサーバーを「マスター KDC サーバー」と呼びます。さらに、各レルムにはセカンダリサーバーを含めるようにしてください。セカンダリサーバーは、マスター KDC からリフレッシュまたは伝播される主体データベースの複製コピーを含むスレーブ *KDC* サーバーにすることができます。セカンダリサーバーは、マルチマスター構成を形成するもう 1 つのマスター KDC にすることもできま

す。マスター KDC サーバーとスレーブ KDC サーバーはどちらも、認証の確立に使用されるチケットを作成します。

レルムにはまた、Kerberos アプリケーションサーバー も含めることができます。このサーバーは、`ftp`、`ssh`、NFS などの Kerberos サービスへのアクセスを提供します。

次の図は、仮想レルムに含まれる可能性のあるものを示しています。

図 7 一般的な Kerberos レルム



Kerberos ユーティリティー

MIT が配布する Kerberos V5 製品と同様に、Oracle Solaris リリースの Kerberos サービスには次が含まれています。

- 鍵配布センター (KDC):
  - Kerberos データベース管理デーモン – `kadmind`。
  - Kerberos チケット処理デーモン – `krb5kdc`。

    - データベース管理プログラム – `kadmin` (マスターのみ)、`kadmin.local`、および `kdb5_util`。
    - データベース伝播ソフトウェア – `kprop` (スレーブのみ) および `kpropd`。
- 資格を管理するためのユーザープログラム – `kinit`、`klist`、および `kdestroy`。
- Kerberos パスワードを変更するためのユーザープログラム – `kpasswd`。
- ネットワークアプリケーション – `ftp`、`rcp`、`rlogin`、`rsh`、`scp`、`sftp`、`ssh`、および `telnet`。
- リモートアプリケーションデーモン – `ftpd`、`rlogind`、`rshd`、`sshd`、および `telnetd`。
- キータブ管理ユーティリティー – `ktutil`。
- Generic Security Service Application Programming Interface (GSS-API) – 新しいメカニズムが追加されるたびにアプリケーションを再コンパイルしなくても、アプリケーションが複数のセキュリティーメカニズムを使用できるようにします。GSS-API では、アプリケーションを多くのオペレーティングシステムに移植可能にする標準インタフェースが使用されます。GSS-API はアプリケーションに、認証だけでなく、整合性およびプライバシセキュリティーサービスを組み込む機能を提供します。`ftp` と `ssh` は、どちらも GSS-API を使用しています
- RPCSEC_GSS Application Programming Interface (API) – NFS サービスが Kerberos 認証を使用できるようにします。RPCSEC_GSS API は、使用されているメカニズムには依存しないセキュリティーサービスを提供します。RPCSEC_GSS は、GSS-API 層の最上位に位置しています。GSS_API ベースのセキュリティーメカニズムは、プラグイン可能なので、RPCSEC_GSS を使用するアプリケーションで使用できます。

さらに、Oracle Solaris の Kerberos サービスには、次のものが含まれています。

- PAM 用の Kerberos V5 サービスモジュール – Kerberos サービスのための認証、アカウント管理、セッション管理、およびパスワード管理を提供します。これらのモジュールにより、Kerberos 認証はユーザーに対して透過的に実行されます。
- Kerberos V5 ユーザーごとの PAM スタック – `/etc/security/pam_policy` ディレクトリ内にある、さまざまなシナリオのための PAM 構成ファイルを提供します。
- カーネルモジュール – NFS サービスで使用される、Kerberos サービスのカーネルベースの実装を提供します。これにより、パフォーマンスが大幅に向上します。

詳細は、第7章「Kerberos サービスのリファレンス」を参照してください。

Kerberos セキュリティーサービス

Kerberos サービスは、ユーザーの認証を行うほかに、次の 2 つのセキュリティーサービスを提供します。

- 「整合性」 – 認証が、あるネットワーク上のクライアントが本人であるかどうかを確認するのと同様に、整合性は、クライアントの送信データが有効で、伝送の間に

改ざんされていないことを確認します。整合性の確認は、データの暗号チェックサムによって行われます。整合性にはユーザー認証も含まれます。

- 「プライバシ」 – プライバシによって、セキュリティーがさらに向上します。プライバシは、伝送データの整合性を検証するだけでなく、伝送前にデータを暗号化して盗聴を防ぎます。プライバシにもユーザー認証が含まれます。

Kerberos 暗号化タイプ

暗号化タイプは、暗号処理が実行されるときに使用する暗号アルゴリズムとモードを特定します。サポートされている暗号化タイプのリストについては、`krb5.conf(4)` および `kdb5_util(1M)` のマニュアルページを参照してください。

クライアントが KDC にチケットを要求する場合、KDC はクライアントとサーバーで互換性のある暗号化タイプの鍵を使用する必要があります。Kerberos プロトコルでは、クライアントは KDC に、チケット応答のクライアントの部分に特定の暗号化タイプを使用するようリクエストできます。このプロトコルでは、サーバーが KDC に暗号化タイプを指定することは許可されません。

暗号化タイプを変更する前に、次の問題を考慮してください。

- KDC では、主体データベースのサーバー主体エントリに関連する最初の鍵/暗号化タイプはサーバーによってサポートされているものとします。
- KDC 上で、主体に対して生成される鍵が、その主体を認証するシステムと互換性があることを確認する必要があります。デフォルトで、`kadmin` コマンドは、サポートされるすべての暗号化タイプの鍵を生成します。主体を使用するシステムがこの暗号化タイプのデフォルトのセットをサポートしていない場合、主体の作成時に暗号化タイプを制限する必要があります。暗号化タイプを制限するために推奨される 2 つの方法として、`kadmin addprinc` で `-e` フラグを使用するか、または `kdc.conf` ファイル内の `supported_enctypes` パラメータをこのサブセットに設定します。`supported_enctypes` パラメータは、Kerberos レルム内のほとんどのシステムが暗号化タイプのデフォルトセットのサブセットをサポートしている場合に使用します。`supported_enctypes` を設定すると、`kadmin addprinc` が特定のレルムに対して主体を作成するときに使用する暗号化タイプのデフォルトセットが指定されます。
- システムがサポートする暗号化タイプを決定する場合は、システム上で実行されている Kerberos のバージョンと、サーバー主体が作成される対象のサーバーアプリケーションでサポートされている暗号化アルゴリズムの両方を考慮してください。たとえば、`nfs/hostname` サービス主体を作成する場合は、暗号化タイプをそのホスト上の NFS サーバーがサポートしているタイプに制限するようにしてください。
- `kdc.conf` ファイル内の `master_key_enctype` パラメータを使用すると、主体データベース内のエントリを暗号化するマスター鍵の暗号化タイプを制御できます。KDC 主体データベースが既に作成済みの場合は、このパラメータを使用しないで

ください。データベースの作成時に master_key_enctype パラメータを使用すると、マスター鍵のデフォルトの暗号化タイプを aes256-cts-hmac-sha1-96 から別の暗号化タイプに変更できます。スレーブ KDC を構成するときに、すべてのスレーブ KDC が選択された暗号化タイプをサポートしていること、およびそれぞれの kdc.conf ファイル内に同一の master_key_enctype エントリが存在することを確認してください。また、kdc.conf で supported_enctypes が設定されている場合は、master_key_enctype が supported_enctypes 内のいずれかの暗号化タイプに設定されていることも確認してください。これらの問題のいずれかが適切に処理されない場合、マスター KDC はスレーブ KDC と連携できない可能性があります。

- クライアント上では、krb5.conf ファイル内のパラメータを使用して、KDC に対する暗号化タイプのリクエストを制御できます。default_tkt_enctypes パラメータは、クライアントが KDC にチケット認可チケット (TGT) をリクエストするときに使用する暗号化タイプを指定します。TGT は、より効果的な方法でほかのサーバーチケットを取得するためにクライアントにより使用されます。default_tkt_enctypes を設定すると、クライアントが TGT を使用してサーバーチケットを要求する (TGS 要求と呼ばれる) ときに、クライアントと KDC 間の通信を保護するために使用される暗号化タイプを一部制御できます。default_tkt_enctypes で指定された暗号化タイプは、KDC 上に格納されている主体データベース内の主体鍵の暗号化タイプの少なくとも 1 つに一致している必要があることに注意してください。一致しない場合、TGT 要求は失敗します。default_tkt_enctypes は相互運用性の問題の原因になる場合があるため、ほとんどの状況ではこのパラメータを設定しないようにしてください。デフォルトでは、クライアントコードは、サポートされるすべての暗号化タイプと KDC に、KDC が主体データベースで検索した鍵に基づいて暗号化タイプを選択するようリクエストします。
- default_tgs_enctypes パラメータは、サーバーチケットを取得するために使用される TGS リクエストでクライアントがリクエストする暗号化タイプを制限します。このパラメータは、クライアントとサーバーが共有するセッション鍵を作成するときに KDC が使用する暗号化タイプも制限します。たとえば、セキュアな NFS の実行中にクライアントで 3DES 暗号化だけを使用する場合は、default_tgs_enctypes = des3-cbc-sha1 を設定するようにしてください。クライアント主体とサーバー主体の主体データベース内に des-3-cbc-sha1 鍵が存在することを確認してください。default_tkt_enctype と同様に、資格が KDC とサーバーの両方で正しく設定されていないと相互運用性の問題の原因になる場合があるため、ほとんどの状況ではこのパラメータを設定しないようにしてください。
- サーバー上では、kdc.conf ファイル内の permitted_enctypes パラメータを使用して、サーバーが受け入れる暗号化タイプを制御できます。さらに、サーバーが keytab エントリを作成するときに使用する暗号化タイプを指定できます。KDC は使用する鍵または暗号化タイプを決定するときにサーバーアプリケーションと通信しないため、これらの方法のいずれかを使用して暗号化タイプを制御することはできるだけ避け、代わりに、KDC に使用する暗号化タイプを決定させるようにしてください。

FIPS 140-2 アルゴリズムと Kerberos 暗号化タイプ

Oracle Solaris では、Kerberos を FIPS 140-2 モードで実行するように構成できます。FIPS 140-2 に準拠しないレガシーアプリケーションまたはシステムがレルムに含まれる場合、そのレルムは FIPS 140-2 モードで実行できません。

FIPS 140-2 モードで実行したときの Kerberos は、FIPS 140-2 プロバイダのコンシューマと呼ばれます。Oracle Solaris でのプロバイダは、暗号化フレームワークです。暗号化フレームワーク用の FIPS 140-2 検証済みの Kerberos 暗号化タイプは、`des3-cbc-sha1` だけです。これはデフォルトではありません。手順については、77 ページの「FIPS 140-2 モードで実行するように Kerberos を構成する方法」を参照してください。

注記 - FIPS 140-2 で検証された暗号化のみを使用するという厳格な要件がある場合は、Oracle Solaris 11.3 SRU 5.6 リリースを実行している必要があります。Oracle は、この特定のリリースでの暗号化フレームワークに対する FIPS 140-2 の検証を完了しました。現在の Oracle Solaris リリースは、この検証された基盤の上に構築されており、パフォーマンス、機能、および信頼性に対応するソフトウェアの機能強化を含んでいます。これらの機能強化を利用するために、可能な場合は常に Oracle Solaris 11.3 を FIPS 140-2 モードで構成するようにしてください。

Kerberos 資格によってサービスへのアクセスが提供されるしくみ

識別情報を証明するチケットとそれに対応するセッション鍵を提供できる場合は、リモートサービスによってアクセスが許可されます。セッション鍵には、ユーザーやアクセスするサービスに特有の情報が含まれています。ユーザーすべてのチケットとセッション鍵は、ユーザーが最初にログインするときに KDC によって作成されます。チケットとそれに対応するセッション鍵が 1 つの資格となります。複数のネットワークサービスを使用する場合には、ユーザーは多数の資格を収集できます。ユーザーは特定のサーバーで動作するサービスごとに 1 つの資格を必要とします。たとえば、`boston` という名前のサーバー上の `nfs` サービスにアクセスするには 1 つの資格が必要です。別のサーバー上の `nfs` サービスにアクセスするには、別の資格が必要です。

特定のサーバー上の特定のサービスにアクセスする場合、ユーザーは 2 つの資格を取得する必要があります。最初の資格は、チケット認可チケット (TGT) のためです。チケット認可サービスは、この資格の暗号を解除すると、ユーザーからアクセスを要求されているサーバーの資格をさらに作成します。ユーザーは、この 2 つめの資格を使

用してサーバー上のサービスへのアクセスを要求します。サーバーがこの資格の暗号を解除すると、ユーザーはアクセスを許可されます。

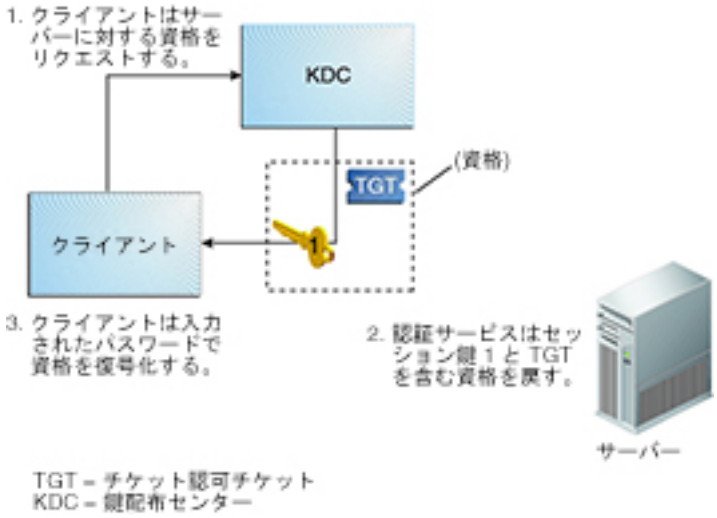
資格を作成して格納するプロセスは透過的です。資格は KDC によって作成され、要求者に送信されます。資格は、受信されると資格キャッシュに格納されます。

チケット認可サービスに対する資格の取得

1. 認証処理を開始するために、クライアントが特定のユーザー主体の要求を認証サーバーに送信します。この要求の送信では暗号は使用されません。このリクエストにはセキュアな情報が含まれていないため、暗号化の使用は必要ありません。
2. 認証サービスは要求を受信すると、ユーザーの主体名を KDC データベースから検索します。主体がデータベースのエントリに一致すると、認証サービスはその主体の非公開鍵を取得します。認証サービスは次に、クライアントおよびチケット認可サービスによって使用されるセッション鍵 (これをセッション鍵 1 と呼びます) と、チケット認可サービスのチケット (チケット 1) を生成します。このチケットを「チケット認可チケット (TGT)」ともいいます。セッション鍵とチケットはユーザーの非公開鍵を使って暗号化され、情報がクライアントに返送されます。
3. クライアントは、これらの情報を使用して、ユーザー主体の非公開鍵でセッション鍵 1 とチケット 1 を復号化します。非公開鍵はユーザーと KDC データベースにしか認識されていないはずであるため、このパケット内の情報は安全です。クライアントはこの情報を資格キャッシュに格納します。

この処理中に、ユーザーは通常、パスワードを要求されます。非公開鍵を作成するために使用された、KDC データベースに格納されているパスワードが、ユーザーが指定したパスワードと同じであると、認証サービスから送信された情報は正しく復号化されます。クライアントはこれで、チケット認可サービスで使用される資格を取得したため、サーバーに対する資格をリクエストする準備ができました。

図 8 チケット認可サービスに対する資格の取得

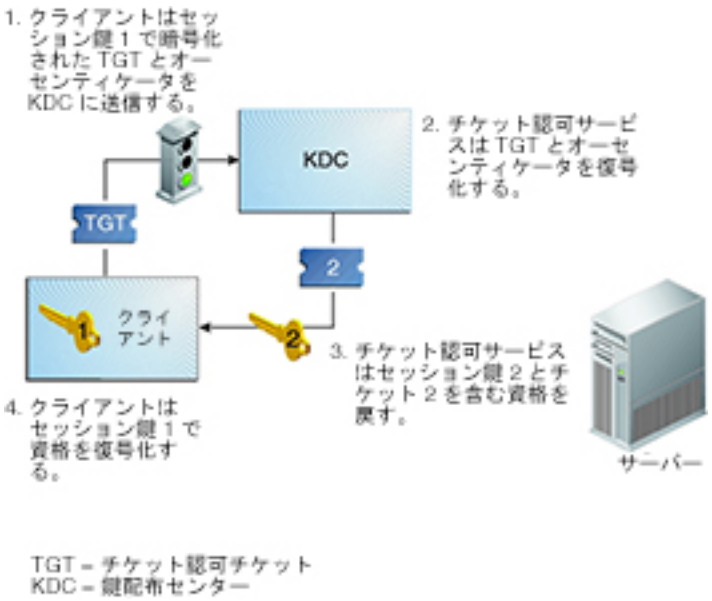


Kerberos サーバーに対する資格の取得

1. 特定のサーバーにアクセスするには、クライアントがその前にサーバーに対する資格を認証サービスから取得している必要があります。59 ページの「チケット認可サービスに対する資格の取得」を参照してください。次にクライアントは、チケット認可サービスに要求を送信します。この要求には、サービス主体名、チケット 1 およびセッション鍵 1 で暗号化されたオーセンティケータが含まれています。チケット 1 は、チケット認可サービスのサービス鍵を使用して認証サービスによって暗号化されたものです。
2. チケット認可サービスのサービス鍵はチケット認可サービスに認識されているため、チケット 1 を復号化できます。チケット 1 の情報にはセッション鍵 1 が含まれているため、チケット認可サービスはオーセンティケータの暗号を解除できます。この時点で、ユーザー主体はチケット認可サービスによって認証されます。
3. 認証に成功すると、チケット認可サービスは、ユーザー主体およびサーバーのためのセッション鍵 (セッション鍵 2) と、サーバーのためのチケット (チケット 2) を生成します。次にセッション鍵 2 とチケット 2 はセッション鍵 1 を使って暗号化されます。セッション鍵 1 を知っているのはクライアントとチケット認可サービスだけであるため、この情報は安全であり、ネットワークを介して安全に送信されます。

4. クライアントはこの情報パケットを受信すると、前に資格キャッシュに格納したセッション鍵 1 を使用して情報を復号化します。クライアントは、サーバーに対して使用する資格を取得したことになります。次にクライアントは、そのサーバーの特定のサービスにアクセスする要求を行います。

図 9 サーバーに対する資格の取得



特定の Kerberos サービスへのアクセスの取得

1. クライアントが特定のサービスへのアクセスを要求するには、まず認証サーバーからチケット認可サービスに対する資格を取得し、チケット認可サービスからサーバー資格を取得する必要があります。59 ページの「チケット認可サービスに対する資格の取得」および60 ページの「Kerberos サーバーに対する資格の取得」を参照してください。次に、クライアントは、チケット 2 と別のオーセンティケータを含む要求をサーバーに送信します。オーセンティケータはセッション鍵 2 を使用して暗号化されます。
2. チケット 2 は、サービスのサービス鍵を使用してチケット認可サービスによって暗号化されています。サービス鍵はサービス主体が知っているため、サービスはチケット 2 を復号化し、セッション鍵 2 を取得できます。次に、セッション鍵 2

を使用してオーセンティケータが復号化されます。オーセンティケータが正しく復号化されると、サービスへのアクセスがクライアントに許可されます。

図 10 特定のサービスへのアクセス権の取得

Oracle Solaris Kerberos と MIT Kerberos の大きな違い

Oracle Solaris に含まれているが、MIT Kerberos には含まれていない拡張機能は次のとおりです。

- Oracle Solaris リモートアプリケーションの Kerberos サポート
- KDC データベースの増分伝播
- クライアント構成スクリプト
- 地域対応のエラーメッセージ
- Oracle Solaris 監査レコードのサポート
- GSS-API を使用した Kerberos のスレッドに対して安全な使用
- 暗号化のための暗号化フレームワークの使用

♦ ♦ ♦ 第 3 章

3 Kerberos サービスの計画

この章では、管理者が Kerberos サービスを構成または配備する前に解決しておく必要のあるいくつかのインストールおよび構成オプションについて説明します。

- 63 ページの「Kerberos 配備の計画」
- 63 ページの「Kerberos レルムの計画」
- 67 ページの「KDC の計画」
- 69 ページの「Kerberos クライアントの計画」
- 71 ページの「Kerberos での UNIX 名と UNIX 資格の使用の計画」

Kerberos 配備の計画

Kerberos サービスをインストールする前に、いくつかの構成についての問題を解決する必要があります。初期インストールのあとに構成を変更することは可能ですが、変更によっては実装が困難になる場合があります。さらに、変更によっては KDC の再構築が必要になる場合もあるため、Kerberos を配備する前に長期的な目標を考慮してください。

Kerberos の基盤の配備には、KDC のインストール、ホストの鍵の作成、ユーザーの移行などのタスクが含まれます。Kerberos 配備の再構成は、初期の配備を実行するのと同程度に困難になる場合があるため、再構成しなくても済むように配備を慎重に計画してください。

Kerberos レルムの計画

レルム は、ドメインに似た論理ネットワークです。レルムは、同一マスター KDC に登録されるシステムのグループを定義します。DNS ドメイン名を確立する場合と同様に、Kerberos サービスを構成する前に、レルム名、各レルムの数とサイズ、レルム間認証のためのレルムとその他のレルムとの関係などの問題を解決する必要があります。

Kerberos レルム名

レルム名には、任意の ASCII 文字列を使用できます。通常、レルム名が大文字である点を除き、レルム名は DNS ドメイン名と同じです。この規則により、なじみのある名前を維持しながら、Kerberos サービスに関する問題と DNS 名前空間に関する問題を区別しやすくなります。どのような文字列でも使用できますが、それにより、構成や保守に必要な作業が増える可能性があります。標準のインターネット命名構造に従ったレルム名を使用してください。

Kerberos レルムの数

インストールするレルムの数は、次の要因によって異なります。

- サポートするクライアント数。1 つのレルムに配置するクライアントが多すぎると、管理が複雑になり、レルムの分割が必要になることがあります。サポートできるクライアント数は、主に次の要因によって決まります。
  - 各クライアントが生成する Kerberos トラフィックの量
  - 物理ネットワークの帯域幅
  - ホストの処理速度

  インストールごとに制限が違ってくるため、最大クライアント数を決定する規則はありません。
- クライアント間の距離。クライアントが地理的に異なる領域に配置されている場合は、小さなレルムをいくつか設定することが望ましい方法です。
- KDC としてインストールできるホスト数。レルムあたり少なくとも 2 つの KDC サーバー (1 つのマスターサーバーと少なくとも 1 つのスレーブサーバー) を作成するように計画してください。

Kerberos レルムと管理ドメインがそろっているようにすることをお勧めします。Kerberos V レルムは、そのレルムの対応する DNS ドメインの複数のサブドメインにまたがることができます。

Kerberos レルムの階層

複数のレルムを構成してレルム間認証を行う場合は、レルム間の接続方法を決定する必要があります。レルム間に階層関係を設定すると、関連付けたドメインに自動パスが作成されます。階層チェーン内のすべてのレルムが正しく構成されていれば、これらの自動パスによって管理の負荷を軽減できます。ただし、ドメインのレベル数が多

い場合は、非常に多くのトランザクションが必要になるため、自動パスを使用しないようにすることをお勧めします。

また、信頼関係を直接確立することもできます。直接の信頼関係は、2 つの階層レルム間に存在するレベル数が多すぎる場合や、階層関係が存在しない場合に役立ちます。接続は、この接続を使用するすべてのホスト上の `/etc/krb5/krb5.conf` ファイルで定義する必要があるため、ある程度の追加の作業が必要になります。直接の信頼関係は、推移的関係とも呼ばれます。図については、53 ページの「Kerberos レルム」を参照してください。構成手順については、140 ページの「レルム間認証の構成」を参照してください。

ホスト名の Kerberos レルムへのマッピング

ホスト名のレルム名へのマッピングは、`krb5.conf` ファイルの `domain_realm` セクションで定義されます。これらのマッピングは、必要に応じてドメイン全体およびホスト単位に定義できます。

また、DNS を使用して KDC に関する情報を検索することもできます。DNS を使用すると、変更のたびにすべての Kerberos クライアント上の `krb5.conf` ファイルを編集する必要がないため、情報の変更が容易になります。

注記 - このガイドの手順では、DNS の使用を前提にしています。

詳細は、`krb5.conf(4)` のマニュアルページを参照してください。

Oracle Solaris の Kerberos クライアントは、Active Directory サーバーと適切に相互運用します。Active Directory サーバーは、ホストのマッピングにレルムを提供するように構成できます。

Kerberos クライアント名とサービス主体名

Oracle Solaris の Kerberos では、`name-service/switch` サービスを使用しません。代わりに、Kerberos サービスは DNS を使用してホスト名を解決します。そのため、すべてのホスト上で DNS が有効になっている必要があります。DNS では、主体に各ホストの完全修飾ドメイン名 (FQDN) が含まれている必要があります。たとえば、ホスト名が `boston`、DNS ドメイン名が `example.com`、レルム名が `EXAMPLE.COM` である場合、ホストの主体名は `host/boston.example.com@EXAMPLE.COM` になります。このガイドの例では、DNS が構成されていて、各ホストに対して FQDN が使用されていることが必要です。

Kerberos サービスは DNS を介してホストの別名を正規化し、関連するサービスのサービス主体を構築する際には正規化された形式 (正規名) を使用します。そのため、サービス主体を作成するとき、サービス主体名のホスト名コンポーネントはそのサービスを提供するシステムのホスト名の正規化形式です。

次の例は、Kerberos サービスがホスト名を正規化する方法を示しています。ユーザーがコマンド `ssh alpha.example.com` (ここで、`alpha.example.com` は正規名 `beta.example.com` に対する DNS ホストの別名) を実行すると、Kerberos サービスは `alpha.example.com` を `beta.example.com` に正規化します。KDC は、このチケットをサービス主体 *host*`/beta.example.com` に対するリクエストとして処理します。

ホストの FQDN を含む主体名の場合は、`/etc/resolv.conf` ファイル内の DNS ドメイン名を表す文字列に必ず一致するようにしてください。Kerberos サービスでは、主体に FQDN を指定するときに、DNS ドメイン名は小文字にする必要があります。DNS ドメイン名には大文字と小文字を使用できますが、ホスト主体を作成する場合は小文字だけを使用します。たとえば、DNS ドメイン名は `example.com`、`Example.COM`、またはその他のどのようなバリエーションであってもかまいません。ホストの主体名は、`host/boston.example.com@EXAMPLE.COM` でなければなりません。

さらに、DNS クライアントサービスが実行されていない場合に多数のデーモンまたはコマンドが起動しないように、サービス管理機能 (SMF) が構成されています。`kdb5_util`、`kadmind`、`kpropd` デーモン、および `kprop` コマンドは、DNS サービスに依存するように構成されています。Kerberos サービスおよび SMF から使用可能な機能を完全に使用するには、すべてのホスト上で DNS クライアントサービスを有効にする必要があります。

Kerberos レルム内でのクロック同期

Kerberos 認証システムに参加するすべてのホストの内部クロックを、指定された最大時間内に同期する必要があります。「クロックスキュー」と呼ばれるこの機能も、Kerberos セキュリティー検査の 1 つです。参加しているホスト間でクロックスキューを超過すると、要求が拒否されます。

すべてのクロックを同期化するときは、Network Time Protocol (NTP) ソフトウェアを使用します。詳細は、142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。クロック同期のその他の方法も使用できますが、何らかの形式の同期が必要です。

Kerberos でサポートされる暗号化タイプ

暗号化タイプとは、Kerberos サービスで使用される、暗号化アルゴリズム、暗号化モード、およびハッシュアルゴリズムを特定する識別子です。Kerberos サービスの鍵

には、このサービスがその鍵で暗号化操作を実行するときに使用される暗号化アルゴリズムとモードを指定する暗号化タイプが関連付けられています。サポートされている暗号化タイプのリストについては、krb5.conf(4) および kdb5_util(1M) のマニュアルページを参照してください。

暗号化タイプを変更する場合は、新しい主体データベースを作成するときに行います。KDC、サーバー、クライアント間の相互作用のために、既存のデータベースでの暗号化タイプの変更は困難です。詳細は、56 ページの「Kerberos 暗号化タイプ」を参照してください。

des などの弱い暗号化タイプは、デフォルトでは許可されません。下位互換性または相互運用性のために弱い暗号化タイプを使用する必要がある場合は、/etc/krb5/krb5.conf ファイルの libdefaults セクション内の allow_weak_crypto エントリを true に設定します。

KDC の計画

KDC は特定のポートを使用し、チケットの増大する負荷を処理するために追加のサーバーを必要とし、またサーバーの同期を維持するために伝播の技法を必要とします。さらに、暗号化タイプが集中管理されます。KDC の初期構成には、いくつかのオプションがあります。

KDC と管理サービス用のポート

デフォルトでは、ポート 88 とポート 750 を KDC が使用し、ポート 749 を KDC 管理デーモンが使用します。別のポート番号を使用できます。ただし、ポート番号を変更する場合は、すべてのクライアント上で /etc/services および /etc/krb5/krb5.conf ファイルを変更する必要があります。さらに、各 KDC 上で /etc/krb5/kdc.conf ファイルを更新する必要があります。

スレーブ KDC の数

スレーブ KDC は、マスター KDC と同様に、クライアントの資格を生成します。マスターが使用できなくなると、スレーブ KDC がバックアップとして使用されます。レルムあたり少なくとも 1 つのスレーブ KDC を作成するように計画してください。

次の要因によっては、追加のスレーブ KDC が必要になることがあります。

- レルム内の物理セグメント数。通常は、レルム内のほかのセグメントが動作しない場合でも、少なくとも各セグメントで機能するように、ネットワークを設定する必

要があります。この設定を実現するには、KDC をすべてのセグメントからアクセス可能にする必要があります。この場合、KDC はマスターまたはスレーブのどちらでも構いません。

- レルム内のクライアント数。スレーブ KDC サーバーを追加すると、現在のサーバーの負荷を軽減することができます。

追加するスレーブ KDC が多くなりすぎないようにしてください。KDC データベースを各サーバーに伝播する必要があるため、インストール済みの KDC サーバーが増えると、レルム全体にわたってデータを更新するためにかかる時間も長くなる場合があります。また、各スレーブには KDC データベースのコピーが保存されるため、スレーブが多くなるほど、セキュリティー侵害の危険性が高くなります。

1 つ以上のスレーブ KDC をマスター KDC とスワップされるように構成します。少なくとも 1 つのスレーブ KDC をこのように構成する利点は、何らかの理由でマスター KDC に障害が発生した場合に、マスター KDC になるように事前に構成されたシステムが存在することです。手順については、144 ページの「マスター KDC とスレーブ KDC の入れ替え」を参照してください。

Kerberos データベースの伝播

マスター KDC に格納されているデータベースは、定期的にスレーブ KDC に伝播する必要があります。データベースの伝播は増分的に構成することができます。増分プロセスでは、データベース全体ではなく、更新された情報のみがスレーブ KDC に伝播されます。データベースの伝播については、148 ページの「Kerberos データベースの管理」を参照してください。

増分伝播を使用しない場合、最初に解決すべき問題の 1 つは、スレーブ KDC の更新頻度です。すべてのクライアントから使用可能な最新の情報を確保する必要性を、更新を完了するために必要な時間と比較検討しなければならなくなります。

1 つのレルムに多くの KDC が配置されている場合は、1 つまたは複数のスレーブからもデータを伝播すると、伝播プロセスを並行して行うことができます。この方法によって、更新にかかる時間は短縮されますが、同時に管理の複雑さが増します。詳細は、157 ページの「Kerberos のための並列伝播の設定」を参照してください。

KDC の構成オプション

KDC の構成には、いくつかの方法があります。もっとも簡単な方法は、`kdcmgr` ユーティリティーを使用して KDC を自動的に、または対話形式で構成する方法です。自動的な方法では、コマンド行オプションを使用して構成パラメータを定義する必要があります。この方法はスクリプトに特に適しています。対話的な方法では、必要なす

べての情報を入力するよう促されます。このコマンドを使用するための手順へのポインタについては、75 ページの「KDC サーバーの構成」を参照してください。

また、LDAP を使用して Kerberos のデータベースファイルを管理することもできます。手順については、93 ページの「LDAP ディレクトリサーバー上での KDC サーバーの構成」を参照してください。LDAP によって、Kerberos データベースとその既存のディレクトリサーバーの設定の間で調整が必要なサイトでの管理が簡略化されます。

Kerberos クライアントの計画

Kerberos クライアントは、自動的にインストールすることも、スクリプトからインストールすることも、構成ファイルを編集することによって手動で構成することもできます。保護されたネットワークログインのために、PAM フレームワークには `pam_ldap` モジュールが用意されています。`pam_ldap(5)` のマニュアルページを参照してください。また、クライアントがサービスをリクエストする場合は、そのサービスをマスター KDC 以外のサーバーが許可することもできます。

Kerberos クライアントの自動インストールの計画

Kerberos クライアントは、Oracle Solaris Automated Installer (AI) 機能を使用して、迅速かつ容易に構成できます。AI サーバー管理者は、Kerberos 構成プロファイルを作成して AI クライアントに割り当てます。さらに、AI サーバーはクライアント鍵を配信します。そのため、Kerberos クライアントは、インストール時に、セキュアなサービスをホストできる完全にプロビジョニングされた Kerberos システムになります。Automated Installer の使用によって、システム管理や保守のコストを削減できます。

`kclient` コマンドを使用すると、あらゆるタイプの KDC のクライアントのための自動インストールを作成できます。

- Oracle Solaris KDC のクライアントではないクライアントのために AI を使用できます。KDC ベンダーのリストについては、`kclient(1M)` のマニュアルページを参照してください。

  すべての KDC タイプに対して、事前に生成された keytab の転送がサポートされます。また、Oracle Solaris KDC と MS AD では自動登録もサポートされます。
- AI のための Kerberos 構成プロファイルを作成するには、`kclient` コマンドを実行します。詳細は、`kclient(1M)` のマニュアルページおよび70 ページの「Kerberos クライアントの構成オプション」を参照してください。AI を使用して Kerberos クライアントを構成する手順については、『Oracle Solaris 11.3 システムのインストール』 の 「AI を使用して Kerberos クライアントを構成する方法」を参照してください。

Kerberos クライアントの構成オプション

Kerberos クライアントは、`kclient` 構成ユーティリティーを使用するか、またはファイルを手動で編集することによって構成できます。このユーティリティーは、対話型モードおよび非対話型モードで動作します。対話型モードでは、Kerberos 固有のパラメータ値の入力を求められるため、クライアントの構成時に変更を行うことができます。非対話型モードでは、パラメータ値を含むファイルを指定します。また、非対話型モードでコマンド行オプションを追加することもできます。対話型モードでも非対話型モードでも必要とする手順は手動構成より少ないため、手順が迅速になるとともに、エラーも発生しにくくなります。

次の設定が有効な場合は、Kerberos クライアントの明示的な構成がまったく必要なくなります。

- DNS が、KDC 用の SRV レコードを返すように構成されている。
- レルム名が DNS ドメイン名に一致するか、または KDC がリフェラルをサポートする。
- Kerberos クライアントが KDC サーバーの鍵とは異なる鍵を必要としない。

次の理由により、Kerberos クライアントの明示的な構成が引き続き必要になる場合があります。

- 構成不要のプロセスは、直接構成されるクライアントに比べて多くの DNS 検索を実行するため、直接の構成より効率が低下します。
- リフェラルが使用されていない場合、構成不要のロジックは、レルムを決定するためにホストの DNS ドメイン名に依存します。この構成では若干のセキュリティーリスクが導入されますが、このリスクは `dns_lookup_realm` を有効にするよりはるかに軽微です。
- `pam_krb5` モジュールは、キータブファイル内のホスト鍵のエントリに依存します。この要件は `krb5.conf` ファイルで無効にすることができますが、セキュリティー上の理由からそれはお勧めできません。詳細は、70 ページの「Kerberos クライアントログインのセキュリティー」および `krb5.conf(4)` のマニュアルページを参照してください。

クライアント構成の詳細は、111 ページの「Kerberos クライアントの構成」を参照してください。

Kerberos クライアントログインのセキュリティー

ログイン時に、クライアントは `pam_krb5` モジュールを使用して、最新の TGT を発行した KDC が `/etc/krb5/krb5.keytab` ファイル内に格納されているクライアントのホスト主体を発行した KDC と同じであることを確認します。`pam_krb5` モジュールは、認証スタックで構成されるときに KDC を確認します。クライアントのホス

ト主体を格納しない DHCP クライアントなどの一部の構成では、このチェックを無効にする必要があります。このチェックを無効にするには、krb5.conf ファイル内の verify_ap_req_nofail オプションを false に設定する必要があります。詳細は、123 ページの「チケット認可チケットの確認の無効化」を参照してください。

Kerberos での信頼できる委任されたサービス

一部のアプリケーションでは、クライアントが、ほかのサービスへの接続時にそれに代わって動作するサーバーに権限を委託することが必要な場合があります。そのクライアントは、中間サーバーに資格を転送する必要があります。サーバーへのサービスチケットを取得するクライアントの機能では、委任された資格の受け入れについてサーバーを信頼できるかどうかに関する情報がクライアントに伝達されません。kadmin コマンドの ok_to_auth_as_delegate オプションは、中間サーバーにそのような資格の受け入れを信頼して任せられるかどうかに関するローカルレルムポリシーを KDC からクライアントに送る方法を提供します。

クライアントへの KDC 応答の暗号化された部分には、ok_to_auth_as_delegate オプションが設定された資格チケットフラグのコピーを含めることができます。クライアントは、この設定を使用して、このサーバーに (プロキシまたは転送された TGT のどちらかを付与することによって) 資格を委任するかどうかを判定できます。このオプションを設定する場合は、そのサービスで委任された資格の使用が必要かどうかだけでなく、そのサービスが実行されるサーバーのセキュリティーや配置についても考慮してください。

Kerberos での UNIX 名と UNIX 資格の使用の計画

Kerberos サービスは、GSS 資格名から UNIX ユーザー ID (UID) へのマッピングを、NFS などこのマッピングを必要とする GSS アプリケーションのために提供します。GSS 資格名は Kerberos サービスを使用する場合の Kerberos 主体名と等価です。また、デフォルトの Kerberos レルム内に有効なユーザーアカウントを持っていない UNIX ユーザーを、PAM フレームワークを使用して自動的に移行できます。

GSS 資格の UNIX 資格へのマッピング

デフォルトのマッピングアルゴリズムでは、Kerberos 主体のプライマリ名を使用して UID を検索します。この検索は、デフォルトレルムか、または /etc/krb5/krb5.conf ファイル内の auth_to_local_realm パラメータによって許可された任意のレルム内で実行されます。たとえば、ユーザー主体名 jdoe@EXAMPLE.COM は、パスワードテーブルを使用して jdoe という名前の UNIX ユーザーの UID にマップされま

す。ユーザー主体名 jdoe/admin@EXAMPLE.COM は、admin というインスタンスコンポーネントが含まれているため、マップされません。

ユーザー資格のデフォルトマッピングが十分な場合、GSS 資格テーブルにデータを入れておく必要がありません。インスタンスコンポーネントを含む主体名をマップする場合など、デフォルトのマッピングが十分でない場合は、ほかの方法が必要です。詳細については、次を参照してください。

- 134 ページの「資格テーブルを作成および変更する方法」
- 135 ページの「レルム間の資格マッピングを提供する方法」
- 209 ページの「GSS 資格の UNIX 資格へのマッピングの監視」

gsscred テーブル

デフォルトのマッピングでは十分でないとき、NFS サーバーは gsscred テーブルを使用して、Kerberos ユーザーを識別します。NFS サービスは、UNIX UID を使用してユーザーを識別します。UNIX ID は、ユーザー主体または資格には含まれません。gsscred テーブルによって、GSS 資格からパスワードファイルの UNIX UID への追加のマッピングが可能になります。このテーブルは、KDC データベースを生成したあとに作成および開始する必要があります。

クライアントリクエストが到着すると、NFS サービスは、資格名を UNIX UID にマップしようとします。このマッピングに失敗した場合、gsscred テーブルが確認されます。

Kerberos レルムへのユーザーの自動的な移行

デフォルトの Kerberos レルム内に有効なユーザーアカウントを持っていない UNIX ユーザーを、PAM フレームワークを使用して自動的に移行できます。具体的には、pam_krb5_migrate.so モジュールを PAM サービスの認証スタックに追加します。それによってサービスが構成され、Kerberos 主体を持っていないユーザーがシステムへのパスワードログインに成功した場合は常に、そのユーザーに対して Kerberos 主体が自動的に作成されるようになります。その場合、新しい主体パスワードは UNIX パスワードと同じになります。pam_krb5_migrate.so モジュールの使用については、124 ページの「Kerberos レルム内のユーザーを自動的に移行するように構成する方法」を参照してください。

♦ ♦ ♦ 第 4 章

Kerberos サービスの構成

この章では、KDC サーバー、ネットワークアプリケーションサーバー、NFS サーバー、および Kerberos クライアントの構成手順について説明します。これらの手順の多くは `root` アクセスを必要とするため、システム管理者または上級ユーザーが実行するようにしてください。レルム間の構成手順など、KDC サーバーに関するトピックについても説明します。

この章で扱う内容は、次のとおりです。

- 73 ページの「Kerberos サービスの構成」
- 75 ページの「KDC サーバーの構成」
- 93 ページの「LDAP ディレクトリサーバー上での KDC サーバーの構成」
- 111 ページの「Kerberos クライアントの構成」
- 130 ページの「Kerberos ネットワークアプリケーションサーバーの構成」
- 132 ページの「Kerberos NFS サーバーの構成」
- 138 ページの「Kerberos サービスへのアクセスのための遅延実行の構成」
- 140 ページの「レルム間認証の構成」
- 142 ページの「KDC と Kerberos クライアントのクロックの同期化」
- 144 ページの「マスター KDC とスレーブ KDC の入れ替え」
- 148 ページの「Kerberos データベースの管理」
- 158 ページの「Kerberos データベースの stash ファイルの管理」
- 161 ページの「Kerberos サーバー上のセキュリティーの強化」

Kerberos サービスの構成

構成プロセスの一部の手順はほかの手順に依存するため、特定の順序で実行する必要があります。多くの場合、これらの手順に従うことにより、Kerberos サービスに必要なサービスを設定できます。その他の手順は互いに依存しないため、任意のタイミングで実行できます。次のタスクマップで、推奨する Kerberos のインストール順序を示します。

注記 - これらのセクションの例では、Oracle Solaris 用の FIPS 140-2 検証済みではないデフォルトの暗号化タイプを使用します。FIPS 140-2 モードで実行するには、データベース、サーバー、およびクライアント通信用の暗号化タイプを `des3-cbc-sha1` 暗号化タイプに制限する必要があります。KDC を作成する前に、77 ページの「FIPS 140-2 モードで実行するように Kerberos を構成する方法」のファイルを編集してください。

表 2 タスクマップ: Kerberos サービスの構成

タスク	説明	参照先
1. Kerberos のインストールを計画します。	ソフトウェアの構成プロセスを開始する前に、構成に関する問題を解決します。事前の計画により、あとで時間やその他のリソースが節約されます。	第3章「Kerberos サービスの計画」
2. KDC サーバーを構成します。	レルムのマスター KDC サーバーとスレーブ KDC サーバー、および KDC データベースを構成および構築します。	75 ページの「KDC サーバーの構成」
2a.(オプション) FIPS 140-2 モードで実行するように Kerberos を構成します。	FIPS 140-2 検証済みアルゴリズムの使用のみを有効にします。	77 ページの「FIPS 140-2 モードで実行するように Kerberos を構成する方法」
2b.(オプション) Kerberos を LDAP 上で動作するように構成します。	LDAP ディレクトリサーバーを使用するように KDC を構成します。	93 ページの「LDAP ディレクトリサーバー上での KDC サーバーの構成」
3. クロック同期ソフトウェアをインストールします。	ネットワーク上のすべてのシステムに時間を提供する中央のクロックを作成します。	142 ページの「KDC と Kerberos クライアントのクロックの同期化」
4.(オプション) 入れ替え可能な KDC を構成します。	マスター KDC とスレーブ KDC を簡単に入れ替えできるようにします。	144 ページの「入れ替え可能なスレーブ KDC を構成する方法」
4.(オプション) KDC のセキュリティーを強化します。	KDC サーバーに対するセキュリティー侵害を回避します。	161 ページの「KDC サーバーへのアクセスの制限」

追加の Kerberos サービスの構成

必要な手順が完了したら、必要に応じて次の手順を実行します。

表 3 タスクマップ: 追加の Kerberos サービスの構成

タスク	説明	参照先
レルム間認証を構成します。	レルム間の通信を使用可能にします。	140 ページの「レルム間認証の構成」
Kerberos アプリケーションサーバーを構成します。	サーバーが Kerberos 認証を使用して `ftp` などのサービスをサポートできるようにします。	130 ページの「Kerberos ネットワークアプリケーションサーバーの構成」
遅延実行サービスを構成します。	`cron` ホストがいつでもタスクを実行できるようにします。	138 ページの「Kerberos サービスへのアクセスのための遅延実行の構成」

タスク	説明	参照先
Kerberos NFS サーバーを構成します。	Kerberos 認証を必要とするファイルシステムを、サーバーが共有できるようにします。	132 ページの「Kerberos NFS サーバーの構成」
Kerberos クライアントを構成します。	クライアントが Kerberos サービスを使用できるようにします。	111 ページの「Kerberos クライアントの構成」
認証に役立つようにクロックを同期します。	クロック同期プロトコルを構成します。	142 ページの「KDC と Kerberos クライアントのクロックの同期化」
Kerberos データベースを管理します。	Kerberos データベースを保守します。	148 ページの「Kerberos データベースの管理」
Kerberos データベースの stash ファイルを管理します。	Kerberos データベースの鍵を処理します。	158 ページの「Kerberos データベースの stash ファイルの管理」

KDC サーバーの構成

Kerberos ソフトウェアをインストールしたら、鍵配布センター (KDC) サーバーを構成する必要があります。マスター KDC と少なくとも 1 つのスレーブ KDC を構成することによって、資格を発行するサービスが提供されます。これらの資格は Kerberos サービスの基盤であるため、ほかのタスクを試行する前に KDC を構成しておく必要があります。

マスター KDC とスレーブ KDC のもっとも大きな違いは、マスター KDC だけがデータベース管理リクエストを処理できることです。たとえば、パスワードの変更や新しい主体の追加は、マスター KDC で行います。これらの変更は、スレーブ KDC に伝播されます。資格の生成は、スレーブ KDC とマスター KDC が行います。スレーブ KDC は、マスター KDC が応答できない場合の冗長性を提供します。

マスター KDC サーバー、データベース、および追加のサーバーを構成および構築するには、次のさまざまな方法を選択できます。

- 自動 – スクリプトに推奨されます
- 対話型 – ほとんどのインストールにはこれで十分です
- 手動 – より複雑なインストールに必要です
- LDAP を使用した手動 – KDC で LDAP を使用する場合に必要です

表 4 タスクマップ: KDC サーバーの構成

タスク	説明	参照先
KDC パッケージをインストールします。	KDC を作成するために必要なパッケージ。	76 ページの「KDC パッケージをインストールする方法」
(オプション) FIPS 140-2 モードで実行するように Kerberos を構成します。	FIPS 140-2 検証済みアルゴリズムの使用のみを有効にします。	77 ページの「FIPS 140-2 モードで実行するように Kerberos を構成する方法」

タスク	説明	参照先
スクリプトを使用してマスター KDC を構成します。	初期構成を簡略化します。	78 ページの「kdcmgr を使用してマスター KDC を構成する方法」 例6「引数なしでの kdcmgr コマンドの実行」
スクリプトを使用してスレーブ KDC サーバーを構成します。	初期構成を簡略化します。	80 ページの「kdcmgr を使用してスレーブ KDC を構成する方法」
マスター KDC サーバーを手動で構成します。	初期インストール中に KDC 構成ファイル内のすべてのエントリを制御できます。	81 ページの「マスター KDC を手動で構成する方法」
スレーブ KDC サーバーを手動で構成します。	初期インストール中に KDC 構成ファイル内のすべてのエントリを制御できます。	86 ページの「スレーブ KDC を手動で構成する方法」
2 番目のマスター KDC サーバーを手動で構成します。	Oracle Directory Server Enterprise Edition (DSEE) LDAP サーバーを含むマルチマスター構成を作成します。	90 ページの「2 番目のマスター KDC を手動で構成する方法」
KDC サーバー上の主体鍵を置き換えます。	より強力な暗号化タイプを使用するために、レガシー KDC サーバー上のセッション鍵を更新します。	92 ページの「マスターサーバー上のチケット認可サービス鍵の置き換え」
LDAP を使用するようにマスター KDC を手動で構成します。	LDAP を使用するようにマスター KDC サーバーおよび Kerberos クライアントを構成します。	93 ページの「LDAP ディレクトリサーバー上での KDC サーバーの構成」

▼ KDC パッケージをインストールする方法

Kerberos クライアントソフトウェアは、デフォルトでシステムにインストールされます。鍵配布センター (KDC) をインストールするには、KDC パッケージを追加する必要があります。

始める前に システムにパッケージを追加するには、ソフトウェアインストールに関連する権利プロファイルが割り当てられている必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **KDC パッケージをインストールします。**

```
$ pkg install system/security/kerberos-5
```

詳細は、pkg(1) のマニュアルページを参照してください。

2. **(オプション) Kerberos サービスを一覧表示します。**
サーバーパッケージの追加により、システムには 2 つの Kerberos サービスが追加されます。Kerberos クライアントソフトウェアと同様に、これらのサービスは、デフォルトでは無効になっています。これらのサービスを有効にする前に、Kerberos を構成します。

```
$ svcs -a krb5
```

```
STATE          STIME    FMRI
disabled       Sep_10   svc:/network/security/krb5kdc:default
disabled       Sep_10   svc:/network/security/krb5_prop:default
$ svcs -a | grep kerb
STATE          STIME    FMRI
disabled       Sep_07   svc:/system/kerberos/install:default
```

▼ FIPS 140-2 モードで実行するように Kerberos を構成する方法

マスター KDC および Kerberos 主体の鍵を作成する前に、次の手順を実行します。81 ページの「マスター KDC を手動で構成する方法」で説明したように、マスター KDC を手動で構成する際は、次の手順を実行するようにしてください。

始める前に Kerberos を FIPS 140-2 モードで実行するには、システムの FIPS 140-2 モードを有効にする必要があります。『Oracle Solaris 11.3 での暗号化と証明書の管理』 の 「FIPS 140-2 が有効になったブート環境を作成する方法」を参照してください。

1. **マスター KDC 上で、KDC 用に暗号化タイプを編集します。**

   `kdc.conf` ファイルの `[realms]` セクションで、KDC データベースのマスター鍵のタイプを設定します。

   ```
   # pfedit  /etc/krb5/kdc.conf
   ...
   master_key_type = des3-cbc-sha1-kd
   ```

2. **同じファイルで、ほかの暗号化タイプを明示的に禁止します。**

   コマンドを実行して暗号化を設定することもできるため、構成ファイルでコマンドへの非 FIPS 140-2 アルゴリズム引数の使用を禁止するようにしてください。

   ```
           supported_enctypes = des3-cbc-sha1-kd:normal
   ```

3. **`krb5.conf` ファイルの `[libdefaults]` セクションで、トランザクション用の暗号化タイプを編集します。**

   これらのパラメータによって、Kerberos サーバー、サービス、およびクライアント用の暗号化タイプが制限されます。

   ```
   # pfedit /etc/krb5/krb5.conf
           default_tgs_enctypes = des3-cbc-sha1-kd
           default_tkt_enctypes = des3-cbc-sha1-kd
           permitted_enctypes = des3-cbc-sha1-kd
   ```

4. **同じファイルで、弱い暗号化タイプを明示的に禁止します。**

   ```
           allow_weak_enctypes = false
   ```

注意事項 56 ページの「Kerberos 暗号化タイプ」を参照してください。

▼ kdcmgr を使用してマスター KDC を構成する方法

kdcmgr スクリプトは、マスターおよびスレーブ KDC をインストールするためのコマンド行インタフェースを提供します。マスターの場合は、Kerberos データベースのパスワードと管理者のパスワードを作成する必要があります。スレーブ KDC 上で、これらのパスワードを指定してインストールを完了する必要があります。これらのパスワードについては、kdcmgr(1M) のマニュアルページを参照してください。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **マスター KDC を作成します。**
   コマンド行で、kdcmgr コマンドを実行し、管理者とレルムを指定します。
   マスター鍵と呼ばれる Kerberos データベースパスワードと、管理主体のパスワードの入力を求められます。スクリプトからパスワードの入力が求められます。

```
kdc1# kdcmgr -a kws/admin -r EXAMPLE.COM create master

Starting server setup
---------------------------------------

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf

Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     /** Type strong password **/
Re-enter KDC database master key to verify: xxxxxxxx

Authenticating as principal root/admin@EXAMPLE.COM with password.
WARNING: no policy specified for kws/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "kws/admin@EXAMPLE.COM":     /** Type strong password **/
Re-enter password for principal "kws/admin@EXAMPLE.COM": xxxxxxxx
Principal "kws/admin@EXAMPLE.COM" created.

Setting up /etc/krb5/kadm5.acl.

--------------------------------------------------
Setup COMPLETE.

kdc1#
```

注記 - これらのパスワードを安全な場所に保存および保管してください。

2. **(オプション) マスター KDC のステータスを表示します。**

```
# kdcmgr status
```

3. **このシステムのクロックをレルム内のほかのクロックと同期します。**

認証が成功するには、すべてのクロックが、krb5.conf ファイルの libdefaults セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、krb5.conf(4) のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

注記 - マスター KDC をクロック同期サーバーにすることはできません。

- **まだクロック同期サーバーが存在しない場合は、1 台構成したあとに、この手順を実行します。**
- **クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。**
  - **PTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Precision Time Protocol の管理」の手順に従います。**
  - **NTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Network Time Protocol の管理」の手順に従います。**

例 6　引数なしでの kdcmgr コマンドの実行

この例では、スクリプトから入力を要求されたら、管理者はレルム名と管理主体を指定します。

```
kdc1# kdcmgr create master

Starting server setup
---------------------------------------

Enter the Kerberos realm: EXAMPLE.COM

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf

Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     /** Type strong password **/
Re-enter KDC database master key to verify: xxxxxxxx

Enter the krb5 administrative principal to be created: kws/admin

Authenticating as principal root/admin@EXAMPLE.COM with password.
WARNING: no policy specified for kws/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "kws/admin@EXAMPLE.COM":     /** Type strong password **/
Re-enter password for principal "kws/admin@EXAMPLE.COM": xxxxxxxx
Principal "kws/admin@EXAMPLE.COM" created.
```

```
Setting up /etc/krb5/kadm5.acl.

-------------------------------------------------
Setup COMPLETE.

kdc1#
```

▼ kdcmgr を使用してスレーブ KDC を構成する方法

始める前に マスター KDC サーバーが構成されています。

root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **スレーブ KDC を作成します。**

   コマンド行で、kdcmgr コマンドを実行し、管理者、レルム、およびマスター KDC を指定します。

   スクリプトから、78 ページの「kdcmgr を使用してマスター KDC を構成する方法」で作成した 2 つのパスワード (管理主体のパスワードと KDC データベースのパスワード) の入力が求められます。

```
kdc2# kdcmgr -a kws/admin -r EXAMPLE.COM create -m kdc1 slave

Starting server setup
--------------------------------------

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf
Obtaining TGT for kws/admin ...
Password for kws/admin@EXAMPLE.COM: xxxxxxxx

Setting up /etc/krb5/kadm5.acl.

Setting up /etc/krb5/kpropd.acl.

Waiting for database from master...
Waiting for database from master...
Waiting for database from master...
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key
Enter KDC database master key: xxxxxxxx

-------------------------------------------------
Setup COMPLETE.

kdc2#
```

2. **(オプション) KDC のステータスを表示します。**

```
# kdcmgr status
```

3. **このシステムのクロックをレルム内のほかのクロックと同期します。**
   認証が成功するには、すべてのクロックが、`krb5.conf` ファイルの `libdefaults` セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、`krb5.conf(4)` のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

   - **まだクロック同期サーバーが存在しない場合は、このサーバー上に PTP または NTP を構成します。**
     - **PTP を構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の 「Precision Time Protocol の管理」の手順に従います。**
     - **NTP を構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の 「Network Time Protocol の管理」の手順に従います。**
   - **クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。**

4. **マスター KDC に戻って、クロック同期サーバーのクライアントにします。**

▼ マスター KDC を手動で構成する方法

この手順では、増分伝播を構成します。この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- マスター KDC = `kdc1.example.com`
- `admin` 主体 = `kws/admin`

始める前に ホストが DNS を使用するように構成されています。マスター KDC を入れ替え可能にする場合の手順については、144 ページの「マスター KDC とスレーブ KDC の入れ替え」を参照してください。

`root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **KDC パッケージをインストールします。**
   76 ページの「KDC パッケージをインストールする方法」の手順に従います。

2. **(オプション) FIPS 140-2 モードで実行するように Kerberos を構成します。**
   FIPS 140-2 モードでレルムを実行する予定の場合は、マスター KDC サーバーを構成する際に、暗号化タイプを指定します。77 ページの「FIPS 140-2 モードで実行するように Kerberos を構成する方法」を参照してください。

3. **Kerberos 構成ファイル `krb5.conf` を編集します。**
   このファイルについては、`krb5.conf(4)` のマニュアルページを参照してください。
   この例では、管理者は `default_realm`、`kdc`、`admin_server`、およびすべての `domain_realm` エントリの行を変更し、`help_url` エントリを編集します。

```
kdc1# pfedit /etc/krb5/krb5.conf
...
[libdefaults]
     default_realm = EXAMPLE.COM

[realms]
     EXAMPLE.COM = {
        kdc = kdc1.example.com
        admin_server = kdc1.example.com
     }

[domain_realm]
     .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
     default = FILE:/var/krb5/kdc.log
     kdc = FILE:/var/krb5/kdc.log
```

注記 - 古い Kerberos システムと通信する必要がある場合は、暗号化タイプの制限が必要になることがあります。暗号化タイプの制限に関する問題については、56 ページの「Kerberos 暗号化タイプ」を参照してください。

4. **KDC 構成ファイル `kdc.conf` 内のレルムを指定します。**
   このファイルについては、`kdc.conf(4)` のマニュアルページを参照してください。
   この例では、管理者はレルム名の定義に加えて、増分伝播とロギングのデフォルト値を変更します。

```
kdc1# pfedit /etc/krb5/kdc.conf
[kdcdefaults]
     kdc_ports = 88,750

[realms]
     EXAMPLE.COM = {
        profile = /etc/krb5/krb5.conf
        database_name = /var/krb5/principal
        acl_file = /etc/krb5/kadm5.acl
        kadmind_port = 749
        max_life = 8h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        sunw_dbprop_enable = true
        sunw_dbprop_master_ulogsize = 1000
```

```
}
```

注記 - 古い Kerberos システムと通信する必要がある場合は、暗号化タイプの制限が必要になることがあります。暗号化タイプの制限に関する問題については、56 ページの「Kerberos 暗号化タイプ」を参照してください。

5. kdb5_util コマンドを使用して KDC データベースを作成します。

kdb5_util コマンドは、KDC データベースを作成します。-s オプションを指定すると、kadmind と krb5kdc デーモンが起動する前に、KDC の認証に使用される stash ファイルが作成されます。詳細は、kdb5_util(1M)、kadmind(1M)、および krb5kdc(1M) のマニュアルページを参照してください。

```
kdc1# /usr/sbin/kdb5_util create -s
Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM'
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     /** Type strong password **/
Re-enter KDC database master key to verify: xxxxxxxx
```

ヒント - この手順が失敗した場合は、KDC 主体がその FQDN で識別されることを確認してください。

```
# getent hosts IP-address-of-KDC
IP-address-of-KDC  kdc       /** This entry does not include FQDN **/
```

次に、その FQDN を /etc/hosts ファイル内の最初の KDC のエントリとして追加します。次に例を示します。

```
10.1.2.3 kdc1.example.com kdc
```

6. 管理主体をデータベースに追加します。

必要な数の admin 主体を追加できます。KDC 構成処理を完了するには、1 つ以上の admin 主体を追加する必要があります。この例では、kws/admin 主体を追加します。「kws」の代わりに、適切な主体名で置き換えることができます。

```
kadmin.local: addprinc kws/admin
Enter password
for principal kws/admin@EXAMPLE.COM:      /** Type strong password **/
Re-enter password
for principal kws/admin@EXAMPLE.COM: xxxxxxxx
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

詳細は、kadmin(1M) のマニュアルページを参照してください。

7. Kerberos アクセス制御リストファイル kadm5.acl を編集します。

生成されたあと、/etc/krb5/kadm5.acl ファイルには、KDC を管理することを許可されたすべての主体名が含まれている必要があります。

```
kws/admin@EXAMPLE.COM   *
```

前のエントリにより、EXAMPLE.COM レルム内の kws/admin 主体は KDC 内の主体やポリシーを変更できるようになります。デフォルトの主体エントリはアスタリスク (*) です。これは、すべての admin 主体に一致します。このエントリはセキュリティーリスクになる場合があります。すべての admin 主体とその権利を明示的に記載するように、このファイルを変更してください。詳細は、kadm5.acl(4) のマニュアルページを参照してください。

8. **KDC および kadmin サービスを有効にします。**

```
kdc1# svcadm enable -r network/security/krb5kdc
kdc1# svcadm enable -r network/security/kadmin
```

9. **さらに主体を追加します。**

```
kdc1# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin:
```

   a. **マスター KDC の host 主体を作成し、それをキータブファイルに追加します。**

   ホスト主体は、変更をスレーブ KDC に伝播するために、kprop などの Kerberos アプリケーションによって使用されます。この主体はまた、ssh などのネットワークアプリケーションを使用して KDC サーバーにセキュアなリモートアクセスを提供するためにも使用されます。主体のインスタンスがホスト名である場合、FQDN は、ネームサービスでのドメイン名の大文字小文字には関係なく小文字で指定する必要があることに注意してください。

```
kadmin: addprinc -randkey host/kdc1.example.com
Principal "host/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

   keytab ファイルにホスト主体を追加すると、自動的に、sshd などのアプリケーションサーバーがこの主体を使用できるようになります。

```
kadmin: ktadd host/kdc1.example.com
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-256 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-128 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

   b. **(オプション) clntconfig 主体を作成し、それに kclient タスクを実行するのに十分な特権を付与します。**

   この主体は、Kerberos クライアントのインストール中に kclient ユーティリティーで使用されます。このユーティリティーを使用しない場合は、主体に追加する必要はありません。kclient ユーティリティーのユーザーは、このパスワードを使用する必要があります。

```
kadmin: addprinc clntconfig/admin
```

```
Enter password for principal clntconfig/admin@EXAMPLE.COM:     /** Type strong password **/
Re-enter password for principal clntconfig/admin@EXAMPLE.COM: xxxxxxxx
Principal "clntconfig/admin@EXAMPLE.COM" created.
kadmin: quit
```

注記 - このパスワードを安全な場所に保存および保管してください。

```
# pfedit /etc/krb5/kadm5.acl
...
clntconfig/admin@EXAMPLE.COM  acdilm
```

詳細は、kclient(1M) のマニュアルページを参照してください。

10. このシステムのクロックをレルム内のほかのクロックと同期します。

認証が成功するには、すべてのクロックが、krb5.conf ファイルの libdefaults セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、krb5.conf(4) のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

注記 - マスター KDC をクロック同期サーバーにすることはできません。

- **まだクロック同期サーバーが存在しない場合は、1 台構成したあとに、この手順を実行します。**
- **クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。**
  - **PTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Precision Time Protocol の管理」の手順に従います。**
  - **NTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Network Time Protocol の管理」の手順に従います。**

11. 追加の KDC を構成します。

冗長性を提供するには、別の KDC を 1 つ以上インストールします。マルチマスター構成では、セカンダリ KDC をスレーブ KDC または追加のマスター KDC にすることができます。

- スレーブ KDC を作成する手順については、80 ページの「kdcmgr を使用してスレーブ KDC を構成する方法」または86 ページの「スレーブ KDC を手動で構成する方法」を参照してください。
- 2 番目のマスター KDC を作成する手順については、90 ページの「2 番目のマスター KDC を手動で構成する方法」を参照してください。

- LDAP 構成で 2 番目のマスター KDC を作成する手順については、104 ページの「OpenLDAP クライアントの Oracle DSEE LDAP ディレクトリサーバー上でマスター KDC を構成する方法」を参照してください。

▼ スレーブ KDC を手動で構成する方法

この手順では、`kdc2` という名前の新しいスレーブ KDC を構成します。また、増分伝播も構成します。この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- マスター KDC = `kdc1.example.com`
- スレーブ KDC = `kdc2.example.com`
- `admin` 主体 = `kws/admin`

始める前に マスター KDC が構成されていることを確認してください。このスレーブをスワップ可能にする場合は、145 ページの「マスター KDC とスレーブ KDC を入れ替えする方法」の手順に従ってください。

KDC サーバー上で `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **スレーブのホスト主体をデータベースに追加し、増分伝播の主体を作成します。**

   a. **`kadmin` コマンドを使用して、マスター KDC にログインします。**

   マスター KDC を構成するときに作成した `admin` 主体名のいずれかを使用します。

   ```
   kdc1# /usr/sbin/kadmin -p kws/admin
   Enter password: xxxxxxxx
   kadmin:
   ```

   詳細は、`kadmin(1M)` のマニュアルページを参照してください。

   b. **データベースにスレーブのホスト主体が追加されていない場合は、追加します。**

   スレーブはホスト主体を持っている必要があります。主体のインスタンスがホスト名である場合、FQDN は、ネームサービスでのドメイン名の大文字小文字には関係なく小文字で指定する必要があることに注意してください。

   ```
   kadmin: addprinc -randkey host/kdc2.example.com
   Principal "host/kdc2.example.com@EXAMPLE.COM" created.
   kadmin:
   ```

c. **マスター KDC からの増分伝播を承認するために、`kiprop` 主体を作成します。**

```
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

d. **`kadmin` を終了します。**

```
kadmin: quit
```

2. **マスター KDC 上で、各スレーブのエントリを Kerberos 構成ファイル `krb5.conf` に追加します。**

   このファイルについては、`krb5.conf(4)` のマニュアルページを参照してください。

```
kdc1# pfedit /etc/krb5/krb5.conf
.
.
[realms]
EXAMPLE.COM = {
kdc = kdc1.example.com
kdc = kdc2.example.com
admin_server = kdc1.example.com
}
```

3. **マスター KDC が増分伝播に対する要求を受信できるように、マスター KDC 上で、`kiprop` エントリを `kadm5.acl` ファイルに追加します。**

```
kdc1# pfedit /etc/krb5/kadm5.acl
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

4. **新しいエントリが使用されるように `kadmin` サービスを再起動します。**

```
kdc1# svcadm restart network/security/kadmin
```

5. **マスター KDC サーバーからすべてのスレーブ KDC に、KDC 管理ファイルをコピーします。**

   各スレーブ KDC には、マスター KDC サーバーに関する最新の情報が必要です。`sftp` または同様の転送メカニズムを使用して、マスター KDC から次のファイルのコピーを取得できます。

   - `/etc/krb5/krb5.conf`
   - `/etc/krb5/kdc.conf`

6. **すべてのスレーブ KDC 上で、マスター KDC のエントリと各スレーブ KDC のエントリをデータベース伝播構成ファイル `kpropd.acl` に追加します。**

```
kdc2# pfedit /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
```

7. **すべてのスレーブ KDC 上で、Kerberos アクセス制御リストファイル `kadm5.acl` が反映されていないことを確認します。**
   変更されていない `kadm5.acl` ファイルは、次の例のようになります。

   ```
   kdc2# pfedit /etc/krb5/kadm5.acl
   */admin@___default_realm___ *
   ```

   ファイルに `kiprop` のエントリがある場合は、それを削除します。

8. **すべてのスレーブ KDC 上で、ポーリング間隔を `kdc.conf` ファイルに定義します。**
   `sunw_dbprop_master_ulogsize` エントリを、スレーブのポーリング間隔を定義するエントリに置き換えます。次のエントリにより、ポーリング時間が 2 分に設定されます。

   ```
   kdc2# pfedit /etc/krb5/kdc.conf
   [kdcdefaults]
         kdc_ports = 88,750

   [realms]
         EXAMPLE.COM= {
           profile = /etc/krb5/krb5.conf
           database_name = /var/krb5/principal
           acl_file = /etc/krb5/kadm5.acl
           kadmind_port = 749
           max_life = 8h 0m 0s
           max_renewable_life = 7d 0h 0m 0s
           sunw_dbprop_enable = true
           sunw_dbprop_slave_poll = 2m
         }
   ```

9. **スレーブの KDC キータブファイルを構成します。**

   a. **`kadmin` コマンドを使用して、スレーブ KDC にログインします。**
      マスター KDC を構成するときに作成した `admin` 主体名のいずれかを使用してログインします。

      ```
      kdc2# /usr/sbin/kadmin -p kws/admin
      Enter password: xxxxxxxx
      kadmin:
      ```

   b. **スレーブの KDC キータブファイルにスレーブのホスト主体を追加します。**
      このエントリにより、`kprop` コマンドやその他の Kerberos アプリケーションが機能できるようになります。主体のインスタンスがホスト名である場合、FQDN は、ネームサービスでのドメイン名の大文字小文字には関係なく小文字で指定する必要があることに注意してください。詳細は、`kprop(1M)` のマニュアルページを参照してください。

      ```
      kadmin: ktadd host/kdc2.example.com
      Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-256 CTS
       mode
      with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
      Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-128 CTS
       mode
      with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
      ```

```
Entry for principal host/kdc2.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **増分伝播が開始されるときに、`kprpod` コマンド自体を認証できるように、スレーブ KDC のキータブファイルに `kiprop` 主体を追加します。**

```
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-256 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES
 cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

10. **スレーブ KDC 上で、Kerberos 伝播デーモンを有効にします。**

```
kdc2# svcadm enable network/security/krb5_prop
```

11. **スレーブ KDC 上で、stash ファイルを作成します。**

```
kdc2# /usr/sbin/kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key: xxxxxxxx
```

詳細は、kdb5_util(1M) のマニュアルページを参照してください。

12. **このシステムのクロックをレルム内のほかのクロックと同期します。**

認証が成功するには、すべてのクロックが、`krb5.conf` ファイルの `libdefaults` セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、krb5.conf(4) のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

- **まだクロック同期サーバーが存在しない場合は、このサーバー上に PTP または NTP を構成します。**
  - **PTP を構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の 「Precision Time Protocol の管理」の手順に従います。**
  - **NTP を構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の 「Network Time Protocol の管理」の手順に従います。**
- **クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。**

13. **マスター KDC に戻って、クロック同期サーバーのクライアントにします。**

14. **スレーブ KDC 上で、KDC デーモンを有効にします。**

```
kdc2# svcadm enable network/security/krb5kdc
```

▼ 2 番目のマスター KDC を手動で構成する方法

この手順では、次のパラメータを使用して、kdc2 という名前の 2 番目のマスター KDC を構成します。

- レルム名 = EXAMPLE.COM
- DNS ドメイン名 = example.com
- マスター KDC = kdc1.example.com
- kdc1 の admin 主体 = kws/admin
- kdc1 の LDAP サーバー = dsserver.example.com
- 2 番目のマスター KDC と LDAP サーバー = kdc2.example.com
- kdc2 の admin 主体 = tester/admin

新しい 2 番目のマスター KDC 上で、この手順のコマンドをすべて実行します。

始める前に マスター KDC が構成されていることを確認してください。

KDC サーバー上で root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **2 番目のマスター KDC にログインし、LDAP TLS バインディングを構成します。**

   TLS 鍵の構成例については、次を参照してください。

   - 104 ページの「OpenLDAP クライアントの Oracle DSEE LDAP ディレクトリサーバー上でマスター KDC を構成する方法」のステップ 1
   - 99 ページの「Oracle Unified Directory LDAP ディレクトリサーバー上でマスター KDC を構成する方法」のステップ 10。

2. **このマスター KDC の Kerberos 構成ファイルに、1 番目のマスター KDC および LDAP サーバーを追加します。**

   このファイルについては、krb5.conf(4) のマニュアルページを参照してください。次の例では、Oracle Directory Server Enterprise Edition サーバーを構成することを想定してします。

```
kdc2# pfedit /etc/krb5/krb5.conf
.
```

```
.
[realms]
     EXAMPLE.COM = {
        kdc = kdc1.example.com
        kdc = kdc2.example.com
        admin_server = kdc2.example.com
        admin_server = kdc1.example.com
        database_module = LDAP
     }
[dbmodules]
     LDAP = {
        db_library =kldap
        ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
        ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
        ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
        ldap_cert_path = /var/ldap
        ldap_servers = ldaps://kdc2.example.com ldaps://dsserver.example.com
     }
...
```

3. **KDC および `kadmin` サービスへの LDAP バインディング用の stash ファイルを作成します。**

   2 番目のマスター KDC 上では、`kadmin` サービスにも LDAP バインディング用の stash ファイルが存在する必要があります。

```
kdc2# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
kdc2# kdb5_ldap_util stashsrvpw "cn=kadmin service,ou=profile,dc=example,dc=com"
```

4. **マスター KDC の stash ファイルを作成します。**

```
kdc2# kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key: xxxxxxxx
```

   詳細は、`kdb5_util`(1M) のマニュアルページを参照してください。

5. **KDC 構成ファイルに Kerberos 構成ファイルの場所を追加します。**

```
# pfedit /etc/krb5/kdc.conf
 ...
 [realms]
         EXAMPLE.COM = {
                 profile = /etc/krb5/krb5.conf
 ...
```

6. **2 番目のマスター KDC の主体を作成します。**

   通常、`kdb5_ldap_util` コマンドは KDC の管理主体を作成します。ただし、これらの主体の大部分は、1 番目のマスター KDC を作成したときに作成されています。この手順では、`kadmin` および `changepw` は 2 番目のマスター KDC 専用の主体です。

```
# kadmin -p tester/admin
kadmin: addprinc -randkey kadmin/kdc2.example.com
kadmin: addprinc -randkey changepw/kdc2.example.com
```

7. **このシステムのクロックをレルム内のほかのクロックと同期します。**

認証が成功するには、すべてのクロックが、krb5.conf ファイルの libdefaults セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、krb5.conf(4) のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

注記 - マスター KDC をクロック同期サーバーにすることはできません。

- **まだクロック同期サーバーが存在しない場合は、1 台構成したあとに、この手順を実行します。**
- **クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。**
  - **PTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Precision Time Protocol の管理」の手順に従います。**
  - **NTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Network Time Protocol の管理」の手順に従います。**

8. **KDC および kadmin サービスを有効にします。**

```
kdc1# svcadm enable -r network/security/krb5kdc
kdc1# svcadm enable -r network/security/kadmin
```

マスターサーバー上のチケット認可サービス鍵の置き換え

チケット認可サービス (TGS) 主体に DES 鍵しかない場合は、この鍵によって、チケット認可チケット (TGT) セッション鍵の暗号化タイプが DES に制限されます。KDC が、より強力な暗号化タイプをサポートするリリースに更新された場合は、TGS 主体がすべてのセッション鍵に対してより強力な暗号化を生成できるように、その主体の DES 鍵を置き換える必要があります。

この鍵はリモートで、またはマスターサーバー上で置き換えることができます。それには、changepw 権限が割り当てられている admin 主体である必要があります。

- いずれかの Kerberos システムから TGS サービス主体鍵を置き換えるには、kadmin コマンドを使用します。

```
kdc1 % /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
```

```
kadmin: cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM
Enter TGS key: xxxxxxxx
Enter new TGS key:      /** Type strong password **/
Re-enter TGS key to verify: xxxxxxxx
```

cpw は、change_password コマンドの別名です。-randkey オプションによって、新しいパスワードの入力を求められます。

- KDC マスターに root としてログオンしている場合は、kadmin.local コマンドを使用できます。新しいデータベースパスワードの入力を求められます。

```
kdc1# kadmin.local -q 'cpw -randkey krbtgt/EXAMPLE.COM@EXAMPLE.COM'
```

注記 - このパスワードを安全な場所に保存および保管してください。

LDAP ディレクトリサーバー上での KDC サーバーの構成

LDAP 上にマスター KDC サーバーとセカンダリサーバーを構成および構築するには、LDAP バックエンドと Kerberos KDC を作成してから、Kerberos と LDAP を相互に認識されるように構成する必要があります。このセクションでは、3 つの別々の LDAP バックエンド OpenLDAP、Oracle Unified Directory (OUD)、および Oracle Directory Server Enterprise Edition を構成し、それらを KDC に接続する方法を示します。さらに、LDAP の people オブジェクトクラスに Kerberos 属性を追加する方法、および LDAP ディレクトリサーバー上の Kerberos レルムを破棄する方法についても説明します。

表 5 タスクマップ: LDAP を使用するための Kerberos の構成

タスク	説明	参照先
OpenLDAP を使用するようにマスター KDC を手動で構成します。	LDAP バックエンドとして OpenLDAP サーバーを使用するように KDC を構成します。	94 ページの「OpenLDAP ディレクトリサーバー上でのマスター KDC の構成」
Oracle Unified Directory (OUD) を使用するようにマスター KDC を手動で構成します。	LDAP バックエンドとして OUD サーバーを使用するように KDC を構成します。	98 ページの「Oracle Unified Directory サーバー上でのマスター KDC の構成」
Oracle Directory Server Enterprise Edition (DSEE) を使用するようにマスター KDC と Kerberos クライアントを手動で構成します。	LDAP バックエンドとして Oracle DSEE サーバーを使用するように KDC と 2 番目のマスター KDC を構成します。	103 ページの「Oracle DSEE LDAP ディレクトリサーバー上でのマスター KDC の構成」
Kerberos 主体を LDAP ディレクトリサーバーに追加します。	Kerberos 属性を LDAP の people オブジェクトクラスに関連付けます。	110 ページの「Oracle DSEE サーバー上で Kerberos 以外のオブジェクトクラス型に Kerberos 主体の属性を混在させる方法」
LDAP ディレクトリサーバー上のレルムを破棄します。	レルムに関連付けられたデータをすべて削除します。	111 ページの「LDAP ディレクトリサーバー上で Kerberos レルムを破棄する方法」

OpenLDAP ディレクトリサーバー上でのマスター KDC の構成

同じサーバー上に KDC と OpenLDAP をインストールすると、パフォーマンスが向上します。

同じサーバー上に KDC と OpenLDAP を構成する際に必要となる主な手順は、次のとおりです。

1. OpenLDAP パッケージのインストール
2. パスへの OpenLDAP ユーティリティーの配置
3. OpenLDAP サーバーの構成と、そのディレクトリの保護
4. OpenLDAP デーモンの起動
5. OpenLDAP サーバーへの組織エントリの追加
6. KDC 構成ファイルへの OpenLDAP サーバーの追加
7. Kerberos データベースの作成
8. OpenLDAP サーバーへの KDC 役割および `kadmin` 役割の追加
9. Kerberos データベースキーの作成
10. マスター KDC のクロックとクロック同期サーバーとの同期
11. KDC および `kadmin` サービスの有効化

▼ OpenLDAP ディレクトリサーバー上でマスター KDC を構成する方法

この手順では、同じシステム上に KDC マスターと OpenLDAP サーバーを構成します。KDC では、OpenLDAP クライアントライブラリが使用され、あとで構成する Kerberos クライアントも使用されます。

始める前に DNS を使用するようにシステムが構成されていることを確認してください。この手順では、LDAP に OpenLDAP を使用します。詳細は、OpenLDAP のホームページを参照してください。

`root` に含まれています。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **`openldap` パッケージをインストールします。**

```
# pkg install library/openldap
```

2. **パスに OpenLDAP ユーティリティーを追加します。**

```
# export PATH="/usr/lib/openldap/bin:$PATH"
```

3. **構成ファイル `slapd.conf` を編集して、OpenLDAP サーバーの構成を開始します。**

ドメインを変更し、スキーマ、アクセス、およびインデックスの情報を追加します。

```
# pfedit /etc/openldap/slapd.conf
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/kerberos.schema

...

access to dn.base=""
    by * read

access to dn.base="cn=Subschema"
    by * read

access to attrs=userPassword,userPKCS12
    by self write
    by * auth

access to attrs=shadowLastChange
    by self write
    by * read

# Providing access to realm container
access to dn.subtree="cn=EXAMPLE.COM,cn=krbcontainer,dc=example,dc=com"
    by dn.exact="cn=kdc service,dc=example,dc=com" read
    by dn.exact="cn=kadmin service,dc=example,dc=com" write
    by * none

# Providing access to principals, if not underneath realm container
access to dn.subtree="ou=users,dc=example,dc=com"
    by dn.exact="cn=kdc service,dc=example,dc=com" read
    by dn.exact="cn=kadmin service,dc=example,dc=com" write
    by * none

access to *
    by * read

...

index   krbPrincipalName

...
```

4. **OpenLDAP ディレクトリ上でアクセス権を変更します。**

```
# chown -R openldap:openldap /var/openldap
# chown openldap:openldap /etc/openldap/slapd.conf
# mkdir /var/openldap/openldap-data
# chown openldap:openldap /var/openldap/openldap-data
```

5. **dn および changetype を削除し、`kerberos.ldif` ファイルに特別な行を追加してから、その行を `kerberos.schema` ファイルにコピーします。**

```
# pfedit /usr/share/lib/ldif/kerberos.ldif
# cp /usr/share/lib/ldif/kerberos.ldif /etc/openldap/schema/kerberos.schema
```

6. **ドメインソケットをサポートするための特定の引数を使用して、OpenLDAP デーモンを起動します。**

```
# /usr/lib/slapd -u openldap -g openldap -f /etc/openldap/slapd.conf \
   -h "ldapi:/// ldaps:///"
```

OpenLDAP デーモンは、ポート 389 を待機します。デフォルトでは、ドメインソケットの場所は /var/openldap/run/ldapi です。

7. **/etc/openldap/ldap.conf ファイルで、クライアントのドメインソケットの場所を指定します。**

```
# pfedit /etc/openldap/ldap.conf
URI     ldapi:/var/openldap/run/ldapi
```

8. **slapd デーモンが IPC を使用して、ドメインソケットを待機していることを確認します。**

```
# /usr/lib/openldap/bin/ldapsearch -Y EXTERNAL -D "cn=directory manager" -b "" \
   -s base '(objectclass=*)' namingContexts
```

9. **OpenLDAP サーバーに組織エントリを追加します。**

```
# pfedit entries.ldif
dn: dc=example,dc=com
objectClass: dcObject
objectClass: organization
o: example.com
dc: example

dn: ou=groups,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: groups

dn: ou=users,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: users

# ldapadd -D "cn=directory manager,dc=example,dc=com" -W -f entries.ldif
```

10. **Kerberos 構成ファイルに OpenLDAP サーバーを追加します。**

```
# pfedit /etc/krb5/krb5.conf
 [realms]
        EXAMPLE.COM = {
                kdc = krb1.example.com
                admin_server = krb1.example.com
                database_module = LDAP
        }

 [dbmodules]
        LDAP = {
                db_library = kldap
                ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
                ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
                ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
                ldap_cert_path = /var/ldap
                ldap_servers = ldapi:///
        }
  ...
```

11. Kerberos データベースに LDAP エントリを作成します。

```
# kdb5_ldap_util -D "cn=directory manager,dc=example,dc=com" create \
   -subtrees ou=users,dc=example,dc=com -r EXAMPLE.COM -s
```

詳細は、kdb5_ldap_util(1M) のマニュアルページを参照してください。

12. OpenLDAP 構成に Kerberos エントリを追加します。

a. 初期プロファイルを作成し、追加します。

```
# pfedit profile.ldif
dn: ou=profile,dc=example,dc=com
ou: profile
objectclass: top
objectclass: organizationalUnit

# ldapadd -D "cn=directory manager,dc=example,dc=com" -W -f profile.ldif
```

b. KDC 役割および `kadmin` 役割を作成し、追加します。

```
# pfedit kdc_roles.ldif
 dn: cn=kdc service,ou=profile,dc=example,dc=com
 cn: kdc service
 sn: kdc service
 objectclass: top
 objectclass: person
 userpassword: nnnnnnnn

 dn: cn=kadmin service,ou=profile,dc=example,dc=com
 cn: kadmin service
 sn: kadmin service
 objectclass: top
 objectclass: person
 userpassword: nnnnnnnn

# ldapadd -D "cn=directory manager,dc=example,dc=com" -W -f kdc_roles.ldif
```

13. KDC および `kadmin` サービスへの LDAP バインディング用の stash ファイルを作成します。

```
# kdb5_ldap_util -D "cn=directory manager,dc=example,dc=com" stashsrvpw \
   cn="kdc service,ou=profile,dc=example,dc=com"
# kdb5_ldap_util -D "cn=directory manager,dc=example,dc=com" stashsrvpw \
   cn="kadmin service,ou=profile,dc=example,dc=com"
```

14. このシステムのクロックをレルム内のほかのクロックと同期します。

認証が成功するには、すべてのクロックが、`krb5.conf` ファイルの `libdefaults` セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、krb5.conf(4) のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

注記 - マスター KDC をクロック同期サーバーにすることはできません。

- **まだクロック同期サーバーが存在しない場合は、1 台構成したあとに、この手順を実行します。**
- **クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。**
  - **PTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Precision Time Protocol の管理」の手順に従います。**
  - **NTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Network Time Protocol の管理」の手順に従います。**

15. **KDC および `kadmin` サービスを有効にします。**

```
# svcadm enable krb5kdc
# svcadm enable kadmin
```

Oracle Unified Directory サーバー上でのマスター KDC の構成

同じサーバー上に KDC と LDAP をインストールすると、パフォーマンスが向上します。

主な手順は次のとおりです。

1. OUD パッケージのインストール
2. OUD サーバーの構成
3. Kerberos 構成ファイルへの OUD サーバーの追加
4. KDC の鍵の作成と、OUD サーバーへの特権ポートの指定
5. OUD サーバー上での KDC 役割およびサービスの構成
6. OUD サーバーの証明書と鍵の作成およびインストール
7. テスト
8. マスター KDC のクロックとクロック同期サーバーとの同期

▼ Oracle Unified Directory LDAP ディレクトリサーバー上でマスター KDC を構成する方法

この手順では、同じシステム上に KDC マスターと Oracle Unified Directory (OUD) サーバーを構成します。KDC では、OpenLDAP クライアントライブラリが使用され、あとで構成する Kerberos クライアントも使用されます。

始める前に DNS を使用するようにシステムが構成されていることを確認してください。この手順では、LDAP に Oracle Unified Directory (OUD) を使用します。詳細は、『Oracle Fusion Middleware Oracle Unified Directory インストレーション・ガイド 11g リリース 2 (11.1.2)』を参照してください。

`root` に含まれています。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **OUD パッケージをダウンロードします。**

   Oracle Unified Directory の Web サイトの指示に従います。

2. **OUD LDAP サーバーを構成します。**

   この構成では、次のパラメータを使用します。

   - リスナーポート: `1389`
   - TLS ポート: `1636` (特権ポート)
   - 管理者コネクタポート: `4444`
   - パスワード: *nnnnnnnn*
   - 証明書: StartTLS と TLS
   - プロセス: `java -server -Dorg.opends.server.scriptName=sta...`

   ```
   # cd Oracle/Middleware/Oracle_OUD1
   # export JAVA_HOME=/usr/jdk/instances/jdk1.6.0
   # ./oud-setup
   ```

3. **LDAP サーバーが待機していることを確認します。**

   ```
   # ldapsearch -p 1389 -D "cn=directory manager" -h $HOSTNAME -b "" -s base objectclass='*'
   ```

4. **OUD 構成に初期プロファイルのエントリを追加します。**

   ```
   # pfedit profile.ldif
     dn: ou=profile,dc=example,dc=com
     ou: profile
     objectclass: top
     objectclass: organizationalUnit
   # ldapmodify -a -h $HOSTNAME -D "cn=directory manager" -f profile.ldif
   ```

5. **`kerberos.ldif` ファイル内のすべての属性タイプから改行を削除し、そのファイルを OUD 構成に追加します。**

```
# pfedit /usr/share/lib/ldif/kerberos.ldif
# ldapmodify -p 1389 -a -h $HOSTNAME -D "cn=directory manager" \
   -f /usr/share/lib/ldif/kerberos.ldif
```

6. **Kerberos 構成ファイルに OUD サーバーを追加します。**

```
# pfedit /etc/krb5/krb5.conf
 [realms]
        EXAMPLE.COM = {
                kdc = krb1.example.com
                admin_server = krb1.example.com
                database_module = LDAP
        }

 [dbmodules]
        LDAP = {
                db_library = kldap
                ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
                ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
                ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
                ldap_cert_path = /var/ldap
                ldap_servers = ldap://krb1:1389
        }
  ...
```

7. **KDC および `kadmin` サービスへの LDAP バインディング用の鍵および stash ファイルを作成します。**

```
# kdb5_ldap_util -D "cn=directory manager" create -P nnnnnnn -r EXAMPLE.COM -s
# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
# kdb5_ldap_util stashsrvpw "cn=kadmin service,ou=profile,dc=example,dc=com"
```

8. **特権ポートが使用されるように、Kerberos 構成ファイルの `ldap_servers` エントリを変更します。**

```
# pfedit /etc/krb5/krb5.conf
       ldap_servers = ldaps://krb1:1636
```

9. **OUD サーバーに Kerberos エントリを追加します。**

   a. **KDC 役割を作成し、追加します。**

```
# pfedit kdc_roles.ldif
 dn: cn=kdc service,ou=profile,dc=example,dc=com
 cn: kdc service
 sn: kdc service
 objectclass: top
 objectclass: person
 userpassword: nnnnnnnn

 dn: cn=kadmin service,ou=profile,dc=example,dc=com
 cn: kadmin service
 sn: kadmin service
 objectclass: top
 objectclass: person
 userpassword: nnnnnnnn

# ldapmodify -p 1389 -a -h $HOSTNAME -D "cn=directory manager" -f kdc_roles.ldif
```

b. 管理ユーザーを作成し、追加します。

```
# pfedit example.ldif
dn: dc=example,dc=com
changetype: modify
replace: aci
aci: (target ="ldap:///dc=example,dc=com")(targetattr !=
 "userPassword")(version 3.0;acl "Anonymous read-search access";
 allow (read, search, compare)(userdn = "ldap:///anyone");)
aci: (target="ldap:///dc=example,dc=com") (targetattr =
  "*")(version 3.0; acl "allow all Admin group"; allow(all) groupdn =
  "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)

dn: ou=Groups, dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: Groups

dn: cn=Directory Administrators, ou=Groups, dc=example,dc=com
cn: Directory Administrators
objectclass: top
objectclass: groupofuniquenames
ou: Groups
uniquemember: uid=kvaughan, ou=People, dc=example,dc=com
uniquemember: uid=rdaugherty, ou=People, dc=example,dc=com
uniquemember: uid=hmiller, ou=People, dc=example,dc=com

dn: ou=People, dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: People
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr =
 "userpassword || telephonenumber || facsimiletelephonenumber")(version 3.0;
 acl "Allow self entry modification";allow (write)(userdn = "ldap:///self");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Accounting)")(version 3.0;
 acl "Accounting Managers Group Permissions";allow (write) (groupdn =
  "ldap:///cn=Accounting Managers,ou=groups,dc=example,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Human Resources)")(version 3.0;
 acl "HR Group Permissions";allow (write)(groupdn = "ldap:///cn=HR Managers,
 ou=groups,dc=example,dc=com
 ");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn ||sn || uid")(targetfilter ="(ou=Product Testing)")(version 3.0;
 acl "QA Group Permissions";allow (write)(groupdn = "ldap:///cn=QA Managers,
 ou=groups,dc=example,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Product Development)")(version 3.0;
 acl "Engineering Group Permissions";allow (write)(groupdn = "ldap:///
 cn=PD Managers,ou=groups,dc=example,dc=com");)

dn: ou=Special Users,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: Special Users
description: Special Administrative Accounts

# ldapmodify -p 1389 -a -h $HOSTNAME -D "cn=directory manager" -f example.ldif
```

c. LDAP エントリに対応する ACL を作成し、追加します。

```
# pfedit kadmin.aci
## Set kadmin ACL for everything under krbcontainer.
dn: cn=krbcontainer,dc=example,dc=com
changetype: modify
replace: aci
aci: (target="ldap:///cn=krbcontainer,dc=example,dc=com") (targetattr="*")
 (version 3.0; acl "kadmin_ACL"; allow (all)
 userdn="ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)

## Set kadmin ACL for everything under the people subtree if there are
## mix-in entries for krb princs:
dn: ou=people,dc=example,dc=com
changetype: modify
replace: aci
aci: (target="ldap:///ou=people,dc=example,dc=com") (targetattr="*")
 (version 3.0; acl "kadmin_ACL"; allow (all)
 userdn="ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)

# ldapmodify -h $HOSTNAME -D "cn=directory manager" -f kadmin.aci
```

10. OUD サーバーの TLS 証明書を生成し、格納します。

この一連のコマンドによって、鍵マネージャープロバイダ、信頼マネージャープロバイダ、および接続ハンドラも作成されます。

```
# export LDAPHOME=~/Oracle/Middleware/asinst_8/OUD
# export LDAPHOME=$PWD
# export LDAP_SERVER_DN=krb1.example.com
# export STORE_PASSWD=xxxxxxxx
# export LDAP_BINDPWF=$LDAPHOME/config/keystore.pin
# export LDAP_ADMIN_PORT=4444
# export LDAP_BINDDN="cn=directory manager"
# export LDAP_SERVER=krb1.example.com
# rm $LDAPHOME/config/keystore
# rm $LDAPHOME/config/truststore
# echo $STORE_PASSWD > LDAP_BINDPWF
# keytool -genkeypair -alias server-cert -keyalg rsa \
    -dname "CN=$LDAP_SERVER_DN" -keystore $LDAPHOME/config/keystore \
    -storepass $STORE_PASSWD -keypass $STORE_PASSWD
# keytool -selfcert -alias server-cert -validity 1825 \
    -keystore $LDAPHOME/config/keystore -storetype JKS -storepass $STORE_PASSWD
# keytool -list -alias server-cert -keystore $LDAPHOME/config/keystore \
    -storepass $STORE_PASSWD
# keytool -exportcert -alias server-cert -file $LDAPHOME/config/server-cert.txt \
    -rfc -keystore $LDAPHOME/config/keystore -storepass $STORE_PASSWD
# cp $LDAPHOME/config/server-cert.txt /var/ldap/certdb.pem
```

11. 鍵マネージャープロバイダ、信頼マネージャープロバイダ、および接続ハンドラを有効にします。

```
# dsconfig -X -n -h $LDAP_SERVER -p $LDAP_ADMIN_PORT -D "$LDAP_BINDDN" \
  -j $LDAP_BINDPWF set-connection-handler-prop --handler-name "LDAPS Connection Handler" \
  --set key-manager-provider:JKS --set trust-manager-provider:JKS \
  --set listen-port:1636 --set enabled:true
# bin/stop-ds
```

12. (オプション) SSL LDAP クエリーを使用して構成を確認します。

```
# /usr/lib/openldap/bin/ldapsearch -v -x -D "$LDAP_BINDDN" -w $LDAP_BINDPW \
    -H ldaps://$LDAP_SERVER_DN:1636 -b "" -s base objectclass='*'
```

13. **このシステムのクロックをレルム内のほかのクロックと同期します。**

    認証が成功するには、すべてのクロックが、`krb5.conf` ファイルの `libdefaults` セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、`krb5.conf(4)` のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

    注記 - マスター KDC をクロック同期サーバーにすることはできません。

    - **まだクロック同期サーバーが存在しない場合は、1 台構成したあとに、この手順を実行します。**
    - **クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。**
        - **PTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Precision Time Protocol の管理」の手順に従います。**
        - **NTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Network Time Protocol の管理」の手順に従います。**

Oracle DSEE LDAP ディレクトリサーバー上でのマスター KDC の構成

同じサーバー上に KDC と LDAP をインストールすると、パフォーマンスが向上します。このセクションでは、マルチマスター構成と呼ばれる 2 つのマスター KDC を作成します。各マスター KDC は LDAP サーバー上に存在します。

主な手順は次のとおりです。

1. 証明書、プロファイル、および Kerberos スキーマを含む 1 番目のマスター Oracle DSEE サーバーの構成
2. 1 番目のマスター KDC の構成
3. 2 番目の Oracle DSEE サーバーの構成
4. 2 番目のマスター KDC の構成
5. マスター KDC のクロックとクロック同期サーバーとの同期
6. Kerberos サービスの有効化

▼ OpenLDAP クライアントの Oracle DSEE LDAP ディレクトリサーバー上でマスター KDC を構成する方法

この手順では、2 つの KDC マスターと 2 つの LDAP サーバーを構成します。1 番目の LDAP サーバーと 1 番目の KDC マスターは同じシステム上に存在し、2 番目の LDAP サーバーと 2 番目の KDC マスターは別々のシステム上に存在します。

この手順では、次の構成パラメータを使用します。

- LDAP インスタンス = `/local/dsInst`
- LDAP ポート = `389 (636: セキュリティー保護済み)`
- 1 番目の KDC/LDAP マスターサーバー = `kdc1.example.com`
- 2 番目の KDC/LDAP マスターサーバー = `kdc2.example.com`
- Kerberos レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- 1 番目のマスターサーバー上の `admin` 主体 = `dav/admin`
- 2 番目のマスターサーバー上の `admin` 主体 = `tester/admin`

始める前に システムは、DNS を使用するように構成されています。同じサーバー上に LDAP と KDC をインストールすると、パフォーマンスが向上します。この手順では、LDAP に Oracle Directory Server Enterprise Edition (DSEE) を使用します。詳細は、『Oracle Fusion Middleware Oracle Directory Server Enterprise Edition インストレーション・ガイド 11g リリース 1 (11.1.1.7.0)』を参照してください。

`root` に含まれています。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **1 番目の Oracle DSEE LDAP サーバーを構成します。**

   LDAP サーバーでは TLS 証明書が使用されます。

   a. **サーバーインスタンスを作成し、その証明書を作成して追加します。**

```
# dsadm create -p 389 -P 636 /local/dsInst
# dsadm start /local/dsInst
# dsconf create-suffix -p 389 -e dc=example,dc=com
# dsconf import -p 389 -e /opt/dsee7/resources/ldif/Example.ldif dc=example,dc=com
# ldapsearch -b "dc=example,dc=com" -s base '(objectclass=*)'
# dsadm show-cert -F der /local/dsInst defaultCert > /tmp/defaultCert.cert.der
# pktool setpin keystore=nss dir=/var/ldap
# chmod a+r /var/ldap/*.db
# pktool import keystore=nss objtype=cert trust="CT" \
   infile=/tmp/defaultCert.cert.der label=defaultCert dir=/var/ldap
# certutil -L -d /var/ldap -n defaultCert -a > /var/ldap/certdb.pem
```

   b. **`ldap.conf` ファイルに組織の情報を追加します。**

```
# pfedit /etc/openldap/ldap.conf
  ...
```

```
#BASE   dc=example,dc=com
#URI    ldap://ldap.example.com ldap://ldap-master.example.com:666

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never

TLS_CACERT      /var/ldap/certdb.pem
...
```

c. 構成された Oracle DSEE サーバーが動作することを確認します。

```
# ldapsearch -Z -P /var/ldap -D "cn=directory manager" \
   -h $HOSTNAME -b "" -s base objectclass='*'
```

d. 初期プロファイルを追加します。

```
# pfedit profile.ldif
  dn: ou=profile,dc=example,dc=com
  ou: profile
  objectclass: top
  objectclass: organizationalUnit
# ldapmodify -a -h $HOSTNAME -D "cn=directory manager" -f profile.ldif
```

e. DSEE サーバーに `kerberos.ldif` ファイルを追加します。

```
# ldapmodify -h $HOSTNAME -D "cn=directory manager" \
   -f /usr/share/lib/ldif/kerberos.ldif
```

2. 1 番目のマスター KDC および Oracle DSEE サーバーを構成します。

a. Kerberos 構成ファイルに Oracle DSEE サーバーを追加します。

```
# pfedit /etc/krb5/krb5.conf
 [realms]
        EXAMPLE.COM = {
                kdc = kdc1.example.com
                admin_server = kdc1.example.com
                database_module = LDAP
        }

 [dbmodules]
        LDAP = {
                db_library = kldap
                ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
                ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
                ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
                ldap_cert_path = /var/ldap
                ldap_servers = ldaps://kdc1
        }
  ...
```

b. マスター KDC およびその鍵を作成します。

```
# kdb5_ldap_util -D "cn=directory manager" create -P nnnnnnnn -r EXAMPLE.COM -s
# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
# kdb5_ldap_util stashsrvpw "cn=kadmin service,ou=profile,dc=example,dc=com"
```

c. KDC および `kadmin` 役割を作成し、Oracle DSEE サーバーに追加します。

```
# pfedit kdc_roles.ldif
 dn: cn=kdc service,ou=profile,dc=example,dc=com
 cn: kdc service
 sn: kdc service
 objectclass: top
 objectclass: person
 userpassword: nnnnnnnn

 dn: cn=kadmin service,ou=profile,dc=example,dc=com
 cn: kadmin service
 sn: kadmin service
 objectclass: top
 objectclass: person
 userpassword: nnnnnnnn

# ldapmodify -a -h $HOSTNAME -D "cn=directory manager" -f kdc_roles.ldif
```

d. Kerberos 管理者の ACL を作成し、Oracle DSEE サーバーに追加します。

```
# pfedit kadmin.aci
 ## Set kadmin ACL for everything under krbcontainer.
 dn: cn=krbcontainer,dc=example,dc=com
 changetype: modify
 replace: aci
 aci: (target="ldap:///cn=krbcontainer,dc=example,dc=com")
  (targetattr="krb*")(version 3.0; acl kadmin_ACL; allow (all)
  userdn = "ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)

 ## Set kadmin ACL for everything under the people subtree if there are
 ## mix-in entries for krb princs:
 dn: ou=people,dc=example,dc=com
 changetype: modify
 replace: aci
 aci: (target="ldap:///ou=people,dc=example,dc=com")(targetattr="*")
  (version 3.0; acl kadmin_ACL; allow (all)
  userdn = "ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)

# ldapmodify -h $HOSTNAME -D "cn=directory manager" -f kadmin.aci
```

e. KDC 構成ファイルに Kerberos 構成ファイルの場所を追加します。

```
# pfedit /etc/krb5/kdc.conf
 ...
 [realms]
        EXAMPLE.COM = {
                profile = /etc/krb5/krb5.conf
 ...
```

f. このシステムのクロックをレルム内のほかのクロックと同期します。

認証が成功するには、すべてのクロックが、`krb5.conf` ファイルの `libdefaults` セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、`krb5.conf(4)` のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

注記 - マスター KDC をクロック同期サーバーにすることはできません。

- まだクロック同期サーバーが存在しない場合は、1 台構成したあとに、この手順を実行します。
- クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。
  - PTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Precision Time Protocol の管理」の手順に従います。
  - NTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Network Time Protocol の管理」の手順に従います。

g. KDC および `kadmin` サービスを有効にします。

```
# svcadm enable krb5kdc
# svcadm enable kadmin
```

3. 2 番目の LDAP サーバーを構成します。

2 番目の LDAP サーバー `kdc2` を構成するときに、1 番目の LDAP サーバー `kdc1` を更新します。

a. 1 番目の LDAP サーバーのレプリカを作成します。

```
kdc1# dsconf enable-repl -h $HOSTNAME -p 389 -d 1 master dc=example,dc=com
```

b. レプリカを構成し、有効にします。

```
kdc2# mkdir -p /local
kdc2# dsadm create -p 389 -P 636 /local/dsInst
kdc2# dsadm start /local/dsInst
kdc2# dsconf create-suffix -p 389 -e dc=example,dc=com
kdc2# dsconf enable-repl -h $HOSTNAME -p 389 -d 2 master dc=example,dc=com
```

c. 相互に認識されるように LDAP サーバーを有効にします。

```
kdc1# dsconf create-repl-agmt -h $HOSTNAME -p 389 dc=example,dc=com kdc2:389
kdc2# dsconf create-repl-agmt -h $HOSTNAME -p 389 dc=example,dc=com kdc1:389
```

4. 1 番目の LDAP サーバーが 2 番目の LDAP サーバーを変更できないことを確認します。

適切に構成された LDAP サーバー上では、2 番目のサーバーのクエリーによって `NOT OK` のステータスがレポートされます。

```
kdc1# dsconf show-repl-agmt-status -h $HOSTNAME -p 389 dc=example,dc=com kdc2:389
kdc1# dsconf accord-repl-agmt -h $HOSTNAME -p 389 dc=example,dc=com kdc2:389
kdc1# dsconf init-repl-dest -h $HOSTNAME -p 389 dc=example,dc=com kdc2:389
```

5. **LDAP バインディング用の stash ファイルを作成し、マスター KDC の stash ファイルを作成します。**

```
kdc1# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
kdc1# kdb5_util stash
```

6. **2 番目の KDC マスターを構成します。**
   管理主体は `tester/admin` です。その他のパラメータは同じ設定のままです。
   2 番目の KDC 上で LDAP TLS バインディングを構成したあとに、`/etc/krb5/krb5.conf` ファイルを更新します。

注記 - KDC と LDAP サーバーが別々のシステム上に存在する場合は、`krb5.conf` ファイル内の `ldap_servers` キーワードに LDAP サーバー名が一覧表示される必要があります。

---

   a. **2 番目の KDC マスター上で、Kerberos 構成ファイルを編集します。**

```
kdc2# pfedit /etc/krb5/krb5.conf
 [realms]
        EXAMPLE.COM = {
                kdc = kdc2.example.com
                kdc = kdc1.example.com
                admin_server = kdc1.example.com
                admin_server = kdc2.example.com
                database_module = LDAP
        }

 [dbmodules]
        LDAP = {
                db_library = kldap
                ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
                ldap_kdc_dn = "cn=kdc service,ou=profile,dc=us,dc=example,dc=com"
                ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
                ldap_cert_path = /var/ldap
                ldap_servers = ldaps://kdc2 ldaps://kdc1
        }
 ...
```

   b. **KDC および `kadmin` サービスへの LDAP バインディング用の stash ファイルを作成し、マスター KDC の stash ファイルを作成します。**

```
kdc2# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
kdc2# kdb5_ldap_util stashsrvpw "cn=kadmin service,ou=profile,dc=us,dc=oracle,dc=com"
kdc2# kdb5_util stash
```

   c. **KDC 構成ファイルに Kerberos 構成ファイルの場所を追加します。**

```
kdc2# pfedit /etc/krb5/kdc.conf
 ...
 [realms]
```

```
        EXAMPLE.COM = {
                profile = /etc/krb5/krb5.conf
...
```

d. 2 番目の KDC マスターの主体に対応する鍵を作成します。

```
kdc2# kadmin -p tester/admin
kadmin: addprinc -randkey -allow_tgs_req kadmin/kdc2.example.com
kadmin: modprinc -requires_preauth kadmin/kdc2.example.com
kadmin: addprinc -randkey -allow_tgs_req +password_changing_service changepw/kdc2.example.com
kadmin: modprinc -requires_preauth changepw/kdc2.example.com
```

7. このシステムのクロックをレルム内のほかのクロックと同期します。

認証が成功するには、すべてのクロックが、`krb5.conf` ファイルの `libdefaults` セクションで定義されているデフォルトの時間内に収まっている必要があります。詳細は、`krb5.conf(4)` のマニュアルページおよび142 ページの「KDC と Kerberos クライアントのクロックの同期化」を参照してください。

注記 - マスター KDC をクロック同期サーバーにすることはできません。

- **まだクロック同期サーバーが存在しない場合は、1 台構成したあとに、この手順を実行します。**
- **クロック同期サーバーが存在する場合は、PTP または NTP の手順に従って、このシステムをそのサーバーのクライアントにします。**
  - **PTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Precision Time Protocol の管理」の手順に従います。**
  - **NTP クライアントを構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の「Network Time Protocol の管理」の手順に従います。**

8. KDC および `kadmin` サービスを有効にします。

```
kdc2# svcadm enable krb5kdc
kdc2# svcadm enable kadmin
```

▼ Oracle DSEE サーバー上で Kerberos 以外のオブジェクトクラス型に Kerberos 主体の属性を混在させる方法

この手順では、krbprincipalaux、krbTicketPolicyAux、および krbPrincipalName 属性が people オブジェクトクラスに関連付けられます。

この手順では、次の構成パラメータを使用します。

- Oracle DSEE Server = dsserver.example.com
- ユーザー主体 = mre@EXAMPLE.COM

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **people オブジェクトクラスの各エントリを用意します。**

   Oracle DSEE サーバー上で、この手順をエントリごとに繰り返します。

```
cat << EOF | ldapmodify -h dsserver.example.com -D "cn=directory manager"
dn: uid=mre,ou=people,dc=example,dc=com
changetype: modify
objectClass: krbprincipalaux
objectClass: krbTicketPolicyAux
krbPrincipalName: mre@EXAMPLE.COM
EOF
```

2. **サブツリー属性をレルムコンテナに追加します。**

   この例では、デフォルトの EXAMPLE.COM コンテナだけでなく、ou=people,dc=example,dc=com コンテナでも主体エントリを検索できるようにします。

```
# kdb5_ldap_util -D "cn=directory manager" modify \
   -subtrees 'ou=people,dc=example,dc=com' -r EXAMPLE.COM
```

3. **(オプション) KDC レコードが DB2 内に格納されている場合は、DB2 エントリを移行します。**

   a. **DB2 エントリをダンプします。**

```
# kdb5_util dump > dumpfile
```

   b. **データベースを LDAP サーバーにロードします。**

```
# kdb5_util load -update dumpfile
```

4. **(オプション) 主体属性を KDC に追加します。**

```
# kadmin.local -q 'addprinc mre'
```

▼ LDAP ディレクトリサーバー上で Kerberos レルムを破棄する方法

この手順は、別の LDAP ディレクトリサーバーがレルムを処理している場合に使用してください。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

- **レルムを破棄します。**

```
# kdb5_ldap_util -D "cn=directory manager" destroy
```

Kerberos クライアントの構成

Kerberos クライアントには、Kerberos サービスを使用する必要のある、KDC サーバーではないネットワーク上のすべてのホストが含まれます。このセクションでは、Kerberos クライアントをインストールするための手順、および `root` 認証を使用して NFS ファイルシステムをマウントする方法について説明します。

クライアントの構成オプションはサーバーオプションと同様であり、さらに Automated Installer (AI) が追加されています。

- AI – 複数の Kerberos クライアントの迅速かつ容易なインストールに推奨されます。
- 自動 – スクリプトに推奨されます
- 対話型 – ほとんどのインストールにはこれで十分です
- 手動 – より複雑なインストールに必要です

次のタスクマップは、このセクションで説明するタスクの一覧です。

表 6 タスクマップ: Kerberos クライアントの構成

タスク	説明	参照先
Automated Installer (AI) を使用してクライアントをインストールします。	Kerberos クライアントをシステムのインストール中に構成する場合に適しています。	『Oracle Solaris 11.3 システムのインストール』 の 「AI を使用して Kerberos クライアントを構成する方法」
同様の Kerberos クライアントのためにインストールプロファイルを作成します。	再利用可能なクライアントのインストールプロファイルを作成します。	112 ページの「Kerberos クライアントのインストールプロファイルの作成方法」

タスク	説明	参照先
スクリプトを使用してクライアントをインストールします。	各クライアントのインストールパラメータが同じ場合に適しています。	113 ページの「Kerberos クライアントを自動的に構成する方法」
プロンプトに応答することによってクライアントをインストールします。	数個のインストールパラメータしか変更する必要がない場合に適しています。	114 ページの「Kerberos クライアントを対話的に構成する方法」
クライアントを手動でインストールします。	各クライアントのインストールに固有のインストールパラメータが必要な場合に適しています。	118 ページの「Kerberos クライアントを手動で構成する方法」
Kerberos クライアントを Active Directory サーバーに参加させます。	自動的に Active Directory サーバーの Kerberos クライアントをインストールします。	117 ページの「Kerberos クライアントを Active Directory サーバーに参加させる方法」
クライアントのチケット認可チケット (TGT) を発行した KDC の確認を無効にします。	Kerberos クライアントがローカルの keytab ファイル内に格納されたホスト主体を持っていない場合の KDC の検証を効率化します。	123 ページの「チケット認可チケットの確認の無効化」
クライアントが `root` ユーザーとして NFS ファイルシステムにアクセスできるようにします。	クライアントが `root` アクセスを使用して NFS ファイルシステムをマウントできるようにします。また、cron ジョブを実行できるように、クライアントが NFS ファイルシステムにアクセスできるようにします。	124 ページの「Kerberos によって保護された NFS ファイルシステムに `root` ユーザーとしてアクセスする方法」

▼ Kerberos クライアントのインストールプロファイルの作成方法

この手順は、Kerberos クライアントをインストールする際に使用される kclient プロファイルを作成します。このプロファイルを使用することにより、入力ミスの可能性を減らします。また、プロファイルを使用すると、対話型のプロセスと比べて、ユーザーの介入も減ります。

注記 - 完全に構成された Kerberos クライアントとして初期ブートするシステムを作成するには、『Oracle Solaris 11.3 システムのインストール』 の 「AI を使用して Kerberos クライアントを構成する方法」を参照してください。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **`kclient` インストールプロファイルを作成します。**

   サンプルの `kclient` プロファイルを次に示します。

```
client# pfedit kcprofile
REALM EXAMPLE.COM
KDC kdc1.example.com
ADMIN clntconfig
FILEPATH /net/denver.example.com/export/install/krb5.conf
NFS 1
```

```
DNSLOOKUP none
```

2. **ファイルを保護して、ほかのクライアントが使用できるように格納します。**

```
client# cp kcprofile /net/denver.example.com/export/install
denver# chown root kcprofile; chmod 644 kcprofile
```

▼ Kerberos クライアントを自動的に構成する方法

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **kclient プロファイルを作成します。**

   112 ページの「Kerberos クライアントのインストールプロファイルの作成方法」のインストールプロファイルを使用します。

2. **プロファイル引数を指定して `kclient` コマンドを実行します。**

   このプロセスを完了するには、`clntconfig` 主体のパスワードを指定する必要があります。このパスワードは、75 ページの「KDC サーバーの構成」でマスター KDC を構成するときに作成しました。詳細は、`kclient(1M)` のマニュアルページを参照してください。

```
client# /usr/sbin/kclient -p /net/denver.example.com/export/install/kcprofile

Starting client setup
-----------------------------------------------------

kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM: xxxxxxxx

nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Copied /net/denver.example.com/export/install/krb5.conf.

-----------------------------------------------------
Setup COMPLETE.

client#
```

3. **このシステムのクロックをレルム内のほかのクロックと同期します。**

   認証が成功するには、すべてのクロックが、`krb5.conf` ファイルの `libdefaults` セクションで定義されているデフォルトの時間内に収まっている必要があります。

- **PTP を構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の 「Precision Time Protocol の管理」の手順に従います。**
- **NTP を構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の 「Network Time Protocol の管理」の手順に従います。**

例 7 インストールプロファイルを使用した Kerberos クライアントの構成

次の例では、`kcprofile` クライアントプロファイルと 2 つのコマンド行オーバーライドを使用してクライアントを構成します。

```
# /usr/sbin/kclient -p /net/denver.example.com/export/install/kcprofile \
-d dns_fallback -k kdc2.example.com

Starting client setup
-------------------------------------------------

kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:  xxxxxxxx

nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Copied /net/denver.example.com/export/install/krb5.conf.

-------------------------------------------------
Setup COMPLETE.

client#
```

▼ Kerberos クライアントを対話的に構成する方法

この手順では、`kclient` インストールユーティリティーをインストールプロファイルなしで使用します。このクライアントを Active Directory サーバーに参加させる場合は、117 ページの「Kerberos クライアントを Active Directory サーバーに参加させる方法」に進みます。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **`kclient` コマンドを引数なしで実行します。**

   ```
   client# /usr/sbin/kclient
   ```

   このスクリプトによって、次の情報の入力を求められます。

   - Kerberos レルム名
   - KDC マスターホスト名
   - KDC スレーブホスト名
   - ローカルレルムにマップするドメイン
   - Kerberos 認証に使用する PAM サービス名およびオプション

   詳細は、`kclient(1M)` のマニュアルページを参照してください。

2. **KDC サーバーで Oracle Solaris リリースが実行されていない場合は、y と答え、KDC を実行しているサーバーのタイプを定義します。**

   使用可能なサーバーのリストについては、`kclient(1M)` のマニュアルページの `-T` オプションを参照してください。

3. **Kerberos 検索に DNS を使用する場合は、y と答え、使用する DNS 検索オプションを示します。**

   有効なオプションは、`dns_lookup_kdc`、`dns_lookup_realm`、および `dns_fallback` です。これらの値の詳細は、`krb5.conf(4)` のマニュアルページを参照してください。

4. **Kerberos レルムの名前とマスター KDC のホスト名を定義します。**

   この情報は、`/etc/krb5/krb5.conf` 構成ファイルに追加されます。

5. **スレーブ KDC がレルム内に存在する場合は、y と答え、スレーブ KDC のホスト名を指定します。**

   この情報は、クライアントの構成ファイルに追加の KDC エントリを作成するために使用されます。

6. **サービス鍵またはホスト鍵が必要な場合は、y と答えます。**

   通常、クライアントシステムが Kerberos サービスをホストしていないかぎり、サービス鍵またはホスト鍵は必要ありません。

7. **クライアントがクラスタのメンバーである場合は、y と答え、そのクラスタの論理名を指定します。**

   この論理ホスト名はサービス鍵の作成時に使用されます。これは、クラスタから Kerberos サービスをホストするときに必要です。

8. **現在のレルムにマップするドメインまたはホストをすべて識別します。**

   このマッピングにより、ほかのドメインをそのクライアントのデフォルトレルムに含めることができます。

9. **クライアントが Kerberos NFS を使用するかどうかを指定します。**
   クライアントが Kerberos を使用して NFS サービスをホストする場合は、NFS サービス鍵を作成する必要があります。

10. **新しい PAM ポリシーを作成する必要があるかどうかを示します。**
    どの PAM サービスが認証に Kerberos を使用するかを設定するには、サービス名と Kerberos 認証の使用方法を示すフラグを指定します。有効なフラグオプションは次のとおりです。

    - `first` – 最初に Kerberos 認証を使用し、Kerberos 認証が失敗した場合にのみ UNIX を使用します
    - `only` – Kerberos 認証のみを使用します
    - `optional` – オプションで、Kerberos 認証を使用します

    Kerberos のために提供される PAM サービスについては、`/etc/security/pam_policy` 内のファイルを確認してください。

11. **マスターの `/etc/krb5/krb5.conf` ファイルをコピーするかどうかを指定します。**
    このオプションにより、`kclient` の引数が十分でないときに特定の構成情報を使用できるようになります。

例 8 `kclient` スクリプトの実行例

```
...
Starting client setup
--------------------------------------------------

Is this a client of a non-Solaris KDC ? [y/n]: n
No action performed.
Do you want to use DNS for kerberos lookups ? [y/n]: n
No action performed.
Enter the Kerberos realm: EXAMPLE.COM
Specify the KDC host name for the above realm: kdc1.example.com

Note, this system and the KDC's time must be within 5 minutes of each other for
Kerberos to function. Both systems should run some form of time synchronization
system like Network Time Protocol (NTP).
Do you have any slave KDC(s) ? [y/n]: y
Enter a comma-separated list of slave KDC host names: kdc2.example.com

Will this client need service keys ? [y/n]: n
No action performed.
Is this client a member of a cluster that uses a logical host name ? [y/n]: n
No action performed.
Do you have multiple domains/hosts to map to realm ? [y/n]: y
Enter a comma-separated list of domain/hosts to map to the default realm: corphdqtrs.example.
com, \
example.com

Setting up /etc/krb5/krb5.conf.

Do you plan on doing Kerberized nfs ? [y/n]: y
Do you want to update /etc/pam.conf ? [y/n]: y
```

```
Enter a comma-separated list of PAM service names in the following format:
service:{first|only|optional}: xscreensaver:first
Configuring /etc/pam.conf.

Do you want to copy over the master krb5.conf file ? [y/n]: n
No action performed.

-------------------------------------------------
Setup COMPLETE.
```

▼ Kerberos クライアントを Active Directory サーバーに参加させる方法

この手順では、`kclient` コマンドをインストールプロファイルなしで使用します。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **(オプション) クライアントの DNS リソースレコードの作成を有効にします。**

```
client# sharectl set -p ddns_enable=true smb
```

2. **`kclient` コマンドを実行します。**

次の出力は、`kclient` コマンドを実行してクライアントを AD ドメイン `EXAMPLE.COM` に参加させたときの出力例を示しています。

`-T` オプションは、KDC サーバータイプ (この場合は、Microsoft Active Directory (AD) サーバータイプ) を選択します。デフォルトでは、AD サーバーの Administrator 主体のパスワードを指定する必要があります。

```
client# /usr/sbin/kclient -T ms_ad
Starting client setup
-------------------------------------------------

Attempting to join 'CLIENT' to the 'EXAMPLE.COM' domain.
Password for Administrator@EXAMPLE.COM: xxxxxxxx
Forest name found: example.com
Looking for local KDCs, DCs and global catalog servers (SVR RRs).

Setting up /etc/krb5/krb5.conf

Creating the machine account in AD via LDAP.
-------------------------------------------------
Setup COMPLETE.
#
```

詳細は、`kclient(1M)` のマニュアルページを参照してください。

注記 - Active Directory (AD) でクライアントのサイトおよびサービスを自動的に検出するには、Kerberos クライアントと同じローカルサブネット上で AD サーバーを構成する必要があります。

▼ Kerberos クライアントを手動で構成する方法

この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- マスター KDC = `kdc1.example.com`
- スレーブ KDC = `kdc2.example.com`
- NFS サーバー = `denver.example.com`
- クライアント = `client.example.com`
- `admin` 主体 = `kws/admin`
- ユーザー主体 = `mre`
- オンラインヘルプ URL = `http://docs.oracle.com/cd/E23824_01/html/821-1456/aadmin-23.html`

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **Kerberos 構成ファイル `krb5.conf` を編集します。**

   Kerberos 構成ファイル内のレルム名とサーバー名を変更します。

```
kdc1# pfedit /etc/krb5/krb5.conf
[libdefaults]
default_realm = EXAMPLE.COM

[realms]
     EXAMPLE.COM = {
        kdc = kdc1.example.com
        kdc = kdc2.example.com
        admin_server = kdc1.example.com
     }

[domain_realm]
     .example.com = EXAMPLE.COM
     #
     # if the domain name and realm name are equivalent,
     # this entry is not needed
     #
[logging]
     default = FILE:/var/krb5/kdc.log
     kdc = FILE:/var/krb5/kdc.log
```

注記 - 古い Kerberos システムと通信する必要がある場合は、暗号化タイプの制限が必要になることがあります。暗号化タイプの制限に関する問題については、56 ページの「Kerberos 暗号化タイプ」を参照してください。

2. **(オプション) KDC の検出に使用されるプロセスを変更します。**

   デフォルトでは、Kerberos レルムから KDC へのマッピングは次の順番で決められます。

   - `krb5.conf` 内の `realms` セクションでの定義
   - DNS 内の SRV レコードの検索

   この動作は、`krb5.conf` ファイルの `libdefaults` セクションに `dns_lookup_kdc` または `dns_fallback` を追加することによって変更できます。詳細は、`krb5.conf(4)` を参照してください。常にリフェラルが最初に試行されます。

3. **(オプション) ホストのレルムの決定に使用されるプロセスを変更します。**

   デフォルトでは、ホストからレルムへのマッピングは次の順番で決められます。

   - KDC がリフェラルをサポートしている場合は、KDC からクライアントに、ホストが属しているレルムが通知されることがあります。
   - `krb5.conf` ファイル内の `domain_realm` の定義。
   - ホストの DNS ドメイン名。
   - デフォルトレルム。

   この動作は、`krb5.conf` ファイルの `libdefaults` セクションに `dns_lookup_kdc` または `dns_fallback` を追加することによって変更できます。詳細は、`krb5.conf(4)` のマニュアルページを参照してください。常にリフェラルが最初に試行されます。

4. **このシステムのクロックをレルム内のほかのクロックと同期します。**

   認証が成功するには、すべてのクロックが、`krb5.conf` ファイルの `libdefaults` セクションで定義されているデフォルトの時間内に収まっている必要があります。

   - **PTP を構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の 「Precision Time Protocol の管理」の手順に従います。**
   - **NTP を構成するには、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』 の 「Network Time Protocol の管理」の手順に従います。**

5. **Kerberos 主体を作成します。**

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
```

```
kadmin:
```

詳細は、kadmin(1M) のマニュアルページを参照してください。

a. (オプション) ユーザー主体が存在しない場合は、ユーザー主体を作成します。

このホストに関連付けられているユーザーに主体が割り当てられていない場合にのみ、ユーザー主体を作成してください。

```
kadmin: addprinc mre
Enter password for principal mre@EXAMPLE.COM:     /** Type strong password **/
Re-enter password for principal mre@EXAMPLE.COM: xxxxxxxx
kadmin:
```

b. (オプション) root 主体を作成し、その主体をサーバーのキータブファイルに追加します。

注記 - クライアントが NFS マウントしたリモートファイルシステムへの root アクセスを必要としない場合は、この手順をスキップできます。

---

cron ジョブを root として実行する場合など、非対話的な root アクセスが必要な場合は、この手順を実行します。

root 主体は、2 つのコンポーネントから成る主体にしてください。レルム全体にわたる root 主体が作成されないようにするために、2 番目のコンポーネントは Kerberos クライアントシステムのホスト名にしてください。主体のインスタンスがホスト名である場合、FQDN は、ネームサービスでのドメイン名の大文字小文字には関係なく小文字で指定する必要があります。

```
kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES
 cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. host 主体を作成し、その主体をサーバーのキータブファイルに追加します。

host 主体は、認証を提供するためにリモートアクセスサービスによって使用されます。keytab ファイル内にまだ資格が存在しない場合は、この主体により、root が資格を取得できるようになります。

```
kadmin: addprinc -randkey host/denver.example.com
Principal "host/denver.example.com@EXAMPLE.COM" created.
kadmin: ktadd host/denver.example.com
Entry for principal host/denver.example.com with kvno 3, encryption type AES-256 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
```

```
Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES
 cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

d. (オプション) サーバーの NFS サービス主体をサーバーのキータブファイルに追加し、`kadmin` を終了します。

この手順は、クライアントが Kerberos 認証を使用して NFS ファイルシステムにアクセスする必要がある場合にのみ必要です。

```
kadmin: ktadd nfs/denver.example.com
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

6. (オプション) NFS での Kerberos を有効にします。

a. `/etc/nfssec.conf` ファイル内の Kerberos セキュリティーモードを有効にします。

`/etc/nfssec.conf` ファイルで、Kerberos セキュリティーモードをコメントアウトしている「#」を削除します。

```
# pfedit /etc/nfssec.conf
.
.
#
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
```

b. DNS を有効にします。

`svc:/network/dns/client:default` サービスが有効でない場合は、それを有効にします。詳細は、`resolv.conf(4)` のマニュアルページを参照してください。

```
# svcadm enable network/dns/client:default
```

c. gss サービスを再起動します。

```
# svcadm restart network/rpc/gss
```

7. (オプション) クライアントが TGT を自動的に更新するか、または Kerberos チケットの有効期限切れに関してユーザーに警告するようにするには、`/etc/krb5/warn.conf` ファイル内にエントリを作成します。

詳細は、warn.conf(4) のマニュアルページおよび 129 ページの「すべてのチケット認可チケットの自動的な更新」を参照してください。

例 9 マルチマスター KDC と連携するための Oracle Solaris クライアントの構成

Microsoft Active Directory (AD) Kerberos サービスは、マルチマスターサーバー上で動作する KDC を提供します。Oracle Solaris クライアントが情報を更新するには、/etc/krb5/krb5.conf ファイル内の admin_server または kpasswd_server のどちらかの宣言にすべてのサーバーが記載されている必要があります。この例は、クライアントが kdc1 と kdc2 で共有される KDC に関する情報を更新できるようにする方法を示しています。

```
[realms]
EXAMPLE.COM = {
kdc = kdc1.example.com
kdc = kdc2.example.com
admin_server = kdc1.example.com
admin_server = kdc2.example.com
}
```

例 10 Oracle Solaris 以外の KDC のための Kerberos クライアントの構成

/etc/krb5/krb5.conf ファイル内の realms セクションに 1 行を追加することによって、Oracle Solaris 以外の KDC と連携するように Kerberos クライアントを設定できます。この行を追加すると、クライアントが Kerberos パスワード変更サーバーとの通信に使用するプロトコルが変更されます。次のコード例は、この行の形式を示しています。

```
[realms]
EXAMPLE.COM = {
kdc = kdc1.example.com
kdc = kdc2.example.com
admin_server = kdc1.example.com
kpasswd_protocol = SET_CHANGE
}
```

例 11 ホストおよびドメイン名の Kerberos レルムへのマッピングのための DNS TXT レコード

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
1989020501   ;serial
10800        ;refresh
3600         ;retry
3600000      ;expire
86400 )      ;minimum

IN      NS      kdc1.example.com.
kdc1                    IN      A       192.146.86.20
kdc2                    IN      A       192.146.86.21

_kerberos.example.com.              IN      TXT     "EXAMPLE.COM"
_kerberos.kdc1.example.com.         IN      TXT     "EXAMPLE.COM"
_kerberos.kdc2.example.com.         IN      TXT     "EXAMPLE.COM"
```

例 12 Kerberos サーバーの場所を記録する DNS SRV レコード

この例では、それぞれ KDC、admin サーバー、および kpasswd サーバーの場所のレコードを定義します。

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
1989020501   ;serial
10800        ;refresh
3600         ;retry
3600000      ;expire
86400 )      ;minimum

IN      NS      kdc1.example.com.
kdc1                                IN      A       192.146.86.20
kdc2                                IN      A       192.146.86.21

_kerberos._udp.EXAMPLE.COM          IN      SRV 0 0 88  kdc2.example.com
_kerberos._tcp.EXAMPLE.COM          IN      SRV 0 0 88  kdc2.example.com
_kerberos._udp.EXAMPLE.COM          IN      SRV 1 0 88  kdc1.example.com
_kerberos._tcp.EXAMPLE.COM          IN      SRV 1 0 88  kdc1.example.com
_kerberos-adm._tcp.EXAMPLE.COM      IN      SRV 0 0 464 kdc1.example.com
_kpasswd._udp.EXAMPLE.COM           IN      SRV 0 0 464 kdc1.example.com
```

チケット認可チケットの確認の無効化

デフォルトでは、Kerberos は、ローカルの /etc/krb5/krb5.keytab ファイル内に格納されているホスト主体の KDC がチケット認可チケット (TGT) を発行した KDC と同じであることを確認します。この確認 verify_ap_req_nofail によって、DNS へのなりすまし攻撃が防止されます。

ただし、ホスト主体を使用できないクライアント構成では、この確認を無効にする必要があります。次の構成では、この確認を無効にする必要があります。

- クライアントの IP アドレスが動的に割り当てられる (DHCP クライアントなど)。
- クライアントはサービスをホストするように構成されていないため、ホスト主体が作成されていない。
- クライアントにホスト鍵が保存されていない。

TGT の確認を無効にするには、krb5.conf ファイル内の verify_ap_req_nofail オプションを false に設定します。verify_ap_req_nofail オプションは、krb5.conf ファイルの [libdefaults] または [realms] セクションに入力できます。[libdefaults] セクションに入力すると、その設定はすべてのレルムに使用されます。

```
client # pfedit /etc/krb5/krb5.conf
[libdefaults]
default_realm = EXAMPLE.COM
verify_ap_req_nofail = false
...
```

このオプションを [realms] セクションに入力すると、その設定は定義されたレルムにのみ適用されます。このオプションの詳細は、krb5.conf(4) のマニュアルページを参照してください。

▼ Kerberos によって保護された NFS ファイルシステムに root ユーザーとしてアクセスする方法

この手順を実行すると、特に、NFS ファイルシステムが -o sec=krb5p, root=client1.example.com のようなオプションで共有されるときに、クライアントは Kerberos 認証を必要とする NFS ファイルシステムに、root 主体によってアクセスできるようになります。

1. **kadmin コマンドを実行します。**

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin:
```

2. **NFS クライアントの root 主体を作成します。**

この主体は、Kerberos 認証を必要とする、NFS マウントされたファイルシステムにスーパーユーザーと同様にアクセスするために使用されます。root 主体は、2 つのコンポーネントから成る主体にしてください。レルム全体にわたる root 主体が作成されないようにするために、2 番目のコンポーネントは Kerberos クライアントシステムのホスト名にしてください。主体のインスタンスがホスト名である場合、FQDN は、ネームサービスでのドメイン名の大文字小文字には関係なく小文字で指定する必要があることに注意してください。

```
kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.
kadmin:
```

3. **root 主体をサーバーのキータブファイルに追加し、kadmin を終了します。**

この手順は、クライアントに NFS サービスを使用してマウントされたファイルシステムへの root アクセスを許可するために必要です。この手順はまた、cron ジョブを root として実行する場合など、非対話的な root アクセスのためにも必要です。

```
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ Kerberos レルム内のユーザーを自動的に移行するように構成する方法

Kerberos 主体を持っていないユーザーを、PAM を使用して既存の Kerberos レルムに自動的に移行できます。UNIX 資格の認識や Kerberos レルム内での再認証を処理するに

は、移行サーバーおよびマスターサーバー上のシステムごとの PAM 構成ファイルをカスタマイズします。

PAM については、第1章「プラグイン可能認証モジュールの使用」および pam.conf(4) のマニュアルページを参照してください。

この手順では、ログインサービス名が自動移行を使用するように構成されます。この例では、次の構成パラメータを使用します。

- レルム名 = EXAMPLE.COM
- マスター KDC = kdc1.example.com
- 移行サービスをホストするマシン = server1.example.com
- 移行サービス主体 = host/server1.example.com

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **server1 のホストサービス主体が存在することを確認します。**

   server1 の keytab ファイル内のホストサービス主体は、マスター KDC にサーバーを認証するために使用されます。

   ```
   server1 # klist -k
   Keytab name: FILE:/etc/krb5/krb5.keytab
   KVNO Principal
   ---- -------------------------------------------------
   3 host/server1.example.com@EXAMPLE.COM
   ...
   ```

   このコマンドのオプションについては、klist(1) のマニュアルページを参照してください。

2. **server1 が一覧表示されない場合は、それをレルム EXAMPLE.COM の Kerberos クライアントとして構成します。**

   手順については、111 ページの「Kerberos クライアントの構成」の例を参照してください。

3. **server1 の PAM ポリシーを変更します。**

   詳細は、22 ページの「ユーザーごとの PAM ポリシーの割り当て」を参照してください。

   a. **server1 上でどの Kerberos ポリシーが使用されているかを判定します。**

      ```
      % grep PAM_POLICY /etc/security/policy.conf
      # PAM_POLICY specifies the system-wide PAM policy (see pam_user_policy(5))
      ...
      PAM_POLICY=krb5_first
      ```

b. その PAM ポリシーファイルをコピーしたあと、新しいポリシーファイルを変更して、各認証スタックのあとに `pam_krb5_migrate.so.1` モジュールを付加します。

```
server1 # cd /etc/security/pam_policy/; cp krb5_first krb5_firstmigrate
server1 # pfedit /etc/security/pam_policy/krb5_firstmigrate.
# login service (explicit because of pam_dial_auth)
#
login auth requisite    pam_authtok_get.so.1
...
login auth required  pam_unix_auth.so.1
login    auth optional  pam_krb5_migrate.so.1
#
# rlogin service (explicit because of pam_rhost_auth)
#
rlogin auth sufficient  pam_rhosts_auth.so.1
...
rlogin auth required    pam_unix_auth.so.1
rlogin   auth optional       pam_krb5_migrate.so.1
#
# Kerberized rlogin service
#
krlogin auth required  pam_unix_cred.so.1
krlogin auth required  pam_krb5.so.1
krlogin auth optional  pam_krb5_migrate.so.1
#
# rsh service (explicit because of pam_rhost_auth)
#
rsh auth sufficient  pam_rhosts_auth.so.1
rsh auth required    pam_unix_cred.so.1
rsh auth optional    pam_krb5_migrate.so.1
#
# Kerberized rsh service
#
krsh auth required  pam_unix_cred.so.1
krsh auth required  pam_krb5.so.1
krsh auth optional  pam_krb5_migrate.so.1
#
# Kerberized telnet service
#
ktelnet auth required  pam_unix_cred.so.1
ktelnet auth required  pam_krb5.so.1
ktelnet auth optional  pam_krb5_migrate.so.1
#
# PPP service (explicit because of pam_dial_auth)
#
ppp auth requisite    pam_authtok_get.so.1
...
ppp auth required  pam_unix_auth.so.1
ppp auth optional  pam_krb5_migrate.so.1
#
# GDM Autologin (explicit because of pam_allow).  These need to be
# here as there is no mechanism for packages to amend pam.conf as
# they are installed.
#
gdm-autologin auth  required    pam_unix_cred.so.1
gdm-autologin auth  sufficient  pam_allow.so.1
gdm-autologin auth optional    pam_krb5_migrate.so.1
#
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
```

```
#
OTHER auth requisite    pam_authtok_get.so.1
...
OTHER auth required  pam_unix_auth.so.1
OTHER auth optional  pam_krb5_migrate.so.1
#
# passwd command (explicit because of a different authentication module)
#
passwd auth required  pam_passwd_auth.so.1
#
# cron service (explicit because of non-usage of pam_roles.so.1)
#
cron account required pam_unix_account.so.1
#
# cups service (explicit because of non-usage of pam_roles.so.1)
#
cups account required pam_unix_account.so.1
#
# GDM Autologin (explicit because of pam_allow) This needs to be here
# as there is no mechanism for packages to amend pam.conf as they are
# installed.
#modified
gdm-autologin account  sufficient  pam_allow.so.1
#
.
.
.
```

c. (オプション) 即座のパスワード変更を強制的に適用します。

新しく作成された Kerberos アカウントについて、`pam_krb5_migrate` エントリに `expire_pw` オプションを追加することにより、パスワードの有効期限を現在の時間に設定します。詳細は、`pam_krb5_migrate(5)` のマニュアルページを参照してください。

```
service-name auth optional     pam_krb5_migrate.so.1 expire_pw
```

d. この構成ファイルで、`OTHER` アカウントスタックを変更して、Kerberos パスワードの期限が切れた場合はアクセスをブロックします。

```
# Definition for Account management
# Used when service name is not explicitly mentioned for account management
# Re-ordered pam_krb5 causes password expiration in Kerberos to block access
#
OTHER account requisite   pam_roles.so.1
OTHER account required pam_krb5.so.1
OTHER account required pam_unix_account.so.1
OTHER account required pam_tsol_account.so.1
# OTHER account required pam_krb5.so.1
#
.
.
.
```

e. 変更された構成ファイルを使用するように `policy.conf` ファイル内の `PAM_POLICY` エントリを変更します。

```
server1 # pfedit /etc/security/policy.conf
```

```
...
# PAM_POLICY=krb5_first
PAM_POLICY=krb5_firstmigrate
```

詳細は、policy.conf ファイルを参照してください。

4. **マスター KDC 上で、kadm5.acl アクセス制御ファイルを更新します。**

   次のエントリにより、root ユーザーを除くすべてのユーザーの host/server1.example.com サービス主体に移行および照会権限が付与されます。U 権限を使用して、移行してはいけないユーザーを一覧表示します。これらのエントリは、permit all または ui エントリの前に置く必要があります。詳細は、kadm5.acl(4) のマニュアルページを参照してください。

```
kdc1# pfedit /etc/krb5/kadm5.acl
host/server1.example.com@EXAMPLE.COM U root
host/server1.example.com@EXAMPLE.COM ui *
*/admin@EXAMPLE.COM *
```

5. **マスター KDC 上で、kadmind デーモンが k5migrate PAM サービスを使用できるようにします。**

   k5migrate サービスファイルが /etc/pam.d ディレクトリ内にない場合は、このディレクトリにそのサービスファイルを追加します。詳細は、pam.d(4) のマニュアルページを参照してください。

   この変更によって、移行が必要なアカウントに対する UNIX ユーザーパスワードの検証が可能になります。

```
kdc1# pfedit /etc/pam.d/k5migrate
...
# Permits validation of migrated UNIX accounts
auth    required        pam_unix_auth.so.1
account required        pam_unix_account.so.1
```

注記 - k5migrate は PAM サービスの名前です。このファイルは、k5migrate という名前である必要があります。

---

6. **作成した構成を本番環境に配置する前にテストします。**

   - **通常のユーザーとして、変更された各 PAM サービスをテストします。**
   - **root として、変更された各 PAM サービスをテストします。**
   - **パスワード変更を強制的に適用してから、変更された PAM サービスをテストします。**

すべてのチケット認可チケットの自動的な更新

管理を容易にするために、チケット認可チケット (TGT) の有効期限切れに関するチケットの更新および警告メッセージを構成できます。管理者はすべてのユーザーへの警告を設定することができ、ユーザーは独自の警告をカスタマイズできます。詳細は、`warn.conf(4)` および `ktkt_warnd(1M)` のマニュアルページを参照してください。

注記 - `ktkt_warn` サービスは、デフォルトでは無効になっています。既存の Kerberos クライアント上のサービスを有効にするには、`svcadm enable ktkt_warn` コマンドを実行します。

例 13 すべてのユーザーへの TGT 有効期限切れメッセージの構成

この例は、TGT の更新およびメッセージシステムを構成するためのいくつかの方法を示しています。

```
# pfedit /etc/krb5/warn.conf
##
## renew the TGT 30 minutes before expiration and send message to users terminal
##
mre@EXAMPLE.COM renew:log terminal 30m
##
## send a warning message to a specific email address 20 minutes before TGT expiration
##
mre@EXAMPLE.COM mail 20m mre@example2.com
##
# renew the TGT 20 minutes before expiration and send an email message on failure
##
bricker@EXAMPLE.COM renew:log-failure mail 20m -
##
## catch-all: any principal not matched above will get an email warning
* mail 20m -
```

メッセージを構成したら、新しいクライアント上で `kclient` コマンドを実行します。

```
client# /usr/sbin/kclient -p /net/denver.example.com/export/install/kcprofile
```

既存のクライアント上で、このサービスを有効にします。

```
# svcadm enable network/security/ktkt_warn
```

例 14 ユーザーへの TGT 有効期限メッセージの構成

各ユーザーは、個々の `warnd` 構成ファイルを構成できます。このファイルの名前は `/var/user/`*`$USER`*`/krb-warn.conf` です。このファイルが存在すると、管理者ファイルは読み取られなくなります。

```
% pfedit /var/user/mre/krb-warn.conf
mre@EXAMPLE.COM renew:log mail 25m &
```

TGT は有効期限切れの 25 分前に更新され、その更新がログに記録され、その時点で Kerberos ユーザー `mre` にメールが送信されます。

Kerberos ネットワークアプリケーションサーバーの構成

ネットワークアプリケーションサーバーとは、ftp、rcp、rlogin、rsh、ssh、telnet のネットワークアプリケーションのうちの 1 つ以上を使用してアクセスを提供するホストのことです。いくつかの手順を実行するだけで、サーバー上でこれらのアプリケーションの Kerberos バージョンを有効にすることができます。

▼ Kerberos ネットワークアプリケーションサーバーを構成する方法

この手順では、次の構成パラメータを使用します。

- アプリケーションサーバー = boston
- admin 主体 = kws/admin
- DNS ドメイン名 = example.com
- レルム名 = EXAMPLE.COM

始める前に マスター KDC が構成され、142 ページの「KDC と Kerberos クライアントのクロックの同期化」の説明どおりにクロックが同期されていることを確認します。処理を十分にテストするには、複数のクライアントが必要です。

アプリケーションサーバー上で root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **新しいサーバーのホスト主体が存在するかどうかを判定します。**

   次のコマンドは、ホスト主体の存在有無を報告します。

   ```
   boston # klist -k | grep host
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   4 host/boston.example.com@EXAMPLE.COM
   ```

   このコマンドが主体を返した場合は、完了です。主体が返されなかった場合は、次の手順を使用して新しい主体を作成します。

2. **マスター KDC を構成するときに作成した admin 主体名のいずれかを使用してサーバーにログインします。**

   ```
   boston # /usr/sbin/kadmin -p kws/admin
   Enter password: xxxxxxxx
   kadmin:
   ```

3. **サーバーの `host` 主体を作成します。**

```
kadmin: addprinc -randkey host/boston.example.com
Principal "host/boston.example.com" created.
kadmin:
```

`host` 主体は、次のように使用されます。

- `rsh` や `ssh` などのリモートコマンドを使用しているときにトラフィックを認証します。
- `pam_krb5` により、`host` 主体を使用してユーザーの Kerberos 資格が信頼できる KDC から取得されたことを確認し、KDC へのなりすまし攻撃を防ぎます。
- `root` ユーザーが `root` 主体の存在なしで Kerberos 資格を自動的に取得できるようにします。この機能は、共有が Kerberos 資格を必要とする場合に手動の NFS マウントを実行するときに役立つことがあります。

リモートアプリケーションを使用するトラフィックに Kerberos サービスによる認証が必要な場合は、この主体が必要です。サーバーに複数のホスト名が関連付けられている場合は、ホスト名の FQDN 形式を使用して、ホスト名ごとに主体を作成します。

4. **サーバーの `host` 主体をサーバーのキータブファイルに追加し、`kadmin` を終了します。**

`kadmin` コマンドが実行中でない場合は、次のようなコマンドでリブートします。`/usr/sbin/kadmin -p` *kws*`/admin`

サーバーに複数のホスト名が関連付けられている場合は、ホスト名ごとに主体を keytab に追加します。

```
kadmin: ktadd host/boston.example.com
Entry for principal host/boston.example.com with kvno 3, encryption type AES-256 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ FTP の実行時に Generic Security Service を Kerberos とともに使用する方法

Generic Security Service (GSS) は、Kerberos ネットワークアプリケーションで、認証、整合性、およびプライバシのために使用できます。次の手順は、ProFTPD の GSS サービスを有効にする方法を示しています。

始める前に FTP サーバー上で `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **FTP サーバーの主体を追加し、その FTP サーバーのキータブファイルを作成します。**

   それらの変更を前に行なっている場合は、これらの手順が必要ないことがあります。

   a. **`kadmin` コマンドを起動します。**

   ```
   ftpserver1 # /usr/sbin/kadmin -p kws/admin
   Enter password: xxxxxxxx
   kadmin:
   ```

   b. **FTP サーバーの `ftp` サービス主体を追加します。**

   ```
   kadmin: addprinc -randkey ftp/ftpserver1.example.com
   ```

   c. **`ftp` サービス主体を新しいキータブファイルに追加します。**

   新しい keytab ファイルにより、サーバーの keytab ファイル内のすべての情報を公開しなくても、FTP サービスでこの情報を使用できるようになります。

   ```
   kadmin: ktadd -k /etc/krb5/ftp.keytab ftp/ftpserver1.example.com
   ```

   詳細は、kadmin(1M) のマニュアルページの `ktadd` コマンドを参照してください。

2. **新しいキータブファイルの所有権を変更します。**

   ```
   ftpserver1 # chown ftp:ftp /etc/krb5/ftp.keytab
   ```

3. **FTP サーバーの GSS を有効にします。**

   `/etc/proftpd.conf` ファイルに次の変更を行います。

   ```
   # pfedit /etc/proftpd.conf
   LoadModule     mod_gss.c

   GSSEngine      on
   GSSKeytab      /etc/krb5/ftp.keytab
   ```

4. **FTP サーバーを再起動します。**

   ```
   # svcadm restart network/ftp
   ```

Kerberos NFS サーバーの構成

NFS サービスは UNIX ユーザー ID (UID) を使用してユーザーを識別しており、GSS 資格を直接使用することはできません。この資格を UID に変換するために、ユーザー資格を UNIX UID にマップする資格テーブルを作成することが必要になる場合があります。デフォルトの資格マッピングについては、71 ページの「GSS 資格の UNIX 資格へのマッピング」を参照してください。このセクションでは、Kerberos NFS サーバーの構成手順、資格テーブルの管理手順、および NFS マウントしたファイルシステムに対

して Kerberos セキュリティーモードを有効にするタスクを中心に説明します。次のタスクマップは、このセクションで説明するタスクの一覧です。

表 7 タスクマップ: Kerberos NFS サーバーの構成

タスク	説明	参照先
Kerberos NFS サーバーを構成します。	Kerberos 認証を必要とするファイルシステムを、サーバーが共有できるようにします。	133 ページの「Kerberos NFS サーバーを構成する方法」
資格テーブルを作成し、それを変更します。	デフォルトのマッピングが十分でない場合に GSS 資格を UNIX UID にマップするための資格テーブルを作成し、エントリを追加します。	134 ページの「資格テーブルを作成および変更する方法」
別のレルムのユーザー資格を UNIX UID にマップします。	資格テーブルの情報を更新します。	例15「異なるドメイン内の主体の Kerberos 資格テーブルへの追加」
類似する 2 つのレルム間の資格マッピングを作成します。	各レルムがパスワードファイルを共有している場合に、UID をあるレルムから別のレルムにマップします。	135 ページの「レルム間の資格マッピングを提供する方法」
Kerberos 認証を使用してファイルシステムを共有します。	セキュリティーモードを使用してファイルシステムを共有し、Kerberos 認証を常に行います。	136 ページの「複数の Kerberos セキュリティーモードで安全な NFS 環境を設定する方法」

▼ Kerberos NFS サーバーを構成する方法

この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- NFS サーバー = `denver.example.com`
- `admin` 主体 = `kws/admin`

始める前に NFS サーバー上で `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

マスター KDC が構成され、142 ページの「KDC と Kerberos クライアントのクロックの同期化」の説明どおりにクロックが同期されていることを確認します。処理を十分にテストするには、複数のクライアントが必要です。

1. **NFS サーバーを Kerberos クライアントとして構成します。**
   111 ページの「Kerberos クライアントの構成」の手順に従ってください。

2. **NFS サービス主体を追加します。**
   `kadmin` コマンドを使用します。

```
denver # /usr/sbin/kadmin -p kws/admin
```

```
Enter password: xxxxxxxx
kadmin:
```

a. **NFS サービス主体を作成します。**

主体のインスタンスがホスト名である場合、FQDN は、ネームサービスでのドメイン名の大文字小文字には関係なく小文字で指定する必要があることに注意してください。

NFS データへのアクセスに使用されるシステム上の一意のインタフェースごとに、上記の手順を繰り返します。ホストに一意の名前を持ったインタフェースが複数存在する場合、一意の名前は、それぞれに NFS サービス主体を持つ必要があります。

```
kadmin: addprinc -randkey nfs/denver.example.com
Principal "nfs/denver.example.com" created.
kadmin:
```

b. **サーバーの NFS サービス主体をサーバーのキータブファイルに追加し、`kadmin` を終了します。**

ステップ 2a で作成した一意のサービス主体ごとに、この手順を繰り返します。

```
kadmin: ktadd nfs/denver.example.com
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS
 mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

3. **必要に応じて、特殊な GSS 資格マップを作成します。**

通常、Kerberos サービスは、GSS 資格と UNIX UID 間の適切な対応表を生成します。デフォルトのマッピングは、71 ページの「GSS 資格の UNIX 資格へのマッピング」で説明されています。デフォルトのマッピングが十分でない場合は、134 ページの「資格テーブルを作成および変更する方法」で詳細を参照してください。

4. **NFS ファイルシステムを Kerberos セキュリティーモードで共有します。**

詳細は、136 ページの「複数の Kerberos セキュリティーモードで安全な NFS 環境を設定する方法」を参照してください。

▼ 資格テーブルを作成および変更する方法

`gsscred` 資格テーブルは、Kerberos 資格を UNIX UID にマップするために NFS サーバーによって使用されます。デフォルトでは、主体名のプライマリの部分が UNIX の

ログイン名に一致します。このテーブルは、デフォルトのマッピングが十分でない場合に作成します。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **/etc/gss/gsscred.conf で示されるセキュリティーメカニズムが files であることを確認します。**

```
# cat /etc/gss/gsscred.conf
...
#
files
#
#
# Syslog (auth.debug) a message for GSS cred to Unix cred mapping
#SYSLOG_UID_MAPPING=yes
```

2. **gsscred コマンドを使用して資格テーブルを作成します。**

```
# gsscred -m kerberos_v5 -a
```

gsscred コマンドは、svc:/system/name-service/switch:default サービス内の passwd エントリに示されているすべてのソースから情報を収集します。資格テーブルにローカルのパスワードエントリを含めない場合は、files エントリを一時的に削除できます。詳細は、gsscred(1M) のマニュアルページを参照してください。

3. **(オプション) 資格テーブルにエントリを追加します。**

たとえば、NFS サーバー上の root 役割として、主体 sandy/admin を UID 3736 にマップするためのエントリを追加します。-a オプションにより、このエントリが資格テーブルに追加されます。

```
# gsscred -m kerberos_v5 -n sandy/admin -u 3736 -a
```

例 15 異なるドメイン内の主体の Kerberos 資格テーブルへの追加

この例では、完全修飾ドメイン名 (FQDN) を使用して、異なるドメイン内の主体を指定します。

```
# gsscred -m kerberos_v5 -n sandy/admin@EXAMPLE.COM -u 3736 -a
```

▼ レルム間の資格マッピングを提供する方法

この手順では、同じパスワードファイルを使用するレルム間の適切な資格マッピングを提供します。この例では、CORP.EXAMPLE.COM と SALES.EXAMPLE.COM の各レルムは同じパスワードファイルを使用します。*username*@CORP.EXAMPLE.COM と *username*@SALES.EXAMPLE.COM の資格は、同じ UID にマップされます。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

- **クライアントシステム上で、krb5.conf ファイルに default_realm および auth_to_local_realm エントリを追加します。**

```
# pfedit /etc/krb5/krb5.conf
[libdefaults]
default_realm = CORP.EXAMPLE.COM
.
[realms]
CORP.EXAMPLE.COM = {
.
auth_to_local_realm = SALES.EXAMPLE.COM
.
}
```

注意事項 資格マッピングの問題のトラブルシューティングに関するヘルプについては、209 ページの「GSS 資格の UNIX 資格へのマッピングの監視」を参照してください。

▼ 複数の Kerberos セキュリティーモードで安全な NFS 環境を設定する方法

この手順では、複数のセキュリティーモードを使用して、NFS サーバーがセキュアな NFS アクセスを提供できるようにします。クライアントが NFS サーバーとセキュリティーモードについてネゴシエーションを行うとき、そのクライアントは、サーバーによって提供された最初のモードを使用します。このモードは、そのサーバーで共有されているファイルシステムの以降のすべてのクライアントリクエストに使用されます。

始める前に NFS サーバー上で root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **キータブファイルに NFS サービス主体が存在することを検証します。**
   klist コマンドを指定すると、キータブファイルが存在するかどうかが出力され、その主体が表示されます。キータブファイルが存在しない場合、または NFS サービス主体が存在しない場合は、133 ページの「Kerberos NFS サーバーを構成する方法」のすべての手順が完了していることを検証する必要があります。

```
# klist -k
Keytab name: FILE:/etc/krb5/krb5.keytab
KVNO Principal
---- --------------------------------------------------------
3 nfs/denver.example.com@EXAMPLE.COM
3 nfs/denver.example.com@EXAMPLE.COM
```

```
3 nfs/denver.example.com@EXAMPLE.COM
3 nfs/denver.example.com@EXAMPLE.COM
```

詳細は、klist(1) のマニュアルページを参照してください。

2. **/etc/nfssec.conf ファイル内の Kerberos セキュリティーモードを有効にします。**

   /etc/nfssec.conf ファイルで、Kerberos セキュリティーモードをコメントアウトしている「#」を削除します。

```
# pfedit /etc/nfssec.conf
.
.
#
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
```

3. **適切なセキュリティーモードを使用してファイルシステムを共有します。**

   - krb5 認証、NFS 経由で送信された機密データの整合性およびプライバシ保護を提供するには、krb5p を選択します。サーバーの処理リソースを超えないかぎり、このモードを使用します。
   - TCP/IP が NFS データに対して提供する最小限の保護に加えて、krb5 認証および整合性保護を提供するには、krb5i を選択します。
   - krb5 認証のみに対しては krb5 を選択します。このセキュリティーモードは、提供されるセキュリティーモードの保護は最小限ですが、プロセッサへの影響はもっとも小さくなります。

```
share -F nfs -o sec=mode  file-system
```

   mode ファイルシステムを共有するときに使用するセキュリティーモードを指定します。複数のセキュリティーモードを使用するときは、デフォルトとして、リストの最初のモードが使用されます。

   file-system 共有するファイルシステムへのパスを定義します。

   指定されたファイルシステムのファイルにアクセスするすべてのクライアントは、Kerberos 認証が必要です。ファイルにアクセスするには、NFS クライアント上にユーザー主体が認証される必要があります。

4. **(オプション) デフォルト以外のセキュリティーモードを使用してファイルシステムをマウントします。**

   デフォルトのセキュリティーモードが受け入れ可能な場合は、この手順を実行しないでください。

   - **オートマウンタが使用される場合は、デフォルト以外のセキュリティーモードに入るように auto_master データベースを編集します。**

file-system `auto_home -nosuid,sec=`*mode*

- **`mount` コマンドを手動で発行することにより、デフォルト以外のモードを使用してファイルシステムにアクセスします。**

  `# mount -F nfs -o sec=`*mode* *file-system*

例 16 1 つの Kerberos セキュリティーモードでファイルシステムを共有する

この例では、NFS サービスを使用していずれかのファイルにアクセスするには、その前に `krb5p` セキュリティーモードでの認証が成功している必要があります。

```
# share -F nfs -o sec=krb5p /export/home
```

例 17 複数の Kerberos セキュリティーモードでファイルシステムを共有する

次の例では、3 つの Kerberos セキュリティーモードがすべて選択されています。使用されるモードは、クライアントと NFS サーバーの間でネゴシエーションが行われます。コマンドの最初のモードが失敗すると、次のモードが試行されます。詳細は、`nfssec(5)` のマニュアルページを参照してください。

```
# share -F nfs -o sec=krb5p:krb5i:krb5 /export/home
```

Kerberos サービスへのアクセスのための遅延実行の構成

デフォルトの Kerberos 環境では、制限された時間が経過すると資格は期限切れになります。`cron` や `at` などの、いつでも実行される場合があるプロセスでは、この制限された時間によって問題が発生します。この手順では、Kerberos による認証されたサービスを必要とする遅延実行プロセスをサポートするように Kerberos 環境を構成する方法について説明します。Oracle Solaris は PAM モジュールを提供し、サービス鍵を使用し、さらに `kclient` 構成オプションを使用することによって Kerberos 認証による遅延実行を可能にし、またそれを代替の解決方法よりセキュアなものにしています。

注記 - `cron` サーバーが危険にさらされた場合、攻撃者はユーザーになりすまして、`cron` サーバー用に構成されたターゲットサービスへのアクセスを取得する可能性があります。そのため、ユーザーに対して中間サービスを提供する、この手順で構成された `cron` ホストをより機密性の高いシステムと見なしてください。

▼ Kerberos サービスにアクセスするための cron ホストを構成する方法

この手順では、次の構成パラメータを使用します。

- cron ホスト = host1.example.com
- NFS サーバー = host2.example.com
- LDAP サーバー = host3.example.com

1. **Kerberos をサポートするように cron サービスを構成します。**

   - **cron ホストで Kerberos が構成されていない場合は、システム上で kclient コマンドを実行します。**
     詳細は、kclient(1M) のマニュアルページを参照してください。
     たとえば、次のコマンドは、EXAMPLE.COM レルム内にクライアントを構成します。このコマンドでは、include メカニズムを使用して、/etc/pam.d/cron サービスファイル内に pam_gss_s4u ファイルを含めています。

     ```
     # kclient -s cron:optional -R EXAMPLE.COM
     ```

   - **cron ホストですでに Kerberos が構成されている場合は、そのホスト上で cron サービスのための PAM 構成を手動で変更する必要があります。**
     cron サービスのための PAM 構成に pam_gss_s4u ファイルが含まれていることを確認します。

     ```
     # cd /etc/pam.d ; cp cron cron.orig
     # pfedit cron
           # PAM include file for optional set credentials
           # through Kerberos keytab and GSS-API S4U support
           auth include         pam_gss_s4u
     ```

2. **cron ホストが代理として機能できるようにします。**
   例:

   ```
   # kadmin -p kws/admin
   Enter password: xxxxxxxx
   kadmin: modprinc +ok_as_delegate host/host1.example.com@EXAMPLE.COM
   Principal host/host1.example.com@EXAMPLE.COM modified.
   ```

3. **cron ホストが、cron ジョブを作成したユーザーの代わりに自身に対してチケットをリクエストできるようにします。**

   ```
   kadmin: modprinc +ok_to_auth_as_delegate host/host1.example.com@EXAMPLE.COM
   Principal host/host1.example.com@EXAMPLE.COM modified.
   kadmin: quit
   ```

4. **LDAP では、代理として使用するサービスを指定するように cron ホストを構成します。**

たとえば、cron ホストが host2 (Kerberos NFS サーバー) 上のユーザーのホームディレクトリにアクセスできるようにするには、その NFS ホストを cron サーバーの LDAP 定義内の krbAllowedToDelegateTo パラメータに追加します。

a. 代理の割り当てを作成します。

```
# pfedit /tmp/delghost.ldif
dn: krbprincipalname=host/host1.example.com@EXAMPLE.COM,cn=EXAMPLE.COM,
cn=krbcontainer,dc=example,dc=com
changetype: modify
krbAllowedToDelegateTo: nfs/host2.example.com@EXAMPLE.COM
```

b. その割り当てを LDAP に追加します。

```
# ldapmodify -h host3 -D "cn=directory manager" -f delghost.ldif
```

レルム間認証の構成

複数のレルムを接続して、レルム間でユーザーを認証することができます。レルム間の認証は、2 つのレルム間で共有される秘密鍵を作成することによって実行されます。レルム間の関係は、階層または直接のどちらかです。詳細は、64 ページの「Kerberos レルムの階層」を参照してください。

▼ 階層関係のレルム間認証を設定する方法

この手順の例では、CORP.EAST.EXAMPLE.COM と EAST.EXAMPLE.COM の間に双方向のレルム間認証を確立します。この手順では、両方のレルム内のマスター KDC 上で実行する必要があります。

始める前に 各レルムのマスター KDC が構成されています。認証プロセスを十分にテストするには、複数のクライアントが必要です。

両方の KDC サーバー上で root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. 2 つのレルムに対して、TGT のサービス主体を作成します。

マスター KDC を構成したときに作成した admin 主体名を使用して、ログインする必要があります。

```
# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: addprinc krbtgt/CORP.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
```

```
Enter password for principal krbtgt/CORP.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM:      /** Type strong
password **/
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@CORP.EAST.EXAMPLE.COM
Enter password for principal krbtgt/EAST.EXAMPLE.COM@CORP.EAST.EXAMPLE.COM:      /** Type strong
password **/
kadmin: quit
```

注記 - これらのパスワードを安全な場所に保存および保管してください。

2. **Kerberos 構成ファイルにエントリを追加して、すべてのレルムのドメイン名を定義します。**

```
# pfedit /etc/krb5/krb5.conf
[libdefaults]
.
.
[domain_realm]
.corp.east.example.com = CORP.EAST.EXAMPLE.COM
.east.example.com = EAST.EXAMPLE.COM
```

この例では、CORP.EAST.EXAMPLE.COM および EAST.EXAMPLE.COM レルムのドメイン名が定義されます。このファイルは上から下に検索されるため、サブドメインはファイル内でドメイン名の前にある必要があります。

3. **Kerberos 構成ファイルをこのレルムのすべてのクライアントにコピーします。**
レルム間認証が機能するには、すべてのシステム (スレーブ KDC やその他のサーバーを含む) が /etc/krb5/krb5.conf のマスター KDC のバージョンを使用する必要があります。

4. **2 番目のレルムでこの手順を繰り返します。**

注記 - 各サービス主体のパスワードは、2 つの KDC で同一である必要があります。そのため、サービス主体 krbtgt/CORP.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM のパスワードは両方のレルムで同じである必要があります。

▼ 直接接続のレルム間認証を確立する方法

この手順の例では、CORP.EAST.EXAMPLE.COM と SALES.WEST.EXAMPLE.COM の 2 つのレルムを使用します。レルム間認証は、双方向に確立されます。この手順は、2 つのレルムのマスター KDC 上で完了する必要があります。

始める前に 各レルムのマスター KDC が構成されています。認証プロセスを十分にテストするには、複数のクライアントが必要です。

両方の KDC サーバー上で root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **2 つのレルムに対して、TGT のサービス主体を作成します。**
   マスター KDC を構成したときに作成した admin 主体名を使用して、ログインする必要があります。

```
# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: addprinc krbtgt/CORP.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM
Enter password for principal
krbtgt/CORP.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM:      /** Type strong password **/
kadmin: addprinc krbtgt/SALES.WEST.EXAMPLE.COM@CORP.EAST.EXAMPLE.COM
Enter password for principal
krbtgt/SALES.WEST.EXAMPLE.COM@CORP.EAST.EXAMPLE.COM:      /** Type strong password **/
kadmin: quit
```

2. **Kerberos 構成ファイルにエントリを追加して、リモートレルムへの直接パスを定義します。**
   この例は、CORP.EAST.EXAMPLE.COM レルム内のクライアントを示しています。SALES.WEST.EXAMPLE.COM レルムで適切な定義を追加するには、レルム名をスワップします。

```
# pfedit /etc/krb5/krb5.conf
[libdefaults]
.
.
[capaths]
CORP.EAST.EXAMPLE.COM = {
SALES.WEST.EXAMPLE.COM = .
}

SALES.WEST.EXAMPLE.COM = {
CORP.EAST.EXAMPLE.COM = .
}
```

3. **Kerberos 構成ファイルを現在のレルムのすべてのクライアントにコピーします。**
   レルム間認証が機能するには、すべてのシステム (スレーブ KDC やその他のサーバーを含む) が Kerberos 構成ファイル /etc/krb5/krb5.conf の新しいバージョンを使用する必要があります。

4. **2 番目のレルムに対してこの手順を繰り返します。**

KDC と Kerberos クライアントのクロックの同期化

Kerberos 認証システムに参加するすべてのホストは、指定した最大時間内に収まるように内部クロックを同期化する必要があります (「クロックスキュー」)。この必要条件は、Kerberos セキュリティーの検査の 1 つです。参加しているホスト間のクロックスキューが超過すると、クライアントの要求が拒否されます。

クロックスキューはまた、アプリケーションサーバーが、再実行されたリクエストを認識して拒否するために Kerberos プロトコルメッセージを追跡する必要がある時間

の長さも決定します。そのため、クロックスキュー値が長いほど、アプリケーションサーバーが収集する情報も多くなります。

最大クロックスキューのデフォルト値は、300 秒 (5 分) です。このデフォルトは、`krb5.conf` ファイルの `libdefaults` セクションで変更できます。

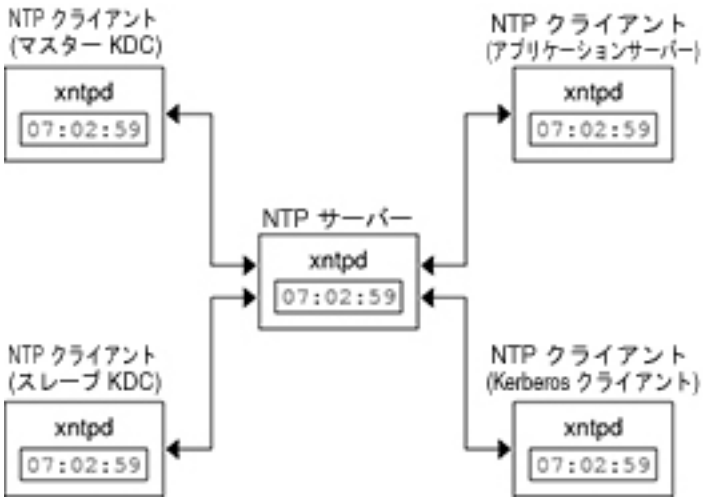
注記 - セキュリティー上の理由から、クロックスキュー値は 300 秒より大きくしないでください。

KDC と Kerberos クライアント間で同期されたクロックを維持することが重要であるため、PTP (Precision Time Protocol) または NTP (Network Time Protocol) ソフトウェアを使用してクロックを同期してください。これらのクロック同期ソフトウェアプロトコルの詳細は、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』を参照してください。

NTP ソフトウェアは、ほとんどの Oracle Solaris システム上にデフォルトでインストールされています。`pkg install ptp` コマンドを使用すると、PTP ソフトウェアをインストールできます。

次の図は、NTP クロック同期の例を示しています。

図 11 NTP を使用したクロック同期



KDC および Kerberos クライアントがクロックを同期化するには、次の手順を実行します。

1. Kerberos ネットワーク上の PTP または NTP サーバーの設定。このサーバーは、マスター KDC 以外の任意のシステムにすることができます。
2. ネットワーク上に KDC と Kerberos クライアントを構成する際に、それらをクロック同期サーバーのクライアントにします。マスター KDC に戻って、クロック同期サーバーのクライアントとして KDC を構成します。
3. すべてのシステム上でのクロック同期サービスの有効化。

マスター KDC とスレーブ KDC の入れ替え

このセクションの手順によって、マスター KDC とスレーブ KDC のスワップが容易になります。マスター KDC とスレーブ KDC の入れ替えは、マスター KDC に何らかの理由で障害が発生した場合、またはマスター KDC を再インストールする必要がある場合 (新しいハードウェアをインストールした場合など) にだけ行なってください。

▼ 入れ替え可能なスレーブ KDC を構成する方法

この手順は、マスター KDC にするために使用可能な状態にするスレーブ KDC サーバー上で増分伝播を使用しているレルム内で実行します。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **KDC をインストールするときに、マスター KDC および入れ替え可能なスレーブ KDC に対して別名を使用します。**

   KDC に対してホスト名を定義するときは、各システムの別名が DNS に登録されている必要があります。また、`/etc/krb5/krb5.conf` ファイル内でホストを定義するときも、それらのエイリアス名を使用します。

2. **スレーブ KDC をインストールします。**

   スワップの前に、このサーバーが、レルム内のその他のすべてのスレーブ KDC と同様に動作するようにしてください。詳細は、86 ページの「スレーブ KDC を手動で構成する方法」を参照してください。

3. **インストールしたら、マスター KDC コマンドを移動します。**

マスター KDC コマンドをこのスレーブ KDC から実行してはいけません。

```
kdc4# mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc4# mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc4# mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
```

▼ マスター KDC とスレーブ KDC を入れ替える方法

この手順では、旧マスター KDC サーバー名は、`kdc1` です。新しいマスター KDC になるスレーブ KDC には `kdc4` という名前が付けられ、元のマスター KDC がスレーブ KDC になります。この手順は、増分伝播を使用していることを想定しています。

始める前に 144 ページの「入れ替え可能なスレーブ KDC を構成する方法」の手順を完了します。

`root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **新しいマスター KDC 上で、`kadmin` を起動します。**

```
kdc4# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin:
```

2. **`kadmind` サービスの新しい主体を作成します。**
次の例では、`addprinc` が 2 行で表示されていますが、1 行に入力する必要があります。

```
kadmin: addprinc -randkey -allow_tgs_req +password_changing_service -clearpolicy \
changepw/kdc4.example.com
Principal "changepw/kdc4.example.com@EXAMPLE.COM" created.
kadmin: addprinc -randkey -allow_tgs_req -clearpolicy kadmin/kdc4.example.com
Principal "kadmin/kdc4.example.com@EXAMPLE.COM" created.
kadmin: quit
```

3. **新しいマスター KDC 上で、同期を強制します。**
次の手順は、スレーブサーバー上で強制的に KDC を完全に更新します。

   a. **`krb5kdc` サービスを無効にし、そのログファイルを削除します。**

```
kdc4# svcadm disable network/security/krb5kdc
kdc4# rm /var/krb5/principal.ulog
```

   b. **更新が完了したことを確認します。**

```
kdc4# /usr/sbin/kproplog -h
```

   c. **KDC サービスを再起動します。**

```
kdc4# svcadm enable -r network/security/krb5kdc
```

d. 新しいマスター KDC サーバーの更新ログを再初期化します。

```
kdc4# svcadm disable network/security/krb5kdc
kdc4# rm /var/krb5/principal.ulog
```

4. 元のマスター KDC 上で、`kadmin` および `krb5kdc` サービスを無効にします。

`kadmin` サービスを無効にすると、KDC データベースを一切変更できなくなります。

```
kdc1# svcadm disable network/security/kadmin
kdc1# svcadm disable network/security/krb5kdc
```

5. 元のマスター KDC 上で、伝播を要求するためのポーリング時間を指定します。

`/etc/krb5/kdc.conf` 内の `sunw_dbprop_master_ulogsize` エントリをコメントアウトし、スレーブのポーリング間隔を定義するエントリを追加します。このエントリにより、ポーリング時間が 2 分に設定されます。

```
kdc1# pfedit /etc/krb5/kdc.conf
[kdcdefaults]
     kdc_ports = 88,750

[realms]
     EXAMPLE.COM= {
        profile = /etc/krb5/krb5.conf
        database_name = /var/krb5/principal
        acl_file = /etc/krb5/kadm5.acl
        kadmind_port = 749
        max_life = 8h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        sunw_dbprop_enable = true
#            sunw_dbprop_master_ulogsize = 1000
        sunw_dbprop_slave_poll = 2m
     }
```

6. 元のマスター KDC 上で、マスター KDC コマンドおよび `kadm5.acl` ファイルを移動します。

マスター KDC のコマンドは、元のマスター KDC から実行しないでください。

```
kdc1# mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc1# mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc1# mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
kdc1# mv /etc/krb5/kadm5.acl /etc/krb5/kadm5.acl.save
```

7. DNS マッピングを変更します。

a. DNS サーバー上で、マスター KDC の別名を変更します。

サーバーを変更するために、`example.com` ゾーンファイルを編集して `masterkdc` のエントリを変更します。

```
masterkdc IN CNAME kdc4
```

b. 新しい別名の情報を再ロードします。

```
# svcadm refresh network/dns/server
```

8. **新しいマスター KDC 上で、マスター KDC コマンドとスレーブ `kpropd.acl` ファイルを移動します。**

   マスター KDC コマンドは、144 ページの「入れ替え可能なスレーブ KDC を構成する方法」のステップ 3 で移動しました。

```
kdc4# mv /usr/lib/krb5/kprop.save /usr/lib/krb5/kprop
kdc4# mv /usr/lib/krb5/kadmind.save /usr/lib/krb5/kadmind
kdc4# mv /usr/sbin/kadmin.local.save /usr/sbin/kadmin.local
kdc4# mv /etc/krb5/kpropd.acl /etc/krb5/kpropd.acl.save
```

9. **新しいマスター KDC 上で、Kerberos アクセス制御リストファイル `kadm5.acl` を作成します。**

   作成された `/etc/krb5/kadm5.acl` ファイルには、KDC を管理できる主体名がすべて含まれている必要があります。このファイルにはまた、増分伝播に対するリクエストを発行できるすべてのスレーブも記載されているべきです。詳細は、`kadm5.acl(4)` のマニュアルページを参照してください。

```
kdc4# pfedit /etc/krb5/kadm5.acl
kws/admin@EXAMPLE.COM    *
kiprop/kdc1.example.com@EXAMPLE.COM p
```

10. **新しいマスター KDC 上で、`kdc.conf` ファイル内の更新ログのサイズを指定します。**

    `sunw_dbprop_slave_poll` エントリをコメントアウトし、`sunw_dbprop_master_ulogsize` を定義するエントリを追加します。このエントリにより、ログサイズが 1000 エントリに設定されます。

```
kdc4# pfedit /etc/krb5/kdc.conf
[kdcdefaults]
     kdc_ports = 88,750

[realms]
     EXAMPLE.COM= {
        profile = /etc/krb5/krb5.conf
        database_name = /var/krb5/principal
        acl_file = /etc/krb5/kadm5.acl
        kadmind_port = 749
        max_life = 8h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        sunw_dbprop_enable = true
#            sunw_dbprop_slave_poll = 2m
        sunw_dbprop_master_ulogsize = 1000
     }
```

11. **新しいマスター KDC 上で、`kadmin` および `krb5kdc` サービスを有効にします。**

```
kdc4# svcadm enable -r network/security/krb5kdc
kdc4# svcadm enable -r network/security/kadmin
```

12. **元のマスター KDC をスレーブ KDC にします。**

    a. **`kiprop` サービス主体を追加します。**

krb5.keytab ファイルに kiprop 主体を追加すると、増分伝播サービスに対して kpropd デーモンが自身を認証できるようになります。

```
kdc1# /usr/sbin/kadmin -p kws/admin
Authenticating as principal kws/admin@EXAMPLE.COM with password.
Enter password: xxxxxxxx
kadmin: ktadd kiprop/kdc1.example.com
Entry for principal kiprop/kdc1.example.com with kvno 3,
encryption type AES-256 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3,
encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES
 cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

b. krb5.conf ファイル内の各 KDC のエントリを伝播構成ファイルに追加します。

```
kdc1# pfedit /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
host/kdc4.example.com@EXAMPLE.COM
```

c. kpropd および krb5kdc サービスを有効にします。

```
kdc1# svcadm enable -r network/security/krb5_prop
kdc1# svcadm enable -r network/security/krb5kdc
```

Kerberos データベースの管理

Kerberos データベースは、Kerberos のもっとも重要なコンポーネントであるため、適切に管理する必要があります。このセクションでは、データベースのバックアップと復元、増分または並列伝播の設定、stash ファイルの管理など、Kerberos データベースを管理するためのいくつかの手順について説明します。データベースを初期設定する手順については、81 ページの「マスター KDC を手動で構成する方法」を参照してください。

Kerberos データベースのバックアップと伝播

マスター KDC の Kerberos データベースをスレーブ KDC に伝播する処理は、構成タスクの中でもっとも重要なものの 1 つです。伝播の頻度が低いと、マスター KDC とスレーブ KDC が同期しなくなります。そのため、マスター KDC が停止した場合、スレーブ KDC には最新のデータベース情報が存在しません。また、負荷を分散するためにスレーブ KDC がマスター KDC として構成されている場合も、そのスレーブ KDC をマスター KDC として使用するクライアントは最新情報を取得できません。

伝播が十分な頻度で実行されることを確認するか、または Kerberos データベースを変更する頻度に基づいてサーバーの増分伝播を構成してください。増分伝播は、その他の伝播方法より優先されます。データベースの手動伝播では必要な管理オーバーヘッドが増え、また完全伝播は非効率的です。

注意 - 定期的にスケジュールされた伝播の前に Kerberos データベースに重要な更新を追加した場合は、データ損失を回避するために、データベースを手動で伝播するようにしてください。

kpropd.acl ファイル

スレーブ KDC の `kpropd.acl` ファイルの各行には、ホスト主体名と、伝播された最新のデータベースの受信元となるシステムが指定されています。マスター KDC を使用してすべてのスレーブ KDC に伝播する場合は、各スレーブ KDC の `kpropd.acl` ファイルに対してマスター KDC の主体名だけを指定する必要があります。

ただし、このガイドで説明されている Kerberos のインストールやそれ以降の構成手順では、マスター KDC とスレーブ KDC 上で同じ `kpropd.acl` ファイルを使用するように指示しています。このファイルには、すべての KDC ホスト主体名が含まれます。この構成を使用すると、伝播元の KDC が一時的に使用できなくなったときでも、任意の KDC から伝播することができます。すべての KDC 上に同一コピーが存在すると保守が容易になります。

kprop_script コマンド

`kprop_script` コマンドは、`kprop` コマンドを使用して Kerberos データベースをほかの KDC に伝播します。`kprop_script` コマンドをスレーブ KDC 上で実行すると、そのスレーブ KDC の Kerberos データベースのコピーがほかの KDC に伝播されます。`kprop_script` は引数として、伝播する KDC を示す、スペースで区切られたホスト名のリストを受け入れます。

`kprop_script` を実行すると、Kerberos データベースのバックアップが `/var/krb5/slave_datatrans` ファイルに作成され、指定した KDC にそのファイルがコピーされます。Kerberos データベースは、伝播が完了するまでロックされます。

Kerberos データベースのバックアップ

マスター KDC を構成するときは、`cron` ジョブ内に `kprop_script` コマンドを設定して、Kerberos データベースを `/var/krb5/slave_datatrans` ダンプファイルに自動的にバックアップし、それをスレーブ KDC に伝播します。ただし、他のファイルと

同様に、Kerberos データベースは壊れることがあります。データベースの次回の自動伝播によって最新のコピーがインストールされるため、スレーブ KDC 上ではデータ破損は問題になりません。ただし、マスター KDC のデータが壊れた場合は、壊れたデータベースが次回の伝播ですべてのスレーブ KDC に伝播されます。また、壊れたバックアップによって、マスター KDC 上の以前の壊れていないバックアップファイルも上書きされます。

このシナリオから保護するには、`cron` ジョブを設定して `slave_datatrans` ダンプファイルを定期的に別の場所にコピーするか、または `kdb5_util` の `dump` コマンドを使用して別の個別のバックアップコピーを作成します。これにより、データベースが壊れた場合は、`kdb5_util` の `load` コマンドを使用してマスター KDC 上の最新のバックアップを復元できます。

次の点も重要です。データベースダンプファイルには主体鍵が含まれているため、許可されないユーザーがアクセスできないように、ファイルを保護する必要があります。デフォルトでは、データベースダンプファイルの読み取り権および書き込み権は、`root` にのみ与えられます。不正なアクセスから保護するには、転送されているデータを暗号化する `kprop` コマンドを使用してデータベースダンプファイルを伝播します。また、`kprop` はデータをスレーブ KDC だけに伝播するため、データベースダンプファイルが間違って許可されないホストに送信される可能性が最小限になります。

例 18 Kerberos データベースの手動でのバックアップ

データベースをバックアップするには、`kdb5_util` コマンドの `dump` コマンドを使用します。このコマンドは、`root` によって所有されているディレクトリで実行します。

```
# /usr/sbin/kdb5_util dump
```

次の例では、Kerberos データベースは `dumpfile` と呼ばれるファイルにバックアップされます。`-verbose` オプションが指定されているため、各主体はバックアップされるときに出力されます。主体が指定されていないため、データベース全体がバックアップされます。

```
# kdb5_util dump -verbose /var/user/kadmin/dumpfile
kadmin/kdc1.corp.example.com@CORP.EXAMPLE.COM
krbtgt/CORP.EXAMPLE.COM@CORP.EXAMPLE.COM
kadmin/history@CORP.EXAMPLE.COM
pak/admin@CORP.EXAMPLE.COM
pak@CORP.EXAMPLE.COM
changepw/kdc1.corp.example.com@CORP.EXAMPLE.COM
```

次の例では、ダンプには `pak` および `pak/admin` 主体のみが含まれます。

```
# kdb5_util dump -verbose pakfile pak/admin@CORP.EXAMPLE.COM pak@CORP.EXAMPLE.COM
pak/admin@CORP.EXAMPLE.COM
pak@CORP.EXAMPLE.COM
```

詳細は、`kdb5_util(1M)` のマニュアルページを参照してください。

▼ Kerberos データベースのバックアップを復元する方法

始める前に KDC マスター上で root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **マスター上で、KDC デーモンを終了します。**

```
kdc1# svcadm disable network/security/krb5kdc
kdc1# svcadm disable network/security/kadmin
```

2. **kdb_util コマンドの load コマンドを使用して、Kerberos データベースを復元します。**

   たとえば、例18「Kerberos データベースの手動でのバックアップ」の dumpfile のバックアップをロードします。

```
# /usr/sbin/kdb5_util load /var/user/kadmin/dumpfile
```

3. **KDC デーモンを起動します。**

```
kdc1# svcadm enable -r network/security/krb5kdc
kdc1# svcadm enable -r network/security/kadmin
```

例 19 Kerberos データベースの復元

次の例では、database1 というデータベースが、dumpfile ファイルから現在のディレクトリに復元されます。-update オプションが指定されていないため、新しいデータベースが作成されます。

```
# kdb5_util load -d database1 dumpfile
```

▼ サーバーのアップグレード後に Kerberos データベースを変換する方法

KDC データベースが古いリリースを実行しているサーバー上で作成された場合は、データベースを変換すると、改善されたデータベースフォーマットを利用できます。

始める前に この手順は、データベースが古い形式を使用している場合にのみ使用します。

KDC マスター上で root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **マスター上で、KDC デーモンを終了します。**

```
kdc1# svcadm disable network/security/krb5kdc
kdc1# svcadm disable network/security/kadmin
```

2. **データベースの一時的なコピーを格納するためのディレクトリを作成します。**

```
kdc1# mkdir /var/krb5/tmp
kdc1# chmod 700 /var/krb5/tmp
```

3. **KDC データベースをダンプします。**

```
kdc1# kdb5_util dump /var/krb5/tmp/prdb.txt
```

4. **現在のデータベースファイルのコピーを保存します。**

```
kdc1# cd /var/krb5
kdc1# mv princ* tmp/
```

5. **データベースをロードします。**

```
kdc1# kdb5_util load /var/krb5/tmp/prdb.txt
```

6. **KDC デーモンを起動します。**

```
kdc1# svcadm enable -r network/security/krb5kdc
kdc1# svcadm enable -r network/security/kadmin
```

▼ マスター KDC を再構成して増分伝播を使用する方法

この手順を使って、増分伝播を使用するように既存のマスター KDC を再構成します。この手順では、次の構成パラメータを使用します。

- レルム名 = `EXAMPLE.COM`
- DNS ドメイン名 = `example.com`
- マスター KDC = `kdc1.example.com`
- スレーブ KDC = `kdc2.example.com`
- `admin` 主体 = `kws/admin`

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **`kdc.conf` ファイルに増分伝播を構成します。**
増分伝播を有効にして、KDC マスターのログ内に保持される更新数を選択します。詳細は、`kdc.conf(4)` のマニュアルページを参照してください。

```
kdc1# pfedit /etc/krb5/kdc.conf
[kdcdefaults]
     kdc_ports = 88,750

[realms]
```

```
EXAMPLE.COM= {
   profile = /etc/krb5/krb5.conf
   database_name = /var/krb5/principal
   acl_file = /etc/krb5/kadm5.acl
   kadmind_port = 749
   max_life = 8h 0m 0s
   max_renewable_life = 7d 0h 0m 0s
   sunw_dbprop_enable = true
   sunw_dbprop_master_ulogsize = 1000
 }
```

2. **kiprop 主体を作成します。**

   kiprop 主体は、マスター KDC サーバーの認証およびマスター KDC からの更新の承認に使用されます。

```
kdc1# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

3. **マスター KDC 上で、kadm5.acl に kiprop エントリを追加します。**

   このエントリにより、マスター KDC が kdc2 サーバーから増分伝播の要求を受け取ることができるようになります。

```
kdc1# pfedit /etc/krb5/kadm5.acl
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

4. **root の crontab ファイル内の kprop 行をコメントにします。**

   この手順により、マスター KDC が KDC データベースのコピーを伝播しなくなります。

```
kdc1# crontab -e
#ident  "@(#)root       1.20    01/11/06 SMI"
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com
```

5. **kadmind を再起動します。**

```
kdc1# svcadm restart network/security/kadmin
```

6. **増分伝播を使用するすべてのスレーブ KDC サーバーを再構成します。**

   完全な手順については、154 ページの「スレーブ KDC を再構成して増分伝播を使用する方法」を参照してください。

▼ スレーブ KDC を再構成して増分伝播を使用する方法

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **kdc.conf にエントリを追加します。**
   最初の新しいエントリにより、増分伝播が有効になります。2 番目の新しいエントリにより、ポーリング時間が 2 分に設定されます。

```
kdc2# pfedit /etc/krb5/kdc.conf
[kdcdefaults]
kdc_ports = 88,750

[realms]
EXAMPLE.COM= {
profile = /etc/krb5/krb5.conf
database_name = /var/krb5/principal
acl_file = /etc/krb5/kadm5.acl
kadmind_port = 749
max_life = 8h 0m 0s
max_renewable_life = 7d 0h 0m 0s
sunw_dbprop_enable = true
sunw_dbprop_slave_poll = 2m
}
```

2. **krb5.keytab ファイルに kiprop 主体を追加します。**

```
kdc2# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.example.com with kvno 3,
encryption type AES-256 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3,
encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc
mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

3. **kpropd を再起動します。**

```
kdc2# svcadm restart network/security/krb5_prop
```

▼ KDC サーバーが同期しているかを検査する方法

増分伝播が構成されている場合は、この手順により、スレーブ KDC に関する情報が更新されたことが確認されます。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **KDC マスターサーバー上で、`kproplog` コマンドを実行します。**

```
kdc1# /usr/sbin/kproplog -h
```

2. **KDC スレーブサーバー上で、`kproplog` コマンドを実行します。**

```
kdc2# /usr/sbin/kproplog -h
```

3. **最後のシリアル番号と最後のタイムスタンプ値が一致することを確認します。**

例 20 KDC サーバーが同期されていることの確認

次の例は、マスター KDC サーバー上での `kproplog` コマンドの実行結果です。

```
kdc1# /usr/sbin/kproplog -h

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
Log version #: 1
Log state: Stable
Entry block size: 2048
Number of entries: 2500
First serial #: 137966
Last serial #: 140465
First time stamp: Wed Dec 4 00:59:27 2013
Last time stamp: Wed Dec 4 01:06:13 2013
```

次の例は、スレーブ KDC サーバー上での `kproplog` コマンドの実行結果です。

```
kdc2# /usr/sbin/kproplog -h

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
Log version #: 1
Log state: Stable
Entry block size: 2048
Number of entries: 0
First serial #: None
Last serial #: 140465
First time stamp: None
Last time stamp: Wed Dec 4 01:06:13 2013
```

最終シリアル番号と最終時間表示が同じであることに注意してください。これは、スレーブがマスター KDC サーバーと同期していることを示しています。

スレーブ KDC サーバーの出力で、スレーブ KDC サーバーの更新ログに更新エントリがないことに注意してください。エントリがないのは、スレーブ KDC サーバーはマスター KDC サーバーとは異なり、一連の更新を保持しないためです。また、最初のシリアル番号や最初のタイムスタンプに関する情報は関連情報ではないため、KDC スレーブサーバーにはこれらの情報も含まれていません。

Kerberos データベースのスレーブ KDC への手動での伝播

cron ジョブは通常、Kerberos データベースをスレーブ KDC に伝播します。定期的な cron ジョブの外部でスレーブ KDC をマスター KDC と同期する必要がある場合は、/usr/lib/krb5/kprop_script と kprop コマンドの 2 つのオプションがあります。詳細は、このスクリプトおよび kprop(1M) のマニュアルページを確認してください。

注意 - スレーブ KDC 上で増分伝播が有効になっている場合は、これらのコマンドを使用しないでください。

▼ Kerberos データベースをスレーブ KDC に手動で伝播する方法

1. **スレーブ KDC 上で増分伝播が有効になっていないことを確認します。**

```
slave# grep sunw_dbprop_enable /etc/krb5/kdc.conf
sunw_dbprop_enable = true
```

2. **この値が `true` である場合は、増分伝播を無効にして `krb5_prop` サービスを再起動します。**

```
slave# cp /etc/krb5/kdc.conf /etc/krb5/kdc.conf.sav
slave# pfedit /etc/krb5/kdc.conf
...
sunw_dbprop_enable = false
...

slave# svcadm restart krb5_prop
```

3. **マスター KDC 上で、次のコマンドのいずれかを使用して、マスター KDC データベースをスレーブ KDC に伝播します。**

   - **`kprop_script` コマンドは、スレーブ KDC を同期する前にデータベースをバックアップします。**

     ```
     master# /usr/lib/krb5/kprop_script slave-KDC
     ```

   - **`kprop` コマンドは、最初に Kerberos データベースの新しいバックアップを作成することなく、現在のデータベースバックアップを伝播します。**

     ```
     master# /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave-KDC
     ```

4. **(オプション) 手動伝播が完了したら、元の `krb5.conf` ファイルを復元します。**

```
slave# mv /etc/krb5/kdc.conf.sav /etc/krb5/kdc.conf
```

Kerberos のための並列伝播の設定

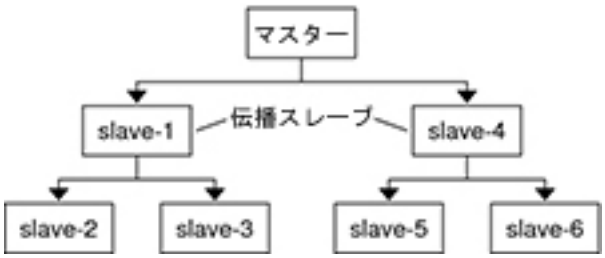
ほとんどの場合、マスター KDC は、Kerberos データベースをスレーブ KDC に伝播するときにだけ使用されます。使用するサイトに複数のスレーブ KDC が存在する場合は、伝播処理の負荷を分散させることもできます。この概念は、「並列伝播」と呼ばれます。

注意 - 増分伝播を使用している場合は、並列伝播を構成しないでください。

並列伝播を使用すると、特定のスレーブ KDC がマスター KDC の伝播の負荷を分散できます。処理を分散すると、伝播をより早く実行でき、マスター KDC の作業を軽減することができます。

たとえば、サイトに 1 つのマスター KDC と 6 つのスレーブ KDC (図12に示されています) があり、`slave-1` から `slave-3` は 1 つの論理グループで構成され、`slave-4` から `slave-6` は別の論理グループで構成されているとします。並列伝播を設定するには、マスター KDC にデータベースを `slave-1` と `slave-4` に伝播させるようにできます。これらのスレーブ KDC がグループ内のスレーブ KDC にデータベースを伝播するようにします。

図 12 Kerberos での並列伝播の構成例



並列伝播を設定するための構成手順

並列伝播を有効にするための大まかな構成手順は次のとおりです。

1. マスター KDC 上で、その `cron` ジョブ内の `kprop_script` エントリを変更して、それ以降の伝播を実行する KDC スレーブ (伝播スレーブ) のみの引数を含むようにします。

2. 各伝播スレーブ上で、その cron ジョブに kprop_script エントリを追加します。これには、伝播するスレーブの引数が含まれている必要があります。並列伝播を正しく行うには、伝播スレーブが新しい Kerberos データベースから伝播されたあとに、cron ジョブが実行されるように設定する必要があります。

注記 - 伝播スレーブが伝播されるために必要な時間は、ネットワーク帯域幅や Kerberos データベースのサイズなどの要因によって異なります。

---

3. 各スレーブ KDC 上で、伝播元の KDC のホスト主体名をその kpropd.acl ファイルに追加することによって、伝播される適切なアクセス権を設定します。

例 21 Kerberos での並列伝播の設定

図12の例を使用すると、マスター KDC の kprop_script エントリは次のようになります。

```
0 3 * * * /usr/lib/krb5/kprop_script slave-1.example.com slave-4.example.com
```

slave-1 の kprop_script エントリは、次のようになります。

```
0 4 * * * /usr/lib/krb5/kprop_script slave-2.example.com slave-3.example.com
```

このスレーブの伝播は、マスターからの伝播が完了してから 1 時間後に開始します。

伝播スレーブの kpropd.acl ファイルには、次のエントリが含まれます。

```
host/master.example.com@EXAMPLE.COM
```

slave-1 から伝播されるスレーブ KDC の kpropd.acl ファイルには、次のエントリが含まれます。

```
host/slave-1.example.com@EXAMPLE.COM
```

Kerberos データベースの stash ファイルの管理

「*stash* ファイル」には、Kerberos データベースのマスター鍵が含まれます。このファイルは、Kerberos データベースを作成すると自動的に作成されます。stash ファイルが壊れた場合は、kdb5_util ユーティリティーの stash コマンドを使用して、置き換えることができます。stash ファイルの削除が必要になるのは、kdb5_util の destroy コマンドを使用して Kerberos データベースを削除したあとだけです。stash ファイルはデータベースとともに自動的に削除されるわけではないため、stash ファイルを手動で削除する必要があります。

stash ファイルを削除するには、rm コマンドを使用します。

```
# rm stash-file
```

この例では、*stash-file* は stash ファイルのパスを示します。デフォルトでは、stash ファイルは /var/krb5/.k5.*realm* にあります。

注記 - stash ファイルを再作成する場合は、kdb5_util コマンドの -f オプションを使用します。

▼ Kerberos データベースの新しいマスター鍵を作成、使用、および格納する方法

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **新しいマスター鍵を作成します。**

   このコマンドでは、ランダムに生成された新しいマスター鍵を追加します。-s オプションは、新しいマスター鍵がデフォルトのキータブに格納されるよう要求します。

```
# kdb5_util add_mkey -s

Creating new master key for master key principal 'K/M@EXAMPLE.COM'
You will be prompted for a new database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     /** Type strong password **/
Re-enter KDC database master key to verify: xxxxxxxx
```

2. **新しいマスター鍵が存在することを確認します。**

```
# kdb5_util list_mkeys
Master keys for Principal: K/M@EXAMPLE.COM
KVNO: 2, Enctype: AES-256 CTS mode with 96-bit SHA-1 HMAC, No activate time set
KVNO: 1, Enctype: AES-128 CTS mode with 96-bit SHA-1 HMAC, Active on: Fri Dec 31 18:00:00
 CST 2011 *
```

   この出力内のアスタリスクは、現在アクティブになっているマスター鍵を特定します。

3. **新しく作成されたマスター鍵がアクティブになる時間を設定します。**

```
# date
Fri Oct 9 17:57:00 CDT 2015
# kdb5_util use_mkey 2 'now+2days'
# kdb5_util list_mkeys
Master keys for Principal: K/M@EXAMPLE.COM
KVNO: 2, Enctype: AES-256 CTS mode with 96-bit SHA-1 HMAC,
Active on: Fri Oct 9 17:57:15 CDT 2015
KVNO: 1, Enctype: AES-128 CTS mode with 96-bit SHA-1 HMAC,
Active on: Fri Dec 31 18:00:00 CST 2011 *
```

   この例では、新しいマスター鍵がすべての KDC に伝播される時間を見越して、日付は 2 日後に設定されています。この日付は、環境に合わせて適宜調整してください。

4. **(オプション) 新しい主体の作成後、新しいマスター鍵が使用されていることを確認します。**

```
# kadmin.local -q 'getprinc tamiko' | egrep 'Principal|MKey'
Authenticating as principal root/admin@EXAMPLE.COM with password.
Principal: tamiko@EXAMPLE.COM
MKey: vno 2
```

この例では、MKey: vno 2 は、新しく作成されたマスター鍵 2 によって主体の秘密鍵が保護されていることを示しています。

5. **新しいマスター鍵を使ってユーザー主体の秘密鍵を再暗号化します。**

コマンドの終わりにパターンの引数を追加した場合は、そのパターンに一致する主体が更新されます。更新される主体を特定するには、このコマンドに -n オプションを追加します。

```
# kdb5_util update_princ_encryption -f -v
Principals whose keys WOULD BE re-encrypted to master key vno 2:
updating: host/kdc1.example.com@EXAMPLE.COM
skipping:   tamiko@EXAMPLE.COM
updating: kadmin/changepw@EXAMPLE.COM
updating: kadmin/history@EXAMPLE.COM
updating: kdc/admin@EXAMPLE.COM
updating: host/kdc2.example.com@EXAMPLE.COM
6 principals processed: 5 updated, 1 already current
```

6. **古いマスター鍵を削除します。**

主体の秘密鍵の保護にマスター鍵が使用されなくなった場合は、マスター鍵主体からそれを削除できます。いずれかの主体で引き続き使用されている鍵はこのコマンドで削除されません。適切なマスター鍵が削除されることを確認するには、このコマンドに -n オプションを追加します。

```
# kdb5_util purge_mkeys -f -v
Purging the following master key(s) from K/M@EXAMPLE.COM:
KVNO: 1
1 key(s) purged.
```

7. **古いマスター鍵が削除されていることを確認します。**

```
# kdb5_util list_mkeys
Master keys for Principal: K/M@EXAMPLE.COM
KVNO: 2, Enctype: AES-256 CTS mode with 96-bit SHA-1 HMAC,
Active on: Sun Oct 4 17:57:15 CDT 2015 *
```

8. **stash ファイルを更新します。**

```
# kdb5_util stash
Using existing stashed keys to update stash file.
```

9. **stash ファイルが更新されていることを確認します。**

```
# klist -kt /var/krb5/.k5.EXAMPLE.COM
Keytab name: FILE:.k5.EXAMPLE.COM
KVNO Timestamp              Principal
---- --------------- ------------------------------------------------------------
2 10/11/2015 18:03 K/M@EXAMPLE.COM
```

Kerberos サーバー上のセキュリティーの強化

このセクションでは、Kerberos アプリケーションサーバーおよび KDC サーバー上のセキュリティーの強化に関するアドバイスを提供します。

KDC サーバーへのアクセスの制限

マスター KDC およびスレーブ KDC には、KDC データベースのローカルコピーがあります。データベースを保護するためにこれらのサーバーへのアクセス権を制限することは、Kerberos 全体のセキュリティーにとって重要です。

- KDC をサポートするハードウェアへの物理的なアクセスを制限します。

  KDC サーバーとそのモニターがセキュアな施設内に設置されていることを確認してください。通常のユーザーがどのような方法でもこのサーバーにアクセスできてはいけません。
- KDC データベースのバックアップを、ローカルディスクまたはスレーブ KDC に格納します。

  KDC のバックアップをテープに作成する場合、そのテープのセキュリティーを十分に確保してください。キータブファイルのコピーも、同様に作成します。

  推奨される方法として、これらのファイルをほかのシステムとは共有されていないローカルファイルシステム上に格納します。格納先のファイルシステムは、マスター KDC または任意のスレーブ KDC から選択できます。

辞書ファイルを使用したパスワードセキュリティーの強化

Kerberos サービスで辞書ファイルを使用すると、その辞書にある単語が新しい資格のパスワードとして使用されることを防止できます。辞書の用語がパスワードとして使用されないようにすると、パスワードの推測が困難になります。デフォルトでは、`/var/krb5/kadm5.dict` ファイルが使用されますが、これは空です。

KDC 構成ファイル `kdc.conf` に、このサービスに辞書ファイルを使用することを指示するための行を追加する必要があります。この例では、管理者は `spell` ユーティリティーに含まれている辞書を使用したあと、Kerberos サービスを再起動します。構成ファイルの詳細は、`kdc.conf(4)` のマニュアルページを参照してください。

```
kdc1# pfedit /etc/krb5/kdc.conf
[kdcdefaults]
     kdc_ports = 88,750
```

```
[realms]
     EXAMPLE.COM = {
        profile = /etc/krb5/krb5.conf
        database_name = /var/krb5/principal
        acl_file = /etc/krb5/kadm5.acl
        kadmind_port = 749
        max_life = 8h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        sunw_dbprop_enable = true
        sunw_dbprop_master_ulogsize = 1000
        dict_file = /usr/share/lib/dict/words
     }
kdc1#
kdc1# svcadm restart -r network/security/krb5kdc
kdc1# svcadm restart -r network/security/kadmin
```

♦ ♦ ♦ 第 5 章

5 Kerberos 主体とポリシーの管理

この章では、主体とそれに関連するポリシーを管理する手順について説明します。また、ホストのキータブファイルの管理方法についても説明します。

この章で扱う内容は、次のとおりです。

- 163 ページの「Kerberos 主体とポリシーの管理方法」
- 164 ページの「Kerberos 主体の管理」
- 168 ページの「Kerberos ポリシーの管理」
- 170 ページの「keytab ファイルの管理」

主体およびポリシーに関する背景情報については、第2章「Kerberos サービスについて」および第3章「Kerberos サービスの計画」を参照してください。

Kerberos 主体とポリシーの管理方法

マスター KDC 上の Kerberos データベースには、レルムの Kerberos 主体、そのパスワード、ポリシーなどの管理情報がすべて含まれています。主体を作成および削除したり、その属性を変更したりするには、`kadmin` コマンドを使用します。

`kadmin` コマンドには、対話型のコマンド行インタフェースが用意されています。このインタフェースを使用して、Kerberos 主体、ポリシー、およびキータブファイルを管理することができます。また、主体の作成を自動化するスクリプトを実行することもできます。`kadmin` コマンドには、ローカルバージョンとリモートバージョンがあります。

- `kadmin` – Kerberos 認証を使用して、ネットワーク上の任意の場所から安全に操作できます
- `kadmin.local` – マスター KDC 上で直接実行する必要があります

この 2 つのバージョンの機能は同じです。リモートバージョンを使用するには、その前にローカルバージョンを使用してデータベースを十分に構成しておく必要があります。

このセクションでは、kadmin.local コマンドのスクリプト機能について説明します。

新しい Kerberos 主体の自動作成

スクリプト内で kadmin.local コマンドを使用すると、新しい Kerberos 主体の作成を自動化できます。自動化は、データベースに多数の新しい主体を追加する場合に役立ちます。

次のシェルスクリプト行は、新しい主体の作成を自動化する方法を示しています。

```
awk '{ print "ank +needchange -pw", $2, $1 }' < /tmp/princnames |
time /usr/sbin/kadmin.local> /dev/null
```

前の例は、読みやすさのために 2 行に分割されています。

- このスクリプトは、princnames という名前のファイルを読み込みます。このファイルには主体名とそのパスワードが含まれており、それらを Kerberos データベースに追加します。

  princnames ファイルをあらかじめ作成する必要があります。このファイルの各行には、主体とそのパスワードを 1 つ以上の空白で区切って指定します。
- ank コマンドは、新しい鍵を追加します。ank は、add_principal コマンドの別名です。
- +needchange オプションは、その主体であるユーザーが最初のログイン時に新しいパスワードの入力を求められるように主体を構成します。

  パスワード変更を求めることは、princnames ファイル内のパスワードがセキュリティーリスクにならないようにするのに役立ちます。

より複雑なスクリプトも作成できます。たとえば、ネームサービスの情報を使用して、主体名に対応するユーザー名の一覧を取得できます。必要な作業とその方法は、使用環境要件とスクリプト使用技術によって決まります。

Kerberos 主体の管理

このセクションでは、kadmin コマンドを使用して主体を管理する例を示します。詳細は、kadmin(1M) のマニュアルページを参照してください。

Kerberos 主体とその属性の表示

次の例は、主体とその属性を一覧表示する方法を示しています。ワイルドカードを使用してリストを構築できます。使用できるワイルドカードについては、kadmin(1M) のマニュアルページにある expression の定義を確認してください。

例 22 Kerberos 主体の表示

この例では、list_principals サブコマンドを使用して、kadmin* に一致するすべての主体を一覧表示します。引数を指定しない場合、list_principals は、Kerberos データベースで定義されているすべての主体を一覧表示します。

```
# /usr/sbin/kadmin
kadmin: list_principals kadmin*
kadmin/changepw@EXAMPLE.COM
kadmin/kdc1.example.com@EXAMPLE.COM
kadmin/history@EXAMPLE.COM
```

例 23 Kerberos 主体の属性の表示

次の例では、jdb/admin 主体の属性を表示します。

```
kadmin: get_principal jdb/admin
Principal: jdb/admin@EXAMPLE.COM

Expiration date: [never]
Last password change: [never]

Password expiration date: Fri Sep 13 11:50:10 PDT 2013
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
Last modified: Thu Aug 15 13:30:30 PST 2013 (host/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 1
Key: vno 1, AES-256 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 1, AES-128 CTS mode with 96-bit SHA-1 HMAC, no salt
Key: vno 1, Triple DES with HMAC/sha1, no salt
Key: vno 1, ArcFour with HMAC/md5, no salt
Attributes: REQUIRES_HW_AUTH
Policy: [none]
kadmin: quit
```

新しい Kerberos 主体の作成

新しい主体に新しいポリシーが必要な場合は、主体を作成する前に新しいポリシーを作成する必要があります。ポリシーの作成については、例30「新しい Kerberos パスワードポリシーの作成」を参照してください。ほとんどの Kerberos ポリシーでパスワード要件が指定されます。

例 24 新しい Kerberos 主体の作成

次の例では、pak という名前の新しい主体を作成し、その主体のポリシーを testuser に設定します。その他の必要な値 (暗号化タイプなど) には、デフォルト値を使用します。

```
# /usr/sbin/kadmin
kadmin: add_principal -policy testuser pak
Enter password for principal "pak@EXAMPLE.COM": xxxxxxxx
Re-enter password for principal "pak@EXAMPLE.COM": xxxxxxxx
Principal "pak@EXAMPLE.COM" created.
kadmin: quit
```

通常、Kerberos データベースを管理する権限は少数のユーザーにしか与えられません。この新しい主体に管理権限が必要な場合は、167 ページの「主体の Kerberos 管理権限の変更」に進んでください。

Kerberos 主体の変更

次の例は、Kerberos 主体のパスワード属性を変更する方法を示しています。

例 25 Kerberos 主体のパスワード再試行の最大数の変更

```
# /usr/sbin/kadmin
kadmin: modify_principal jdb
kadmin: maxtries=5
kadmin: quit
```

例 26 Kerberos 主体のパスワードの変更

```
# /usr/sbin/kadmin
kadmin: change_password jdb
Enter password for principal "jdb": xxxxxxxx
Re-enter password for principal "jdb": xxxxxxxx
Password for "jdb@EXAMPLE.COM" changed.
kadmin: quit
```

Kerberos 主体の削除

次の例は、主体を削除する方法を示しています。続行する前に、オンラインメッセージに従ってください。

```
# /usr/sbin/kadmin
kadmin: delete_principal pak
Are you sure you want to delete the principal "pak@EXAMPLE.COM"? (yes/no): yes
Principal "pak@EXAMPLE.COM" deleted.
Make sure that you have removed this principal from all ACLs before reusing.
```

```
kadmin: quit
```

すべての ACL からこの主体を削除するには、167 ページの「主体の Kerberos 管理権限の変更」を参照してください。

主体の Kerberos 管理権限の変更

Kerberos データベースを管理する権限が与えられている少数のユーザーは、Kerberos アクセス制御リスト (ACL) で指定されます。このリストは、ファイル `/etc/krb5/kadm5.acl` 内のエントリとして保存されます。詳細は、`kadm5.acl(4)` のマニュアルページを参照してください。

`kadm5.acl` ファイルにエントリを追加するには、`pfedit` コマンドを使用します。

```
# pfedit /usr/krb5/kadm5.acl
```

`kadm5.acl` ファイル内のエントリの形式は次のとおりです。

principal privileges [*principal-target*]

- *principal* – 権限を与える主体を指定します。主体名の任意の部分に「`*`」のワイルドカードを含めることができます。このワイルドカードは、主体のグループに同じ権限を指定するときに役立ちます。たとえば、`admin` インスタンスを持つすべての主体を指定する場合は、`*/admin@`*realm* を使用します。

  `admin` インスタンスの一般的な使用法は、個別の Kerberos 主体に個別の権限 (Kerberos データベースへの管理アクセスなど) を与えるためであることに注意してください。たとえば、ユーザー `jdb` が `jdb/admin` という名前の、管理使用のための主体を持っている場合があります。2 つの主体を持つことにより、ユーザー `jdb` は、管理権限が必要な場合にのみ `jdb/admin` チケットを取得します。
- *privileges* – 主体が実行できる操作を指定します。このフィールドは、次のリストにある 1 つ以上の文字の文字列で構成されます。文字が大文字であるか、または指定されていない場合、その操作は許可されません。文字が小文字である場合、その操作は許可されます。
  - [A]a – 主体またはポリシーの追加を許可します[しません]。
  - [C]c – 主体のパスワードの変更を許可します[しません]。
  - [D]d – 主体またはポリシーの削除を許可します[しません]。
  - [I]i – Kerberos データベースへの照会を許可します[しません]。
  - [L]l – 主体またはポリシーの一覧表示を許可します[しません]。
  - [M]m – 主体またはポリシーの変更を許可します[しません]。
  - `x` または `*` – すべての権限 (`admcil`) を許可します。
- *principal-target* – このフィールドで主体が指定されている場合は、主体の権限がこの主体にのみ適用されます。主体のグループに権限を割り当てるには、*principal-target* で「`*`」のワイルドカードを使用します。

例 27 Kerberos 主体の権限の変更

kadm5.acl ファイル内の次のエントリは、admin インスタンスを持つ EXAMPLE.COM レルム内の任意の主体に Kerberos データベースに関するすべての特権を与えます。

```
*/admin@EXAMPLE.COM *
```

kadm5.acl ファイル内の次のエントリは、jdb@EXAMPLE.COM 主体に、root インスタンスを持つすべての主体の一覧表示とそれらの主体に関する照会を行う権限を与えます。

```
jdb@EXAMPLE.COM li */root@EXAMPLE.COM
```

Kerberos ポリシーの管理

このセクションでは、kadmin コマンドを使用して Kerberos ポリシーを管理する例を示します。ほとんどの Kerberos ポリシーでパスワード要件が指定されます。

ポリシーを管理するための手順は、主体を管理するための手順とほぼ同じです。詳細は、kadmin(1M) のマニュアルページを参照してください。

例 28 Kerberos ポリシーのリストの表示

この例では、list_policies サブコマンドを使用して、*user* に一致するすべてのポリシーを一覧表示します。引数を指定しない場合、list_policies は、Kerberos データベースで定義されているすべてのポリシーを一覧表示します。

```
# kadmin
kadmin: list_policies *user*
testuser
financeuser
kadmin: quit
```

例 29 Kerberos ポリシーの属性の表示

この例では、get_policy サブコマンドを使用して、financeuser ポリシーの属性を表示します。

```
# /usr/sbin/kadmin.local
kadmin.local: get_policy financeuser
Policy: financeuser
Maximum password life: 13050000
Minimum password life: 10886400
Minimum password length: 8
Minimum number of password character classes: 2
Number of old keys kept: 3
Reference count: 8
Maximum password failures before lockout: 5
Password failure count reset interval: 200
Password lockout duration: 300
```

```
kadmin: quit
```

「Reference count」は、このポリシーが割り当てられている主体の数です。

例 30 新しい Kerberos パスワードポリシーの作成

この例では、add_policy サブコマンドを使用して build11 ポリシーを作成します。このポリシーでは、パスワード内に少なくとも 3 種類以上の文字クラスが必要です。

```
# kadmin
kadmin: add_policy -minclasses 3 build11
kadmin: quit
```

例 31 Kerberos アカウントのロックアウトポリシーの処理

この例では、300 秒の間に認証失敗が 3 回発生すると、900 秒のアカウントのロックアウトがトリガーされます。

```
kadmin: add_policy -maxfailure 3 -failurecountinterval "300 seconds"\
-lockoutduration "900 seconds" default
```

15 分以内にロックを解除するには、管理アクションが必要になります。

```
# /usr/sbin/kadmin -p kws/admin
Enter password: xxxxxxxx
kadmin: modify_principal -unlock principal
```

例 32 Kerberos ポリシーの変更

この例では、modify_policy サブコマンドを使用して、build11 ポリシーでの最小パスワード長を 8 文字に変更します。

```
# kadmin
kadmin: modify_policy -minlength 8 build11
kadmin: quit
```

例 33 Kerberos ポリシーの削除

この例では、delete_policy サブコマンドを使用して build11 ポリシーを削除します。

1. 管理者はそのポリシーを、それを使用するすべての主体から削除します。

   ```
   # kadmin
   kadmin: modify_principal -policy build11 *admin*
   ```

2. 次に、管理者はそのポリシーを削除します。

   ```
   kadmin: delete_policy build11
   Are you sure you want to delete the policy "build11"? (yes/no): yes
   kadmin: quit
   ```

そのポリシーが主体に割り当てられている場合は、`delete_policy` コマンドが失敗します。

keytab ファイルの管理

サービスを提供するすべてのホストには、キータブファイル (「鍵テーブル」の短縮名) と呼ばれるローカルファイルが必要です。キータブには、「サービス鍵」と呼ばれる該当するサービスの主体が格納されます。サービス鍵サービス鍵は、KDC に対してサービス自身を認証するときに使用され、Kerberos とそのサービスだけが認識します。たとえば、Kerberos NFS サーバーを使用している場合は、そのサーバーに、`nfs` サービス主体のサービス鍵を含む keytab ファイルが必要です。

keytab ファイルにサービス鍵を追加するには、`kadmin` プロセスで `ktadd` コマンドを使用して、適切なサービス主体をホストの keytab ファイルに追加します。サービス主体を keytab ファイルに追加しているため、その主体がすでに Kerberos データベース内に存在する必要があります。Kerberos サービスを提供するアプリケーションサーバーの場合、keytab ファイルは、デフォルトでは `/etc/krb5/krb5.keytab` です。

キータブはユーザーのパスワードに似ています。ユーザーが自分のパスワードを保護する必要があるのと同様に、アプリケーションサーバーも自身の keytab ファイルを保護する必要があります。キータブファイルは常時ローカルディスクに格納し、`root` ユーザー以外は読み取れないようにしてください。また、キータブファイルは、セキュリティー保護されていないネットワークを介して送信しないでください。

特殊な状況では、ホストの keytab ファイルに `root` 主体を追加することが必要になる場合があります。Kerberos クライアント上のユーザーが、`root` と同等のアクセスを必要とする Kerberos NFS ファイルシステムをマウントできるようにする場合は、そのクライアントの `root` 主体をクライアントの keytab ファイルに追加する必要があります。そうしない場合は、`root` アクセスで Kerberos NFS ファイルシステムをマウントするたびに、ユーザーは `kinit` コマンドを `root` として使用して、クライアントの `root` 主体の資格を取得する必要があります。これは、オートマウンタを使用している場合でも同様です。

注意 - `root` としての NFS サーバーのマウントはセキュリティーリスクになります。

`ktutil` コマンドを使用して keytab ファイルを管理することもできます。`kadmin` とは異なり、このコマンドは Kerberos データベースとは対話しないため、この対話型のコマンドを使用すると、Kerberos 管理権限を持っていなくてもローカルホストの keytab ファイルを管理できます。主体が keytab ファイルに追加されたら、`ktutil` を使用して keytab ファイル内の鍵リストを表示したり、サービスの認証を一時的に無効にしたりできます。

注記 - kadmin の ktadd コマンドを使用して keytab ファイル内の主体を変更すると、新しい鍵が生成され、keytab ファイルに追加されます。

keytab ファイルへの Kerberos サービス主体の追加

主体が Kerberos データベース内に存在することを確認したら、その主体を keytab ファイルに追加できます。詳細は、165 ページの「Kerberos 主体とその属性の表示」を参照してください。

keytab ファイルに主体を追加する必要があるホスト上で、kadmin プロセスで ktadd コマンドを実行します。詳細は、kadmin(1M) のマニュアルページを参照してください。

```
# /usr/sbin/kadmin
kadmin: ktadd [-e enctype] [-k keytab] [-q] [principal | -glob principal-exp]
```

-e *enctype*	krb5.conf ファイルで定義された暗号化タイプの一覧をオーバーライドします。
-k *keytab*	キータブファイルを指定します。デフォルトでは、/etc/krb5/krb5.keytab が使用されます。
-q	冗長な情報を表示しません。
principal	キータブファイルに追加する主体を指定します。host、root、nfs、ftp の各サービス主体を追加できます。
-glob *principal-exp*	主体表現を指定します。*principal-exp* に一致するすべての主体が、キータブファイルに追加されます。主体表現の規則は、kadmin の list_principals コマンドと同じです。使用できる式については、kadmin(1M) のマニュアルページにある expression の定義を確認してください。

例 34 サービス主体のキータブファイルへの追加

この例では、KDC が denver のネットワークサービスを認証できるように、denver の host 主体が denver の keytab ファイルに追加されます。

```
denver # /usr/sbin/kadmin
kadmin: ktadd host/denver.example.com
Entry for principal host/denver.example.com with kvno 3,
encryption type AES-256 CTS
mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3,
encryption type AES-128 CTS mode
with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3,
```

```
encryption type Triple DES cbc mode
with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

キータブファイルからのサービス主体の削除

keytab ファイルから主体を削除できます。keytab ファイルから主体を削除する必要があるホスト上で、まず主体のリストを表示します。172 ページの「keytab ファイル内の主体の表示」を参照してください。

次に、`kadmin` プロセスで `ktadd` コマンドを実行します。詳細は、`kadmin(1M)` のマニュアルページを参照してください。

```
# /usr/sbin/kadmin
kadmin: ktremove [-k keytab] [-q] principal [kvno | all | old ]
```

-k *keytab*	キータブファイルを指定します。デフォルトでは、`/etc/krb5/krb5.keytab` が使用されます。
`-q`	冗長な情報を表示しません。
principal	キータブファイルから削除する主体を指定します。
kvno	指定された主体のうち、鍵のバージョン番号が *kvno* と一致する主体のすべてのエントリを削除します。
`all`	指定された主体のすべてのエントリを削除します。
`old`	指定された主体のすべてのエントリを削除します。ただし、鍵バージョン番号がもっとも大きい主体を除きます。

例 35 キータブファイルからのサービス主体の削除

この例では、`denver` の keytab ファイルから `denver` の `host` 主体が削除されます。

```
denver # /usr/sbin/kadmin
kadmin: ktremove host/denver.example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver.example.com@EXAMPLE.COM with kvno 3
removed from keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

keytab ファイル内の主体の表示

keytab ファイルを含むホスト上で、`ktutil` コマンドを実行し、keytab を読み取ってから主体を一覧表示します。主体は、鍵リストとも呼ばれます。

注記 - ほかのユーザーが所有するキータブファイルを作成することもできますが、キータブファイルのデフォルトの位置を使用するには root 所有権が必要です。

例 36 キータブファイル内のキー一覧 (主体) の表示

次の例では、denver ホスト上の /etc/krb5/krb5.keytab ファイル内の鍵リストを表示します。

```
denver # /usr/bin/ktutil
ktutil: read_kt /etc/krb5/krb5.keytab
ktutil: list
slot KVNO Principal
---- ---- ---------------------------------------
1    5 host/denver@EXAMPLE.COM
ktutil: quit
```

ホスト上の Kerberos サービスの一時的な無効化

ネットワークアプリケーションサーバー上のサービス (ssh や ftp など) のための認証メカニズムを一時的に無効にすることが必要になる場合があります。たとえば、保守作業中は、ユーザーがシステムにログインできないようにする必要があります。

注記 - デフォルトでは、ほとんどのサービスが認証を必要とします。サービスが認証を必要としない場合は、認証を無効にしても影響はありません。そのサービスは引き続き使用できます。

▼ ホスト上の Kerberos サービスの認証を一時的に無効にする方法

ktutil コマンドを使用すると、kadmin 権限のないユーザーでもサービスを無効にできます。このユーザーはまた、サービスを復元することもできます。詳細は、ktutil(1) のマニュアルページを参照してください。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **現在のキータブファイルを一時ファイルに保存します。**
   この一時ファイルは、ステップ 9 で認証を再度有効にするために使用します。

2. **keytab ファイルを含むホスト上で、ktutil コマンドを起動します。**

注記 - ほかのユーザーが所有するキータブファイルを作成することもできますが、キータブファイルのデフォルトの位置を使用するには root 所有権が必要です。

```
# /usr/bin/ktutil
```

3. **keytab ファイルを鍵リストバッファーに読み込みます。**

   ```
   ktutil: read_kt keytab
   ```

4. **鍵リストバッファーを表示します。**

   ```
   ktutil: list
   ```

   現在のキー一覧バッファーが表示されます。無効にするサービスのスロット番号を記録します。

5. **鍵リストバッファーから特定のサービス主体を削除することによって、ホストのサービスを一時的に無効にします。**

   ```
   ktutil: delete_entry slot-number
   ```

   ここで、*slot-number* は、list の出力にある削除されるサービス主体のスロット番号を指定します。

6. **鍵リストバッファーを新しい keytab ファイルに書き込みます。**

   ```
   ktutil: write_kt new-keytab
   ```

7. **ktutil コマンドを終了します。**

   ```
   ktutil: quit
   ```

8. **新しい keytab ファイルを使用して、主体の認証を無効にします。**

   ```
   # mv new-keytab keytab
   ```

9. **(オプション) サービスを再度有効にするには、一時的な keytab ファイルを元の場所にコピーします。**

   ```
   # cp original-keytab keytab
   ```

例 37 Kerberos ホストの一時的な無効化

この例では、denver ホスト上の host サービスが一時的に無効になります。denver 上のホストサービスを再度有効にするため、管理者は保存されたキータブファイルを元の場所にコピーします。

```
denver # cp /etc/krb5/krb5.keytab /etc/krb5/krb5.keytab.save
denver # /usr/bin/ktutil
ktutil:read_kt /etc/krb5/krb5.keytab
ktutil:list
slot KVNO Principal
```

```
---- ---- ---------------------------------------
1    8 root/denver@EXAMPLE.COM
2    5 host/denver@EXAMPLE.COM
ktutil:delete_entry 2
ktutil:list
slot KVNO Principal
---- ---- ---------------------------------------
1    8 root/denver@EXAMPLE.COM
ktutil:write_kt /etc/krb5/nodenverhost.krb5.keytab
ktutil: quit
denver # cp /etc/krb5/nodenverhost.krb5.keytab /etc/krb5/krb5.keytab
```

ホストは、保存されているファイルをユーザーが元の場所にコピーするまで使用できません。

```
denver # cp /etc/krb5/krb5.keytab.save /etc/krb5/krb5.keytab
```

♦ ♦ ♦ 第 6 章

6 Kerberos アプリケーションの使用

この章では、Oracle Solaris で提供されている Kerberos に基づくコマンドおよびサービスを使用する方法について説明します。この章のこれらのコマンドの説明を読み進める前に、読者が各コマンドの元のバージョンをすでに熟知していることが求められます。

この章はアプリケーションユーザーを対象にしているため、チケットの取得、表示、破棄など、チケットに関する情報が含まれています。この章にはまた、Kerberos パスワードの選択や変更に関する情報も含まれています。

この章で扱う内容は、次のとおりです。

- 177 ページの「Kerberos チケットの管理」
- 180 ページの「Kerberos パスワードの管理」
- 182 ページの「Kerberos のユーザーコマンド」

Kerberos の概要については、第2章「Kerberos サービスについて」を参照してください。

Kerberos チケットの管理

このセクションでは、チケットの取得、表示、および破棄を行う方法を説明します。チケットの概要については、46 ページの「Kerberos サービスの動作」を参照してください。

Oracle Solaris では、Kerberos は `login` コマンドに組み込まれています。ただし、チケットを自動的に取得するには、PAM サービスを該当するログインサービス用に構成する必要があります。詳細は、`pam_krb5(5)` のマニュアルページを参照してください。

`ssh` コマンドを設定するとチケットのコピーをほかのシステムに転送できるため、これらのシステムへのアクセスを取得するためにチケットを明示的に要求する必要はありません。セキュリティー上の理由から、管理者がこれを防止している可能性があります。詳細は、`ssh(1)` のマニュアルページにあるエージェント転送に関する説明を参照してください。

チケットの有効期限については、190 ページの「チケットの有効期限」を参照してください。

Kerberos チケットの作成

PAM が正しく構成されている場合は、ログインするとチケットが自動的に作成されるため、チケットを取得するために特殊な操作を行う必要はありません。ただし、チケットが期限切れになった場合は、チケットを作成する必要があります。また、デフォルトの主体に加えて別の主体を使用することが必要になる場合 (`ssh -l` を使用してほかのユーザーとしてシステムにログインする場合など) もあります。

チケットを作成するには、`kinit` コマンドを使用します。

```
% /usr/bin/kinit
```

`kinit` コマンドからはパスワードの入力を求めるプロンプトが表示されます。`kinit` コマンドの完全な構文については、`kinit(1)` のマニュアルページを参照してください。

例 38 Kerberos チケットの作成

この例は、自分のシステム上でチケットを作成しているユーザー `kdoe` を示しています。

```
% kinit
Password for kdoe@CORP.EXAMPLE.COM: xxxxxxxx
```

この例では、ユーザー `kdoe` が `-l` オプションを使用して 3 時間有効なチケットを作成します。

```
% kinit -l 3h kdoe@EXAMPLE.ORG
Password for kdoe@EXAMPLE.ORG: xxxxxxxx
```

Kerberos チケットの表示

すべてのチケットが同じ属性を持つわけではありません。たとえば、1 つのチケットが**転送可能**であり、別のチケットが**遅延**であり、3 番目のチケットが転送可能と遅延の両方である場合があります。現在のチケットが何で、どのような属性を持つかを知るには、`klist` コマンドで `-f` オプションを使用します。

```
% /usr/bin/klist -f
```

次の記号はチケットに関連付けられる属性です。`klist` によって表示されます。

A 事前認証済み (Preauthenticated)

D	遅延可能 (Postdatable)
d	遅延 (Postdated)
F	転送可能 (Forwardable)
f	転送済み (Forwarded)
I	初期 (Initial)
i	無効 (Invalid)
P	プロキシ可能 (Proxiable)
p	プロキシ (Proxy)
R	更新可能 (Renewable)

チケットに指定できる属性については、188 ページの「チケットの種類」を参照してください。

例 39 Kerberos チケットの表示

この例は、ユーザー kdoe が、転送可能 (F) かつ遅延 (d) である初期チケットを持っているが、まだ検証されていない (i) ことを示しています。

```
% /usr/bin/klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: kdoe@EXAMPLE.COM

Valid starting                Expires                Service principal
09 Feb 14 15:09:51  09 Feb 14 21:09:51  nfs/EXAMPLE.COM@EXAMPLE.COM
renew until 10 Feb 14 15:12:51, Flags: Fdi
```

次の例は、ユーザー kdoe が、別のホストからそのユーザーのホストに転送済み (f) であった 2 つのチケットを持っていることを示しています。これらのチケットは転送可能 (F) です。

```
% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: kdoe@EXAMPLE.COM

Valid starting                Expires                Service principal
07 Feb 14 06:09:51  09 Feb 14 23:33:51  host/EXAMPLE.COM@EXAMPLE.COM
renew until 10 Feb 14 17:09:51, Flags: fF

Valid starting                Expires                Service principal
08 Feb 14 08:09:51  09 Feb 14 12:54:51  nfs/EXAMPLE.COM@EXAMPLE.COM
renew until 10 Feb 14 15:22:51, Flags: fF
```

次の例は、-e オプションを使用してセッション鍵の暗号化タイプとチケットを表示させる方法を示しています。-a オプションは、ネームサービスが変換を実行できる場合に、システムアドレスをシステム名にマップするために使用されます。

```
% klist -fea
Ticket cache: /tmp/krb5cc_74287
Default principal: kdoe@EXAMPLE.COM

Valid starting                Expires                Service principal
07 Feb 14 06:09:51  09 Feb 14 23:33:51  krbtgt/EXAMPLE.COM@EXAMPLE.COM
renew until 10 Feb 14 17:09:51, Flags: FRIA
Etype(skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC
Addresses: client.example.com
```

Kerberos チケットの破棄

現在のセッション中に取得したすべての Kerberos チケットを破棄するには、kdestroy コマンドを使用します。このコマンドは資格キャッシュを破棄します。これにより、すべての資格とチケットが破棄されます。この破棄は通常は必要ありませんが、kdestroy を実行すると、ログインしていない期間中に資格キャッシュが危険にさらされる可能性が低くなります。

チケットを破棄するには、kdestroy コマンドを使用します。

```
% /usr/bin/kdestroy
```

kdestroy コマンドは、そのユーザーのすべてのチケットを破棄します。このコマンドを使用して、特定のチケットを選択して破棄することはできません。

システムから離れようとしている場合は、kdestroy コマンドを使用するか、またはスクリーンセーバーで画面をロックするようにしてください。

Kerberos パスワードの管理

Kerberos サービスが構成されている場合、ユーザーは、通常の Oracle Solaris パスワードと Kerberos パスワードの 2 つのパスワードを持っています。これらのパスワードは同じでも、異なっていても構いません。

パスワードの変更

PAM が正しく構成されている場合、Kerberos パスワードは 2 つの方法で変更できます。

- passwd コマンドを使用します。Kerberos サービスが構成されていると、passwd コマンドでも新しい Kerberos パスワードを求めるプロンプトが自動的に表示されます。

`passwd` を使用すると、UNIX パスワードと Kerberos パスワードの両方を一度に設定できます。ただし、`passwd` で変更できるパスワードは 1 つだけであり、ほかのパスワードは変更されないままにします。

注記 - `passwd` コマンドの動作は、PAM モジュールの構成方法によって異なります。構成によっては、両方のパスワードを変更しなければならない場合があります。あるサイトでは UNIX パスワードの変更が必須であり、別のサイトでは Kerberos パスワードの変更が必須であるという場合があります。

- `kpasswd` コマンドを使用します。`kpasswd` は、Kerberos パスワードのみを変更します。UNIX パスワードを変更する場合は、`passwd` コマンドを使用する必要があります。

  `kpasswd` の主な使用法として、有効な UNIX ユーザーではない Kerberos 主体のパスワードの変更があります。たとえば、`jdoe/admin` は Kerberos 主体であるが、実際の UNIX ユーザーではないため、そのパスワードは `kpasswd` を使用して変更する必要があります。

パスワードを変更したら、そのパスワードをネットワーク経由で伝播する必要があります。伝播に必要な時間は、Kerberos ネットワークのサイズに応じて、数分から 1 時間以上まで変動することがあります。パスワードを変更したあとすぐに新しい Kerberos チケットを取得する場合は、新しいパスワードをまず試してください。新しいパスワードが有効でない場合は、以前のパスワードを使用して再度試してください。

Kerberos ポリシーによって、パスワードに関する基準が定義されます。ユーザーごとにポリシーを設定することも、デフォルトポリシーを適用することもできます。ポリシーについては、168 ページの「Kerberos ポリシーの管理」を参照してください。Kerberos パスワードに関して設定できる基準を一覧表示した例については、例 29「Kerberos ポリシーの属性の表示」を参照してください。パスワードの文字クラスは、小文字、大文字、数字、区切り文字、およびその他のすべての文字です。

Kerberos でのリモートログイン

Oracle Solaris の場合と同様に、Kerberos には、リモートアクセスのための `ssh` コマンドが用意されています。インストールが終了すると、`ssh` コマンドは、ネットワークリクエストを受け入れる唯一のネットワークサービスになります。そのため、`rlogin` や `telnet` などの、Kerberos のために変更されたその他のネットワークサービスは無効になっています。詳細は、51 ページの「Kerberos ネットワークプログラム」および182 ページの「Kerberos のユーザーコマンド」を参照してください。

Kerberos のユーザーコマンド

Kerberos V5 製品は、シングルサインオンシステムです。つまり、ネットワークアプリケーションの使用時、パスワードを 1 回しか入力する必要がありません。Kerberos は、なじみのある一連の既存の各ネットワークアプリケーションに組み込まれているため、Kerberos V5 アプリケーションは認証を実行し、オプションで自動的に暗号化を行います。Kerberos V5 アプリケーションは、既存の UNIX ネットワークアプリケーションに Kerberos 機能が追加されたバージョンです。

注記 - 一般に、リモートログインのニーズに対しては `ssh` アプリケーションで十分です。`ssh(1)` のマニュアルページを参照してください。非推奨のネットワークアプリケーションの有効化については、51 ページの「Kerberos ネットワークプログラム」を参照してください。

既存のネットワークアプリケーションでの Kerberos 機能については、次のマニュアルページの特に「**オプション**」のセクションを参照してください。

- `ftp(1)`
- `rcp(1)`
- `rlogin(1)`
- `rsh(1)`
- `telnet(1)`

Kerberos アプリケーションを使用してリモートシステムに接続すると、アプリケーション、KDC、およびリモートシステムが一連の迅速なネゴシエーションを実行します。これらのネゴシエーションが完了し、アプリケーションがユーザーの代わりにリモートシステムに対してその識別情報を証明すると、リモートシステムはユーザーにアクセスを許可します。

非推奨のリモートログインコマンドの使用例については、*Oracle Solaris 11.1* の管理*:* セキュリティーサービスの「Kerberos アプリケーションの使用 (タスク)」を参照してください。

♦ ♦ ♦ 第 7 章

7 Kerberos サービスのリファレンス

この章では、Kerberos 製品に組み込まれている多数のファイル、コマンド、およびデーモンを示します。

この章では、次のリファレンス情報について説明します。

- 183 ページの「Kerberos ファイル」
- 184 ページの「Kerberos コマンド」
- 186 ページの「Kerberos デーモン」
- 186 ページの「Kerberos の用語」
- 56 ページの「Kerberos 暗号化タイプ」

Kerberos ファイル

Kerberos サービスによって、次のファイルが使用されます。

`~/.k5login`	Kerberos アカウントへのアクセスを許可する主体のリスト
`/etc/krb5/kadm5.acl`	KDC 管理者の主体名とその Kerberos 管理権限を含む Kerberos アクセス制御リストファイル
`/etc/krb5/kdc.conf`	KDC 構成ファイル
`/etc/krb5/kpropd.acl`	Kerberos データベース伝播構成ファイル
`/etc/krb5/krb5.conf`	Kerberos レルム構成ファイル
`/etc/krb5/krb5.keytab`	ネットワークアプリケーションサーバーのための keytab ファイル
`/etc/krb5/warn.conf`	Kerberos チケットの有効期限切れの警告と自動更新の構成ファイル

`/etc/pam.conf`	PAM 構成ファイル
`/tmp/krb5cc_`*UID*	デフォルトの資格キャッシュ (*UID* はユーザーの 10 進数 UID)
`/tmp/ovsec_adm.`*xxxxxx*	パスワード変更操作の間だけ有効な一時資格キャッシュ (*xxxxxx* はランダムな文字列)
`/var/krb5/.k5.`*REALM*	KDC マスター鍵のコピーを含む KDC stash ファイル
`/var/krb5/kadmin.log`	`kadmind` のログファイル
`/var/krb5/kdc.log`	KDC のログファイル
`/var/krb5/principal`	Kerberos 主体データベース
`/var/krb5/principal.kadm5`	ポリシー情報を含む Kerberos 管理データベース
`/var/krb5/principal.kadm5.lock`	Kerberos 管理データベースのロックファイル
`/var/krb5/principal.ok`	Kerberos データベースが正常に初期化されたときに作成される、Kerberos 主体データベースの初期化ファイル
`/var/krb5/principal.ulog`	増分伝播の更新を含む Kerberos 更新ログ
`/var/krb5/slave_datatrans`	`kprop_script` スクリプトが伝播のために使用する KDC のバックアップファイル
`/var/krb5/slave_datatrans_`*slave*	指定されたスレーブに対して完全更新が実行されたときに作成される一時ダンプファイル
`/var/user/`*$USER*`/krb-warn.conf`	ユーザーごとのチケットの有効期限切れの警告と自動更新の構成ファイル

Kerberos コマンド

Kerberos 製品には、次のコマンドが含まれています。これらのコマンドは、各マニュアルページに記載されています。

ftp(1)	ファイル転送プロトコルアプリケーション
kdestroy(1)	Kerberos チケットを破棄する
kinit(1)	Kerberos チケット認可チケットを取得してキャッシュする
klist(1)	現在の Kerberos チケットを表示する
kpasswd(1)	Kerberos パスワードを変更する
ktutil(1)	Kerberos keytab ファイルを管理する
kvno(1)	Kerberos 主体の鍵バージョン番号を一覧表示する
rcp(1)	リモートファイルコピーアプリケーション
rlogin(1)	リモートログインアプリケーション
rsh(1)	リモートシェルアプリケーション
scp(1)	セキュアなリモートファイルコピーアプリケーション
sftp(1)	セキュアなファイル転送アプリケーション
ssh(1)	リモートログインのための Secure Shell
telnet(1)	Kerberos `telnet` アプリケーション
kprop(1M)	Kerberos データベース伝播プログラム
gsscred(1M)	`gsscred` テーブルエントリを管理する
kadmin(1M)	主体、ポリシー、および keytab ファイルを管理するために使用される、Kerberos 認証を必要とするリモート Kerberos データベース管理プログラム
kadmin.local(1M)	主体、ポリシー、および keytab ファイルを管理するために使用される、マスター KDC 上で動作するローカル Kerberos データベース管理プログラム
kclient(1M)	インストールプロファイルとともに、またはインストールプロファイルなしで使用される、Kerberos クライアントのインストールスクリプト
kdb5_ldap_util(1M)	Kerberos データベースの LDAP コンテナを作成する

kdb5_util(1M)	Kerberos データベースと stash ファイルを作成する
kdcmgr(1M)	Kerberos のマスターおよびスレーブ KDC を構成する
kproplog(1M)	更新ログ内の更新エントリのサマリーを一覧表示する

Kerberos デーモン

Kerberos 製品によって、次のデーモンが使用されます。

/usr/lib/inet/proftpd	セキュアなファイル転送プロトコルデーモン
/usr/lib/ssh/sshd	Secure Shell デーモン
/usr/lib/krb5/kadmind	Kerberos データベース管理デーモン
/usr/lib/krb5/kpropd	Kerberos データベース伝播デーモン
/usr/lib/krb5/krb5kdc	Kerberos チケット処理デーモン
/usr/lib/krb5/ktkt_warnd	Kerberos チケットの有効期限切れの警告と自動更新のデーモン
/usr/sbin/in.rlogind	リモートログインデーモン
/usr/sbin/in.rshd	リモートシェルデーモン
/usr/sbin/in.telnetd	telnet デーモン

Kerberos の用語

次の Kerberos の用語は、Kerberos のドキュメント全体にわたって使用されます。Kerberos の概念を理解するには、これらの用語を理解する必要があります。

Kerberos 固有の用語

KDC を管理するには、このセクションで説明する用語を理解する必要があります。

鍵配布センター (*KDC*) は、資格の発行に責任を負う Kerberos のコンポーネントです。資格は、KDC データベースに格納されている情報に基づいて作成されます。各レルムには 2 つ以上の KDC サーバー (マスターと 1 つ以上のスレーブ) が必要です。すべての KDC が資格を生成できますが、KDC データベースを変更できるのはマスター KDC だけです。

stash ファイルには、KDC のマスター鍵が含まれています。サーバーがリブートされると、この鍵を使用して KDC が自動的に認証されて、`kadmind` および `krb5kdc` コマンドがブートされます。このファイルにはマスター鍵が入っているため、このファイルやバックアップは安全な場所に保管する必要があります。ファイルは、`root` の読み取り専用のアクセス権で作成されます。ファイルをセキュリティー保護するには、アクセス権を変更しないでください。ファイルの保護が破られると、この鍵を使用して KDC データベースのアクセスや変更が可能になります。

認証固有の用語

認証処理を理解するには、このセクションで説明する用語を理解する必要があります。プログラマやシステム管理者はこれらの用語に精通している必要があります。

クライアントとは、ユーザーのワークステーション上で動作するソフトウェアのことです。クライアントで動作する Kerberos ソフトウェアは、処理中に多数の要求を作成します。そのため、このソフトウェアとユーザーの動作を区別することが重要です。

サーバーとサービスという用語は多くの場合、同じ意味で使用されます。明確にするために、サーバーという用語は、Kerberos ソフトウェアが実行されている物理マシンを定義するために使用されます。サービスという用語は、サーバー上でサポートされている特定の機能 (`ftp` や `nfs` など) に対応します。サーバーがサービスの一部として記述されることがよくありますが、これはこれらの用語の定義をあいまいにします。そのため、サーバーという用語は、物理マシンを指します。サービスという用語は、ソフトウェアを指します。

Kerberos 製品では、2 種類の鍵が使用されます。鍵の 1 つの種類は、パスワードから派生する鍵です。これは、各ユーザー主体に与えられ、そのユーザーと KDC だけが認識しています。鍵のもう 1 つの種類はランダム鍵です。これは、パスワードに関連付けられていないため、ユーザー主体が使用するのには適していません。ランダム鍵は通常、keytab 内にエントリがあり、セッション鍵が KDC によって生成されたサービス主体に使用されます。サービスは非対話形式での実行を許可するキータブファイ

ル内の鍵にアクセスできるため、サービス主体はランダム鍵を使用できます。セッション鍵は、クライアントとサービス間のトランザクションを保護するために KDC によって生成され、クライアントとサービス間で共有されます。

チケットとは、ユーザーの識別情報をサーバーまたはサービスにセキュアに渡すために使用される情報パケットのことです。チケットは、単一クライアントと特定サーバー上の特定サービスだけに有効です。チケットには、次のものが含まれています。

- サービスの主体名
- ユーザーの主体名
- ユーザーのホストの IP アドレス
- タイムスタンプ
- チケットの有効期限を定義する値
- セッション鍵のコピー

これらのすべてのデータは、サーバーのサービス鍵に暗号化されます。KDC は、このチケットを資格に組み込んで発行します。チケットは、発行されてから有効期限まで再使用できます。

資格とは、チケットとそれに対応するセッション鍵を含む情報のパケットのことです。資格は要求する主体の鍵で暗号化されます。一般的に、KDC はクライアントからのチケット要求に応じて資格を生成します。

オーセンティケータとは、サーバーがクライアントのユーザー主体を認証するために使用する情報のことです。オーセンティケータは、ユーザーの主体名、タイムスタンプ、およびその他のデータを含みます。チケットとは異なり、オーセンティケータは (通常はサービスへのアクセスがリクエストされたときの) 1 回だけ使用されます。オーセンティケータは、クライアントとサーバーが共有するセッション鍵を使用して暗号化されます。通常、クライアントが、オーセンティケータを作成し、サーバーまたはサービスに対して認証するためにサーバーまたはサービスのチケットとともに送信します。

チケットの種類

チケットには、チケットがどのように使用されるかを決めるプロパティーがあります。これらのプロパティーは、チケットの作成時にチケットに割り当てられます。ただし、チケットのプロパティーはあとから変更できます。たとえば、チケットを**転送可能**の状態から**転送済み**の状態に変更できます。チケットのプロパティーを表示するには、`klist` コマンドを使用します178 ページの「Kerberos チケットの表示」を参照してください。

チケットは、次の 1 つまたは複数のプロパティーで表されます。

転送可能/転送済み

転送可能チケットは、あるホストから別のホストに送信できます。これにより、クライアントは自身を再認証する必要がなくなります。たとえば、ユーザー `david` がユーザー `jennifer` のホスト上にいる間に転送可能チケットを取得した場合、`david` は、新しいチケットを取得しなくても (それにより自分自身を再認証しなくても) 自分のホストにログインできます。転送可能チケットの例については、例38「Kerberos チケットの作成」を参照してください。

初期

初期チケットは、チケット認可チケットに基づかずに、直接発行されるチケットです。パスワードを変更するアプリケーションなどの一部のサービスは、クライアントが自身の秘密鍵を知っていることを示せるように、チケットに「初期」とマークするよう要求できます。初期チケットは、クライアントが最近、古くなっている可能性があるチケット認可チケットに依存することなく自身を認証したことを示します。

無効

無効チケットは、まだ使用可能になっていない遅延チケットです。無効チケットは、有効になるまでアプリケーションサーバーから拒否されます。チケットが検証されるには、その開始時間が過ぎたあとに `VALIDATE` フラグを設定して、クライアントがチケット認可サービスリクエストでそのチケットを KDC に提示する必要があります。

遅延可能/遅延

遅延チケットは、作成されたあと、指定された一定の時間が経過するまで有効にならないチケットです。たとえばこのようなチケットは、夜遅く実行するバッチジョブに使用するのに便利です。チケットが盗まれてもバッチジョブが実行される予定の時刻まで使用できないためです。遅延チケットは、無効チケットとして発行され、開始時間を過ぎて、クライアントが KDC による検査を要求したときに有効になります。遅延チケットは通常、チケット認可チケットの有効期限まで有効です。ただし、チケットに更新可能が指定されている場合、その最長有効期限は通常、チケット認可チケットの最長有効期限と同じに設定されます。

プロキシ可能/プロキシ

場合によっては、主体はサービスに、自身の代わりに操作を実行するよう許可する必要がある場合があります。プロキシの主体名は、チケットの作成時に指定する必要があります。Oracle Solaris では、プロキシ可能チケットやプロキシチケットはサポートされません。

プロキシ可能チケットは、1 つのサービスに対してのみ有効である点を除き、転送可能チケットと同じです。転送可能チケットは、サービスにクライアントの識別情報の完全な使用を許可します。したがって、転送可能チケットは一種のスーパープロキシと考えられます。

更新可能
: 有効期限が非常に長いチケットはセキュリティーリスクになるため、チケットを更新可能として指定できます。更新可能チケットには 2 つの有効期限があります。 1 つはチケットの現在のインスタンスの有効期限で、もう 1 つは任意のチケットの最長有効期限 (1 週間) です。クライアントがチケットの使用を継続するときは、最初の有効期限が切れる前にチケットの有効期限を更新します。たとえば、すべてのチケットの最長有効期限が 10 時間のときに、あるチケットが 1 時間しか有効になれないとします。このチケットを保持するクライアントが 1 時間を超えて使用する場合は、その時間内にチケットの有効期限を更新する必要があります。チケットが最長有効期限 (10 時間) に達すると、チケットの有効期限が自動的に切れ、それ以上更新できなくなります。

チケットの属性を表示する方法については、178 ページの「Kerberos チケットの表示」を参照してください。

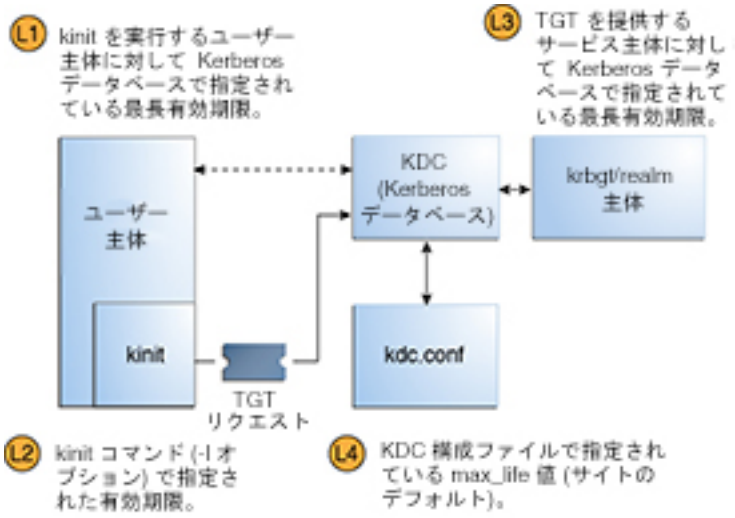
チケットの有効期限

主体がチケット (チケット認可チケット (TGT) を含む) を取得すると、チケットの有効期限は、次の有効期限の値の最小値として設定されます。

- `kinit` を使用してチケットを取得する場合は、`kinit` の `-l` オプションで指定された有効期限の値。デフォルトでは、`kinit` は最長有効期限の値を使用します。
- `kdc.conf` ファイルで指定された最長有効期限の値 (`max_life`)。
- チケットを提供するサービス主体に対し Kerberos データベースに指定されている最長有効期限値。`kinit` の場合、サービス主体は `krbtgt/`*realm* です。
- チケットを要求するユーザー主体に対し Kerberos データベースに指定されている最長有効期限値。

次の図は、TGT の有効期限が決定される方法と、4 つの有効期限の値が生成される場所を示しています。どの主体がチケットを取得した場合も、チケットの有効期限は同様に決定されます。2 つの違いとして、`kinit` が有効期限の値を提供しない点と、チケットを提供するサービス主体が `krbtgt/`*realm* 主体の代わりに最長有効期限の値を提供する点があります。

図 13 TGT の有効期限が決定される方法



チケットの有効期限 = L1、L2、L3、および L4 の最小値

更新可能チケットの有効期限は、次のように、4 つの更新可能な有効期限の値の最小値から決定されます。

- `kinit` を使用してチケットを取得または更新する場合、`kinit` の `-r` オプションに指定した更新可能有効期限値。
- `kdc.conf` ファイルで指定された更新可能な最長有効期限の値 (`max_renewable_life`)。
- チケットを提供するサービス主体に対し Kerberos データベースに指定されている更新可能最長有効期限値。`kinit` の場合、サービス主体は `krbtgt/`*realm* です。
- チケットを要求するユーザー主体に対し Kerberos データベースに指定されている更新可能最長有効期限値。

Kerberos 主体名

チケットは主体名で識別され、主体名はユーザーやサービスを識別します。次の例は、標準的な主体名を示しています。

`changepw/kdc1.example.`	パスワードを変更するときに、KDC にアクセスできるマスター KDC の主体。

com@EXAMPLE.COM	
clntconfig/admin@EXAMPLE.COM	kclient インストールユーティリティーで使用される主体。
ftp/boston.example.com@EXAMPLE.COM	ftp サービスによって使用される主体。この主体は host 主体の代わりに使用できます。
host/boston.example.com@EXAMPLE.COM	Kerberos アプリケーション (klist や kprop など) およびサービス (ftp や telnet など) によって使用される主体。この主体は host またはサービス主体と呼ばれます。主体は NFS マウントの認証に使用されます。この主体はまた、クライアントが受け取った TGT が正しい KDC から発行されたものであることを確認するためにも使用されます。
K/M@EXAMPLE.COM	マスター鍵名の主体。各マスター KDC には、1 つのマスター鍵名の主体が関連付けられます。
kadmin/history@EXAMPLE.COM	この主体に含まれる鍵を使用して、ほかの主体のパスワード履歴が保管されます。各マスター KDC には、これらの主体のいずれかが割り当てられます。
kadmin/kdc1.example.com@EXAMPLE.COM	kadmind を使用して KDC にアクセスできるマスター KDC サーバーの主体。
kadmin/changepw.example.com@EXAMPLE.COM	Oracle Solaris リリースが動作していないクライアントからのパスワード変更要求の受け入れに使用される主体。
krbtgt/EXAMPLE.COM@EXAMPLE.COM	この主体を使用して、チケット認可チケットを生成します。
krbtgt/EAST.EXAMPLE.COM@WEST.EXAMPLE.COM	この主体は、レルム間チケット認可チケットの例です。
nfs/boston.example.	NFS サービスによって使用される主体。この主体は host 主体の代わりに使用できます。

com@EXAMPLE.COM	
root/boston.example.com@EXAMPLE.COM	クライアントの root アカウントに関連付けられた主体。この主体は、root 主体と呼ばれ、NFS がマウントされたファイルシステムへの root アクセスを提供します。
username@EXAMPLE.COM	ユーザー用の主体。
username/admin@EXAMPLE.COM	KDC のデータベースを管理するために使用できる admin 主体。

♦ ♦ ♦ 第 8 章

8 Kerberos エラーメッセージとトラブルシューティング

この章では、Kerberos サービスを使用するときに発生するエラーメッセージの解決策を説明します。また、さまざまな問題のトラブルシューティングのヒントについても説明します。

この章で扱う内容は、次のとおりです。

- 195 ページの「Kerberos のエラーメッセージ」
- 207 ページの「Kerberos のトラブルシューティング」
- 210 ページの「Kerberos サービスでの DTrace の使用」

Kerberos のエラーメッセージ

このセクションでは、Kerberos のエラーメッセージ、エラーの発生原因、およびその対処方法について説明します。

Kerberos 共通エラーメッセージ (A - M)

このセクションでは、Kerberos コマンド、Kerberos デーモン、PAM フレームワーク、GSS インタフェース、NFS サービス、および Kerberos ライブラリに共通するエラーメッセージを、英語版メッセージのアルファベット順 (A - M) に示します。

`Bad lifetime value (有効期間の値が無効です。)`

原因: 指定した有効期限値が無効または間違った形式です。

対処方法: 指定された値が、`kinit(1)` のマニュアルページの時間形式のセクションに適合していることを確認してください。

`Bad start time value (開始時間の値が無効です。)`

原因: 指定した開始時間が無効または間違った形式です。

対処方法: 指定された値が、kinit(1) のマニュアルページの時間形式のセクションに適合していることを確認してください。

`Cannot contact any KDC for requested realm (要求されたレルムの KDC に接続できません。)`

原因: 要求されたレルムの KDC が応答しません。

対処方法: 1 つ以上の KDC (マスターまたはスレーブ) にアクセスできること、または `krb5kdc` デーモンが KDC 上で動作していることを確認してください。`/etc/krb5/krb5.conf` ファイルに指定されている構成済みの KDC (`kdc =` *kdc-name*) を確認してください。

`Cannot determine realm for host: host is '`*hostname*`' (ホストのレルムを判断できません: ホストは 'hostname' です。)`

原因: Kerberos がホストのレルム名を判断できません。

対処方法: デフォルトのレルム名を指定するか、Kerberos 構成ファイル (`krb5.conf`) にドメイン名のマッピングを設定してください。

`Cannot find a kadmin KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`' (krb5.conf(4) 内に kadmin KDC エントリが見つからないか、レルム 'realmname' の DNS サービスロケーションレコードが見つかりません)`

`Cannot find a kpassword KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`' (krb5.conf(4) 内に kpassword KDC エントリが見つからないか、レルム 'realmname' の DNS サービスロケーションレコードが見つかりません。)`

`Cannot find a kpassword KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`' (krb5.conf(4) 内にマスター KDC エントリが見つからないか、レルム 'realmname' の DNS サービスロケーションレコードが見つかりません。)`

`Cannot find a kpassword KDC entry in krb5.conf(4) or DNS Service Location records for realm '`*realmname*`' (krb5.conf(4) 内に KDC エントリが見つからないか、レルム 'realmname' の DNS サービスロケーションレコードが見つかりません。)`

原因: `krb5.conf` ファイルまたは DNS サーバーレコードのどちらかが正しく構成されていません。

対処方法: Kerberos 構成ファイル (`/etc/krb5/krb5.conf`) または KDC の DNS サーバーレコードが適切に構成されていることを確認してください。

`Cannot find address for '`*hostname*`': '`*error-string*`' ('hostname' のアドレスが見つかりません: 'error-string')`

原因: 指定されたホスト名の DNS レコード内にアドレスが見つかりません。

対処方法: DNS 内のホストレコードを修正するか、DNS 検索プロセス内のエラーを訂正してください。

`cannot initialize realm` *realm-name* `(レルム realm-name を初期化できません。)`

原因: KDC に stash ファイルが存在しない可能性があります。

対処方法: KDC に stash ファイルが存在することを確認してください。存在しない場合は、`kdb5_util` コマンドを使用して stash ファイルを作成し、再度 `krb5kdc` コマンドを実行します。

`Cannot resolve KDC for requested realm (要求されたレルムの KDC を解決できません。)`

原因: Kerberos がレルムの KDC を判断できません。

対処方法: Kerberos 構成ファイル (`krb5.conf`) の `realm` セクションに KDC が指定されていることを確認してください。

`Cannot resolve network address for KDCs '`*hostname*`' discovered via DNS Service Location records for realm '`*realm-name*`' (レルム 'realm-name' の DNS サービスロケーションレコードを介して検出された KDC の 'hostname' のネットワークアドレスを解決できません。)`

`Cannot resolve network address for KDCs '`*hostname*`' specified in krb5.conf(4) for realm '`*realm-name*`' (レルム 'realm-name' の krb5.conf(4) 内に指定された KDC の 'hostname' のネットワークアドレスを解決できません。)`

原因: `krb5.conf` ファイルまたは DNS サーバーレコードのどちらかが正しく構成されていません。

対処方法: Kerberos 構成ファイル (`/etc/krb5/krb5.conf`) および KDC の DNS サーバーレコードが適切に構成されていることを確認してください。

`Can't open/find Kerberos configuration file (Kerberos 構成ファイルを開けません / 見つかりません。)`

原因: Kerberos 構成ファイル (`krb5.conf`) を使用できませんでした。

対処方法: `krb5.conf` ファイルが、正しい場所に配置されていることを確認してください。また、このファイルに正しいアクセス権が与えられていることを確認して

ください。このファイルに対する書き込み権は root、読み込み権はすべてのユーザーに与える必要があります。

`Client '`*principal*`' pre-authentication failed (クライアントの 'principal' の事前認証に失敗しました。)`

原因: その主体の認証に失敗しました。

対処方法: ユーザーが正しいパスワードを使用していることを確認してください。

`Client or server has a null key (クライアントまたはサーバーの鍵が空です。)`

原因: クライアントまたはサーバーの鍵が空です。

対処方法: `kadmin` の `cpw` コマンドを使用して、主体の鍵の値を入力してください。

`Communication failure with server while initializing kadmin interface (kadmin インタフェースを初期化中に、サーバーとの通信の失敗です。)`

原因: マスター KDC として指定されたサーバーで `kadmind` デーモンが動作していませんでした。

対処方法: マスター KDC に正しいサーバー名が指定されていることを確認してください。正しいサーバー名が指定されている場合は、指定したマスター KDC 上で `kadmind` が動作していることを確認してください。

`Credentials cache file permissions incorrect (資格キャッシュファイルのアクセス権が正しくありません。)`

原因: 資格キャッシュ (`/tmp/krb5cc_`*uid*) に対する読み取り権または書き込み権が適切ではありません。

対処方法: 資格キャッシュに対する読み取り権および書き込み権があることを確認してください。

`Credentials cache I/O operation failed` *XXX* `(資格キャッシュ入出力操作が失敗しました。 XXX)`

原因: システムの資格キャッシュ (`/tmp/krb5cc_`*uid*) に書き込むときに、Kerberos で問題が発生しました。

対処方法: 資格キャッシュが削除されていないことを確認するとともに、`df` コマンドを使用してデバイス上に空き領域があることを確認してください。

`Decrypt integrity check failed (復号化で整合性チェックが失敗しました。)`

原因: チケットが無効である可能性があります。

対処方法: 次の条件の両方を確認してください。

- 資格が有効であることを確認してください。`kdestroy` を使用してチケットを破棄し、`kinit` を使用して新しいチケットを作成します。
- 対象ホストのキータブファイルに対して、正しいバージョンのサービス鍵が割り当てられていることを確認してください。`kadmin` を使用して、Kerberos データベース内のサービス主体 (`host/`*FQDN-hostname* など) の鍵バージョン番号を表示します。また、ターゲットホスト上で `klist -k` コマンドを使用して、その鍵バージョン番号が同じであることも確認してください。

`Decrypt integrity check failed for client 'principal' and server 'hostname' (クライアント 'principal' およびサーバー 'hostname' で復号化の整合性チェックに失敗しました。)`

原因: チケットが無効である可能性があります。

対処方法: 資格が有効であることを確認してください。`kdestroy` コマンドを使用してチケットを破棄し、`kinit` コマンドを使用して新しいチケットを作成します。

`failed to obtain credentials cache (資格キャッシュを取得できませんでした。)`

原因: `kadmin` の初期化中に、`kadmin` が `admin` 主体の資格を取得しようとしましたが、失敗しました。

対処方法: `kadmin` コマンドを実行したときに、正しい主体とパスワードを使用したことを確認してください。

`Field is too long for this implementation (この実装ではフィールドが長すぎます。)`

原因: Kerberos アプリケーションから送信されたメッセージのサイズが長すぎます。このエラーは、トランスポートプロトコルが UDP の場合に発生します。UDP では、デフォルトの最大メッセージ長は 65535 バイトです。また、Kerberos サービスから送信されるプロトコルメッセージの各フィールドにも制限があります。

対処方法: KDC サーバーの `/etc/krb5/kdc.conf` ファイルでトランスポートを UDP に制限していないことを確認してください。

`GSS-API (or Kerberos) error (GSS-API (または Kerberos) エラー)`

原因: このメッセージは、汎用 GSS-API または Kerberos のエラーメッセージで、いくつかの問題の組み合わせによって発生した可能性があります。

対処方法: `/var/krb5/kdc.log` ファイルを確認して、このエラーが発生したときに詳細なエラーメッセージが記録されているかどうかを確認してください。

`Improper format of Kerberos configuration file (Kerberos 構成ファイルのフォーマットが不適切です。)`

原因: Kerberos 構成ファイルに無効なエントリがあります。

対処方法: `krb5.conf` ファイル内のすべての関係式に、= 記号と値が使用されていることを確認してください。また、各下位セクションが角カッコで囲まれていることも確認してください。

`Invalid credential was supplied (無効な資格が指定されました。)`
`Service key not available (サービス鍵が使用できません。)`

原因: 資格キャッシュ内のサービスチケットが間違っている可能性があります。

対処方法: このサービスを使用しようとする前に、現在の資格キャッシュを破棄し、`kinit` コマンドを再実行してください。

`Invalid flag for file lock mode (ファイルロックモードのフラグが無効です。)`

原因: Kerberos の内部エラーが発生しました。

対処方法: バグを報告してください。

`Invalid message type specified for encoding (エンコードに対し無効なメッセージタイプが指定されました。)`

原因: Kerberos アプリケーションによって送信されたメッセージタイプが Kerberos で認識されませんでした。

対処方法: 使用するサイトまたはベンダーで開発した Kerberos アプリケーションを使用している場合は、Kerberos が正しく使用されていることを確認してください。

`kadmin: Bad encryption type while changing host/`*`FQDN`*`'s key (host/FQDN の鍵の変更中に不正な暗号化タイプが見つかりました。)`

原因: 新しいリリースの基本リリースには、デフォルトの暗号化タイプが追加されました。以前のバージョンの ソフトウェアを実行している KDC がサポートしない暗号化タイプをクライアントが要求する場合があります。

対処方法: `aes256` 暗号化タイプを含まないように、クライアント上の `krb5.conf` の `permitted_enctypes` を設定します。この手順は、新しいクライアントが追加されるごとに実行する必要があります。

`KDC can't fulfill requested option (KDC は要求したオプションを処理できません。)`

原因: 要求されたオプションを KDC が許可しませんでした。遅延または転送可能オプションが要求されましたが、KDC が許可しませんでした。または、TGT の更新が要求されましたが、更新可能な TGT が存在しない可能性があります。

対処方法: KDC が許可しないオプションまたは使用できない種類のチケットを要求していないかどうかを確認してください。

`KDC reply did not match expectation: KDC not found. Probably got an unexpected realm referral (KDC の応答が期待したものと一致しませんでした。KDC が見つかりません。おそらく予期しないレルムのリフェラルを受け取りました。)`

原因: KDC の応答に予期した主体名が含まれていないか、応答内のその他の値が正しくありません。

対処方法: 通信先の KDC が RFC4120 に準拠していること、送信している要求が Kerberos V5 要求であること、および KDC が利用可能であることを確認してください。

`kdestroy:Could not obtain principal name from cache (キャッシュから主体名を取得できません。)`

原因: 資格キャッシュが欠落しているか、または破壊されています。

対処方法: 指定したキャッシュ位置が正しいことを確認してください。必要に応じて、`kinit` を使用して削除し、新しい TGT を取得してください。

`kdestroy:Could not obtain principal name from cache (キャッシュの破棄中に資格キャッシュが見つかりませんでした。)`

原因: 資格キャッシュ (`/tmp/krb5c_`*uid*) がないか、または壊れています。

対処方法: 指定したキャッシュ位置が正しいことを確認してください。必要に応じて、`kinit` を使用して削除し、新しい TGT を取得してください。

`kdestroy:Could not obtain principal name from cache (TGT 期限切れの警告が削除されません。)`

原因: 資格キャッシュが欠落しているか、または破壊されています。

対処方法: 指定したキャッシュ位置が正しいことを確認してください。必要に応じて、`kinit` を使用して削除し、新しい TGT を取得してください。

`Kerberos authentication failed (Kerberos 認証に失敗しました。)`

原因: Kerberos パスワードが正しくないか、または UNIX パスワードと一致していません。

対処方法: パスワードが同期されていない場合は、認証を完了するために Kerberos パスワードを指定する必要があります。ユーザーが元のパスワードを忘れた可能性があります。

`Key version` *number* `is not available for principal` *principal* `(鍵バージョン number が主体 principal に使用できません。)`

原因: 鍵の鍵バージョンが、アプリケーションサーバー上の鍵のバージョンに一致しません。

対処方法: `klist -k` コマンドを使用して、アプリケーションサーバー上の鍵のバージョンを確認してください。

`Key version number for principal in key table is incorrect (鍵テーブルの主体の鍵バージョン番号が正しくありません。)`

原因: サーバーのキータブファイル内の主体の鍵バージョンが Kerberos データベース内のバージョンと異なります。サービスの鍵が変更されたか、旧サービスチケットを使用している可能性があります。

対処方法: `kadmin` などによってサービスの鍵が変更されている場合は、新しい鍵を抽出して、サービスが動作しているホストのキータブファイルに格納する必要があります。

または、古い鍵を含む古いサービスチケットを使用している可能性があります。`kdestroy` コマンドを実行し、次に `kinit` コマンドを再度実行してください。

`kinit: gethostname failed (gethostname が失敗しました。)`

原因: ローカルネットワーク構成でのエラーは、`kinit` が失敗する原因になります。

対処方法: ホストが正しく構成されていることを確認してください。

`login: load_modules: can not open module /usr/lib/security/pam_krb5.so.1 (load_modules: /usr/lib/security/pam_krb5.so.1 モジュールを開けません。)`

原因: Kerberos PAM モジュールが存在しないか、有効な実行可能バイナリではありません。

対処方法: Kerberos PAM モジュールが `/usr/lib/security` ディレクトリに存在し、有効な実行可能バイナリであることを確認してください。また、`login` の PAM 構

成ファイルに pam_krb5.so.1 への正しいパスが含まれていることも確認してください。

`Looping detected getting initial creds: '`*client-principal*`' requesting ticket '`*service-principal*`'. Max loops is` *value*`. Make sure a KDC is available. (最初の資格の取得中にループが検出されました: 'client-principal' がチケット 'service-principal' を要求しています。最大ループ回数は value です。KDC が利用可能であることを確認してください。)`

原因: Kerberos が初期チケットを複数回取得しようとしましたが、失敗しました。

対処方法: 認証要求に対して 1 つ以上の KDC が応答していることを確認してください。

`Master key does not match database (マスター鍵がデータベースと一致しません。)`

原因: 読み込まれたデータベースのダンプが、マスター鍵を含むデータベースから作成されませんでした。マスター鍵は /var/krb5/.k5.*REALM* 内に格納されています。

対処方法: ロードされたデータベースダンプ内のマスター鍵が、/var/krb5/.k5.*REALM* 内に格納されているマスター鍵に一致することを確認してください。

`Matching credential not found (一致する資格が見つかりません。)`

原因: 要求に一致する資格が見つかりませんでした。資格キャッシュで使用できない資格を要求しています。

対処方法: kdestroy を使用してチケットを破棄し、kinit を使用して新しいチケットを作成します。

`Message out of order (メッセージの順序が違います。)`

原因: 順次プライバシを使用しているときに送信されたメッセージが正しくない順序で到着しました。一部のメッセージが転送中に失われました。

対処方法: Kerberos セッションを再初期化する必要があります。

`Message stream modified (メッセージストリームが変更されました。)`

原因: 計算されたチェックサムとメッセージのチェックサムが一致しません。メッセージが転送中に変更された可能性があります。これは、セキュリティー漏洩を示す場合があります。

対処方法: メッセージがネットワーク経由で正しく送信されていることを確認してください。このメッセージはまた、送信されている間にメッセージの改ざんが発生した可能性も示す場合があるため、チケットを破棄し、使用している Kerberos サービスを再初期化してください。

Kerberos 共通エラーメッセージ (N - Z)

このセクションでは、Kerberos コマンド、Kerberos デーモン、PAM フレームワーク、GSS インタフェース、NFS サービス、および Kerberos ライブラリに共通するエラーメッセージを、英語版メッセージのアルファベット順 (N - Z) に示します。

`No credentials cache file found (資格キャッシュファイルが見つかりません。)`

原因: Kerberos が資格キャッシュ (`/tmp/krb5cc_`*uid*) を見つけることができません。

対処方法: 資格ファイルが存在し、読み込み可能であることを確認してください。そうでない場合は、`kinit` コマンドを再度実行してみてください。

`No credentials were supplied, or the credentials were unavailable or inaccessible (資格が提供されていません。あるいは、資格を使用またはアクセスできません。)`

`No credentials cache found (資格キャッシュが見つかりません。)`

原因: ユーザーの資格キャッシュが間違っているか、または存在しません。

対処方法: ユーザーは、サービスを開始しようとする前に `kinit` を実行する必要があります。

`No credentials were supplied, or the credentials were unavailable or inaccessible (資格が提供されていません。あるいは、資格を使用またはアクセスできません。)`

`No principal in keytab ('`*filename*`') matches desired name` *principal* `(キータブ ('filename') 内の主体が目的の名前 principal と一致しません)`

原因: サーバーの認証を試みている間にエラーが発生しました。

対処方法: ホスト主体またはサービス主体が、サーバーのキータブファイル内にあることを確認してください。

`Operation requires “`*privilege*`” privilege (操作には privilege 特権が必要です。)`

原因: 使用されていた `admin` 主体に、`kadm5.acl` ファイル内で適切な権限が割り当てられていません。

対処方法: 適切な特権を持つ主体を使用してください。または、使用されていた主体を、適切な権限が割り当てられるように構成してください。通常は、名前の一部に `/admin` が含まれる主体には、適切な特権が割り当てられています。

`PAM-KRB5 (auth): krb5_verify_init_creds failed: Key table entry not found (PAM-KRB5 (auth): krb5_verify_init_creds に失敗しました: 鍵テーブルエントリが見つかりません)`

原因: リモートアプリケーションは、ローカル `/etc/krb5/krb5.keytab` ファイル内にあるホストのサービス主体を読み込もうとしましたが、サービス主体が存在しませんでした。

対処方法: ホストのキータブファイルにホストのサービス主体を追加してください。

`Permission denied in replay cache code (再実行キャッシュコードでアクセス権がありません。)`

原因: システムの再実行キャッシュを開けませんでした。サーバーは、現在のユーザー ID と異なるユーザー ID で最初に実行された可能性があります。

対処方法: 再実行キャッシュに適切なアクセス権が割り当てられていることを確認してください。再実行キャッシュは、Kerberos サーバーアプリケーションが動作するホストに格納されます。`root` 以外のユーザーの場合、再実行キャッシュファイルの名前は `/var/krb5/rcache/rc`*_service_name_uid* です。root ユーザーの場合、再実行キャッシュファイルの名前は `/var/krb5/rcache/root/rc`*_service_name* です。

`Protocol version mismatch (プロトコルバージョンが一致していません。)`

原因: Kerberos V4 要求が KDC に送信された可能性があります。Kerberos サービスでは、Kerberos V5 プロトコルだけがサポートされます。

対処方法: アプリケーションが Kerberos V5 プロトコルを使用していることを確認してください。

`Request is a replay (要求は再送です。)`

原因: この要求は、すでにこのサーバーに送信され、処理が完了しています。チケットが盗まれた可能性があり、ほかのユーザーがチケットを再使用しようとしています。

対処方法: しばらくしてから要求を再発行してください。

`Requested principal and ticket don't match: Requested principal is '`*service-principal*`' and TGT principal is '`*TGT-principal*`' (要求された主体とチケッ`

トが一致しません: 要求された主体は 'service-principal' で、TGT 主体は 'TGT-principal' です。)

原因: 接続するサービス主体と使用するサービスチケットが一致しません。

対処方法: DNS が適切に機能することを確認してください。別のベンダーのソフトウェアを使用する場合は、そのソフトウェアが主体名を正しく使用していることを確認します。

`Server refused to negotiate authentication, which is required for encryption. Good bye.`

原因: リモートアプリケーションは、クライアントからの Kerberos 認証を受け取ることができないか、またはそのように構成されていません。

対処方法: 認証をネゴシエーションできるアプリケーションを提供するか、認証を有効にする適切なフラグを使用してアプリケーションを構成します。

`Server rejected authentication (during sendauth exchange)` (サーバーが認証を拒否しました (sendauth 交換で)。)

原因: 通信しようとしているサーバーが認証を拒否しました。ほとんどの場合、このエラーは Kerberos データベースを伝播するときに発生します。`kpropd.acl` ファイル、DNS、またはキータブファイルに問題が発生している可能性があります。

対処方法: `kprop` 以外のアプリケーションを実行しているときにこのエラーが発生した場合は、サーバーのキータブファルが正しいかどうかを調査してください。

`Target name principal '`*principal*`' does not match` *service-principal* (ターゲット名の主体 'principal' が service-principal と一致しません。)

原因: 使用されているサービス主体が、アプリケーションサーバーで使用しているサービス主体と一致しません。

対処方法: アプリケーションサーバーで、サービス主体がキータブファイルに含まれていることを確認してください。クライアントでは、正しいサービス主体が使用されていることを確認してください。

`The ticket isn't for us` (チケットはわれわれのものではありません。)

`Ticket/authenticator do not match.` (チケット/オーセンティケータが一致しません。)

原因: 要求内の主体名がサービス主体の名前と一致しなかった可能性があります。その原因は、サービスが非 FQDN 名を期待しているときにチケットが主体の FQDN 名とともに送信されたか、またはサービスが FQDN 名を期待しているときに非 FQDN 名が送信されたかのいずれかです。

対処方法: kprop 以外のアプリケーションを実行しているときにこのエラーが発生した場合は、サーバーのキータブファルが正しいかどうかを調査してください。

`Truncated input file detected (不完全な入力ファイルを検出しました。)`

原因: 操作に使用されたデータベースダンプファイルが完全ではありません。

対処方法: ダンプファイルを作成し直すか、別のデータベースダンプファイルを使用します。

Kerberos のトラブルシューティング

このセクションでは、Kerberos ソフトウェアに関するトラブルシューティング情報を提供します。

鍵バージョン番号に関する問題

KDC で使用される鍵バージョン番号 (KVNO) と、システム上でホストされるサービスの `/etc/krb5/krb5.keytab` に格納されているサービス主体鍵が一致しないことがあります。keytab ファイルを新しい鍵で更新せずに KDC 上で新しい鍵セットが作成されると、KVNO が同期しなくなる場合があります。問題を診断したら、`krb5.keytab` ファイルをリフレッシュしてください。

1. keytab エントリを一覧表示します。

   各主体の KVNO が各エントリ内の最初の項目です。

   ```
   # klist -k
   Keytab name: FILE:/etc/krb5/krb5.keytab
   KVNO Principal
   ---- --------------------------------------------------------------------------
   2 host/denver.example.com@EXAMPLE.COM
   2 host/denver.example.com@EXAMPLE.COM
   2 host/denver.example.com@EXAMPLE.COM
   2 nfs/denver.example.com@EXAMPLE.COM
   2 nfs/denver.example.com@EXAMPLE.COM
   2 nfs/denver.example.com@EXAMPLE.COM
   2 nfs/denver.example.com@EXAMPLE.COM
   ```

2. `host` 鍵を使用して、最初の資格を取得します。

   ```
   # kinit -k
   ```

3. KDC で使用される KVNO を確認します。

   ```
   # kvno nfs/denver.example.com
   ```

```
nfs/denver.example.com@EXAMPLE.COM: kvno = 3
```

ここに示されている KVNO は 2 ではなく 3 です。

krb5.conf ファイルの形式に関する問題

krb5.conf ファイルが正しくフォーマットされない場合は、次のエラーメッセージが端末ウィンドウに表示されるか、またはログファイルに記録されることがあります。

```
Improper format of Kerberos configuration file while initializing krb5 library
```

形式が正しくないと、関連付けられているサービスが攻撃に対して脆弱になることがあります。この問題を修正しないと、Kerberos 機能を使用できません。

Kerberos データベースの伝播の問題

Kerberos データベースの伝播が失敗した場合は、スレーブ KDC とマスター KDC の間で、およびマスター KDC からスレーブ KDC サーバーに /usr/bin/rlogin -x を試してみてください。

KDC がデフォルトでセキュアになっている場合、rlogin コマンドは無効になっているため、この問題のトラブルシューティングには使用できません。KDC 上で rlogin を有効にするには、eklogin サービスを有効にする必要があります。

```
# svcadm enable svc:/network/login:eklogin
```

問題のトラブルシューティングを完了したら、eklogin サービスを無効にしてください。

```
# svcadm disable svc:/network/login:eklogin
```

リモートアクセスが機能しない場合は、KDC 上の keytab ファイルの問題である可能性があります。リモートアクセスが機能する場合、rlogin と伝播ソフトウェアは同じ host/*host-name* 主体を使用しているため、keytab ファイルやネームサービスの問題ではありません。この場合、kpropd.acl ファイルが正しいことを確認してください。

Kerberos NFS ファイルシステムのマウントの問題

- Kerberos NFS ファイルシステムのマウントに失敗した場合には、NFS サーバーに /var/rcache/root ファイルが存在することを確認してください。ファイルシステムの所有者が root でない場合は、削除してから再度マウントします。
- Kerberos NFS ファイルシステムへのアクセスに問題がある場合は、使用するシステムと NFS サーバー上で gssd サービスが有効になっていることを確認してください。

- Kerberos NFS ファイルシステムにアクセスしようとしたときに `invalid argument` または `bad directory` のどちらかのエラーメッセージが表示される場合は、NFS ファイルシステムをマウントしようとするときに完全修飾 DNS 名を使用していないことが問題である可能性があります。マウントされているホストが、サーバーのキータブファイル内のサービス主体名に含まれるホスト名と一致していません。

  また、複数の Ethernet インタフェースを実装したサーバーに DNS を設定するときに、ホスト単位に複数のアドレスレコードを割り当てずに、インタフェース単位に名前を割り当てた場合にも、この問題が発生します。Kerberos サービスの場合は、次のようにホスト単位に複数のアドレスレコードを設定する必要があります[1]:

```
my.host.name.   A       1.2.3.4
A       1.2.4.4
A       1.2.5.4

my-en0.host.name.       A       1.2.3.4
my-en1.host.name.       A       1.2.4.4
my-en2.host.name.       A       1.2.5.4

4.3.2.1         PTR     my.host.name.
4.4.2.1         PTR     my.host.name.
4.5.2.1         PTR     my.host.name.
```

この例の設定では、インタフェースごとに 1 つの参照が割り当てられます。また、サーバーのキータブファイル内で、3 つのサービス主体の代わりに、1 つのサービス主体を使用できます。

`root` ユーザーの認証の問題

システム上で `root` になろうとしたときに認証に失敗したが、`root` 主体はすでにホストの keytab ファイルに追加している場合は、2 つの領域を調査してください。まず、キータブファイル内の `root` 主体が、そのインスタンスとして完全指定形式のシステム名であることを確認します。完全指定形式名の場合は、`/etc/resolv.conf` ファイルを確認して、システムが DNS クライアントとして正しく設定されていることを確認してください。

GSS 資格の UNIX 資格へのマッピングの監視

資格マッピングをモニターするには、最初に、`/etc/gss/gsscred.conf` ファイルの次の行をコメント解除します。

```
SYSLOG_UID_MAPPING=yes
```

[1]Ken Hornstein、「Kerberos FAQ」、[http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html#kerbdns]、2010 年 3 月 10 日にアクセス。

次に、gssd サービスで /etc/gss/gsscred.conf ファイルを読み取ります。

```
# pkill -HUP gssd
```

これで、gssd によってリクエストされた資格マッピングをモニターできます。syslog.conf ファイルが debug の重要度レベルで auth システム機能用に構成されている場合、これらのマッピングは syslog デーモンによって記録されます。

注記 - rsyslog サービスインスタンスが有効になっている場合、これらのマッピングは rsyslog デーモンによって記録されます。

Kerberos サービスでの DTrace の使用

Kerberos メカニズムは、さまざまなプロトコルメッセージを複号化するためのいくつかの DTrace プローブをサポートしています。リストについては、付録A Kerberos 用の DTrace プローブを参照してください。DTrace により、特権ユーザーは暗号化されていない Kerberos やアプリケーションデータを容易に調査できるため、DTrace プローブには、ほかのプロトコルインスペクタに対する明確な優位性があります。

次の例は、Kerberos DTrace プローブで表示できる情報を示しています。

例 40 DTrace を使用した Kerberos メッセージの追跡

次のスクリプトは、DTrace プローブを使用して、システムによって送受信された Kerberos メッセージに関する詳細情報を表示します。表示されるフィールドは、krb_message-recv および krb_message-send プローブに割り当てられた構造体に基づいていることに注意してください。詳細は、280 ページの「Kerberos DTrace プローブの定義」を参照してください。

```
kerberos$target:::krb_message-recv
{
    printf("<- krb message recved: %s\n", args[0]->krb_message_type);
    printf("<- krb message remote addr: %s\n", args[1]->kconn_remote);
    printf("<- krb message ports: local %d remote %d\n",
            args[1]->kconn_localport, args[1]->kconn_remoteport);
    printf("<- krb message protocol: %s transport: %s\n",
            args[1]->kconn_protocol, args[1]->kconn_type);
}

kerberos$target:::krb_message-send
{
    printf("-> krb message sent: %s\n", args[0]->krb_message_type);
    printf("-> krb message remote addr: %s\n", args[1]->kconn_remote);
    printf("-> krb message ports: local %d remote %d\n",
            args[1]->kconn_localport, args[1]->kconn_remoteport);
    printf("-> krb message protocol: %s transport: %s\n",
            args[1]->kconn_protocol, args[1]->kconn_type);
    printf("\n");
}
```

```
kerberos$target:::krb_error-read
{
    printf("<- krb error code: %s\n", args[1]->kerror_error_code);
    printf("<- krb error client: %s server: %s\n", args[1]->kerror_client,
            args[1]->kerror_server);
    printf("<- krb error e-text: %s\n", args[1]->kerror_e_text);
    printf("\n");
}
```

前のスクリプトは、コマンド行から、または krb5kdc デーモンで呼び出すことができます。コマンド行から krb-dtrace.d という名前のスクリプトを呼び出す例を次に示します。このコマンドにより、Kerberos はメッセージを送受信するようになります。Kerberos DTrace プローブを含む Kerberos mech_krb5.so ライブラリを強制的にロードするために LD_NOLAZYLOAD=1 が必要であることに注意してください。

```
# LD_NOLAZYLOAD=1 dtrace -s ./krb\-dtrace.d -c kinit
dtrace: script './krb-dtrace' matched 4 probes
kinit: Client 'root@DEV.ORACLE.COM' not found in Kerberos database while g
etting initial credentials
dtrace: pid 3750 has exited
 CPU     ID                    FUNCTION:NAME
   2  74782 k5_trace_message_send:krb_message-send -> krb message sent: KR
B_AS_REQ(10)
-> krb message remote addr: 10.229.168.163
-> krb message ports: local 62029 remote 88
-> krb message protocol: ipv4 transport: udp


   2  74781 k5_trace_message_recv:krb_message-recv <- krb message recved:
KRB_ERROR(30)
<- krb message remote addr: 10.229.168.163
<- krb message ports: local 62029 remote 88
<- krb message protocol: ipv4 transport: udp

   2  74776     krb5_rd_error:krb_error-read <- krb error code: KDC_ERR_C_
PRINCIPAL_UNKNOWN(6)
<- krb error client: root@DEV.ORACLE.COM server: krbtgt/DEV.ORACLE.COM@DEV
.ORACLE.COM
<- krb error e-text: CLIENT_NOT_FOUND
```

krb5kdc デーモンでこのスクリプトを使用するには、svc:/network/security/krb5kdc:default サービスを有効にしてオンラインにする必要があります。mech_krb5.so ライブラリによって krb5kdc デーモンがロードされるため、次のコマンドでは LD_NOLAZYLOAD=1 を使用していないことに注意してください。

```
# dtrace -s ./krb\-dtrace.d -p $(pgrep -x krb5kdc)
```

例 41 DTrace を使用した Kerberos 事前認証タイプの表示

次の例は、クライアントによってどのような事前認証が選択されるかを示しています。最初の手順では、次のように DTrace スクリプトを作成します。

```
cat krbtrace.d
kerberos$target:::krb_message-recv
{
printf("<- krb message recved: %s\n", args[0]->krb_message_type);
printf("<- krb message remote addr: %s\n", args[1]->kconn_remote);
```

```
printf("<- krb message ports: local %d remote %d\n",
args[1]->kconn_localport, args[1]->kconn_remoteport);
printf("<- krb message protocol: %s transport: %s\n",
args[1]->kconn_protocol, args[1]->kconn_type);
}

kerberos$target:::krb_message-send
{
printf("-> krb message sent: %s\n", args[0]->krb_message_type);
printf("-> krb message remote addr: %s\n", args[1]->kconn_remote);
printf("-> krb message ports: local %d remote %d\n",
args[1]->kconn_localport, args[1]->kconn_remoteport);
printf("-> krb message protocol: %s transport: %s\n",
args[1]->kconn_protocol, args[1]->kconn_type);
printf("\n");
}

kerberos$target:::krb_kdc_req-make
{
printf("-> krb kdc_req make msg type: %s\n", args[0]->krb_message_type);
printf("-> krb kdc_req make pre-auths: %s\n", args[1]->kdcreq_padata_types);
printf("-> krb kdc_req make auth data: %s\n", args[1]->kdcreq_authorization_data);
printf("-> krb kdc_req make client: %s server: %s\n", args[1]->kdcreq_client,
args[1]->kdcreq_server );
}

kerberos$target:::krb_kdc_req-read
{
/* printf("<- krb kdc_req msg type: %s\n", args[0]->krb_message_type); */
printf("<- krb kdc_req client: %s server: %s\n", args[1]->kdcreq_client,
args[1]->kdcreq_server );
printf("\n");
}

kerberos$target:::krb_kdc_rep-read
{
/* printf("<- krb kdc_rep msg type: %s\n", args[0]->krb_message_type); */
printf("<- krb kdc_rep client: %s server: %s\n", args[1]->kdcrep_client,
args[1]->kdcrep_enc_server );
printf("\n");
}

kerberos$target:::krb_ap_req-make
{
printf("-> krb ap_req make server: %s client: %s\n", args[2]->kticket_server,
args[2]->kticket_enc_client );
}

kerberos$target:::krb_error-read
{
printf("<- krb error code: %s\n", args[1]->kerror_error_code);
printf("<- krb error client: %s server: %s\n", args[1]->kerror_client,
args[1]->kerror_server);
printf("<- krb error e-text: %s\n", args[1]->kerror_e_text);
printf("\n");
}
```

次に、次のコマンドを入力することによって、Kerberos システム上で特権ユーザーとして `krbtrace.d` スクリプトを実行します。

```
# LD_BIND_NOW=1 dtrace -qs krbtrace.d -c "kinit -k"
.
.
-> krb kdc_req make pre-auths: FX_COOKIE(133) ENC_TIMESTAMP(2) REQ_ENC_PA_REP(149)
```

出力に事前認証のタイプが表示されます。事前認証のさまざまなタイプの詳細は、RFC 4120 を参照してください。

例 42 DTrace を使用した Kerberos のエラーメッセージのダンプ

```
# dtrace -n 'krb_error-make {
printf("\n{");
printf("\n\tctime = %Y", (uint64_t)(args[1]->kerror_ctime * 1000000000));
printf("\n\tcusec = %d", args[1]->kerror_cusec);
printf("\n\tstime = %Y", (uint64_t)(args[1]->kerror_stime * 1000000000));
printf("\n\tsusec = %d", args[1]->kerror_susec);
printf("\n\terror_code = %s", args[1]->kerror_error_code);
printf("\n\tclient = %s", args[1]->kerror_client);
printf("\n\tserver = %s", args[1]->kerror_server);
printf("\n\te_text = %s", args[1]->kerror_e_text);
printf("\n\te_data = %s", "");
printf("\n}");
}'
dtrace: description 'krb_error-make ' matched 1 probe
CPU     ID                    FUNCTION:NAME
0  78307     krb5_mk_error:krb_error-make
{
ctime = 2012 May 10 12:10:20
cusec = 0
stime = 2012 May 10 12:10:20
susec = 319090
error_code = KDC_ERR_C_PRINCIPAL_UNKNOWN(6)
client = testuser@EXAMPLE.COM
server = krbtgt/EXAMPLE.COM@EXAMPLE.COM
e_text = CLIENT_NOT_FOUND
e_data =
}
```

例 43 DTrace を使用した SSH サーバーのサービスチケットの表示

```
# LD_PRELOAD_32=/usr/lib/gss/mech_krb5.so.1 dtrace -q -n '
kerberos$target:::krb_kdc_req-make {
printf("kdcreq_server: %s",args[1]->kdcreq_server);
}' -c "ssh local@four.example.com" -o dtrace.out
Last login: Wed Sep 10 10:10:20 2014
Oracle Solaris 11 X86    July 2014
$ ^D
# cat dtrace.out
kdcreq_server: host/four.example.com@EXAMPLE.COM
```

例 44 DTrace を使用した、初期 TGT のリクエスト時に使用できない KDC のアドレスとポートの表示

```
# LD_BIND_NOW=1 dtrace -q -n '
kerberos$target:::krb_message-send {
printf("%s:%d\n",args[1]->kconn_remote, args[1]->kconn_remoteport)
}' -c "kinit local4"

10.10.10.14:88
10.10.10.14:750
10.10.10.14:88
10.10.10.14:750
10.10.10.14:88
```

```
10.10.10.14:750
kinit(v5): Cannot contact any KDC for realm 'EXAMPLE.COM'
while getting initial credentials
```

例 45 DTrace を使用した Kerberos 主体からのリクエストの表示

```
# LD_BIND_NOW=1 dtrace -qs /opt/kdebug/mykdtrace.d \
-c 'kadmin -p kdc/admin -w test123 -q listprincs'
Authenticating as principal kdc/admin with password.
krb kdc_req msg type: KRB_AS_REQ(10)
krb kdc_req make client: kdc/admin@TEST.NET server:
kadmin/interop1.example.com@TEST.NET
krb message sent: KRB_AS_REQ(10)
krb message recved: KRB_ERROR(30)
Err code: KDC_ERR_PREAUTH_REQUIRED(25)
Err msg client: kdc/admin@TEST.NET server: kadmin/interop1.example.com@TEST.NET
Err e-text: NEEDED_PREAUTH
krb kdc_req msg type: KRB_AS_REQ(10)
krb kdc_req make client: kdc/admin@TEST.NET server:
kadmin/interop1.example.com@TEST.NET
krb message sent: KRB_AS_REQ(10)
krb message recved: KRB_AS_REP(11)
kadmin: Database error! Required KADM5 principal missing while
initializing kadmin interface
krb kdc_req msg type: KRB_AS_REQ(10)
krb kdc_req make client: kdc/admin@TEST.NET server:
kadmin/interop2.example.com@TEST.NET
krb message sent: KRB_AS_REQ(10)
krb message recved: KRB_ERROR(30)
Err code: KDC_ERR_S_PRINCIPAL_UNKNOWN(7)
Err msg client: kdc/admin@TEST.NET server: kadmin/interop2.example.com@TEST.NET
Err e-text: SERVER_NOT_FOUND
krb kdc_req msg type: KRB_AS_REQ(10)
krb kdc_req make client: kdc/admin@TEST.NET server:
kadmin/interop2.example.com@TEST.NET
```

次のスクリプトは、前の出力を生成するために使用されました。

```
kerberos$target:::krb_message-recv
{
printf("krb message recved: %s\n", args[0]->krb_message_type);
}

kerberos$target:::krb_message-send
{
printf("krb message sent: %s\n", args[0]->krb_message_type);
}

kerberos$target:::krb_kdc_req-make
{
printf("krb kdc_req msg type: %s\n", args[0]->krb_message_type);
printf("krb kdc_req make client: %s server: %s\n", args[1]->kdcreq_client,
args[1]->kdcreq_server );
}

kerberos$target:::krb_ap_req-make
{
printf("krb ap_req make server: %s client: %s\n", args[2]->kticket_server,
args[2]->kticket_enc_client );
}

kerberos$target:::krb_error-read
```

```
{
printf("Err code: %s\n", args[1]->kerror_error_code);
printf("Err msg client: %s server: %s\n", args[1]->kerror_client,
args[1]->kerror_server);
printf("Err e-text: %s\n", args[1]->kerror_e_text);
}
```

♦ ♦ ♦ 第 9 章

9 簡易認証セキュリティー層の使用

この章では、簡易認証セキュリティー層 (SASL) について説明します。

- 217 ページの「SASL について」
- 217 ページの「SASL のリファレンス」

SASL について

簡易認証セキュリティー層 (SASL) は、ネットワークプロトコルに認証サービスとセキュリティーサービス (オプション) を提供するフレームワークです。アプリケーションは、SASL ライブラリ `/usr/lib/libsasl.so` を呼び出します。SASL ライブラリは、アプリケーションと各種 SASL メカニズムを結び付ける層の役割を果たします。このメカニズムは、認証プロセスや、オプションのセキュリティーサービスの提供時に使用されます。SASL のバージョンは Cyrus SASL から派生したものですが、少し変更されています。

SASL は、次のサービスを提供します。

- 任意のプラグインをロードする
- アプリケーションから必要なセキュリティーオプションを判断してセキュリティーメカニズムの選択を支援する
- アプリケーションで使用可能なプラグインを一覧表示する
- 特定の認証の試みに対して、使用可能なメカニズムの一覧から最適なメカニズムを選択する
- アプリケーションと選択されたメカニズムの間で認証データのルーティングを指定する
- アプリケーションに返す SASL ネゴシエーションに関する情報を提供する

SASL のリファレンス

次のセクションでは、SASL の実装についての情報を提供します。

SASL プラグイン

SASL プラグインは、セキュリティーメカニズム、ユーザーの正規化、および補助プロパティーの検索をサポートします。デフォルトでは、動的にロードされる 32 ビットのプラグインは `/usr/lib/sasl` にインストールされ、64 ビットのプラグインは `/usr/lib/sasl/`*`$ISA`* にインストールされます。次のセキュリティーメカニズムプラグインが提供されます。

`crammd5.so.1` CRAM-MD5。認証のみをサポートし、承認はサポートしません

`digestmd5.so.1` DIGEST-MD5。認証、整合性、機密性のほか、承認もサポートします

`gssapi.so.1` GSSAPI。認証、整合性、機密性のほか、承認もサポートします。GSSAPI セキュリティーメカニズムには、機能している Kerberos 基盤が必要です。

`plain.so.1` PLAIN。認証と承認をサポートします。

さらに、EXTERNAL セキュリティーメカニズムプラグインと INTERNAL ユーザー正規化プラグインが `libsasl.so.1` に組み込まれています。EXTERNAL メカニズムは、認証と承認をサポートします。外部のセキュリティーソースによって提供された場合には、整合性と機密性がサポートされます。INTERNAL プラグインは、必要に応じて、レルム名をユーザー名に追加します。

Oracle Solaris リリースは、現時点ではどの `auxprop` プラグインも提供していません。CRAM-MD5 および DIGEST-MD5 メカニズムプラグインをサーバー側で完全に機能させるには、平文のパスワードを取得するための `auxprop` プラグインをユーザーが提供する必要があります。PLAIN プラグインには、パスワードを検証するための追加のサポートが必要です。パスワード検証のサポートには、サーバーアプリケーションへのコールバック、`auxprop` プラグイン、`saslauthd`、`pwcheck` のいずれかを使用できます。`saslauthd` および `pwcheck` デーモンは、Oracle Solaris リリースでは提供されていません。相互運用性を向上させるために、サーバーアプリケーションは、`mech_list` SASL オプションによって完全に機能するメカニズムに制限してください。

SASL の環境変数

デフォルトでは、クライアントの認証名は `getenv("LOGNAME")` に設定されます。この変数は、クライアントまたはプラグインでリセットできます。

SASL のオプション

libsasl およびプラグインの動作は、/etc/sasl/*app*.conf ファイルで設定できるオプションを使用してサーバー側で変更できます。変数 *app* は、サーバー定義のアプリケーション名です。サーバーのドキュメントでは、*app* でアプリケーション名が指定されているはずです。

サポートしているオプションは、次のとおりです。

auto_transition　ユーザーが平文認証に成功すると、自動的にそのユーザーをほかのメカニズムに切り替えます。

auxprop_login　使用する補助プロパティープラグインの名前を一覧表示します。

canon_user_plugin　使用する canon_user プラグインを選択します。

mech_list　サーバーアプリケーションが使用可能なメカニズムを列挙します。

pwcheck_method　パスワードを検証するために使用するメカニズムを列挙します。現時点では、auxprop が唯一の使用可能な値です。

reauth_timeout　迅速に再認証するために認証情報がキャッシュされる時間の長さを分単位で設定します。このオプションは、DIGEST-MD5 プラグインで使用されます。このオプションを 0 に設定すると、再認証が無効になります。

次のオプションはサポートされません。

plugin_list　使用可能なメカニズムを一覧表示します。このオプションは、プラグインの動的な読み込みの動作を変えるため、使用されなくなりました。

saslauthd_path　saslauthd デーモンとの通信に使用される saslauthd ドアの場所を定義します。saslauthd デーモンは、Oracle Solaris リリースに組み込まれていません。このため、このオプションも組み込まれていません。

keytab　GSSAPI プラグインによって使用される keytab ファイルの場所を定義します。代わりに KRB5_KTNAME 環境変数を使用して、デフォルトの keytab の場所を設定します。

次のオプションは、Cyrus SASL では用意されていません。ただし、Oracle Solaris リリースでは追加されています。

`use_authid`	GSS クライアントセキュリティーコンテキストの作成時に、デフォルトの資格を使用するのではなく、クライアントの資格を取得します。デフォルトでは、デフォルトのクライアント Kerberos 識別情報が使用されます。
`log_level`	サーバーのログのレベルを設定します。

♦ ♦ ♦ 第 10 章

10 Oracle Solaris でのマルチファクタ認証へのスマートカードの使用

この章では、Oracle Solaris 11.3 システムへの認証時にスマートカードを識別情報の証拠として使用する方法についての情報を提供します。このシステムが Sun Ray Software (SRS) を実行していることはありません。

注記 - SRS を実行しているシステムでは、統合された Sun Ray スマートカードの認証手順に従います。

この章で扱う内容は、次のとおりです。

- 221 ページの「ツーファクタ認証とスマートカード」
- 230 ページの「スマートカードのログインのための Oracle Solaris システムの構成」
- 247 ページの「スマートカードログインのための Oracle Solaris システムの有効化」
- 248 ページの「Web ブラウザおよび電子メールでのスマートカード使用の有効化」
- 251 ページの「スマートカードの使用」

ツーファクタ認証とスマートカード

スマートカードは、機密のコンピュータおよび Web サイトにログインする際、識別情報の 2 つ目の証明を提供します。

ツーファクタ認証について

ツーファクタ認証 (2FA) では、ログインの検証を強化する独立した認証ステップが追加されます。2FA は、ユーザーがコンピュータシステムにアクセスできるようになる

前に、ユーザーが認証メカニズムにいくつかの別々の証拠を示す必要がある、マルチファクタ認証 (MFA) の一種です。2FA では、ユーザーは、次の MFA の証拠のカテゴリのうち 2 つ指定する必要があります。

- 知っていること、たとえば個人識別番号 (PIN) またはパスワードなど
- 所有しているもの、たとえばスマートカード、チャレンジレスポンスキーフォブ、またはトークンジェネレータなど
- 身体に固有のもの、たとえば生体認証の指紋、網膜スキャン、声紋など

2 つの識別情報の証明に加え、2FA では通常、ユーザーがアカウントにログインしようとしているユーザー自身であることを確認する必要があります。2FA を実行している Oracle Solaris のコンピュータサーバーでは、識別情報の 2 つの別々の証明、つまりスマートカード (所有しているもの) と PIN (知っていること) が必要になります。

共通アクセスカード (CAC) とも呼ばれるスマートカードは、個人識別情報、認証、データストレージ、およびアプリケーション処理を提供できるマイクロチップが埋め込まれたプラスチック製のカードです。さらに、これらは電子メールの暗号化および暗号化署名、および公開鍵インフラストラクチャー (PKI) 認証ツールの使用を有効にできます。Oracle Solaris においてスマートカードでログインすると、従来のパスワードにのみに依存するネットワークログインプロセスよりもはるかに強力なセキュリティーが提供されます。

米国政府のスマートカード

このガイドでは、CAC は 2FA に使用される米国国防総省 (DoD) のスマートカードのことです。CAC は承認された政府職員に対する標準の身分証明書として発行されます。政府職員は自分の CAC を使用して政府の建物やコンピュータネットワークにアクセスします。CAC には、PKI 証明書を含むカード所有者情報が含まれます。スマートカードリーダー内のソフトウェアは、標準のインターネットプロトコルを介し、カード所有者情報を政府のサーバーにある情報と比較でき、アクセスを付与するか拒否します。

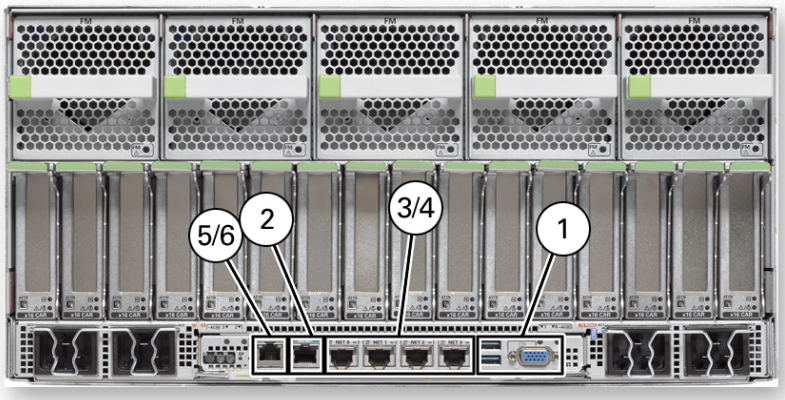
Oracle Solaris はコンピュータ認証に対して次の 4 種類の DoD CAC カードを認識します。

- ジュネーブ条約身分証明カード – 現役/予備軍および公安系公務員用
- ジュネーブ条約付属軍カード – 緊急時の軍属用
- ID および特権共通アクセスカード – 軍事施設に居住する一般市民用
- DoD の ID カード/政府機関の身分証明書 – 軍属の従業員および請負業者用

ローカル、リモート、ILOM のスマートカードログイン

次の図は、スマートカードログインのエントリポイントを示しています。

図 14 スマートカードのエントリポイント



- 1 – システムに直接接続されているスマートカードリーダー。モニターおよびキーボードもこのエントリポイントを使用します。図15を参照してください。
- 2 – シリアルポートを介してシステムに直接接続されるスマートカードリーダー。コンソールおよびターミナルプログラムもシリアルポートを使用します。図15を参照してください。
- 3 – Secure Shell を使用したスマートカードへのリモートネットワークアクセス。図16を参照してください。
- 4 – X11 デスクトップを使用したスマートカードへのリモートネットワークアクセス。図16を参照してください。
- 5/6 – Integrated Lights Out Management (ILOM) ポートが Secure Shell または `https` を使用してスマートカードに接続。図17を参照してください。

Oracle Solaris のスマートカードとスマートカードリーダーでは、ローカルログイン、ネットワーク経由のリモートログイン、および Oracle の Integrated Lights Out Manager (ILOM) を使用したリモートログインの、3 つのタイプのログイン用の 2FA ユーザー認証および否認防止が提供されます。ユーザーは、サーバーへのスマートカードのログインおよび認証を構成したあと、Web ブラウザや電子メールソフトを構成して、セキュアな Web 通信およびセキュアな電子メールを使用することもできます。詳細は、248 ページの「Web ブラウザおよび電子メールでのスマートカード使用の有効化」を参照してください。

次の図はスマートカードを使用する 2FA ログインを示しています。

図 15 スマートカードを使用したローカルログイン

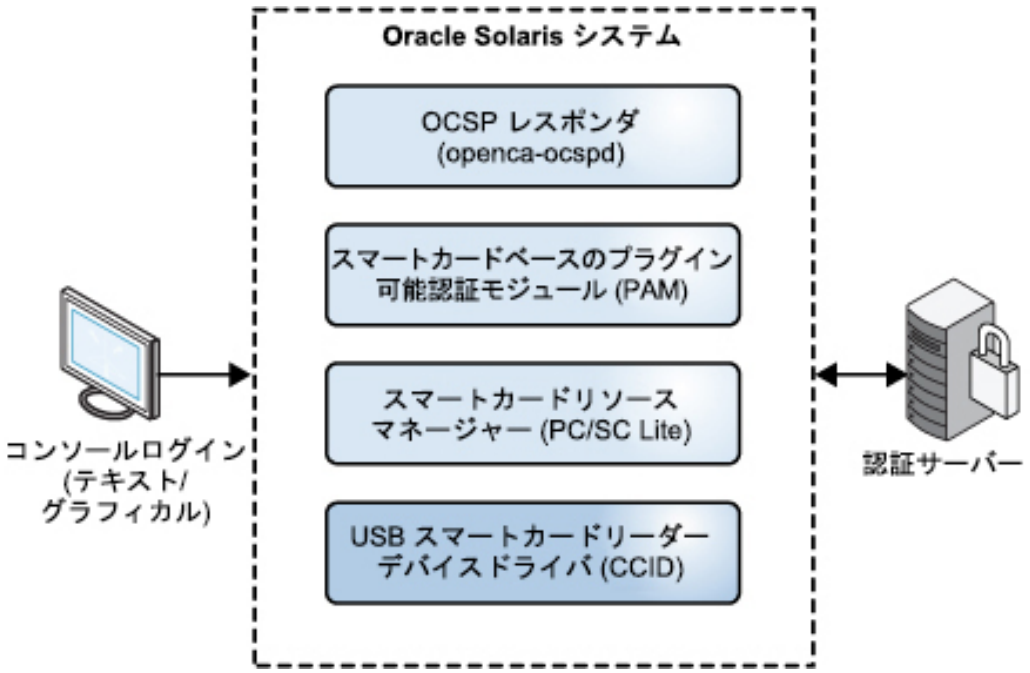


図 16 スマートカードを使用したネットワーク経由のリモートログイン

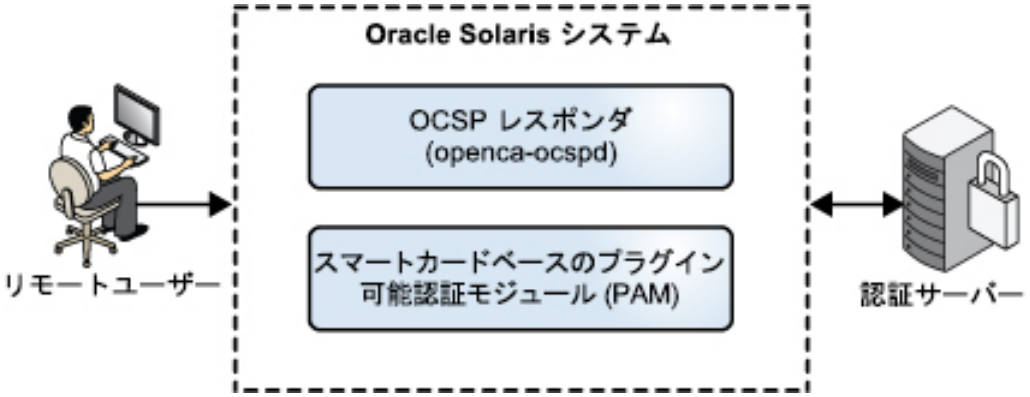
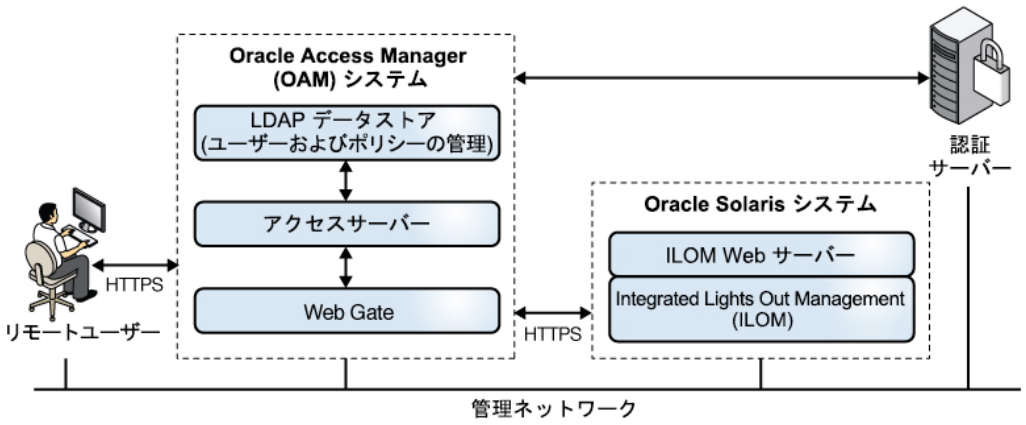


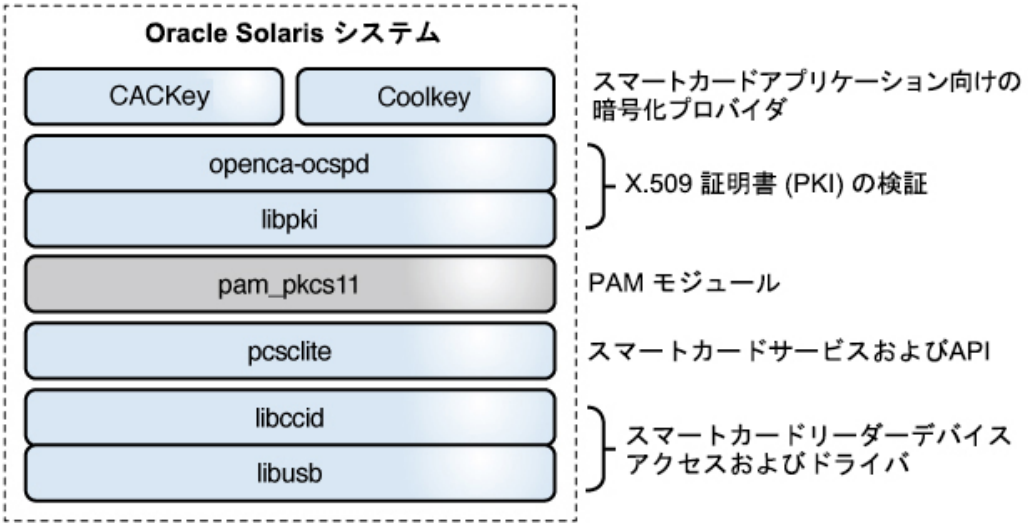
図 17 スマートカードを使用した ILOM ログイン



Oracle Solaris でのツーファクタ認証の実装

Oracle Solaris は、次のソフトウェアスタックを使用してスマートカードでの 2FA を実装します。ほとんどのソフトウェアは、`smartcard` パッケージにあります。CACKey 暗号化プロバイダは、別個のパッケージにあります。IPS パッケージグループはいずれもスマートカードパッケージをインストールしないので、それらをインストールする必要があります。

図 18 Oracle Solaris でのツーファクタ認証のソフトウェア実装



2FA ソフトウェアスタックは次のモジュールで構成されます。

- `libusb` – USB デバイスへのアクセスを有効化するオープンソースライブラリ。`https://sourceforge.net/projects/libusb/files/libusb-1.0/` を参照してください。
- libccid – 汎用 USB CCID (Chip/Smart Card Interface Devices) ドライバおよび ICCD (Integrated Circuit Card Devices) 用のオープンソースライブラリ。
- libpki – 証明書を生成から検証まで管理するオープンソースライブラリ。

  ドキュメントについては、libpki のドキュメントを参照してください。
- pcsclite – ドライバをロードし、`libpcsclite.so` クライアントライブラリにリンクされたランタイムプログラムを処理する要求に応答する SMF サービスとして、`pcscd` デーモンを提供します。

  `libpcsclite.so` クライアントライブラリは、`pam_pkcs11` モジュールと連携してスマートカードドライバを認証ソフトウェアに接続します。

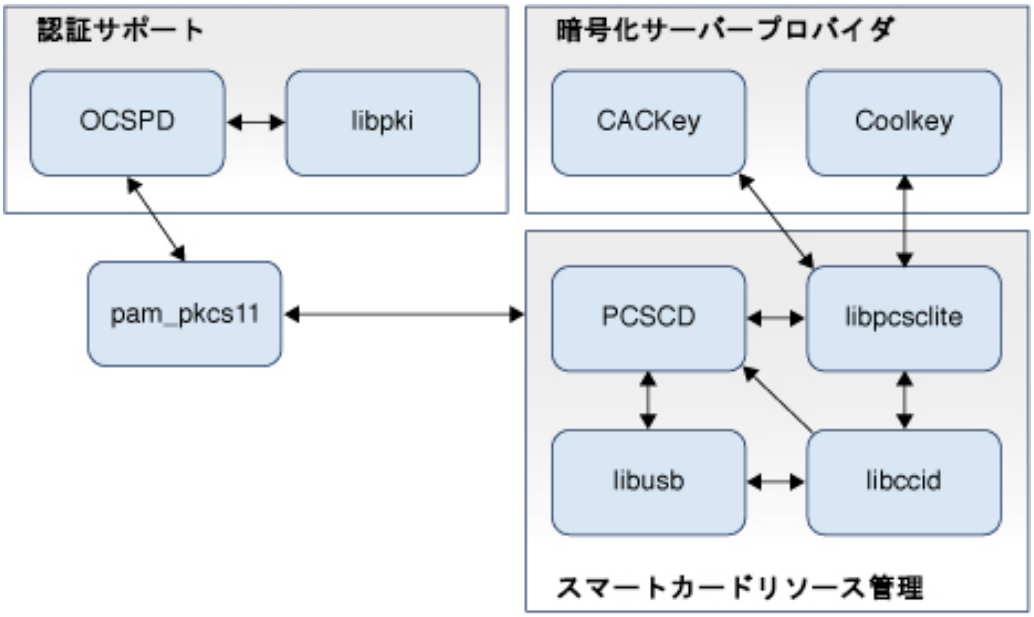
  - DoD CAC 対応のアプリケーションおよび Web サイトをサポートするため、2FA の実装では CACKey ソフトウェアが PKCS #11 モジュールおよび Web ブラウザプラグインにリンクする必要があります。
  - PIV カード対応のアプリケーションおよび Web サイトをサポートするため、2FA の実装では Coolkey ソフトウェアが PKCS #11 モジュールおよび Web ブラウザプラグインにリンクする必要があります。
- openca-ocspd – オープンソースの Online Certificate Status Protocol (OCSP) レスポンダ

OCSP は、X.509 デジタル証明書がまだ有効であるかどうかを検証するインターネットプロトコルです。OCSP メッセージは ASN.1 でエンコードされ、通常 HTTP 経由で通信されます。OCSP サーバーは、証明書の検証要求に応答するので、OCSP レスポンダと呼ばれます。

- pam_pkcs11 – ユーザーの Oracle Solaris システムへの認証に使用される PKCS #11 トークンライブラリ用のプラグイン可能認証モジュール (PAM)。

次の図は、スマートカードのモジュール接続を示しています。

図 19 Oracle Solaris でのツーファクタ認証用のソフトウェア接続



スマートカード向けのソフトウェア暗号化プロバイダ

図18および図19に示されているように、Oracle Solaris では 2 つのプロバイダからのスマートカード暗号化をサポートしています。

- Coolkey – `smartcard` パッケージから使用可能。

  `pcsclite` デーモン (`pcscd`) がスマートカードリーダーやその他の CCID 対応デバイスなどの 1 つ以上の PKI ハードウェアトークンデバイスインタフェースを管理しているとき、Coolkey はトークンの存在を動的に検出します。

- CACKey – 保全許可および Controlled Access Card を持つユーザー用に DISA から入手可能。このソフトウェアは、`solaris` のパブリッシャーからも入手できます。

  CACKey は、PC/SC 準拠のリーダーに接続されたスマートカードに対して標準の PKCS #11 インタフェースを提供します。CACKey は Coolkey と同様の機能を実行しますが、Government Smart Card Interoperability Specification (GSC-IS) v2.1 以降を実装する米国政府のスマートカードのみをサポートします。仕様を確認するには、NIST Computer Security Division の Web サイトに移動し、そのページで「6887」を検索します。カードのリストは、222 ページの「米国政府のスマートカード」を参照してください。

スマートカードのハードウェアリーダー

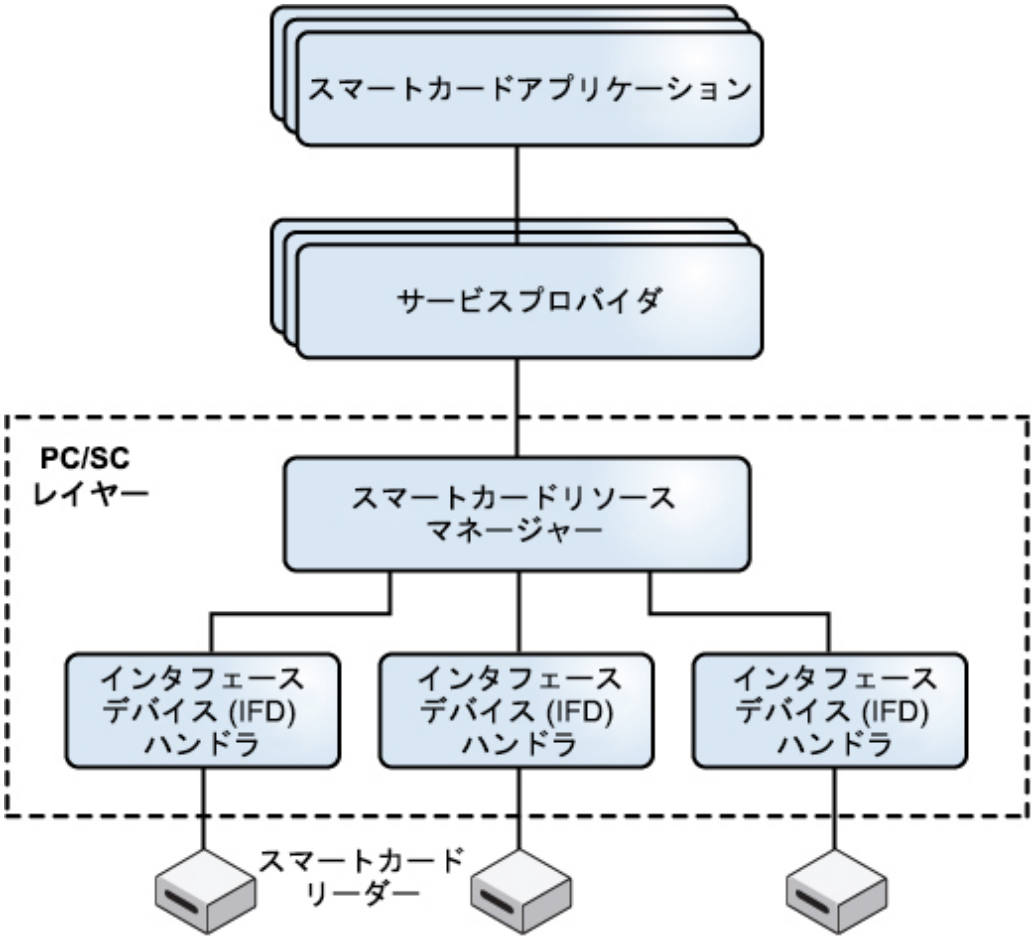
スマートカードを使用したローカルログインでシステムへ認証するために、次のスマートカードハードウェアリーダーを Oracle Solaris システムに接続できます。

- HID/Omnikey、3121
- Identive (以前の SCM Microsystems)、SCR-3310v2 および SCR-3310
- ActivCard/ActivIdentity V3

Oracle Solaris のスマートカードアーキテクチャー

次の図は、Oracle Solaris がローカルに接続されたスマートカードリーダーに接続し、それらのスマートカード上のリソースを暗号化サービスプロバイダおよびスマートカード対応アプリケーションで使用できるようにする方法を示しています。

図 20 ドライバをスマートカードに接続する PC/SC レイヤー

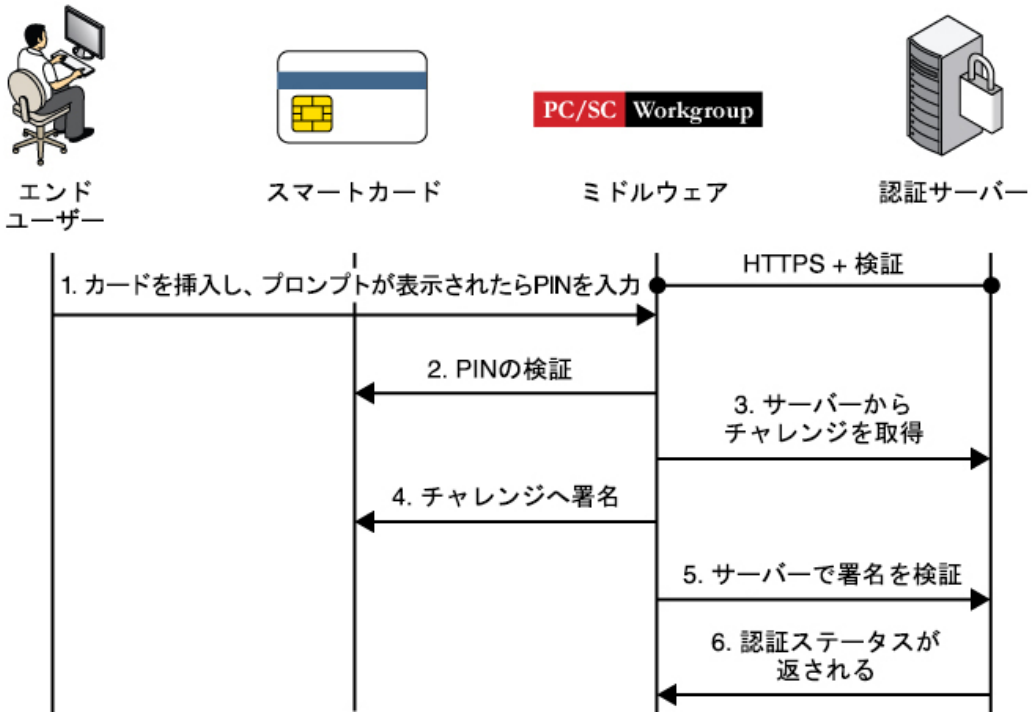


スマートカードリーダーは、スマートカードにアクセスするための PC/SC 業界の標準を使用して Oracle Solaris システム上のスマートカードに接続および通信します。`pam_pkcs11` モジュールは、`smartcard` パッケージのソフトウェアと統合して、2FA 認証を提供します。次に、OCSP レスポンダが既存のスマートカードの Certificate Authentication (CA) サーバーインフラストラクチャーと通信して、ユーザーが入力したスマートカード PIN を認証および検証し、スマートカードにある X.509 証明書を検証します。

注記 - OCSP を使用しない場合は、ローカルファイルを使用して証明書およびユーザーを検証できます。

次の図は、Oracle Solaris がスマートカードの PKI 認証を処理する方法を示しています。

図 21 スマートカードによる PKI 認証



スマートカードのログインのための Oracle Solaris システムの構成

適切なハードウェアおよびソフトウェアがインストールされていると、Oracle Solaris システムはスマートカードを使用してユーザーを認証できます。図21は PKI がスマー

トカードを認証する方法、図18および図19はソフトウェアスタックを示しています。証明書を登録できる LDAP ディレクトリサービスをサイトがすでに実行していることが前提です。

注意 - これらの手順は、Sun Ray では機能しません。SRS を実行しているシステムでは、統合された Sun Ray スマートカードの認証手順に従います。

主なスマートカード構成タスク

主な構成タスクでは、ソフトウェアが含まれている `smartcard` パッケージをインストールし、PAM を構成し、LDAP サーバーをソフトウェアに接続し、ソフトウェアプロバイダを追加および構成し、スマートカードを登録しテストします。ステップの順序は次のとおりです。

1. `smartcard` パッケージをインストールします。米国政府組織の場合は、`pkcs11_cackey` パッケージもインストールします – 232 ページの「スマートカードパッケージのインストール」。
2. スマートカードサーバーおよびスマートカードクライアント上で Secure Shell の OpenSSH バージョンを実行します – 『Oracle Solaris 11.3 での Secure Shell アクセスの管理』 の 「Secure Shell の OpenSSH 実装をインストールして切り替える方法」。
3. (オプション) スマートカードと通信するための `pcsclite` インタフェースを確認します – 232 ページの「スマートカードへの `pcsclite` の使用」。
4. (オプション) デフォルトの構成が不十分な場合 `ccid` XML ファイルを構成します – 233 ページの「スマートカードリーダー用の `libccid` の構成」。
5. スマートカードユーザーのローカルまたはリモートのデスクトップを構成します – 234 ページの「スマートカードを持つユーザーのデスクトップの構成」。
6. 証明書を作成し、証明書の検証用に OCSP レスポンダを構成します 235 ページの「スマートカード用の OCSP 証明書の構成」。

   OCSP は、OCSP が検証する PKI 証明書を管理するために Oracle Solaris の `libpki` モジュールに依存します。証明書を保存して CRL をローカルで使用する場合にはこのステップをスキップできます。
7. X.509 証明書情報をスマートカードから取得するように `pam_pkcs11.conf` を構成します – 240 ページの「スマートカードの X.509 証明書の表示方法」。
8. PAM を `pam_pkcs11` モジュールを使用するように構成します – 242 ページの「スマートカードを使用した 2FA 用に PAM を構成する方法」。
9. ユーザーのスマートカードおよびルート証明書を `ssh` クライアントに登録します – 246 ページの「Secure Shell クライアントをスマートカード用に構成する方法」。

スマートカードパッケージのインストール

`smartcard` パッケージは、システム上でスマートカードで Coolkey 暗号化プロバイダを使用するために必要なすべてのソフトウェアをインストールします。`pkcs11_cackey` パッケージは、米国政府が DoD アクセスに必要とする CACKey 暗号化プロバイダをインストールします。

▼ スマートカードパッケージのインストール方法

始める前に ソフトウェアインストールに関連する権利プロファイルが割り当てられている管理者になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **`smartcard` パッケージがインストールされているかどうかを確認します。**
   次のコマンドの出力は、システムにパッケージがインストールされていないことを示しています。

```
$ pkg info smartcard
pkg: info: no packages matching the following patterns you specified are
installed on the system.  Try querying remotely instead:

        smartcard
```

2. **パッケージをインストールします。**

```
$ pfbash pkg install smartcard
```

3. **CACKey を使用する組織に属している場合、`pkcs11_cackey` パッケージをインストールします。**

```
$ pkg install pkcs11_cackey
```

スマートカードへの `pcsclite` の使用

PC/SC Lite では、スマートカードおよびリーダーと通信するための Windows SCard インタフェースが提供されます。これは Windows SCard と同じ `winscard` API を使用します。PC/SC Lite は、CCID 準拠のスマートカードリーダードライバとして `libccid` ライブラリも使用します。スマートカードの認証には、通常 `pcsclite` ライブラリ (PC/SC Lite) のデフォルトの構成で十分です。

Oracle Solaris では、ライブラリおよびデーモンは、`pkg://library/security/pcsc/pcsclite` パッケージにあります。ライブラリのバージョンでは、パッケージ情報の `Version` フィールドにあります。

```
$ pkg info pcsclite
             Name: library/security/pcsc/pcsclite
```

```
          Summary: Provides smart card services using the SCard API (PC/SC)
...
          Version: version
```

Oracle Solaris は、スマートカード認証に、PC/SC Lite クライアント (libpcsclite.so) および PC/SC Lite サーバーデーモン (pcscd) の両方を使用します。PC/SC Lite は Oracle Solaris 内のサービスのため、svcadm コマンドを使用して pcscd を管理します。

smartcard グループパッケージのインストール後、PC/SC デーモンは無効のままになります。スマートカードのサポートソフトウェアを構成したあと、247 ページの「スマートカードログインのための Oracle Solaris システムの有効化」で pcsc サービスを手動で有効にすることになります。

スマートカードリーダー用の libccid の構成

スマートカードの認証には、通常 libccid のデフォルトの構成で十分です。デバッグ中に構成を変更することがあるかもしれません。

ライブラリのバージョンでは、パッケージ情報の Version フィールドにあります。

```
$ pkg info ccid
             Name: library/security/pcsc-lite/ccid
          Summary: Provides smart card reader drivers for pcsclite (PC/SC)
...
          Version: version
```

CCID ドライバの構成ファイル、Info.plist でデバッグレベルの設定や電圧レベルの変更ができます。これはデフォルトで、/usr/lib/$ISA/pcsc/drivers/ifd-ccid.bundle/Contents ディレクトリにインストールされます。

CCID ドライバは pcsclite デバッグ機能を使用します。デバッグ出力は、pcsclite デバッグを構成する方法に応じて、stdout または syslog に送信されます。

▼ libccid の構成およびデバッグ方法

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **有用なデバッグレベルを設定します。**

   デバッグレベルは、ifdLogLevel フィールドに設定します。これは、4 つの異なるレベルのバイナリ OR の組み合わせです。

   - 1 – Critical: 重要なエラーメッセージ
   - 2 – Info: どのリーダーが検出されたかなどの情報メッセージ

- 4 – Comm: ホストとリーダーとの間で交換されるすべてのバイトのダンプ
- 8 – Periodic: カードが存在する場合の `pcscd` テスト中のアクティビティーの定期的なログ (1/10 秒ごと)

デフォルトでデバッグレベルは 3 (1 + 2) に設定され、critical および info のレベルに対応します。

2. **電圧レベルを変更するには、`ifdDriverOptions` フィールドを修正します。**

   電圧レベルは 4 つの異なるレベルのバイナリ OR の組み合わせです。

   - 0 – カードを 5V で電源オン (デフォルト値)
   - 16 – カードを 3V で電源オン、3V で失敗する場合 5V で電源オン
   - 32 – カードを 1.8V で電源オンにし、続いて 3V、次に 5V で電源オン
   - 48 – スマートカードリーダーが電圧レベルを決定

   デフォルトでは、電圧レベルは 0 に設定され、5V に対応します。

3. **ドライバを再起動して、修正された `Info.plist` 設定を読み取ります。**

   2 つの選択肢があります。

   - **すべての CCID リーダーの接続を外します (CCID ドライバがアンロードされます)。続いて、それらを再度接続します。**
   - **または、`pcsc` サービスデーモンを再起動できます。**

     ```
     # svcadm restart pcsc
     ```

次の手順 このファイルでバージョン番号および USB デバイス番号を構成することもできます。

スマートカードを持つユーザーのデスクトップの構成

システムデスクトップにアクセスする必要のあるユーザーは、管理者によってユーザー用に構成されたローカルまたはリモートの X11 デスクトップが必要です。ローカルデスクトップの場合、管理者は `solaris-desktop` グループパッケージをインストールする必要があります。リモートデスクトップのアクセスには、管理者は XDMCP を構成する必要があります。

▼ ローカルデスクトップを構成する方法

始める前に ソフトウェアインストールに関連する権利プロファイルが割り当てられている管理者になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **solaris-desktop グループパッケージがインストールされていることを確認します。**

```
$ pkg search -H -o pkg.name solaris-desktop
group/system/solaris-desktop
```

2. **solaris-desktop グループパッケージがインストールされていない場合には、インストールします。**

```
$ pfexec pkg install solaris-desktop
```

これでこのシステムは、デスクトップが必要なスマートカードのユーザーが使用できるようになります。

▼ リモート X11 デスクトップを構成する方法

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **solaris-desktop グループパッケージがインストールされていることを確認します。**

```
# pkg search -H -o pkg.name solaris-desktop
group/system/solaris-desktop
```

2. **gdm で XDMCP 接続を有効にします。**

/etc/gdm/custom.conf ファイルで Enable を true に設定します。

```
[xdmcp] Enable=true
```

詳細は、gdm(1M) のマニュアルページを参照してください。

3. **変更をアクティブ化する前に、ログオフして作業を保存するようユーザーに警告します。**

再起動すると現在のすべての gdm セッションが強制終了されます。

4. **gdm デスクトップウィンドウサービスを再起動して、構成の変更を有効にします。**

```
# svcadm restart gdm
```

これでユーザーが自身のスマートカードを使用してリモートデスクトップにアクセスできるようになります。ユーザーの手順については、255 ページの「スマートカードを使用してリモートの GNOME デスクトップに ssh を実行する方法」を参照してください。

スマートカード用の OCSP 証明書の構成

スマートカードの証明書は、2 番目の認証ファクタに使用されます。スマートカードは、ルート CA によって検証された証明書を含む「持ち物」になります。

注記 - 証明書を保存し、CRL をローカルで使用する場合には、このタスクをスキップできます。

▼ 証明書を構成および検証する方法

この手順では、スマートカード認証用にルート証明書を構成し、ocspd デーモンがスマートカード上にある証明書のステータスを確認できることをテストする方法を示します。認証局 (CA) を構成する端末ウィンドウと、ocspd 検証をテストする別の端末ウィンドウの 2 つのウィンドウが必要になります。

始める前に Oracle Solaris で、libpki ライブラリが OpenSSL で推奨される暗号化プロバイダおよび OpenLDAP ライブラリにすでにリンクされています。openca-ocspd レスポンダは、libpki を使用して PKI 証明書を生成から検証まで管理します。

root 役割になっています。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **テスト認証局 (CA) を設定します。**

   たとえば、次のコマンドはローカルの CA を作成します。

```
# cd /root
# mkdir CertAuth
# cd CertAuth
# mkdir certs private
# chmod g-rwx,o-rwx private
# echo '01' > serial
# touch index.txt
```

2. **openssl.conf ファイルをこの CA を指すように構成します。**

   たとえば、次の openssl.conf ファイルは root CA を指します。

```
# cat << 'EOF' > openssl.conf
[ ca ]
default_ca              = CertAuth

[ CertAuth ]
dir                     = /root/CertAuth
certificate             = $dir/cacert.pem
database                = $dir/index.txt
new_certs_dir           = $dir/certs
private_key             = $dir/private/cakey.pem
serial                  = $dir/serial

default_crl_days        = 7
default_days            = 365
default_md              = sha256

policy                  = CertAuth_policy
x509_extensions         = certificate_extensions
copy_extensions         = copy
```

```
[ CertAuth_policy ]
commonName              = supplied
stateOrProvinceName     = optional
countryName             = optional
emailAddress            = optional
organizationName        = optional
organizationalUnitName  = optional

[ certificate_extensions ]
basicConstraints        = CA:false
extendedKeyUsage        = OCSPSigning

[ req ]
default_bits            = 2048
default_keyfile         = /root/CertAuth/private/cakey.pem
default_md              = sha256

prompt                  = no
distinguished_name      = root_ca_distinguished_name

x509_extensions         = root_ca_extensions

[ root_ca_distinguished_name ]
commonName              = CertAuth

[ root_ca_extensions ]
basicConstraints        = CA:true
EOF
```

3. **既存のシェルで `OPENSSL_CONF` 環境変数を宣言します。**

注記 - CA は自分のネットワークからアクセス可能である必要があります。

```
# export OPENSSL_CONF=/root/CertAuth/openssl.conf
```

4. **CA 鍵および CA 証明書を作成します。**

```
# openssl genrsa -out private/cakey.pem 2048
Generating RSA private key, 2048 bit long modulus
...+++
.............................+++
e is 65537 (0x10001)

# openssl req -new -x509 -days 999 -key private/cakey.pem -out cacert.pem
```

5. **新しい端末でテスト証明書署名要求 (CSR) を作成します。**

```
# unset OPENSSL_CONF
# cd /root
# mkdir test_client
# cd test_client

# openssl genrsa -out testkey.pem 2048

# openssl req -new -key testkey.pem -out testreq.pem
  Country Name (2 letter code) []:
  State or Province Name (full name) []:
  Locality Name (eg, city) []:
  Organization Name (eg, company) []:
  Organizational Unit Name (eg, section) []:
  Common Name (e.g. server FQDN or YOUR name) []:test
```

```
  Email Address []:
  A challenge password []:
  An optional company name []:
```

6. **テスト証明書を作成し、取り消しできることを確認します。**

   端末構成を CA に変更します。

```
# export OPENSSL_CONF=/root/CertAuth/openssl.conf

# cd /root/CertAuth

# openssl ca -in /root/test_client/testreq.pem
  Sign the certificate? [y/n]:y
  1 out of 1 certificate requests certified, commit? [y/n]y

# openssl verify -CAfile cacert.pem certs/01.pem

# cp certs/01.pem /root/test_client/testcert.pem

# openssl ca -revoke /root/test_client/testcert.pem

# openssl ca -gencrl -out crl.pem
```

7. **新しい端末で ocspd デーモンの鍵および証明書署名要求を作成します。**

```
# unset OPENSSL_CONF

# cd /etc/ocspd

# ocspd-genreq.sh
  Please Enter the Server's Subject (eg., CN=OCSP Server, O=OpenCA, C=US):[Enter]
  Please Enter the Algorithm (default: RSA-SHA256):[Enter]
  Please Enter the Key Size (default: 2048):[Enter]

# cp /etc/ocspd/req.pem /root/CertAuth/ocspdreq.pem
# chmod a+r /root/CertAuth/ocspdreq.pem
```

注記 - サーバー鍵を暗号化する場合、プロンプトが表示されたらパスワードを使用します。

---

8. **ocspd の証明書を作成します。**

   pem ファイルを `/etc/ocspd` にコピーします。

```
# export OPENSSL_CONF=/root/CertAuth/openssl.conf
# cd /root/CertAuth
# openssl ca -in ocspdreq.pem
  Sign the certificate? [y/n]:y
  1 out of 1 certificate requests certified, commit? [y/n]y

# openssl verify -CAfile cacert.pem certs/02.pem

# cp /root/CertAuth/certs/02.pem /etc/ocspd/certs/cert.pem
# cp /root/CertAuth/cacert.pem /etc/ocspd/certs
# cp /root/CertAuth/crl.pem /etc/ocspd/crls
```

9. **ocsp サービスを有効化し、ocspd デーモンが daemon として実行していることを確認します。**

デフォルトでは、ocspd デーモンは smartcard グループパッケージのインストール後に無効化されます。Oracle Solaris でのスマートカード認証で OCSPD レスポンダを使用するには、このサービスを有効にする必要があります。

```
# svcs ocsp
STATE          STIME    FMRI
disabled       14:21:16 svc:/application/security/ocsp:default

# svcadm enable ocsp

# svcs ocsp
STATE          STIME    FMRI
online         14:27:13 svc:/application/security/ocsp:default

# ps -ef |grep ocspd
  daemon 22814     1   0 14:27:14 ?           0:00 /usr/lib/ocspd -c /etc/ocspd/ocspd.xml -
d
```

10. 通常のユーザーとして、証明書の失効ステータスを確認します。

```
$ openssl ocsp -issuer /etc/ocspd/certs/cacert.pem \
   -CAfile /etc/ocspd/certs/cacert.pem -url http://localhost:2560/ -serial 1
Response verify OK
1: revoked
        This Update: Jun 12 21:03:32 2016 GMT
        Next Update: Jun 12 21:08:32 2016 GMT
        Revocation Time: Jun 12 20:49:22 2016 GMT

$ openssl ocsp -issuer /etc/ocspd/certs/cacert.pem \
  -CAfile /etc/ocspd/certs/cacert.pem -url http://localhost:2560/ -serial 2
Response verify OK
2: good
        This Update: Jun 12 21:03:54 2016 GMT
        Next Update: Jun 12 21:08:54 2016 GMT
```

スマートカード用の PAM の構成

pam_pkcs11 ログインモジュールは、CACKey および Coolkey スマートカード上に存在する証明書である、X.509 証明書ベースのユーザー認証を有効にします。このモジュールは、ネームサービススイッチ (NSS) を使用して PKCS #11 のスマートカードをローカルでアクセス可能な証明書失効リスト (CRL) または Online Certificate Status Protocol (OCSP) のいずれかから管理および検証します。

すべての Oracle Solaris ログインが PAM を経由します。ユーザーに対してスマートカード認証を有効にするには、ユーザーのスマートカードからの情報を PAM ファイルに追加します。

/etc/security/pam_pkcs11 ディレクトリに、次のファイルを作成または修正します。

- pam_pkcs11.conf – CACKey または Coolkey 暗号化モジュールを特定し、スマートカードからの一部の情報を含み、マッピングファイルを指します

- `subject_mapping` – スマートカードの X.509 証明書の `subject` をカードのログインユーザー、またはユーザーが引き受けられる `root` などの追加の役割にマップします
- `cn_map` – スマートカードの X.509 証明書名 (CN) をログインユーザーの CN、またはログインユーザーが引き受けられる `root` などの追加の役割の CN にマップします

続いて、すべてのログインの `auth` PAM スタックが、2 番目の認証ステップを要求するように修正されます。この 2 番目のステップでは、PKCS #11 ライブラリを使用してスマートカードの X.509 証明書を検証し、ユーザーにスマートカード PIN を指定するよう求めます。

▼ スマートカードの X.509 証明書の表示方法

この手順では、CACKey または Coolkey を暗号化モジュールとして使用するスマートカードを認識するように、`pam_pkcs11` を構成します。この構成には、Secure Shell へのスマートカード認証のサポートが含まれます。

この準備のあと、この情報を使用してユーザーのスマートカードアクセスを構成します。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

『Oracle Solaris 11.3 での Secure Shell アクセスの管理』 の 「Secure Shell の OpenSSH 実装をインストールして切り替える方法」を完了し、Secure Shell の OpenSSH バージョンを実行しています。ユーザーのスマートカードが入ったスマートカードリーダーは Oracle Solaris システムに接続されています。システムに `pcsclite` および `ccid` パッケージがインストールされています。

1. **`pcsc` サービスをまだ有効にしていない場合は、有効にします。**

```
# svcadm enable pcsc
```

このサービスは、`pam_pkcs11` モジュールがスマートカードとの通信に使用する `pcscd` デーモンを起動します。

2. **`pam_pkcs11.conf` ファイルを `pam_pkcs11.conf.orig` にコピーします。**

```
# cd /etc/security/pam_pkcs11
# cp pam_pkcs11.conf pam_pkcs11.conf.orig
```

3. **PAM をスマートカードの CACKey または Coolkey 暗号化モジュールを使用するよう構成します。**

適切なモジュールを pam_pkcs11.conf ファイルに追加します。

- **CACKey を暗号化モジュールとして特定し、サポートします。**

```
# pfedit pam_pkcs11.conf
use_pkcs11_module = cackey;
```

この行に続いて、CACKey のサポートを追加します。

```
# CACKey support
pkcs11_module cackey {
     module = /usr/lib/$ISA/libcackey.so;
     description = "CACKey";
     slot_num = 0;
     support_threads = false;
     ca_dir =/etc/security/pam_pkcs11/cacerts;
     crl_dir =/etc/security/pam_pkcs11/crls;
     cert_policy = none;
     crl_policy = none;
  }
```

- **Coolkey を暗号化モジュールとして特定し、サポートします。**

```
# cd /etc/security/pam_pkcs11
# pfedit pam_pkcs11.conf
use_pkcs11_module = coolkey;
```

この行に続いて、Coolkey のサポートを追加します。

```
# Coolkey support
pkcs11_module coolkey {
     module = /usr/lib/$ISA/libcoolkeypk11.so;
     description = "Coolkey";
     slot_num = 0;
     support_threads = false;
     ca_dir =/etc/security/pam_pkcs11/cacerts;
     crl_dir =/etc/security/pam_pkcs11/crls;
     cert_policy = none;
     crl_policy = none;
  }
```

4. **引き続き pam_pkcs11.conf ファイルで use_mappers の定義を見つけて変更します。**
このエントリは、証明書を検証できる証明書パラメータを示します。

```
use_mappers = cn, subject, openssh, null;
```

サポートされているマッパーの完全なリストは、pam_pkcs11.conf ファイルにあります。

5. **mapper subject 定義を見つけて変更します。**

```
# Certificate Subject to login based mapper
# provided file stores one or more "Subject -> login" lines
mapper subject {
     debug = false;
     module = internal;
     ignorecase = false;
     mapfile = file:///etc/security/pam_pkcs11/subject_mapping;
}
```

242 ページの「スマートカードを使用した 2FA 用に PAM を構成する方法」で subject_mapping マップファイルを作成することになります。

6. **mapper cn 定義を見つけて変更します。**

```
mapper cn {
       debug = true;
       module = internal;
       ignorecase = true;
       mapfile = file:///etc/security/pam_pkcs11/cn_map;
  }
```

242 ページの「スマートカードを使用した 2FA 用に PAM を構成する方法」で cn_map マップファイルを作成することになります。

7. **mapper openssh 定義を見つけて変更します。**

```
  # Search public keys from user's $HOME/.ssh/authorized_keys for match
  mapper openssh {
       debug = false;
       module = /usr/lib/pam_pkcs11/$ISA/openssh_mapper.so;
  }
```

8. **pam_pkcs11.conf ファイルを終了します。**

9. **pam_pkcs11.conf ファイルに制限されたアクセス権を設定します。**

```
# chmod 644 pam_pkcs11.conf
```

10. **ユーザーのスマートカードの情報を表示できることを確認します。**

    a. **端末ウィンドウで、pkcs11_inspect コマンドを実行します。**

    ```
    # /usr/lib/pam_pkcs11/pkcs11_inspect
    ```

    b. **プロンプトでユーザーのスマートカード PIN を入力します。**

    PIN の入力後、ユーザーのマートカードからの X.509 証明書情報が表示されるはずです。サンプル出力については、242 ページの「スマートカードを使用した 2FA 用に PAM を構成する方法」のステップ 1を参照してください。

次の手順 スマートカード認証用の PAM の構成を完了するには、242 ページの「スマートカードを使用した 2FA 用に PAM を構成する方法」に進みます。

▼ スマートカードを使用した 2FA 用に PAM を構成する方法

この手順では、スマートカードユーザーを認証するための、pam_pkcs11 モジュールの構成を完了する方法を示します。この手順の例は、米国政府発行の CACKeys の場合の例です。スマートカードユーザーごとにこれらのステップに従う必要があります。

始める前に 240 ページの「スマートカードの X.509 証明書の表示方法」を完了しています。

root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **ユーザーのスマートカードの情報を表示します。**

```
# /usr/lib/pam_pkcs11/pkcs11_inspect
```

PIN の入力後、ユーザーのマートカードからの X.509 証明書情報が次のように表示されるはずです。

```
PIN for token:
Printing data for mapper cn:
LNAME.FNAME.ID
Printing data for mapper subject:
/C=US/O=U.S. Government/OU=DoD/OU=PKI/OU=Division/CN=LNAME.FNAME.ID
Printing data for mapper openssh:
ssh-rsa AAAAB3NzaC1yc2EAAAA ...
... fname.lname@example.org
Printing data for mapper cn:
LNAME.FNAME.ID
Printing data for mapper subject:
/C=US/O=U.S. Government/OU=DoD/OU=PKI/OU=Division/CN=LNAME.FNAME.ID
Printing data for mapper openssh:
ssh-rsa AAAAB3NzaC1yc2EAAAA ...
... fname.lname@example.org
...
Printing data for mapper cn:
DoD Root CA ...
...
Printing data for mapper subject:
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD Root CA

Printing data for mapper cn:
DOD CA-30

Printing data for mapper subject:
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD CA-30
...
```

2. **subject_mapping ファイルを作成します。**

このファイルを /etc/security/pam_pkcs11/subject_mapping.example からコピーします。

```
# cd /etc/security/pam_pkcs11
# cp subject_mapping.example subject_mapping
```

3. **subject_mapping ファイルで、X.509 証明書からのユーザーの subject 値をそれらのログイン名にマップします。**

format 行はマッピング形式を記述します。

Printing data for mapper subject: の最初のインスタンスに続く行からの値を使用します。例:

```
/C=US/O=U.S. Government/OU=DoD/OU=PKI/OU=Division/CN=LNAME.FNAME.ID
```

```
# Mapping file for Certificate Subject
# format: Certificate Subject -> login
#
## User certificates
/C=US/O=U.S. Government/OU=DoD/OU=PKI/OU=Division/CN=LNAME.FNAME.ID -> login
...
## Root certificate authority
...
```

米国政府によって発行されていないスマートカードは、証明書サブジェクトの値が異なります。

注記 - root アカウントに一意のスマートカードがある場合は、root をユーザーとして扱い、X.509 証明書からの root の証明書名を root のログイン名にマップします。

---

4. (オプション) スマートカードユーザーが root などのほかの役割を引き受けることが許可されている場合、カードからの正しい証明書を正しい識別情報にマップします。

- この例では、root CA の証明書は、root の役割の証明書になります。

```
# pfedit subject_mapping
# Mapping file for Certificate Subject
# format: Certificate Subject -> login
#
## User certificates
...
## Certificate name mapped to the root account
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD CA-3 -> root
```

- この例では、root の役割の証明書はルート CA の証明書とは異なります。

```
# pfedit subject_mapping
# Mapping file for Certificate Subject
# format: Certificate Subject -> login
#
## User certificates
...
## Certificate name mapped to the root account
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD CA-30 -> root
```

- この例では、DOD CA-29 証明書サブジェクトが sysadmin の役割にマップします。

```
# pfedit subject_mapping
# Mapping file for Certificate Subject
# format: Certificate Subject -> login
#
## User certificates
...
## Certificate name mapped to the sysadmin role
/C=US/O=U.S. Government/OU=DoD/OU=PKI/CN=DOD CA-29 -> sysadmin
```

米国政府によって発行されていないスマートカードは、証明書名の値が異なります。

5. **cn_map ファイルに CN 値をマップします。**

   - /etc/security/pam_pkcs11/cn_map ファイルを作成します。
   - X.509 証明書からのユーザーの証明書名をユーザーのログイン名にマップします。
   - ユーザーが役割になることができる場合は、その役割に適切な証明書名をマップします。

```
# pfedit cn_map
# Mapping file for Certificate Name
# format: Certificate Name -> login
#
## User certificate names
LNAME.FNAME.ID -> login
... many user entries

## Certificate name mapped to the root account
DOD CA-3 -> root
```

6. **マッピングファイルに制限されたアクセス権を設定します。**

```
# chmod 644 cn_map subject_mapping
```

7. **login PAM 構成ファイルの auth スタックに pam_pkcs11 を最初のモジュールとして追加します。**

```
# cd /etc/pam.d
# cp login login.orig
# pfedit login
# login service (explicit because of pam_dial_auth)
#
## pam_pkcs11 enables smart card logins
auth sufficient         pam_pkcs11.so
auth definitive         pam_user_policy.so.1
...
```

8. **other ファイルを同様に変更します。**

```
# cp other other.orig
# pfedit other
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
#
## pam_pkcs11 enables smart card logins
auth sufficient         pam_pkcs11.so
auth definitive         pam_user_policy.so.1
...
```

9. **PAM 構成をテストします。**

   a. **ローカルデスクトップにログインします。**

   b. **リモートデスクトップにログインします。**

c. **ssh コマンドを使用してログインします。**

d. **ローカルコンソールを使用してログインします。**

PAM およびテストの詳細は、『Oracle Solaris 11.3 での Kerberos およびその他の認証サービスの管理』 の 第 1 章, 「プラグイン可能認証モジュールの使用」を参照してください。

Secure Shell クライアントのスマートカード用の構成

証明書の検証のため、スマートカードは Secure Shell サーバーに接続する必要があります。Secure Shell は OpenSSH に基づき、クライアントに対して必要な PKCS #11 サポートを提供するため、スマートカード用の追加の構成は不要です。Secure Shell サーバーはどの OpenSSH バージョンでも実行できます。『Oracle Solaris 11.3 での Secure Shell アクセスの管理』 の 「Secure Shell の OpenSSH 実装をインストールして切り替える方法」を参照してください。

Secure Shell クライアントから、ユーザーのスマートカード上の非公開鍵が、鍵ベースの認証を使用してリモートの Secure Shell サーバーを認証します。ユーザーは鍵を構成する必要があります。

▼ Secure Shell クライアントをスマートカード用に構成する方法

この手順では、スマートカードユーザーがスマートカードから公開鍵を取得し、その鍵を使用して Secure Shell に対してカードを識別し、それを認識するように Secure Shell を構成します。

始める前に スマートカードが入ったスマートカードリーダーが Oracle Solaris システムに接続されています。システムに `pcsclite` および `ccid` パッケージがインストールされ、`pcscd` デーモンが有効になっています。

1. **スマートカードから公開鍵を取得します。**

   手順については、240 ページの「スマートカードの X.509 証明書の表示方法」を参照してください。

2. **公開鍵部分をホームディレクトリの `.ssh` ディレクトリに追加します。**

   a. **`$HOME/.ssh` ディレクトリに `authorized_keys` ファイルを作成します。**

```
$ cd ; mkdir .ssh ; chmod 755 .ssh
$ cd .ssh ; touch authorized_keys
```

b. 前の出力からの公開鍵情報を `authorized_keys` ファイルにコピーします。

出力の最初の公開鍵を次のように `authorized_keys` ファイルに追加します。

```
Printing data for mapper openssh:
ssh-rsa AAAAB3NzaC1yc2EAAAA ...
... fname.lname@example.org
```

鍵は `ssh-` *key-signing-algorithm* で始まり、電子メールアドレスで終わります。コピー&ペーストする際は空白文字を挿入しないでください。

c. `authorized_keys` ファイルのアクセス権が 600 であることを確認します。

```
$ chmod 600 authorized_keys
```

3. アクセスをテストします。

CCID 準拠のスマートカードリーダーが接続されている PC またはワークステーションから、`ssh` と入力してスマートカードサーバーに接続します。

```
$ ssh username@SSH-server
```

X.509 証明書ベースの CAC またはスマートカードおよび PIN によって認証されます。

注記 - `ssh` ログインプロンプトで、システムは今はスマートカード上の X.509 証明書情報から認証しているため、ここではシステムはパスワードの代わりに PIN を要求するはずです。

---

4. テスト後ただちにログアウトします。

Secure Shell 接続は、サーバーへのセキュアな信頼できるリンクです。ローカル PC またはワークステーションからの攻撃の可能性を防ぐため、アクティブに機能していないときには、ユーザーはサーバーをログアウトするか、スマートカードまたは CAC を取り出す必要があります。

スマートカードログインのための Oracle Solaris システムの有効化

システムでのスマートカードの使用を有効化するなかで、構成は時間のかかる部分です。230 ページの「スマートカードのログインのための Oracle Solaris システムの構成」での構成タスクの完了後、`svc://system/security/pcsc` スマートカードサービスを有効にしてスマートカードの使用を有効にします。

▼ スマートカード認証を有効にする方法

始める前に `root` 役割になっています。230 ページの「スマートカードのログインのための Oracle Solaris システムの構成」のタスクを完了しておきます。確認するには、231 ページの「主なスマートカード構成タスク」を参照してください。

1. **スマートカードサーバーで OCSP のルート CA 証明書をインポートします。**

   注記 - 証明書を保存して CRL をローカルで使用する場合にはこのステップをスキップできます。

   証明書は、236 ページの「証明書を構成および検証する方法」で構成およびテストしています。

   ```
   # pktool import "Root CA certificates" -i CACert.pem
   ```

   ここで *CACert.pem* は base-64 形式のルート CA 証明書ファイルです。

2. **スマートカードユーティリティーを有効にします。**

   ```
   # svcadm enable pcsc
   ```

Web ブラウザおよび電子メールでのスマートカード使用の有効化

ユーザーは、スマートカードからの適切な証明書を使用してセキュアなブラウザおよび電子メールにアクセスできます。証明書は、証明書の階層に依存します。通常、最上位の証明書が中間の証明書に署名し、中間の証明書がサイトの証明書に署名します。最上位の証明書をインポートして信頼すると、その他をインストールする必要がなくなります。

この証明書は SSL/PKI を使用して Web サイトに認証し、電子メールを署名および暗号化します。スマートカードを使用して Firefox Web ブラウザおよび Thunderbird 電子メールに認証するには、ブラウザおよび電子メールを CAC カードからクライアント証明書を読み取るように構成します。

▼ Web および電子メールの使用のためにスマートカード証明書をダウンロードする方法

この手順では、アクセスにスマートカードを必要とするアプリケーションに対して認証する証明書をダウンロードします。証明書のチェーン全体または階層が必要です。証明書を使用するには、249 ページの「認証にスマートカードを使用するように Firefox を構成する方法」および250 ページの「電子メールの署名および暗号化にスマートカードを使用するように Thunderbird を構成する方法」に進みます。

この例では、米国政府の証明書を使用します。その他の組織では、セキュリティー管理者の指示に従って証明書を特定しインストールしてください。

1. **Web ブラウザを開きます。**
2. **`http://iase.disa.mil/pki-pke/Pages/tools.aspx` に進みます。**
3. **「PKI and PKE Tools」セクションで、「Trust Store」ボタンをクリックします。**
4. **「PKI CA Certificate Bundles: PKCS#7」セクションで「For DoD PKI Only - Version *n.n*」の URL をクリックしてダウンロードします。**
5. **ファイルを解凍し、`README.txt` 内の指示に従います。**

▼ 認証にスマートカードを使用するように Firefox を構成する方法

この手順では、スマートカード認証を必要とするサイトで認証するように Firefox を構成します。

注記 - スマートカードソフトウェアは、32 ビット版の Firefox ブラウザで動作します。64 ビット版のブラウザでは動作しません。

始める前に 249 ページの「Web および電子メールの使用のためにスマートカード証明書をダウンロードする方法」を完了している必要があります。

1. **このアプリケーションのみに対して証明書をダウンロードします。**
2. **CAC スマートカードを挿入します。**
   緑色のライトが点滅するはずです。

3. **CAC モジュールを Firefox にセキュリティーデバイスとして追加します。**

   a. **Firefox の「オプション」メニューから、「詳細」セクションに移動し、「セキュリティデバイス」ボタンをクリックしてから、「追加」ボタンをクリックします。**

   b. **モジュール名として「CAC Module」と入力し、適切な CAC モジュールを参照します。**

      - CACKey には `/usr/lib/libcackey.so` を選択します。
      - Coolkey には `/usr/lib/libcoolkeypk11.so` を選択します。

   これで、Firefox は CAC カードからクライアント証明書を読み取ることができるようになりました。

4. **Firefox で、CAC が有効にされた Web サイトに移動して構成をテストします。**

   PIN を入力するように求められ、そのサイトが「PKI 証明書が検出されました」と報告した場合、正しい構成になっています。

▼ 電子メールの署名および暗号化にスマートカードを使用するように Thunderbird を構成する方法

この手順では、スマートカードからの証明書を使用して、電子メールを暗号化および署名するよう Thunderbird を構成します。

始める前に 249 ページの「Web および電子メールの使用のためにスマートカード証明書をダウンロードする方法」を完了しておきます。

1. **このアプリケーションのみに対して証明書をダウンロードします。**

2. **CAC スマートカードを挿入します。**

   緑色のライトが点滅するはずです。

3. **CAC モジュールを Thunderbird にセキュリティーデバイスとして追加します。**

   a. **Thunderbird の「オプション」メニューから、「詳細」セクション、「証明書」タブに移動し、「セキュリティデバイス」ボタンをクリックしてから、「追加」ボタンをクリックします。**

   b. **モジュール名として「CAC Module」と入力し、適切な CAC モジュールを参照します。**

- CACKey には `/usr/lib/libcackey.so` を選択します。
- Coolkey には `/usr/lib/libcoolkeypk11.so` を選択します。

これで、Thunderbird は CAC カードからクライアント証明書を読み取ることができるようになりました。

4. **Thunderbird で自分自身に電子メールを送信して構成をテストします。**
電子メールメッセージを作成する際、「セキュリティ」メニューをプルダウンし、「S/MIME 署名」および「S/MIME 暗号化」を選択してからメッセージを送信します。

スマートカードの使用

注意 - これらの手順は、Sun Ray では機能しません。SRS を実行しているシステムでは、統合された Sun Ray スマートカードの認証手順に従います。

スマートカードでは、パスワードではなく個人識別番号 (PIN) を使用します。スマートカードは、スマートカードの所有者のみに知られている PIN によって悪用から保護されます。スマートカードを使用するには、コンピュータに接続されたスマートカードリーダーにカードを挿入して、プロンプトが表示されたら、PIN を入力します。スマートカードは、スマートカードを所有し、PIN を知っている人のみが使用できます。

コンピュータで使用するには、セッションの期間中 CAC、PIV または X.509 証明書ベースのスマートカードをリーダーに残したままにする必要があります。スマートカードがリーダーから取り出されると、既存のログインセッションにおいて再認証を必要とするすべてのアプリケーションで資格情報を使用できなくなります。

注意 - アクティブでない状態の期間はログアウトしてください。認証されたスマートカードは、サーバーへのセキュアな信頼できるリンクです。ローカルシステムからの攻撃の可能性を防ぐため、アクティブに機能していないときには、ログアウトするか、スマートカードまたは CAC を取り出す必要があります。

次のタスクでは、システムへのさまざまなエントリポイントを使用して、スマートカードで Oracle Solaris にログインする方法について説明しています。

- 252 ページの「スマートカード認証を使用してローカルコンソールからログインする方法」
- 253 ページの「スマートカード認証を使用して役割としてログインする方法」
- 254 ページの「スマートカード認証で ssh を使用してリモートでログインする方法」

- 255 ページの「スマートカードを使用してリモートの GNOME デスクトップに ssh を実行する方法」
- 255 ページの「スマートカードを使用してローカルの GNOME デスクトップにログインする方法」
- 259 ページの「スクリーンセーバーでスマートカードを使用して認証する方法」

▼ スマートカード認証を使用してローカルコンソールからログインする方法

始める前に PC またはワークステーションに接続された CCID 準拠のスマートカードリーダーにスマートカードを挿入しておきます。

1. **プロンプトで、ユーザー名を入力します。**

```
smartcard console login: username
Smartcard authentication starts
Smart card found.
Welcome username!
```

2. **スマートカード PIN を入力します。**

```
Smart card PIN: nnnnnnnn
verifying certificate
Last login: ...
Oracle Corporation   SunOS 5.11
You have new mail.
username@server:~$
```

3. **間違った PIN を入力した場合、再度 PIN を入力します。**

```
Error 2320: Wrong smartcard PIN
date smartcard login: open_pkcs11_login() failed x0000000A0
Login incorrect
Smartcard authentication starts
Smart card found.
Please insert your smart card or enter your username.
Smartcard console login: nnnnnnnn
verifying certificate
Oracle Corporation   SunOS 5.11
You have new mail.
username@server:~$
```

4. **(オプション) 役割を引き受けるは、その役割に su を実行します。**

   たとえば、root の役割になります。

```
$ su -
```

   端末にスマートカード認証の進行状況が表示されます。

```
Smartcard authentication starts
Smart card found.
```

```
Welcome root!
```

5. **スマートカード PIN を入力します。**

```
Smart card PIN: nnnnnnnn
verifying certificate
Oracle Corporation    SunOS 5.11
You have new mail.
root@server: ~#
```

正しい PIN を入力した場合、ログインします。

6. **Control-D と入力してセッションからログアウトします。**

▼ スマートカード認証を使用して役割としてログインする方法

始める前に CCID 準拠のスマートカードリーダーが接続されている PC またはワークステーションにログインしておきます。root アカウントは、独自の証明書を持っています。242 ページの「スマートカードを使用した 2FA 用に PAM を構成する方法」のステップ 3で、特権ユーザーに対する証明書の特定およびマップを開始しています。

1. **端末ウィンドウを開きます。**

2. **ユーザーを役割に切り替えます。**
この例では、root の役割に切り替えます。

```
$ su -
```

端末にスマートカード認証の進行状況が表示されます。

```
Smartcard authentication starts
Smart card found.
Welcome root!
```

3. **スマートカード PIN を入力します。**
正しい PIN を入力した場合、一連の「verifying certificate」メッセージが表示され、ログインします。

```
Smart card PIN: nnnnnnnn
verifying certificate
verifying certificate
...
Oracle Corporation    SunOS 5.11
You have new mail.
root@server: ~#
```

4. **誤った PIN を入力した場合、役割ではなくユーザー名としてログインします。**

```
Error 2320: Wrong smartcard PIN
```

```
su: Authentication failed
username@server: ~$
```

役割に切り替えるには、この手順を繰り返します。

5. **exit と入力してセッションからログアウトします。**

▼ スマートカード認証で ssh を使用してリモートでログインする方法

始める前に スマートカード用に構成されている Secure Shell サーバーがあること。PC またはワークステーションに接続された CCID 準拠のスマートカードリーダーにスマートカードを挿入しておきます。

1. **端末ウィンドウを開きます。**

2. **ssh を使用してリモートサーバーに接続します。**

   - **Oracle Solaris Secure Shell クライアントから Oracle Solaris Secure Shell サーバーに接続する場合には、次のコマンドを入力します。**

     ```
     $ ssh username@SSH-server
     ```

   - **Secure Shell クライアントまたはサーバーのどちらかが Oracle Solaris ではない場合、PKCS #11 ライブラリへのフルパスを指定します。**

     使用する暗号化モジュールを管理者に確認します。

     ```
     $ ssh -I /usr/lib/libcackey.so username@SSH-server
     $ ssh -I /usr/lib/libcoolkeypk11.so username@SSH-server
     ```

   端末にスマートカード認証の進行状況が表示されます。

   ```
   Smartcard authentication starts
   Smart card found.
   Welcome LNAME.FNAME.ID!
   ```

3. **スマートカード PIN を入力します。**

   ```
   Smart card PIN: nnnnnnnn
   ```

   - 正しい PIN を入力した場合、「`verifying certificate`」メッセージが表示され、ログインします。
   - 誤った PIN を入力した場合、エラーメッセージ「`Error 2320: Wrong smartcard PIN`」が表示されます。スマートカード認証が再開したら、正しい PIN を入力します。

4. **ssh セッションを終了するには、`Control-D` と入力します。**

▼ スマートカードを使用してリモートの GNOME デスクトップに ssh を実行する方法

始める前に 管理者は235 ページの「リモート X11 デスクトップを構成する方法」を完了しておきます。

- **コマンド行から、または XDMCP チューザを呼び出すクライアントアプリケーションからログインできます。**
  - **ssh コマンドに -x オプションを使用します。**

    ```
    $ ssh -X GNOME-desktop-server
    ```

    詳細は、ssh(1) のマニュアルページを参照してください。
  - **リモートの X11 デスクトップ表示には、クライアントアプリケーションから XDMCP チューザを呼び出します。**

    使用可能なクライアントアプリケーションには Hummingbird が含まれます。

▼ スマートカードを使用してローカルの GNOME デスクトップにログインする方法

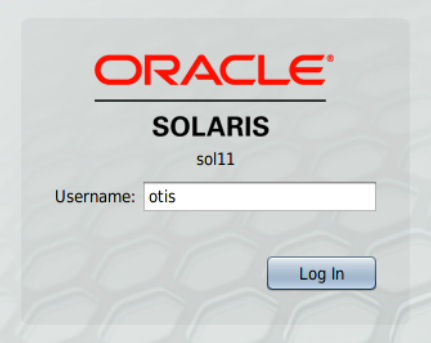
始める前に 管理者は234 ページの「ローカルデスクトップを構成する方法」を完了しておきます。PC またはワークステーションに接続された CCID 準拠のスマートカードリーダーにスマートカードを挿入しておきます。

1. **ログインします。**

   2 つのスマートカードダイアログボックス (「OK」ボタンを押して確認する必要があります) が表示されます。

2. 「OK」キーを押してから、スマートカードを挿入するか、ユーザー名を入力します。

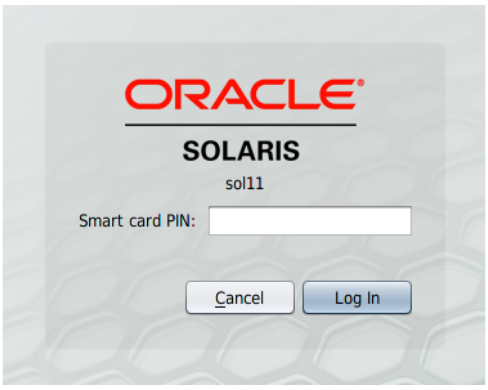
3. 「OK」キーを押してユーザー名を入力します。



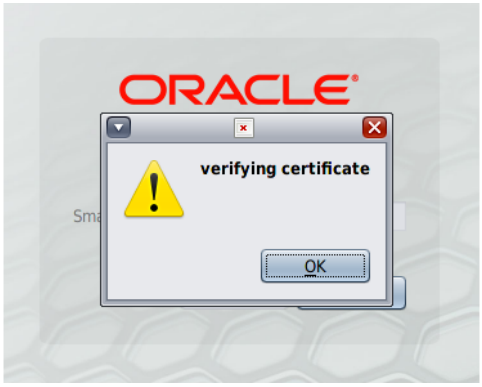
システムはスマートカードを探し、「Smart card found.」を表示します。

4. **「OK」キーを押します。**
「Welcome」ダイアログボックスでもう一度このキーを押します。

5. **プロンプトでスマートカード PIN を入力し、「Log In」ボタンを押します。**



システムは証明書を検証したあと、ユーザーをログインさせます。



6. **前述のステップのいずれかが失敗した場合は、適切なアクションを実行します。**

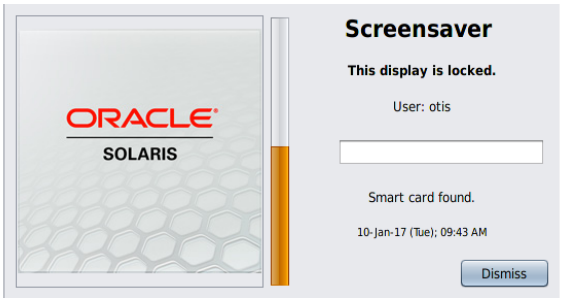
システムにより、エラーについて説明するエラーダイアログボックスが表示されます。

- `Error 2312: open PKCS#11 session failed` – システムがスマートカードを検出できないか、カード上の `subject_mapping` ID がユーザー名と一致しませんでした。管理者に連絡してください。
- `Error 2320: Wrong smartcard PIN` – 間違った PIN が入力されました。「`OK`」ボタンを押して、再度 PIN を入力します。

▼ スクリーンセーバーでスマートカードを使用して認証する方法

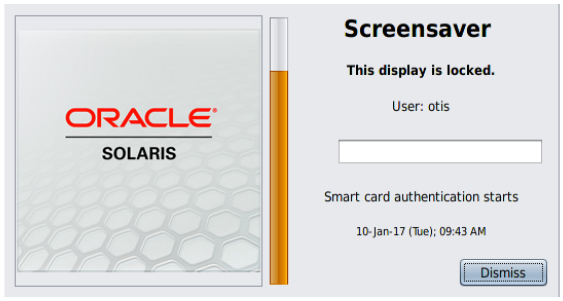
始める前に GNOME スクリーンセーバーが、スマートカード PIN でログインしたデスクトップに表示されています。

1. **カーソルを入力フィールドに置き、スマートカード PIN を入力します。**



`Enter` キーまたは `Return` キーを押します。

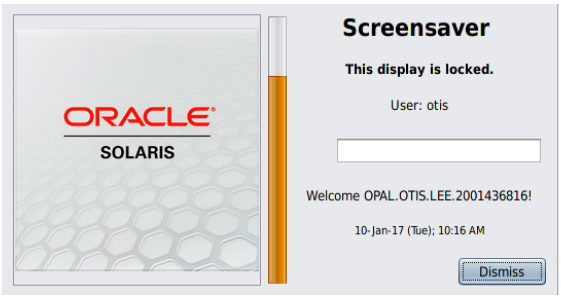
注記 - 「`Dismiss`」ボタンを押さないでください。



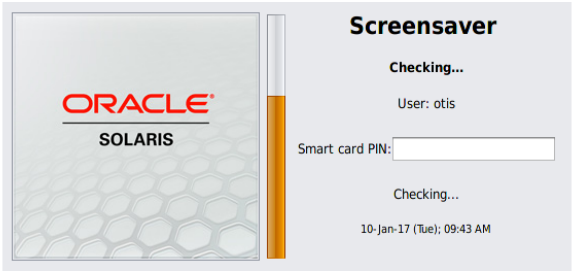
注記 - 画面をロック解除するために別のアカウントが必要な場合にのみ「`Dismiss`」ボタンを押します。その後、別のアカウントのカードを挿入し、入力フィールドにユーザー名を入力し、`Enter` キーまたは `Return` キーを押します。システムは、証明書を見つけるためにカードの読み取りを試行し、この別のアカウントのスマートカードの PIN の入力を要求します。

カードが読み取られているときにはスマートカードリーダーの緑色の LED が点滅し、証明書の `subject_mapping` フィールドが検証されます。

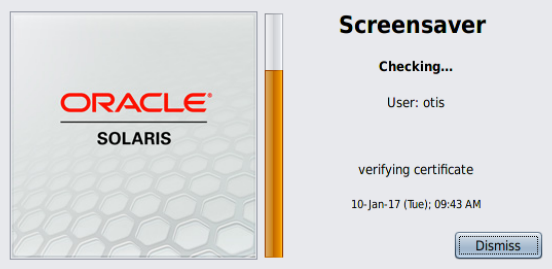
2. **ダイアログボックスにウェルカムメッセージが表示されたら、「`Dismiss`」ボタンを押します。**



3. **スマートカード PIN の入力が求められたら、PIN を入力して `Enter` キーまたは `Return` キーを押します。**



「verifying certificate」ダイアログボックスが表示されます。



ダイアログボックスに「Error 2320: Wrong smartcard PIN」が表示された場合、「Dismiss」ボタンを押して、再度 PIN を入力します。

4. **「Dismiss」ボタンを押してログインします。**

♦ ♦ ♦ 第 11 章

11 Oracle Solaris でのマルチファクタ認証へのワンタイムパスワードの使用

この章では、選択されたユーザーが Oracle Solaris システムにログインする際に、モバイルオーセンティケータからのワンタイムパスワード (OTP) の使用を必要とするように Oracle Solaris を構成する方法について説明します。この章で扱う内容は、次のとおりです。

- 263 ページの「Oracle Solaris での OTP について」
- 264 ページの「Oracle Solaris での OTP 管理」
- 264 ページの「Oracle Solaris での OTP の構成および使用」

Oracle Solaris での OTP について

OTP は Oracle Solaris にログインする際に、識別情報の 2 つ目の証明を提供します。識別情報の 2 つ目の証明を使用することは、ツーファクタ認証 (2FA) と呼ばれます。システムでは、最初に UNIX パスワード、次にモバイルオーセンティケータアプリからの OTP の入力が求められます。システムはこれらの 2 つの認証を検証したあと、ユーザーをログインさせます。詳細は、221 ページの「ツーファクタ認証について」を参照してください。

Oracle Solaris の OTP は、RFC 4226 および RFC 6238 の HMAC ベースおよび時間ベースの OTP の仕様に準拠しているため、これらの仕様に準拠したオーセンティケータと連携できます。

Oracle Solaris は、`system/security/otp` IPS パッケージに OTP を配布します。`solaris-small-server`、`solaris-large-server`、および `solaris-desktop` グループは、次の項目を含むこのパッケージを配布します。

- OTP PAM モジュール – `pam_otp_auth` は OTP を実装します。`pam_otp_auth` がログイン PAM スタック内のモジュールのときには、ユーザーは OTP を入力する必要があります。詳細は、`pam_otp_auth`(5) のマニュアルページを参照してください。

- OTP PAM ポリシー構成ファイル – `otp` および `otp_strict` は、OTP パッケージに用意されたユーザーごとの 2 つの PAM 構成ファイルです。
- OTP 管理コマンド – `otpadm` は、ユーザーの OTP 認証の構成に使用するコマンドです。ユーザーは、このコマンドを使用して自分の鍵を管理できます。詳細は、`otpadm`(1M) のマニュアルページを参照してください。
- OTP 権利プロファイル – OTP Auth Manage All Users 権利プロファイルを使用すると、管理者は `otpadm` コマンドを使用して OTP を管理できます。

個々のユーザーに OTP を割り当てるには、管理者は `useradd` または `usermod` コマンドに `-K pam_policy=otp` オプションを使用します。手順については、264 ページの「Oracle Solaris での OTP の構成および使用」を参照してください。

Oracle Solaris での OTP 管理

OTP Auth Manage All Users 権利プロファイルを持つ管理者ユーザーは、`otpadm` コマンドを使用して、ユーザーの OTP に影響する属性を管理できます。通常のユーザーは自分自身のアカウントの鍵を管理できますが、管理者が OTP パッケージをインストールして、ユーザーに `otp` PAM ポリシーを割り当てる必要があります。

ユーザーごとの OTP を構成するには、User Management 権利プロファイルが割り当てられている必要があります。

管理者は、デフォルト値を継承してはならない OTP 属性を構成する責任があります。自身の秘密鍵を設定するユーザーも、その OTP 属性を設定できます。

注記 - モバイルオーセンティケータアプリおよび Oracle Solaris の `otpadm` ユーティリティーでは、ユーザーがカスタマイズできる属性を制限します。たとえば、ユーザーの OTP コードは 6 桁以上の長さである必要があります。

通常、デフォルトの属性値が適切です。属性の説明については、`otpadm`(1M) のマニュアルページを参照してください。

Oracle Solaris での OTP の構成および使用

OTP の構成には、管理者と OTP ユーザー間の調整が必要です。構成後、OTP ユーザーは自身の UNIX ログイン、およびモバイルオーセンティケータアプリに表示される OTP を使用してサーバーにログインできます。

注意 - OTP の使用を求められる前に、秘密鍵がユーザーのモバイルデバイスにない場合、ユーザーはロックアウトされる可能性があります。

管理者の責任:

1. ログインサーバーに `otp` パッケージがインストールされていることを確認します。
2. ログインサーバーが、正確な時間を維持できることを確認します。

   サーバーは、Precision Time Protocol (PTP) または Network Time Protocol (NTP) クロック同期サービスのクライアントである必要があります。詳細は、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』を参照してください。
3. ユーザーに秘密鍵があることを確認します。ユーザーの秘密鍵は、ユーザーまたは管理者が作成できます。
4. ユーザーに `otp` ユーザーごとの PAM ポリシーを割り当てます。

モバイル認証アプリを持つユーザーの責任:

1. モバイルデバイスにモバイルオーセンティケータアプリをダウンロードします。
2. 秘密鍵を作成するか、管理者がオーセンティケータアプリ用の秘密鍵を作成するときに管理者に協力します。
3. オーセンティケータアプリ上の OTP 構成がログインサーバー上の構成と一致していることを確認します。
4. 管理者がユーザーのモバイルオーセンティケータアプリに `otp` PAM ポリシーを割り当てる前に、そのアプリに秘密鍵を入力します。
5. OTP を求めるプロンプトが表示され、OTP でユーザーがログインすることをテストします。

表 8 タスクマップ: Oracle Solaris での OTP の使用

タスク	説明	参照先
OTP の準備をします。	管理者は、ログインサーバーに OTP をサポートするモジュールがあることを確認します。	266 ページの「OTP の構成方法」
秘密鍵を作成して確認します。	ユーザーはログインサーバー上に秘密鍵を作成し、その秘密鍵を使用してモバイルオーセンティケータを構成します。	266 ページの「OTP の秘密鍵を構成および確認する方法」
秘密鍵を作成してユーザーに送信します。	あるいは、管理者がユーザーの秘密鍵を作成します。	268 ページの「OTP ユーザー用に秘密鍵を設定する方法」
OTP を有効にします。	管理者は、ログイン時にユーザーからの OTP を求めます。	269 ページの「Oracle Solaris システムへのログインに UNIX パスワードおよび OTP を要求する方法」

▼ OTP の構成方法

始める前に `otp` パッケージをインストールするには、ソフトウェアインストールに関連する権利プロファイルを持つ管理者になる必要があります。OTP を管理できるようにするには、OTP Auth Manage All Users 権利プロファイルを持つ管理者になる必要があります。`root` 役割には、これらの権利がすべて含まれています。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **ログインサーバーがクロック同期サービスのクライアントであることを確認します。**

   手順については、『Oracle Solaris 11.3 でのクロック同期と Web キャッシュを使用したシステムパフォーマンスの拡張』を参照してください。

2. **ログインサーバーに `otp` パッケージがインストールされていることを確認します。**

   ```
   login-server $ pkg search -H -o pkg.name -l \<otp\>
   pkg:/system/security/otp
   ```

3. **(オプション) サイトポリシーに対してデフォルトで十分かどうかを判断します。**

   ```
   $ otpadm get
                  mode=timer
             algorithm=hmac-sha1
                digits=6
                  ...
   ```

4. **ユーザーが異なる値を使用する必要がある場合、ユーザーに指示を送ります。**

   - ユーザーがモバイルデバイス上で設定する必要のある属性を含めます。
   - ユーザーが自分の秘密鍵を設定している場合は、ユーザーに指示を送ります。例 46「ユーザーによる長い OTP および強力なアルゴリズムへの変更」および例 47「16 進数の秘密鍵の設定および表示」を参照してください。

▼ OTP の秘密鍵を構成および確認する方法

始める前に あなたはインターネットに接続するモバイルデバイスの所有者で、管理者が266 ページの「OTP の構成方法」を完了しています。

1. **モバイルオーセンティケータアプリをデバイスにダウンロードし、アプリケーションを開きます。**

   アプリストアでオーセンティケータを検索します。モバイルオーセンティケータアプリの 1 例として、Oracle Mobile Authenticator があります。

2. **ログインサーバーで秘密鍵を作成します。**

```
$ otpadm set secret
XXXX nnnn XXXX XXXX nnnn nnnn nnnn XXXX
Enter current code from authenticator: nnnnnnnn
```

サーバーが秘密鍵を表示し、モバイルアプリケーションからのコードの入力が求められます。

3. **表示された秘密鍵をモバイルオーセンティケータに入力します。**

   たとえば、Oracle Mobile Authenticator の画面で、画面右上にあるプラス (+) ボタンを押し、「Enter provided key」を選択し、アカウント (*username@login-server*) の名前を選択し、「Key」の下に秘密鍵を入力します。

4. **モバイルアプリケーションでコードを作成し、それを `otpadm` プロンプトに入力します。**

   `otpadm` プロンプトがオーセンティケータから有効なコードを受け入れると、OTP が構成されて使用する準備ができます。

5. **OTP をテストします。**

   管理者が269 ページの「Oracle Solaris システムへのログインに UNIX パスワードおよび OTP を要求する方法」を完了したあとで、サーバーにログインします。最初にサーバーのログインに対して入力を求められ、次に OTP に対して入力を求められるはずです。OTP を入力すると、ログインできます。

例 46 ユーザーによる長い OTP および強力なアルゴリズムへの変更

管理者は OTP ユーザーに SHA2 アルゴリズムおよび 8 桁パスワードに変更するよう通知します。

1. 電子メールで、管理者はそれらのユーザーに新しいガイドラインに従うように指示します。

   ```
   Users,
   We are changing the mobile authenticator to use a longer password and
   a stronger algorithm.  Please complete the changeover by Friday.
   On the server, open a terminal window and issue the following commands:
   otpadm set algorithm=hmac-sha256 digits=8 secret
   Respond to the prompts and instructions.
   If you have difficulty, notify the administrator.
   ```

2. ユーザーは、モバイルオーセンティケータアプリで `hmac-sha256` アルゴリズムを選択し、桁を 8 に設定します。
3. 各ユーザーは、ログインサーバーで、電子メールからコマンドを実行します。

   ```
   $ otpadm set algorithm=hmac-sha256 digits=8 secret
   1234 abcd 1234 abcd 1234 1234 1234 abcd
   Enter current code from authenticator: nnnnnnnn
   ```

4. 各ユーザーは、モバイルオーセンティケータアプリに秘密鍵を入力します。
5. ユーザーがアプリ上でコードを生成し、ログインサーバープロンプトでそれを入力すると、アプリとサーバーの同期がとられ、構成が完了します。

例 47 16 進数の秘密鍵の設定および表示

ユーザーは、16 進数形式の秘密鍵の入力が求められるモバイルオーセンティケータを所有しています。これらのユーザーは 16 進形式で表示される秘密鍵を作成します。

```
$ otpadm -f hex set secret
 7DDF B236 7023 82A6 F70F 0001 C8B7 F0BE A76C 3F31
```

注意事項 OTP パスワードが失敗した場合、2 つ目に表示される OTP を待機し、試します。

ログインサーバーが OTP 受け入れない場合、モバイルデバイスのクロックとサーバーが同期していることを管理者とともに確認します。

ログインサーバーの時間とモバイルオーセンティケータが同期していない場合、管理者とともに、カウンタベースの OTP ではなく時間ベースの OTP を構成できます。例 49「OTP 認証にタイマーではなくカウンタを使用」を参照してください。

▼ OTP ユーザー用に秘密鍵を設定する方法

始める前に 266 ページの「OTP の構成方法」を完了しておきます。

OTP Auth Manage All Users 権利プロファイルを持つ管理者になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **(オプション) サイトポリシーに対してデフォルトで十分であるかを判断します。**

```
$ otpadm get
                mode=timer
           algorithm=hmac-sha1
              digits=6
                ...
```

2. **ユーザー用に秘密鍵を作成します。**

```
$ pfexec otpadm -u username -f [base32 | hex] set attributes secret
```

たとえば、デフォルトの OTP 属性を使用します。

```
$ pfexec otpadm -u jdoe set secret
```

たとえば、さらに長い OTP を要求します。

```
$ pfexec otpadm -u jdoe set digits=8 secret
```

たとえば、カウンタモードを設定します。

```
$ pfexec otpadm -u jdoe set mode=counter secret
```

デフォルトでは、OTP の秘密鍵は Base32 形式で表示されます。ほとんどのオーセンティケータがこの形式を受け入れますが、一部は 16 進数形式を想定しています。OTP の秘密用に形式を変更するには、例47「16 進数の秘密鍵の設定および表示」を参照してください。

3. **ユーザーへの秘密鍵を取得します。**

   - **秘密鍵をアウトオブバンドでユーザーに送信します。**

     a. **秘密鍵を表示します。**

        ```
        $ pfexec otpadm -u username get secret
        CBA6 5JBR M73T XGZK CNAB 36HG QLE5 PFCR
        ```

     b. **暗号化された電子メールなどのセキュアなチャネルを介してこれを送信します。**

   - **ログインして、秘密鍵を表示し、それをモバイルオーセンティケータアプリに入力するよう、ユーザーに指示します。**

     ```
     username $ otpadm get secret
     CBA6 5JBR M73T XGZK CNAB 36HG QLE5 PFCR
     ```

▼ Oracle Solaris システムへのログインに UNIX パスワードおよび OTP を要求する方法

始める前に 266 ページの「OTP の構成方法」を完了しておきます。

このタスクのステップを完了するには、次の権利プロファイルを持つ管理者になる必要があります。

- User Management 権利プロファイル – ユーザーへの PAM ポリシーの割り当て
- OTP Auth Manage All Users 権利プロファイル – OTP の管理

`root` 役割には、これらの権利がすべて含まれています。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **ユーザーがモバイルオーセンティケータアプリに秘密鍵を入力し確定したことを確認します。**

   管理者とユーザーは次のタスクを完了しています。

- 266 ページの「OTP の秘密鍵を構成および確認する方法」
- 268 ページの「OTP ユーザー用に秘密鍵を設定する方法」

2. **User Management 権利プロファイルを持つ管理者として、ユーザーの PAM ポリシーを `otp` に変更します。**

```
$ pfexec usermod -K pam_policy=otp username
```

3. **OTP ユーザーにログインをテストするよう指示します。**

   ユーザーは、最初に通常のログインパスワード、次に OTP の入力を求められるはずです。

4. **(オプション) ユーザーに対して OTP を無効にするには、次の 2 つのステップを実行します。**

   a. **User Management 権利プロファイルを持つ管理者として、ユーザーの PAM ポリシーとしての `otp` を削除します。**

```
$ pfexec usermod -K pam_policy=  username
```

   b. **OTP Auth Manage All Users 権利プロファイルを持つ管理者として、ユーザーのホームディレクトリから非アクティブの OTP 構成ファイルを削除します。**

```
$ pfexec otpadm -u username expunge
```

例 48 長い OTP および強力なアルゴリズムへの変更の強制

企業で使用されているオーセンティケータアプリは、非常に強力なアルゴリズムと長いパスワードを扱うことができます。強力なセキュリティーポリシーを実装するには、管理者は OTP ユーザーに SHA2 アルゴリズムおよび 8 桁のパスワードに変更するよう通知します。次に、管理者はユーザーの変更を監査します。電子メールおよびユーザーの責任は、例46「ユーザーによる長い OTP および強力なアルゴリズムへの変更」に示されています。

1. ユーザーが OTP 属性を変更する時間を与えたあと、管理者はすべての OTP ユーザーを監査します。

```
$ pfexec otpadm -u username get algorithm digits
              digits=8
          algorithm=hmac-sha256
```

2. ユーザーの構成が以前の出力とは異なる場合、管理者はユーザーがロックアウトされる日付を指定した警告電子メールを送信します。
3. 管理者は、指定された日に、新しい OTP 構成に変更しなかった OTP ユーザーをロックアウトします。

```
$ pfexec usermod -e date username
```

例 49 OTP 認証にタイマーではなくカウンタを使用

この例では、ユーザーはカウンタモードをサポートしているモバイルオーセンティケータを使用しています。管理者は、モバイルオーセンティケータがログインサーバーに同期しない場合、OTP モードを `counter` に設定します。既存のコードの使用を防ぐため、管理者は新しい秘密鍵を設定します。

```
$ otpadm -u username set mode=counter secret
$ otpadm -u username get secret
VOCJ YHTV 2C4O DTDN R34X CGM4 YZVM JJFI
```

管理者は、秘密鍵をアウトオブバンドでユーザーに送信します。秘密を入力する前に、ユーザーはモバイルオーセンティケータアプリがカウンタモードを使用するように設定します。

注意事項 オーセンティケータがユーザーの OTP を確認しない場合、ユーザーは 2 つ目に表示される OTP を待機し、試す必要があります。

ログインサーバーが OTP を受け入れない場合、モバイルデバイスのクロックとサーバーが同期していることを確認します。

♦ ♦ ♦ 第 12 章

12 ネットワークサービスの認証の構成

この章では、Secure RPC を使用して NFS マウントでホストとユーザーを認証する方法について説明します。この章の内容は次のとおりです。

- 273 ページの「Secure RPC について」
- 275 ページの「Secure RPC による認証の管理」

Secure RPC について

Secure RPC (リモート手続き呼び出し) は、認証メカニズムを使用してリモート手続きを保護します。Diffie-Hellman 認証メカニズムは、サービスを要求するホストとユーザーを認証します。この認証メカニズムはデータ暗号化規格 (Data Encryption Standard、DES) 暗号化を使用します。Secure RPC を使用するアプリケーションには、NFS と NIS ネームサービスが含まれます。

NFS サービスと Secure RPC

NFS を使用すると、複数のホストがネットワーク経由でファイルを共有できます。NFS サービスでは、サーバーは、複数のクライアントから利用できるデータとリソースを保持します。クライアントは、サーバーがクライアントと共有するファイルシステムにアクセスできます。クライアントシステムにログインしたユーザーは、ファイルシステムをサーバーからマウントすることによって、そのファイルシステムにアクセスできます。このとき、クライアントシステム上のユーザーには、ファイルはクライアントのローカルファイルシステム上にあるように見えます。NFS のもっとも一般的な使用形態の 1 つは、システムを各オフィスにインストールして、すべてのユーザーファイルを 1 箇所で集中管理することです。`mount` コマンドの `-nosuid` オプションなどのいくつかの NFS 機能を使用すると、権限を持たないユーザーがデバイスやファイルシステムにアクセスすることを禁止できます。

NFS サービスでは Secure RPC を使用して、要求を出したユーザーをネットワーク上で認証します。このプロセスは、*Secure NFS* と呼ばれます。Diffie-Hellman 認証メカニズ

ム AUTH_DH は、DES 暗号化を使用し、承認されたアクセスを保証します。AUTH_DH メカニズムは、AUTH_DES とも呼ばれてきました。詳細については、次を参照してください。

- Secure NFS を設定および管理するには、『Oracle Solaris 11.3 でのネットワークファイルシステムの管理』 の 「Secure NFS システムの管理」を参照してください。

Kerberos 認証

Kerberos は、マサチューセッツ工科大学 (MIT) で開発された認証システムです。このリリースには、RPCSEC_GSS を使用する Kerberos V5 のクライアント側とサーバー側の実装が含まれています。詳細は、133 ページの「Kerberos NFS サーバーを構成する方法」を参照してください。

Secure NFS での DES 暗号化

データ暗号化規格 (Data Encryption Standard、DES) 暗号化機能は 56 ビットの鍵を使用して、データを暗号化します。資格を持つ 2 人のユーザー、すなわち主体が同じ DES 鍵を知っている場合、その鍵を使用してテキストを暗号化または復号化することによって、プライベートに通信できます。DES は比較的高速な暗号化メカニズムです。

DES 鍵を使用する上での問題点は、同じ鍵で暗号化された多数のテキストメッセージを侵入者が収集することによって、鍵が発見されてメッセージが解読される危険性があるということです。このため、Secure NFS などのセキュリティーシステムは鍵を頻繁に変更する必要があります。

Diffie-Hellman 認証と Secure RPC

Diffie-Hellman (DH) のユーザー認証方式は簡単には破られません。クライアントとサーバーは、独自の非公開鍵と公開鍵を使って共通鍵を作り出します。非公開鍵は秘密鍵とも呼ばれます。クライアントとサーバーは、共通鍵を使って相互に通信します。共通鍵は、相互に合意した暗号化機能 (DES など) によって暗号化されます。

認証では、送信側のシステムの共通鍵を使用して現在の時間を暗号化する機能を利用します。受信側のシステムは、その現在の時間を復号し、自分の時間と照合します。クライアントとサーバーで時間が同期している必要があります。クロックを同期するには、Network Time Protocol (NTP) を使用できます。Oracle Solaris ソフトウェアにはデラウェア大学の NTP パブリックドメインソフトウェアが含まれています。ドキュメントは、NTP ドキュメント のWeb サイトから入手できます。

公開鍵と非公開鍵は、NIS データベース内に格納されます。NIS では、これらの鍵を `publickey` マップ内に格納します。このファイルには、すべての潜在的なユーザーの公開鍵と非公開鍵が含まれています。

システム管理者は、NIS マップの設定と、各ユーザーの公開鍵と非公開鍵の生成に責任を負っています。非公開鍵は、ユーザーのパスワードで暗号化されて格納されます。これにより、その非公開鍵はそのユーザーだけが知っていることになります。

Secure RPC による認証の管理

マウントされた NFS ファイルシステムの使用のための認証を求めることにより、ネットワークのセキュリティーが強化されます。

次のタスクマップは、NIS および NFS 用の Secure RPC を構成する手順を示しています。

表 9 タスクマップ: Secure RPC による認証の管理

タスク	説明	参照先
1. キーサーバーを起動します。	ユーザーが認証されるために鍵を作成できるようにします。	275 ページの「Secure RPC キーサーバーを再起動する方法」
2. NIS ホスト上で資格を設定します。	NIS 環境でホスト上の `root` ユーザーが認証されるようにします。	276 ページの「NIS ホストに Diffie-Hellman 鍵を設定する方法」
3. NIS ユーザーに鍵を提供します。	NIS 環境でユーザーが認証されるようにします。	277 ページの「NIS ユーザーに Diffie-Hellman 鍵を設定する方法」
4. 認証によって NFS ファイルを共有します。	NFS サーバーが認証によって共有ファイルシステムを安全に保護できるようにします。	278 ページの「Diffie-Hellman 認証で NFS ファイルを共有する方法」

▼ Secure RPC キーサーバーを再起動する方法

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **`keyserv` デーモンが実行中であることを確認します。**

```
# svcs \*keyserv\*
STATE    STIME    FMRI
disabled Dec_14   svc:/network/rpc/keyserv
```

2. **`nobody` ユーザーがデフォルトの鍵を使用できないことを確認します。**

```
$ pfedit /etc/default/keyserv
# Change the default value of ENABLE_NOBODY_KEYS to NO.
#
#ENABLE_NOBODY_KEYS=YES
ENABLE_NOBODY_KEYS=NO
```

3. **キーサーバーサービスがオンラインになっていない場合は、サービスを有効にします。**

```
# svcadm enable network/rpc/keyserv
```

▼ NIS ホストに Diffie-Hellman 鍵を設定する方法

この手順は、NIS ドメイン内のすべてのホスト上で実行します。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **デフォルトのネームサービスが NIS でない場合は、publickey マップをネームサービスに追加します。**

   a. **ネームサービスの config/default の値が nis でないことを確認します。**

```
# svccfg -s name-service/switch listprop config
config                      application
config/value_authorization  astring       solaris.smf.value.name-service.switch
config/default              astring       files
config/host                 astring       "files nis dns"
config/printer              astring       "user files nis"
```

   config/default の値が nis である場合は、ここで停止できます。

   b. **publickey のネームサービスを nis に設定します。**

```
# svccfg -s name-service/switch setprop config/publickey = astring: "nis"
# svccfg -s name-service/switch:default refresh
```

   c. **publickey 値を確認します。**

```
# svccfg -s name-service/switch listprop
config                      application
config/value_authorization  astring       solaris.smf.value.name-service.switch
config/default              astring       files
config/host                 astring       "files nis dns"
config/printer              astring       "user files nis"
config/publickey            astring       nis
```

   このシステムでは、publickey がデフォルトの files とは異なるため、その値が表示されます。

2. **newkey コマンドを使用して、新しい鍵のペアを作成します。**

```
# newkey -h hostname
```

hostname は、クライアント名です。

例 50 NIS クライアント上で root の新しい鍵を設定する

次の例では、Name Service Security 権利プロファイルを持つ管理者が、セキュアな NIS クライアントとして earth を設定します。

```
# newkey -h earth
Adding new key for unix.earth@example.com
New Password: xxxxxxxx
Retype password: xxxxxxxx
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

▼ NIS ユーザーに Diffie-Hellman 鍵を設定する方法

この手順は、NIS ドメイン内のすべてのユーザーに対して実行します。

始める前に ユーザーの新しい鍵を生成するには、NIS マスターサーバーにログインしている必要があります。Name Service Security 権利プロファイルが割り当てられている必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』の 「割り当てられている管理権利の使用」を参照してください。

1. **ユーザーの新しい鍵を作成します。**

```
# newkey -u username
```

username はユーザー名です。システムはパスワードを求めるプロンプトを出します。汎用パスワードを入力できます。非公開鍵は、汎用パスワードを使用して暗号化されて格納されます。

2. **ユーザーに、ログインして chkey -p コマンドを入力するよう指示します。**
このコマンドでは、そのユーザーは自分だけが知っているパスワードを使用して、自分の非公開鍵を暗号化し直すことができます。

注記 - chkey コマンドを使用すると、新しい鍵のペアをユーザーに作成できます。

例 51 NIS で新しいユーザー鍵を設定して暗号化する

この例では、スーパーユーザーが鍵を設定します。

```
# newkey -u jdoe
Adding new key for unix.12345@example.com
```

```
New Password: xxxxxxxx
Retype password: xxxxxxxx
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

次に、ユーザー jdoe が非公開パスワードで鍵を再暗号化します。

```
% chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@example.com
Please enter the Secure-RPC password for jdoe: xxxxxxxx
Please enter the login password for jdoe: xxxxxxxx
Sending key change request to centralexample...
```

▼ Diffie-Hellman 認証で NFS ファイルを共有する方法

この手順では、アクセスに対する認証を要求することにより、NFS サーバー上で共有されているファイルシステムを保護します。

始める前に Diffie-Hellman の公開鍵認証がネットワークで有効である必要があります。ネットワーク上で認証を有効にするには、276 ページの「NIS ホストに Diffie-Hellman 鍵を設定する方法」を完了します。

このタスクを実行するには、System Management 権利プロファイルが割り当てられている管理者になる必要があります。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **NFS サーバーで、 Diffie-Hellman 認証でファイルシステムを共有します。**

   ```
   # share -F nfs -o sec=dh /filesystem
   ```

   filesystem は、共有されているファイルシステムです。

   -o sec=dh オプションは、ファイルシステムにアクセスするために AUTH_DH 認証が必要になるということです。

2. **NFS クライアントで、Diffie-Hellman 認証でファイルシステムをマウントします。**

   ```
   # mount -F nfs -o sec=dh server:filesystem mount-point
   ```

server	*filesystem* を共有しているシステムの名前です
filesystem	/opt など、共有されているファイルシステムの名前です
mount-point	/opt など、マウントポイントの名前です

   -o sec=dh オプションにより、AUTH_DH 認証を使ってファイルシステムがマウントされます。

♦ ♦ ♦ 付 録 A

Kerberos 用の DTrace プローブ

この付録では、DTrace プローブおよび引数構造体について説明します。その使用法の例については、210 ページの「Kerberos サービスでの DTrace の使用」を参照してください。

Kerberos での DTrace プローブ

プローブとは、DTrace が、一連のアクションを実行するためのリクエストをバインドできるプログラムの場所またはアクティビティーのことです。プローブは、プロバイダによって定義および実装されます。プロバイダは、プローブによるデータのトレースを可能にする、カーネルでロード可能なモジュールです。

これらのプローブは、ユーザー静的定義トレース (USDT) に使用されます。USDT プローブは、ユーザーランド内の Kerberos プロトコルを検査するように設計されています。静的定義トレースのためのカーネルプローブは提供されていません。

適切な DTrace プローブでユーザーが必要とする情報 (スタックトレース、タイムスタンプ、関数の引数など) を記録するスクリプトを作成します。プローブが起動されると、DTrace はそのプローブからデータを収集し、それをユーザーに報告します。プローブのアクションを指定しない場合、DTrace は、プローブが起動した各時間と動作している CPU を記録します。

Kerberos DTrace プローブは、RFC4120: The Kerberos Network Authentication Service (V5) (`http://www.ietf.org/rfc/rfc4120.txt`) で説明されている Kerberos メッセージタイプのあとにモデル化されます。プローブは、`libkrb5/mech_krb5` のコンシューマ (`libgss` 経由で `mech_krb5` を使用するアプリケーションを含む) が使用できます。プローブは、メッセージの作成と消費、および送信と受信の間で分けられます。`libgss` の詳細は、`libgss(3LIB)` のマニュアルページを参照してください。

プローブを使用するには、`kerberos` プロバイダ、プローブの名前 (`krb_message-recv` など)、および引数を指定します。例については、210 ページの「Kerberos サービスでの DTrace の使用」を参照してください。

Kerberos DTrace プローブの定義

KRB_AP_REP 用のプローブ:

```
kerberos$pid:::krb_ap_rep-make
kerberos$pid:::krb_ap_rep-read

args[0]        krbinfo_t *
args[1]        kaprepinfo_t *
```

KRB_AP_REQ 用のプローブ:

```
kerberos$pid:::krb_ap_req-make
kerberos$pid:::krb_ap_req-read

args[0]        krbinfo_t *
args[1]        kapreqinfo_t *
args[2]        kticketinfo_t *
args[3]        kauthenticatorinfo_t *
```

KRB_KDC_REP 用のプローブ:

```
kerberos$pid:::krb_kdc_rep-make
kerberos$pid:::krb_kdc_rep-read

args[0]        krbinfo_t *
args[1]        kdcrepinfo_t *
args[2]        kticketinfo_t *
```

KRB_KDC_REQ 用のプローブ:

```
kerberos$pid:::krb_kdc_req-make
kerberos$pid:::krb_kdc_req-read

args[0]        krbinfo_t *
args[1]        kdcreqinfo_t *
```

KRB_CRED 用のプローブ:

```
kerberos$pid:::krb_cred-make
kerberos$pid:::krb_cred-read

args[0]        krbinfo_t *
args[1]        kcredinfo_t *
```

KRB_ERROR 用のプローブ:

```
kerberos$pid:::krb_error-make
kerberos$pid:::krb_error-read

args[0]        krbinfo_t *
args[1]        kerrorinfo_t *
```

KRB_PRIV 用のプローブ:

```
kerberos$pid:::krb_priv-make
kerberos$pid:::krb_priv-read

args[0]        krbinfo_t *
args[1]        kprivinfo_t *
```

KRB_SAFE 用のプローブ:

```
kerberos$pid:::krb_safe-make
kerberos$pid:::krb_safe-read

args[0]        krbinfo_t *
args[1]        ksafeinfo_t *
```

メッセージを送受信するためのプローブ

```
kerberos$pid:::krb_message-recv
kerberos$pid:::krb_message-send

args[0]        krbinfo_t *
args[1]        kconninfo_t *
```

Kerberos での DTrace 引数構造体

特定の状況では、一部の引数の値が `0` または空になることがあります。Kerberos 引数構造体は、RFC4120: The Kerberos Network Authentication Service (V5) (`http://www.ietf.org/rfc/rfc4120.txt`) と一般的に整合性があるように設計されています。

DTrace での Kerberos メッセージ情報

```
typedef struct krbinfo {
uint8_t krb_version;                /* protocol version number (5) */
string krb_message_type;            /* Message type (AS_REQ(10), ...) */
uint64_t krb_message_id;            /* message identifier */
uint32_t krb_message_length;        /* message length */
uintptr_t krb_message;              /* raw ASN.1 encoded message */
} krbinfo_t;
```

注記 - Kerberos プロトコルには、メッセージ識別子はありません。`krb_message_id` 識別子は Kerberos プロバイダに固有であり、`make/read` と `send/recv` プローブの間でメッセージをリンクするように設計されています。

DTrace での Kerberos 接続情報

```
typedef struct kconninfo {
string kconn_remote;                /* remote host address */
string kconn_local;                 /* local host address */
uint16_t kconn_localport;           /* local port */
uint16_t kconn_remoteport;          /* remote port */
string kconn_protocol;              /* protocol (ipv4, ipv6) */
string kconn_type;                  /* transport type (udp, tcp) */
} kconninfo_t;
```

DTrace での Kerberos オーセンティケータ情報

```
typedef struct kauthenticatorinfo {
string kauth_client;                /* client principal identifier */
string kauth_cksum_type;            /* type of checksum (des-cbc, ...) */
uint32_t kauth_cksum_length;        /* length of checksum */
uintptr_t kauth_cksum_value;        /* raw checksum data */
uint32_t kauth_cusec;               /* client time, microseconds */
uint32_t kauth_ctime;               /* client time in seconds */
string kauth_subkey_type;           /* sub-key type (des3-cbc-sha1, ...) */
uint32_t kauth_subkey_length;       /* sub-key length */
uintptr_t kauth_subkey_value;       /* sub-key data */
uint32_t kauth_seq_number;          /* sequence number */
string kauth_authorization_data;    /* top-level authorization types
(AD-IF-RELEVANT, ... ) */
} kauthenticatorinfo_t;

typedef struct kticketinfo_t {
string kticket_server;              /* service principal identifier */
uint32_t kticket_enc_part_kvno;     /* key version number */
string kticket_enc_part_etype;      /* enc type of encrypted ticket */
string kticket_enc_flags;           /* ticket flags (forwardable, ...) */
string kticket_enc_key_type;        /* key type (des3-cbc-sha1, ...) */
uint32_t kticket_enc_key_length;    /* key length */
uintptr_t kticket_enc_key_value;    /* key data */
string kticket_enc_client;          /* client principal identifier */
string kticket_enc_transited;       /* list of transited Kerberos realms */
string kticket_enc_transited_type;  /* encoding type */
uint32_t kticket_enc_authtime;      /* time of initial authentication */
uint32_t kticket_enc_starttime;     /* ticket start time in seconds */
uint32_t kticket_enc_endtime;       /* ticket end time in seconds */
uint32_t kticket_enc_renew_till;    /* ticket renewal time in seconds */
string kticket_enc_addresses;       /* addresses associated with ticket */
string kticket_enc_authorization_data;  /* list of top-level auth types */
} kticketinfo_t;

typedef struct kdcreqinfo {
string kdcreq_padata_types;         /* list of pre-auth types */
string kdcreq_kdc_options;          /* requested ticket flags */
string kdcreq_client;               /* client principal identifier */
string kdcreq_server;               /* server principal identifier */
uint32_t kdcreq_from;               /* requested start time in seconds */
uint32_t kdcreq_till;               /* requested end time in seconds */
uint32_t kdcreq_rtime;              /* requested renewal time in seconds */
uint32_t kdcreq_nonce;              /* nonce for replay detection */
string kdcreq_etype;                /* preferred encryption types */
string kdcreq_addresses;            /* list of requested ticket addresses */
string kdcreq_authorization_data;   /* list of top-level auth types */
uint32_t kdcreq_num_additional_tickets; /* number of additional tickets */
} kdcreqinfo_t;

typedef struct kdcrepinfo {
string kdcrep_padata_types;         /* list of pre-auth types */
string kdcrep_client;               /* client principal identifier */
uint32_t kdcrep_enc_part_kvno;      /* key version number */
string kdcrep_enc_part_etype;       /* enc type of encrypted KDC reply */
string kdcrep_enc_key_type;         /* key type (des3-cbc-sha1, ...) */
uint32_t kdcrep_enc_key_length;     /* key length */
uintptr_t kdcrep_enc_key_value;     /* key data */
string kdcrep_enc_last_req;         /* times of last request of principal */
uint32_t kdcrep_enc_nonce;          /* nonce for replay detection */
uint32_t kdcrep_enc_key_expiration; /* expiration time of client's key */
```

```
string kdcrep_enc_flags;            /* ticket flags */
uint32_t kdcrep_enc_authtime;       /* time of authentication of ticket */
uint32_t kdcrep_enc_starttime;      /* ticket start time in seconds */
uint32_t kdcrep_enc_endtime;        /* ticket end time in seconds */
uint32_t kdcrep_enc_renew_till;       /* ticket renewal time in seconds*/
string kdcrep_enc_server;           /* server principal identifier */
string kdcrep_enc_caddr;            /* zero or more client addresses */
} kdcrepinfo_t;

typedef struct kapreqinfo {
string kapreq_ap_options;           /* options (use-session-key,... ) */
uint32_t kapreq_authenticator_kvno; /* key version number */
string kapreq_authenticator_etype;  /* enc type of authenticator */
} kapreqinfo_t;

typedef struct kaprepinfo {
uint32_t kaprep_enc_part_kvno;      /* key version number */
string kaprep_enc_part_etype;       /* enc type of encrypted AP reply */
uint32_t kaprep_enc_ctime;          /* client time in seconds */
uint32_t kaprep_enc_cusec;          /* client time, microseconds portion */
string kaprep_enc_subkey_type;      /* sub-key type */
uint32_t kaprep_enc_subkey_length;  /* sub-key length */
uintptr_t kaprep_enc_subkey_value;  /* sub-key data */
uint32_t kaprep_enc_seq_number;     /* sequence number */
} kaprepinfo_t;

typedef struct kerrorinfo {
uint32_t kerror_ctime;              /* client time in seconds */
uint32_t kerror_cusec;              /* client time, microseconds */
uint32_t kerror_stime;              /* server time in seconds */
uint32_t kerror_susec;              /* server time, microseconds */
string kerror_error_code;           /* error code (KRB_AP_ERR_SKEW, ...) */
string kerror_client;               /* client principal identifier */
string kerror_server;               /* server principal identifier */
string kerror_e_text;               /* additional error text */
string kerror_e_data;               /* additional error data */
} kerrorinfo_t;

typedef struct ksafeinfo {
uintptr_t ksafe_user_data;          /* raw application specific data */
uint32_t ksafe_timestamp;           /* time of sender in seconds */
uint32_t ksafe_usec;                /* time of sender, microseconds */
uint32_t ksafe_seq_number;          /* sequence number */
string ksafe_s_address;             /* sender's address */
string ksafe_r_address;             /* recipient's address */
string ksafe_cksum_type;            /* checksum type (des-cbc, ...) */
uint32_t ksafe_cksum_length;        /* length of checksum */
uintptr_t ksafe_cksum_value;        /* raw checksum data */
} ksafeinfo_t;

typedef struct kprivinfo {
uint32_t kpriv_enc_part_kvno;       /* key version number */
string kpriv_enc_part_etype;        /* enc type of encrypted message */
uintptr_t kpriv_enc_user_data;      /* raw application specific data */
uint32_t kpriv_enc_timestamp;       /* time of sender in seconds */
uint32_t kpriv_enc_usec;            /* time of sender, microseconds */
uint32_t kpriv_enc_seq_number;      /* sequence number */
string kpriv_enc_s_address;         /* sender's address */
string kpriv_enc_r_address;         /* recipient's address */
} kprivinfo_t;

typedef struct kcredinfo {
uint32_t kcred_enc_part_kvno;       /* key version number */
string kcred_enc_part_etype;        /* enc type of encrypted message */
```

```
uint32_t kcred_tickets;             /* number of tickets */
uint32_t kcred_enc_nonce;           /* nonce for replay detection */
uint32_t kcred_enc_timestamp;       /* time of sender in seconds */
uint32_t kcred_enc_usec;            /* time of sender, microseconds */
string kcred_enc_s_address;         /* sender's address */
string kcred_enc_r_address;         /* recipient's address */
} kcredinfo_t;
```

認証サービスの用語集

アクセス制御リスト (ACL) アクセス制御リスト (ACL) を使用すると、従来の UNIX ファイル保護よりもきめ細かな方法でファイルセキュリティーを確立できます。たとえば、特定のファイルにグループ読み取り権を設定し、そのグループ内の 1 人のメンバーだけにそのファイルへの書き込み権を与えることが可能です。

アプリケーションサーバー ネットワークアプリケーションサーバーを参照してください。

インスタンス Kerberos では、主体名の 2 番目の部分。インスタンスは、主体のプライマリを修飾します。サービス主体の場合、インスタンスは必ず指定する必要があります。`host/central.example.com` にあるように、インスタンスはホストの完全修飾ドメイン名です。ユーザー主体の場合、インスタンスは省略することができます。ただし、`jdoe` と `jdoe/admin` は、一意の主体です。プライマリ、主体名、サービス主体、ユーザー主体も参照してください。

オーセンティケータ Kerberos では、オーセンティケータは KDC にチケットを要求するときおよびサーバーにサービスを要求するときに、クライアントから渡されます。オーセンティケータには、クライアントとサーバーだけが知っているセッション鍵を使用して生成された情報が含まれます。オーセンティケータは、最新の識別として検査され、そのトランザクションが安全であることを示します。これをチケットとともに使用すると、ユーザー主体を認証できます。オーセンティケータには、ユーザーの主体名、ユーザーのホストの IP アドレス、タイムスタンプが含まれます。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できません。通常、サービスへのアクセスが要求されたときに使用されます。オーセンティケータは、そのクライアントとそのサーバーのセッション鍵を使用して暗号化されます。

鍵 1. 一般には、次に示す 2 種類の主要鍵のどちらか一方です。

- 対称鍵 – 復号化鍵とまったく同じ暗号化鍵。対称鍵はファイルの暗号化に使用されます。
- 非対称鍵または公開鍵 – Diffie-Hellman や RSA などの公開鍵アルゴリズムで使用される鍵。公開鍵には、1 人のユーザーしか知らない非公開鍵、サーバーまたは一般リソースによって使用される公開鍵、およびこれらの 2 つを組み合わせた公開鍵と非公開鍵のペアがあります。非公開鍵は、「秘密鍵」とも呼ばれます。公開鍵は、「共有鍵」や「共通鍵」とも呼ばれます。

2. キータブファイルのエントリ (主体名)。キータブファイルも参照してください。

3. Kerberos では暗号化鍵であり、次の 3 種類があります。

- 「非公開鍵」 – 主体と KDC によって共有される暗号化鍵。システムの外部に配布されます。非公開鍵も参照してください。
- 「サービス鍵」– 非公開鍵と同じ目的で使用されますが、この鍵はサーバーとサービスによって使用されます。サービス鍵も参照してください。
- 「セッション鍵」 – 一時的な暗号化鍵。2 つの主体の間で使用され、その有効期限は 1 つのログインセッションの期間に制限されます。セッション鍵も参照してください。

関係 Kerberos では、`kdc.conf` または `krb5.conf` ファイルに定義される構成変数または関係の 1 つ。

キーストア キーストアは、アプリケーションによる取得のために、パスワード、パスフレーズ、証明書、およびその他の認証オブジェクトを保持します。キーストアはテクノロジ固有にすることも、複数のアプリケーションで使用される場所にすることもできます。

キータブファイル Kerberos では、1 つまたは複数の鍵 (主体) が含まれるキーテーブル。ホストまたはサービスとキータブファイルとの関係は、ユーザーとパスワードの関係と似ています。

機密性 プライバシを参照してください。

クライアント 狭義では、`rlogin` を使用するアプリケーションなど、ユーザーの代わりにネットワークサービスを使用するプロセスを指します。サーバー自身が他のサーバーやサービスのクライアントになる場合もあります。

広義では、a) Kerberos 資格を受け取り、b) サーバーから提供されたサービスを利用するホストを指します。

広義では、サービスを使用する Kerberos 主体を指します。

クライアント主体 (RPCSEC_GSS API) Kerberos では、RPCSEC_GSS でセキュリティー保護されたネットワークサービスを使用するクライアント (ユーザーまたはアプリケーション)。クライアント主体名は、`rpc_gss_principal_t` 構造体の形式で格納されます。

クロックスキュー Kerberos 認証システムに参加しているすべてのホスト上の内部システムクロックに許容できる最大時間。参加しているホスト間でクロックスキューを超過すると、要求が拒否されます。クロックスキューは、`krb5.conf` ファイルに指定できます。

公開鍵の暗号化 暗号化スキームの 1 つ。各ユーザーが 1 つの公開鍵と 1 つの非公開鍵を所有します。公開鍵の暗号化では、送信者は受信者の公開鍵を使用してメッセージを暗号化し、受信者は非公開鍵を使用してそれを復号化します。

更新可能チケット 有効期限の長いチケットではセキュリティーが低下するリスクがあるため、Kerberos チケットを `renewable` として指定できます。更新可能チケットには 2 つの有効期限

があります。 a) チケットの現在のインスタンスの有効期限と、b) 任意のチケットの最長有効期限です。クライアントがチケットの使用を継続するときは、最初の有効期限が切れる前にチケットの有効期限を更新します。たとえば、すべてのチケットの最長有効期限が 10 時間のときに、あるチケットが 1 時間だけ有効だとします。このチケットを保持するクライアントが 1 時間を超えて使用する場合は、チケットの有効期限を更新する必要があります。チケットが最長有効期限に達すると、チケットの有効期限が自動的に切れ、それ以上更新できなくなります。

コンシューマ Oracle Solaris の暗号化フレームワーク機能では、コンシューマは暗号化プロバイダが提供する暗号化サービスのユーザー。コンシューマになりえるものとして、アプリケーション、エンドユーザー、カーネル処理などが挙げられます。Kerberos はコンシューマの例であり暗号化フレームワークはプロバイダです。

サーバー Kerberos では、ネットワーククライアントにリソースを提供する主体。 たとえば、システム `central.example.com` に `ssh` で接続する場合、そのシステムが `ssh` サービスを提供するサーバーになります。サービス主体も参照してください。

サーバー主体 (RPCSEC_GSS API) Kerberos では、サービスを提供する主体。サーバー主体は、*service@host* という書式で ASCII 文字列として格納されます。クライアント主体も参照してください。

サービス 1. ネットワーククライアントに提供されるリソース。多くの場合、複数のサーバーから提供されます。たとえば、システム `central.example.com` に `ssh` で接続する場合、そのシステムが `ssh` サービスを提供するサーバーになります。

2. 認証以外の保護レベルを提供するセキュリティーサービス (整合性またはプライバシ)。整合性とプライバシも参照してください。

サービス鍵 Kerberos サービス主体と KDC によって共有される暗号化鍵。システムの外部に配布されます。鍵も参照してください。

サービス主体 Kerberos では、1 つまたは複数のサービスに認証を提供する主体。サービス主体では、プライマリ名はサービス名 (`ftp` など) で、インスタンスはサービスを提供するシステムの完全指定ホスト名になります。ホスト主体、ユーザー主体も参照してください。

資格 Kerberos では、チケットと照合セッション鍵を含む情報パッケージ。Kerberos 主体の識別情報を認証するときに使用します。チケットとセッション鍵も参照してください。

資格キャッシュ Kerberos では、KDC から受信した資格を含むストレージ領域。通常はファイルです。

主体 1. Kerberos では、ネットワーク通信に参加し、一意の名前を持つ「クライアントまたはユーザー」あるいは「サーバーまたはサービス」のインスタンス。Kerberos トランザクションでは、主体 (サービス主体とユーザー主体) 間、または主体と KDC の間で対話が行われます。つまり、主体とは、Kerberos がチケットを割り当てることができ

る一意のエンティティーのことです。主体名、サービス主体、ユーザー主体も参照してください。

2.(RPCSEC_GSS API) クライアント主体、サーバー主体を参照してください。

主体名 1. Kerberos では、*primary/instance@REALM* という書式の主体の名前。インスタンス、プライマリ、レルムも参照してください。

2.(RPCSEC_GSS API) クライアント主体、サーバー主体を参照してください。

承認 1. Kerberos では、主体がサービスを使用できるかどうか、主体がアクセスできるオブジェクト、各オブジェクトに許可するアクセスの種類を決定するプロセス。

2. ユーザー権利の管理で、役割またはユーザーに割り当てる (権利プロファイルに埋め込む) ことができる一連の操作 (そうしない場合、セキュリティーポリシーによって拒否される) を実行するための権利。承認はカーネルではなく、ユーザーアプリケーションレベルで適用されます。

初期チケット Kerberos では、直接発行されるチケット (既存のチケット認可チケットは使用されない)。パスワードを変更するアプリケーションなどの一部のサービスでは、クライアントが非公開鍵を知っていることを確認するために、「初期」と指定されたチケットを要求することができます。初期チケットは、長期間存在する可能性のあるチケット認可チケットに依存することなく、最近クライアントが認証されたことを示すため、このような保証が重要となります。

スレーブ KDC マスター KDC のコピー。マスター KDC のほとんどの機能を実行できます。各レルムには通常、いくつかのスレーブ KDC と 1 つのマスター KDC を配置します。複数のマスター KDC が含まれるレルムは、マルチマスター構成と呼ばれます。KDC、マスター KDC も参照してください。

整合性 ユーザー認証に加えて、暗号チェックサムを使用して、転送されたデータの有効性を提供するセキュリティーサービス。認証、プライバシも参照してください。

セキュリティーサービス サービスを参照してください。

セキュリティーフレーバ フレーバを参照してください。

セキュリティーポリシー ポリシーを参照してください。

セキュリティーメカニズム 暗号化アルゴリズムの特定の実装 (`aes-256` や `aes-512` など)。

セッション鍵 Kerberos では、認証サービスまたはチケット認可サービスによって生成される鍵。セッション鍵は、クライアントとサービス間のトランザクションのセキュリティーを保護するために生成されます。セッション鍵の有効期限は、単一のログインセッションに制限されます。鍵も参照してください。

遅延チケット Kerberos では、遅延チケットは作成されても指定された時点まで有効になりません。このようなチケットは、夜遅く実行するバッチジョブなどのために効果的です。そのチケットは盗まれても、バッチジョブが実行されるまで使用できないためです。遅延チケットは、無効チケットとして発行され、a) 開始時間を過ぎて、b) クライアントが KDC による検査を要求したときに有効になります。遅延チケットは通常、チケット認可チケットの有効期限まで有効です。ただし、その遅延チケットが「更新可能」と指定されている場合、その有効期限は通常、チケット認可チケットの有効期限に設定されます。無効チケット、更新可能チケットも参照してください。

チケット Kerberos では、ユーザーの識別情報をサーバーやサービスにセキュアに渡すために使用される情報パケット。チケットは、単一クライアントと特定サーバー上の特定サービスだけに有効です。チケットには、サービスの主体名、ユーザーの主体名、ユーザーのホストの IP アドレス、タイムスタンプ、チケットの有効期限を定義する値などが含まれます。チケットは、クライアントとサービスによって使用されるランダムセッション鍵を使用して作成されます。チケットは、作成されてから有効期限まで再使用できます。チケットは、最新のオーセンティケータとともに提示されなければ、クライアントの認証に使用することができません。オーセンティケータ、資格、サービス、セッション鍵も参照してください。

チケットファイル 資格キャッシュを参照してください。

転送可能チケット Kerberos では、チケットの 1 つ。クライアントがリモートホスト上のチケットを要求するときに使用できます。このチケットを使用すれば、リモートホスト上で完全な認証プロセスを実行する必要がありません。たとえば、ユーザー `david` がユーザー `jennifer` のシステム上にいる間に転送可能チケットを取得した場合、`david` は、新しいチケットを取得しなくても (それにより自分自身を再認証しなくても) 自分のシステムにログインできます。プロキシ可能チケットも参照してください。

特権 一般に、コンピュータシステム上で通常のユーザーの能力を超える操作を実行する能力または機能。特権ユーザーまたは特権アプリケーションは、追加の権利が付与されているユーザーまたはアプリケーションです。

特権付きアプリケーション システム制御をオーバーライドできるアプリケーション。このようなアプリケーションは、セキュリティー属性 (特定の UID、GID、承認、特権など) をチェックします。

特権ユーザー コンピュータシステム上で通常ユーザーの権利を超えた権利が割り当てられているユーザー。

認証 Kerberos では、特定の主体の識別情報を検証するプロセス。

ネットワークアプリケー ネットワークアプリケーションを提供するサーバー (`ftp` など)。Kerberos レルムは、複数のネットワークアプリケーションサーバーで構成されます。

ションサーバー

パスフレーズ 非公開鍵がパスフレーズユーザーによって作成されたことを検証するために使用されるフレーズ。望ましいパスフレーズは、10 - 30 文字の長さで英数字が混在しており、単純な文や名前を避けたものです。通信の暗号化と復号化を行う非公開鍵の使用を認証するため、パスフレーズの入力を求めるメッセージが表示されます。

パスワードポリシー パスワードの生成に使用できる暗号化アルゴリズム。パスワードをどれぐらいの頻度で変更すべきか、パスワードの試行を何回まで認めるかといったセキュリティー上の考慮事項など、パスワードに関連した一般的な事柄を指すこともあります。セキュリティーポリシーにはパスワードが必要です。パスワードポリシーでは、パスワードの強度、変更の頻度、および許可されるアルファベット、数字、およびキーボード修飾キーが指示される可能性があります。

非公開鍵 Kerberos の各ユーザー (主体) に与えられ、主体のユーザーと KDC だけが知っている鍵。ユーザー主体の場合、鍵はユーザーのパスワードに基づいています。Kerberos サービスは非公開鍵システムです。鍵も参照してください。

秘密鍵 非公開鍵を参照してください。

プライバシ セキュリティーサービスの 1 つ。送信データを送信前に暗号化します。プライバシには、データの整合性とユーザー認証も含まれます。認証、整合性、サービスも参照してください。

プライマリ Kerberos では、主体名の最初の部分。インスタンス、主体名、レルムも参照してください。

フレーバ 従来は、「セキュリティーフレーバ」と「認証フレーバ」は、認証のタイプ (AUTH_UNIX、AUTH_DES、AUTH_KERB) を指すフレーバとして、同じ意味を持っていました。RPCSEC_GSS もセキュリティーフレーバですが、これは認証に加えて、整合性とプライバシのサービスも提供します。

プロキシ可能チケット Kerberos では、クライアントに対する操作を実行する際に、クライアントの代わりにサービスで使用できるチケット。このことを、サービスがクライアントのプロキシとして動作するといいます。サービスは、チケットを使用して、クライアントの識別情報を所有できます。このサービスは、プロキシ可能チケットを使用して、ほかのサービスへのサービスチケットを取得できますが、チケット認可チケットは取得できません。転送可能チケットと異なり、プロキシ可能チケットは単一の操作に対してだけ有効です。転送可能チケットも参照してください。

ホスト ネットワークを通じてアクセス可能なシステム。

ホスト主体 Kerberos では、サービス主体のインスタンスの 1 つ (プライマリ名は `host`)。さまざまなネットワークサービス (`ftp` や NFS など) を提供するために設定します。`host/central.example.com@EXAMPLE.COM` はホスト主体の例です。サーバー主体も参照してください。

ポリシー 一般には、意思やアクションに影響を与えたり、これらを決定したりする計画や手続き。コンピュータシステムでは、多くの場合セキュリティーポリシーを指します。実際のサイトのセキュリティーポリシーは、処理される情報の重要度や未承認アクセスから情報を保護する手段を定義する規則セットです。

Kerberos ポリシーおよびパスワードポリシーも参照してください。

マスター KDC 各レルムのメイン KDC。Kerberos 管理サーバー `kadmind` と、認証とチケット認可デーモン `krb5kdc` で構成されます。レルムごとに、1 つ以上のマスター KDC を割り当てる必要があります。また、クライアントに認証サービスを提供する複製 (スレーブ) KDC を任意の数だけ割り当てることができます。マルチマスター構成では、レルムに少なくとも 2 つのマスター KDC が含まれています。

無効チケット Kerberos では、まだ使用可能になっていない遅延チケット。無効チケットは、有効になるまでアプリケーションサーバーから拒否されます。これを有効にするには、開始時期が過ぎたあと、TGS 要求で `VALIDATE` フラグをオンにしてクライアントがこのチケットを KDC に提示する必要があります。遅延チケットも参照してください。

ユーザー主体 Kerberos では、特定のユーザーに属する主体。ユーザー主体のプライマリ名はユーザー名であり、その省略可能なインスタンスは対応する資格の使用目的を説明するために使われる名前です (`jdoe`、`jdoe/admin` など)。「ユーザーインスタンス」とも呼ばれます。サービス主体も参照してください。

レルム 1. 1 つの Kerberos データベースといくつかの鍵配布センター (KDC) を配置した論理ネットワーク。

2. 主体名の 3 番目の部分。主体名が `jdoe/admin@CORP.EXAMPLE.COM` の場合、レルムは `CORP.EXAMPLE.COM` です。主体名も参照してください。

admin 主体 Kerberos では、*username*`/admin` という形式 (`jdoe/admin` など) の名前を持つユーザー主体。通常のユーザー主体より多くの特権 (ポリシーの変更など) を持つことができます。主体名とユーザー主体も参照してください。

AES Advanced Encryption Standard。対称ブロックのデータ暗号技術。2000 年の 10 月、米国政府は暗号化標準としてこのアルゴリズムの Rijndael 方式を採用しました。Kerberos 暗号化のデフォルトは `aes256-cts-hmac-sha1-96` です。

Diffie-Hellman プロトコル 公開鍵暗号化としても知られています。1976 年に Diffie 氏と Hellman 氏が開発した非対称暗号鍵協定プロトコルです。このプロトコルを使用すると、セキュアでない伝達手段で、事前の秘密情報がなくても 2 人のユーザーが秘密鍵を交換できます。Diffie-Hellman は Kerberos で使用されます。

FQDN 完全指定形式のドメイン名。`central.example.com` など (単なる `central` は FQDN ではない)。

GSS-API Generic Security Service Application Programming Interface の略。さまざまなモジュールセキュリティーサービス (Kerberos サービスなど) をサポートするネットワーク層。

GSS-API は、セキュリティー認証、整合性、およびプライバシサービスを提供します。認証、整合性、プライバシも参照してください。

KDC 鍵配布センター (Key Distribution Center)。次の 3 つの Kerberos V5 コンポーネントで構成されるシステム。

- 主体と鍵データベース
- 認証サービス
- チケット認可サービス

レルムごとに、少なくとも 1 つのマスター KDC と、1 つ以上のスレーブ KDC を配置する必要があります。

Kerberos 認証サービス、Kerberos サービスが使用するプロトコル、または Kerberos サービスの実装に使用されるコード。

Oracle Solaris の Kerberos は、Kerberos V5 認証にほぼ準拠して実装されています。

「Kerberos」と「Kerberos V5」は技術的には異なりますが、Kerberos のドキュメントでは多くの場合、同じ意味で使用されます。

Kerberos (または Cerberus) は、ギリシャ神話において、ハデスの門を警護する 3 つの頭を持つどう猛な番犬のことです。

Kerberos ポリシー Kerberos サービスでのパスワードの使用方法を管理する一連の規則。ポリシーは、主体のアクセスやチケットのパラメータ (有効期限など) を制限できます。

kvno Kerberos の鍵バージョン番号。特定の鍵に対して、生成順に付けられたシーケンス番号。もっとも大きい kvno が、最新の鍵を示します。

LDAP バインディング LDAP バインディングを使用すると、LDAP サーバー上の LDAP ディレクトリとのデータ交換が可能となるため、LDAP プロトコルを使用して LDAP ディレクトリを検索、比較、および変更するプロセスが有効になります。

NTP Network Time Protocol (NTP)。デラウェア大学で開発されたソフトウェア。ネットワーク環境で、正確な時間またはネットワーククロックの同期化を管理します。NTP を使用して、Kerberos 環境のクロックスキューを管理できます。クロックスキューも参照してください。

PAM プラグイン可能認証モジュール (Pluggable Authentication Module)。複数の認証メカニズムを使用できるフレームワーク。認証メカニズムを使用するサービスはコンパイルし直す必要がありません。PAM は、ログイン時に Kerberos セッションを初期化できます。

PTP Precision Time Protocol (PTP)。正確な時間またはネットワーククロック同期 (あるいはその両方) をネットワーク環境で管理できるソフトウェア。PTP を使用すると、Kerberos 環境のクロックスキューを管理できます。クロックスキューも参照してください。

stash ファイル Kerberos では、stash ファイルに KDC のマスター鍵を暗号化したコピーが含まれます。サーバーがリブートされると、このマスター鍵を使用して KDC が自動的に認証されてから、`kadmind` プロセスと `krb5kdc` プロセスがブートされます。stash ファイルにはマスター鍵が入っているため、このファイルやこのファイルのバックアップは安全な場所に保管する必要があります。暗号が破られると、この鍵を使用して KDC データベースのアクセスや変更が可能になります。

TGS チケット認可サービス。KDC のコンポーネントの 1 つ。チケットを発行します。

TGT チケット認可チケット。KDC によって発行されるチケット。クライアントは、このチケットを使用して、ほかのサービスのチケットを要求することができます。

索引

数字・記号

~/.k5login ファイル
  説明, 183
2FA 参照 ツーファクタ認証 (2FA)
2 番目のマスター KDC
  構成, 90
16 進数の秘密鍵の表示, 268

あ

アカウントのロックアウト
  Kerberos での作成およびロック解除, 169
アクセス
  KDC サーバーへの制限, 161
  Secure RPC 認証, 273
  サーバーへの取得
    Kerberos を使用した, 58
  スマートカード認証, 221
  スマートカードのエントリポイント, 222
  ツーファクタ認証 (2FA), 221, 263
  特定のサービスへの取得, 61
  ワンタイムパスワード (OTP), 263
アクセス制御リスト 参照 ACL
アプリケーションサーバー
  構成, 130
アプリケーションサーバーの構成, 130
暗号化
  DES アルゴリズム, 274
  NIS ユーザーの非公開鍵, 277
  Secure NFS, 274
  アルゴリズム
    Kerberos と, 66
  スマートカードを使用した電子メール, 250
  タイプ
    FIPS 140-2 モードの Kerberos, 77
    Kerberos と, 56, 66
  プライバシサービス, 45
  ホームディレクトリ, 26
  モード
    Kerberos と, 66
インスタンス
  主体名, 52
インストール
  Kerberos
    自動 (AI), 69
  スマートカードパッケージ, 232
インポート
  ルート CA 証明書, 248
エラーメッセージ
  Kerberos, 195
エントリポイント
  スマートカードログイン, 222
オーセンティケータ
  Kerberos, 60, 188

か

階層レルム
  Kerberos, 53, 64
  構成, 140
回避
  表示可能なログインエラーメッセージ, 27
鍵
  Kerberos での定義, 187
  NIS ユーザーに DH 鍵を作成する, 277
  サービス鍵, 170
  セッション鍵
    Kerberos 認証と, 58
鍵配布センター 参照 KDC
鍵リスト 参照 主体

管理
Kerberos
keytab, 170
主体, 164
ポリシー, 168
Kerberos によるパスワード, 180
Secure RPC のタスクマップ, 275
キーサーバー
起動, 275
起動
KDC デーモン, 90
Secure RPC キーサーバー, 275
キャッシュ
資格, 58
業界標準
スマートカード, 228
強制
ログイン時の OTP, 269
共通アクセスカード (CAC) 参照 スマートカード
共通鍵
DH 認証と, 274
クライアント
Kerberos での定義, 187
Kerberos の構成, 111
クライアント名
Kerberos での計画, 65
クロックスキュー
Kerberos 計画と, 66
Kerberos と, 142
クロックの同期
Kerberos クライアント, 113, 119
Kerberos クライアントと, 113, 119
Kerberos 計画と, 66
Kerberos スレーブ KDC と, 81, 89
概要, 142
スレーブ KDC, 81, 89
クロックの同期化
Kerberos スレーブ KDC と, 78, 91, 97
マスター KDC, 78, 91, 97
計画
Kerberos
クライアント名とサービス主体名, 65
クロック同期, 66
構成の決定, 63
スレーブ KDC, 67
データベースの伝播, 68
ポート, 67
レルム, 63
レルムの階層, 64
レルムの数, 64
レルム名, 64
PAM, 21
計算
DH 鍵, 277
権利プロファイル
OTP Auth Manage All Users, 264, 266, 269
User Management, 269
ソフトウェアインストール, 266
ユーザー管理, 264
ユーザーごとの PAM ポリシー, 22, 22
公開鍵
DH 認証と, 274
更新可能チケット
定義, 190
構成
CACKey スマートカード, 240
Coolkey スマートカード, 240
Kerberos
2 番目のマスター KDC サーバー, 90
DSEE を使用したマスター KDC サーバー, 104
DSEE を使用したマルチマスター KDC サーバー, 104
NFS サーバー, 133
OpenLDAP を使用したマスター KDC サーバー, 94
OUD を使用したマスター KDC サーバー, 99
概要, 73
クライアント, 111
スレーブ KDC サーバー, 80, 86
タスクマップ, 73
マスター KDC サーバー, 78, 79, 81
レルム間認証, 140
NIS での DH 鍵, 276
NIS ユーザーの DH 鍵, 277
OTP 属性, 264
OTP のユーザー, 264
PAM, 23
暗号化された電子メール, 250
署名された電子メール, 250

スマートカード, 221, 230
スマートカード証明書用の openssl, 236
スマートカードの証明書, 236
スマートカード用 libccid, 233
スマートカード用の pam_pkcs11, 239
スマートカード用の Secure Shell, 239, 246
スマートカード用の Secure Shell クライアント, 246
スマートカード用の認証局 (CA), 236
スマートカード用のリモート X11 デスクトップ, 234
スマートカード用のローカルデスクトップ, 234
認証された Web サイトアクセス, 249
ワンタイムパスワード (OTP), 263, 266
構成の決定
Kerberos
KDC サーバー, 68
暗号化タイプ, 66
クライアント, 69
クライアント名とサービス主体名, 65
クロック同期, 66
スレーブ KDC, 67
データベースの伝播, 68
ポート, 67
ホスト名のレルムへのマッピング, 65
レルム, 63
レルムの階層, 64
レルムの数, 64
レルム名, 64
PAM, 21
スマートカード, 231
ワンタイムパスワード (OTP), 266, 267
構成ファイル
PAM
pam.d, 24
構文, 34
変更, 25, 31
スマートカード
Info.plist, 233
リモート X11 デスクトップ
/etc/gdm/custom.conf, 235
個人識別検証 (PIV) 参照 スマートカード
コマンド
Kerberos, 184

さ

サーバー
Kerberos での定義, 187
Kerberos を使用したアクセスの取得, 58
資格の取得, 60
レルムと, 53
サービス
Kerberos での定義, 187
特定のサービスへのアクセスの取得, 61
ホスト上での無効化, 173
サービス鍵
Kerberos での定義, 187
keytab ファイルと, 170
サービス主体
keytab ファイルからの削除, 172
keytab ファイルへの追加, 170, 171
説明, 52
名前の計画, 65
削除
keytab ファイルからのサービス主体, 172
ktremove コマンドによる主体, 172
主体 (Kerberos), 166
ホストのサービス, 174
作成
kinit によるチケット, 178
stash ファイル, 89, 91, 91
新しい主体 (Kerberos), 165
新しいポリシー (Kerberos), 165
資格テーブル, 134
資格
TGS に対する取得, 59
キャッシュ, 58
サーバーに対する取得, 60
説明, 188
またはチケット, 47
マッピング, 71
資格テーブル
1 つのエントリの追加, 135
辞書
Kerberos パスワードのための使用, 161
実装
ツーファクタ認証 (2FA), 225
自動インストール (AI)
Kerberos クライアント, 69
自動的な構成

Kerberos
マスター KDC サーバー, 78
暗号化されたホームディレクトリ, 26
主体
clntconfig の作成, 84
host の作成, 84
Kerberos, 52
keytab からのサービス主体の削除, 172
keytab ファイルからの削除, 172
keytab へのサービス主体の追加, 170, 171
管理, 163, 164
サービス主体, 52
削除, 166
作成, 165
作成の自動化, 164
主体名, 52
変更, 166
ユーザー ID の比較, 134
ユーザー主体, 52
リストの表示, 165
主体の作成の自動化, 164
手動での構成
Kerberos
2 番目のマスター KDC サーバー, 90
DSEE を使用したマスター KDC サーバー, 104
DSEE を使用したマルチマスター KDC サーバー, 104
OpenLDAP を使用したマスター KDC サーバー, 94
OUD を使用したマスター KDC サーバー, 99
スレーブ KDC サーバー, 86
マスター KDC サーバー, 81
取得
kinit によるチケット, 178
TGS に対する資格, 59, 59
サーバーに対する資格, 60, 60
スマートカードの公開鍵情報, 240
特定のサービスへのアクセス, 61, 61
ジュネーブ条約付属軍カード, 222
ジュネーブ条約身分証明カード, 222
使用
OTP カウンタモード, 271
OTP の 16 進数の秘密鍵, 268
暗号化および署名された電子メール, 251
オーセンティケータアプリ, 266
スマートカード, 247, 251
セキュアな Web サイト, 250
ワンタイムパスワード (OTP), 264
承認
Kerberos と, 45
証明書
DoD 階層, 249
Firefox、使用, 249
Thunderbird、使用, 250
スマートカードでの使用のためのダウンロード, 249
スマートカード用のインポート, 248
スマートカード用の構成, 236
初期チケット
定義, 189
署名
スマートカードを使用した電子メール, 250
署名された電子メール
構成, 250
シリアルポート
スマートカードのエントリポイント, 222
シングルサインオンシステム, 182
Kerberos と, 45
スマートカード
OCSP レスポンダソフトウェア, 226
OpenSSH の使用, 231
PAM のサポート, 226
PKI 認証, 230
Secure Shell クライアントの構成, 246
Secure Shell の構成, 239, 246
Web サイトへの認証, 248
アーキテクチャー, 228
暗号化プロバイダ, 227
主な構成ステップ, 230, 231
業界標準, 228
共通アクセスカード (CAC), 221
公開鍵情報の取得, 240
構成, 221
サポートされるタイプ, 222
使用, 251
使用の有効化, 247
説明, 221, 221
ソフトウェアモジュール、リスト, 226
電子メールの暗号化および署名, 250

ドライバ, 226, 228
ドライバの接続, 228
取り出し, 247, 251
ハードウェア, 228
米国政府 CaC, 222
ライブラリのサポート, 226
リーダー, 228
リーダーの電圧レベル, 233
ルート CA 証明書のインポート, 248
ログインエントリポイント, 222
ログインの構成, 230
ログインの図, 222
スマートカードの暗号化プロバイダ, 227
スマートカードのドライバ, 226, 228
スマートカードリーダー
システムに直接接続, 222
ドライバ, 228
スレーブ KDC
計画, 67
構成, 86
対話的な構成, 80
定義, 187
マスター KDC と, 53
マスター KDC とのスワップ, 144
またはマスター, 75
制御フラグ
PAM, 37
制限
コンソールログイン, 24
整合性
Kerberos と, 45
セキュリティーサービス, 55
セキュリティーサービス
Kerberos と, 55
セキュリティーモード
複数のモードでの環境の設定, 136
セッション鍵
Kerberos での定義, 187
Kerberos 認証と, 58
設定
管理者による OTP 用の秘密鍵, 268
ユーザーによる OTP の秘密鍵, 266
ソフトウェアインストールに関連する権利プロファイル, 266

た

対話的な構成
Kerberos
スレーブ KDC サーバー, 80
マスター KDC サーバー, 79
ダウンロード
スマートカード証明書, 249
タスクマップ
Kerberos NFS サーバーの構成, 132
Kerberos 構成, 73
Kerberos の保守, 74
PAM, 23
Secure RPC の管理, 275
ワンタイムパスワード (OTP), 265
遅延チケット
説明, 47
定義, 189
チケット
Kerberos での定義, 188
`kinit` による作成, 178
`klist` コマンド, 178
更新可能, 190
更新可能な最長有効期限, 191
種類, 188
初期, 189
遅延, 47
遅延可能, 189
定義, 46
転送可能, 47, 189
破棄, 180
表示, 178
ファイル 参照 資格キャッシュ
プロキシ, 189
プロキシ可能, 189
または資格, 47
無効, 189
有効期限, 190
有効期限切れに関する警告, 121
チケット認可サービス 参照 TGS
チケット認可チケット 参照 TGT
チケットの種類, 188
チケットの有効期限
Kerberos, 190
チケットの有効期限切れに関する警告, 121
チケットファイル 参照 資格キャッシュ

抽出
スマートカードの公開鍵情報, 240
調査
スマートカード, 240
直接レルム, 141
追加
keytab ファイルへのサービス主体 (Kerberos), 171
PAM モジュール, 27
パッケージ
pkcs11_cackey, 232
smartcard, 232
マウントされたファイルシステムへの DH 認証, 275
ツーファクタ認証 (2FA), 221, 263
参照 スマートカード
参照 ワンタイムパスワード (OTP)
使用, 251
スマートカード, 221
スマートカードでの実装, 225
説明, 221
要求, 264
ワンタイムパスワード (OTP), 263
データ暗号化規格 (Data Encryption Standard) 参照 DES 暗号化
データベース
KDC の作成, 83
KDC の伝播, 68
KDC のバックアップと伝播, 148
Secure RPC のための cred, 275
Secure RPC のための publickey, 275
テーブル
gsscred, 72
デーモン
Kerberos のテーブル, 186
keyserv, 275
ocspd, 236, 238
pcscd, 233
デスクトップ
スマートカード用のリモート X11, 234
スマートカード用のローカル, 234
スマートカード用のローカルの構成, 234
リモート X11 の構成, 234
テスト
証明書署名要求 (CSR), 237
ルート CA, 236
デバッグレベル
スマートカード用の libccid, 233
デュアル認証 参照 ツーファクタ認証 (2FA)
電圧レベル
libccid, 234
電子メール
スマートカードを使用した署名および暗号化, 250
転送可能チケット
説明, 47
定義, 189
伝播
KDC データベース, 68
Kerberos データベース, 148
透過性
Kerberos での定義, 46
ドキュメント
スマートカードの libpki, 226
トラブルシューティング
Kerberos, 195
Kerberos でのアカウントのロックアウト, 169
PAM, 33
取り出し
スマートカード, 247, 251

な

認証
DH 認証, 274
Kerberos と, 45
Kerberos の概要, 58
NFS での使用, 273
NFS マウントしたファイル, 278, 278
PAM, 19
Secure RPC, 273
スマートカード, 221
スマートカードユーザー, 251
スマートカードリーダー, 228
スマートカードをサポートするライブラリ, 225
セキュアな Web サイトアクセス, 249
ツーファクタ, 221, 251
ネームサービス, 273
マルチファクタ, 221, 263
用語, 187

レルム間の構成, 140
ワンタイムパスワード (OTP), 263
認証局 (CA)
スマートカード用のインポート, 248
スマートカード用の構成, 236

は

ハードウェア
スマートカードのエントリポイント, 222
スマートカードリーダー, 228
ツーファクタ認証 (2FA), 223
破棄
kdestroy によるチケット, 180
パスワード
Kerberos での辞書, 161
kpasswd コマンドによる変更, 181
passwd コマンドによる変更, 180
UNIX と Kerberos, 180
管理, 180
ポリシーと, 181
バックアップ
Kerberos データベース, 148
スレーブ KDC, 67
パッケージ
smartcard, 225
solaris/library/security/pam/module/pam-pkcs11, 239
solaris/library/security/pcsc-lite/ccid, 233
solaris/library/security/pcsc/pcsclite, 232
system/security/otp, 263
ハッシュ
アルゴリズム
Kerberos と, 66
非階層レルム
Kerberos, 53
非公開鍵, 274
参照 秘密鍵
Kerberos での定義, 187
表示
list コマンドによる鍵リストバッファー, 172, 174
主体のリスト, 165
スマートカードの公開鍵情報, 240
チケット, 178
ファイル
/etc/security/pam_policy/otp, 264
DH 認証での共有, 278
DH 認証によるマウント, 278
kdc.conf, 190
Kerberos, 183
PAM 構成, 34
rsyslog.conf, 32
syslog.conf, 32
ユーザーごとの PAM ポリシー
変更, 29
ファイルシステム
NFS, 273
暗号化されたホームディレクトリ, 26
セキュリティー
認証および NFS, 273
ファイルの共有
DH 認証, 278
プライバシ
Kerberos と, 45
セキュリティーサービス, 55
プライマリ
主体名, 52
ブラウザ 参照 Web ブラウザ
プラグイン
SASL と, 218
プラグイン可能認証モジュール 参照 PAM
プロキシ可能チケット
定義, 189
プロキシチケット
定義, 189
プロバイダ
スマートカードの暗号法, 227
米国政府のスマートカード
CACKey, 227
ツーファクタ認証 (2FA) および, 222
変更
kpasswd によるパスワード, 181
passwd によるパスワード, 180
主体 (Kerberos), 166
ポート
Kerberos KDC 用, 67
保護

ツーファクタ認証の使用, 221, 263
ホスト
Kerberos サービスの無効化, 173
ホスト名
レルムへのマッピング, 65
ポリシー
管理, 163, 168
作成 (Kerberos), 165
パスワードと, 181

ま

マウント
DH 認証によるファイル, 278
マスター KDC
DSEE を使用した構成, 104
OpenLDAP を使用した構成, 94
OUD を使用した構成, 99
自動的な構成, 78
手動での構成, 81
スレーブ KDC と, 53, 75
スレーブ KDC とのスワップ, 144
対話的な構成, 79
定義, 187
マスター KDC とスレーブ KDC のスワップ, 144
マッピング
Kerberos 主体への UID, 72
レルムへのホスト名 (Kerberos), 65
マルチファクタ認証 参照 スマートカード 参照 ワンタイムパスワード (OTP)
無効化
表示可能なログインエラーメッセージ, 27
ホスト上のサービス (Kerberos), 173
無効チケット
定義, 189

や

有効化
スマートカードの使用, 247
スマートカードを使用した認証された Web サイトアクセス, 249
電子メールの暗号化および署名, 250
ユーザー
OTP での認証, 264
OTP の秘密鍵の構成, 266
暗号化されたホームディレクトリの作成, 26
スマートカード情報の表示, 240
スマートカードの構成, 246
スマートカードを使用した認証, 251
ログインエラーメッセージ表示の回避, 27
ワンタイムパスワード構成の検証, 267
ユーザーごとの PAM ポリシー
権利プロファイルでの割り当て, 22
ユーザーへの OTP の割り当て, 270
ユーザー主体
説明, 52
ユーザーの手順
chkey コマンド , 278
NIS ユーザーの非公開鍵の暗号化, 277
ユーザー ID
NFS サービス, 134
用語
Kerberos, 186
Kerberos 固有, 187
認証固有, 187

ら

ライブラリのサポート
スマートカード, 226
リモートデスクトップ
スマートカード用の構成, 235
リモートログイン
スマートカードエントリポイント, 222
ツーファクタ認証 (2FA), 223, 224
ルート CA 証明書
インポート, 248
レルム間認証
構成, 140
レルム (Kerberos)
階層, 64, 140
階層または非階層, 53
構成の決定, 63
サーバーと, 53
主体名, 52
数, 64
直接, 141
内容, 54
名前, 64

ホスト名のマッピング, 65
レルム間認証の構成, 140
ローカルログイン
スマートカードエントリポイント, 222
ツーファクタ認証 (2FA), 223, 224
ロギング
PAM エラー, 32
ログイン
OTP の使用の強制, 269
PAM エラーメッセージの無効化, 27
コンソールの制限, 24
スマートカードエントリポイント, 222
スマートカードの構成, 230
スマートカードの使用, 222

わ

ワンタイムパスワード (OTP)
16 進数の秘密鍵の表示, 268
PAM 構成ファイル, 264
概要, 263
カウンタモード, 271
構成, 263, 266
デフォルト属性, 268
秘密鍵の 16 進数表示, 269
秘密鍵の設定, 266, 268, 269
ユーザーの構成, 264
ユーザーへの送信, 269
ワンタイムパスワード (OTP) でのカウンタモード, 271
ワンタイムパスワード (OTP) の秘密鍵
16 進数表示, 268
ワンタイムパスワード (OTP) 用の秘密鍵
管理者による設定, 268

A

ACL
kadm5.acl ファイル, 166, 167
Kerberos 管理者の指定, 83
Kerberos ファイル, 147
ActivCard
スマートカードハードウェアリーダー, 228
admin_server セクション
krb5.conf ファイル, 82
AUTH_DES 認証 参照 AUTH_DH 認証
AUTH_DH 認証
および NFS, 273
auto_transition オプション
SASL と, 219
auxprop_login オプション
SASL と, 219

B

binding 制御フラグ
PAM, 37

C

CACKey
pam_pkcs11 の構成, 240
スマートカードの暗号化プロバイダ, 227
米国政府暗号化プロバイダ, 227
canon_user_plugin オプション
SASL と, 219
chkey コマンド, 277
clntconfig 主体
作成, 84
Coolkey
pam_pkcs11 の構成, 240
スマートカードの暗号化プロバイダ, 227
crammd5.so.1 プラグイン
SASL と, 218
cred データベース
DH 認証 , 274
cred テーブル
DH 認証と, 275

D

default_realm セクション
krb5.conf ファイル, 82
definitive 制御フラグ
PAM, 37
delete_entry コマンド
ktutil コマンド, 174
DES 暗号化
Secure NFS, 274
DH 認証

NIS クライアント用, 276
NIS での構成, 276
説明, 274
ファイルの共有, 278
ファイルのマウント, 278
Diffie-Hellman 認証 参照 DH 認証
digestmd5.so.1 プラグイン
SASL と, 218
DNS
Kerberos と, 65
DoD の ID カード/政府機関の身分証明書, 222
domain_realm セクション
krb5.conf ファイル, 65, 82
DSEE (LDAP)
を使用したマスター KDC の構成, 104
を使用したマルチマスター KDC の構成, 104
DTrace
Kerberos, 279
Kerberos オーセンティケータ情報の使用, 282
Kerberos 接続情報の使用, 281
Kerberos での引数構造体, 281
Kerberos でのプローブ, 279
Kerberos メッセージ情報の使用, 281

E

/etc/gdm/custom.conf ファイル, 235
/etc/krb5/kadm5.acl ファイル
説明, 183
/etc/krb5/kdc.conf ファイル
説明, 183
/etc/krb5/kpropd.acl ファイル
説明, 183
/etc/krb5/krb5.conf ファイル
説明, 183
/etc/krb5/krb5.keytab ファイル
説明, 183
/etc/krb5/warn.conf ファイル
説明, 183
/etc/pam.conf ファイル
Kerberos と, 184
PAM レガシー構成ファイル, 34
/etc/pam.d ディレクトリ
PAM 構成ファイル, 34
/etc/publickey ファイル
DH 認証と, 275
/etc/security/pam_policy
OTP 構成ファイル, 264
PAM ユーザーごとの構成ファイル, 34
/etc/syslog.conf ファイル
PAM と, 32
EXTERNAL セキュリティーメカニズムプラグイン
SASL と, 218

F

FIPS 140-2
Kerberos と, 58
Kerberos の構成, 77
暗号化タイプ, 58
Firefox 参照 Web ブラウザ
FQDN (完全修飾ドメイン名)
Kerberos, 65
ftp コマンド
Kerberos と, 185

G

gdm プログラム
スマートカード用の構成, 235
GSS-API
Kerberos と, 46
GSS 資格のマッピング, 71
gssapi.so.1 プラグイン
SASL と, 218
gsscred コマンド
説明, 185
gsscred テーブル
使用, 72
gssd デーモン
Kerberos と, 186

H

HID/Omnikey
スマートカードハードウェアリーダー, 228
host 主体
作成, 84
HOTP 参照 ワンタイムパスワード (OTP)

I

ID
  UNIX の Kerberos 主体へのマッピング, 72
ID および特権共通アクセスカード, 222
Identive
  スマートカードハードウェアリーダー, 228
ILOM ログイン
  スマートカードエントリポイント, 222
  ツーファクタ認証 (2FA), 223, 225
in.rlogind デーモン
  Kerberos と, 186
in.rshd デーモン
  Kerberos と, 186
in.telnetd デーモン
  Kerberos と, 186
include 制御フラグ
  PAM, 38
Info.plist ファイル, 233
INTERNAL プラグイン
  SASL と, 218

K

.k5.レルム ファイル
  説明, 184
.k5login ファイル
  説明, 183
kadm5.acl ファイル
  新しい主体と, 166
  エントリの形式, 167
  説明, 183
  マスター KDC のエントリ, 83, 147
kadmin.local コマンド
  主体の作成の自動化, 164
  説明, 185
kadmin.log ファイル
  説明, 184
kadmin コマンド
  host 主体の作成, 84
  Kerberos 用の CLI, 163
  keytab からの主体の削除, 172
  ktadd コマンド, 171
  ktremove コマンド, 172
  SEAM ツールと, 163
  新しい主体の作成, 165
  新しいポリシーの作成, 165
  主体の削除, 166
  主体の変更, 166
  主体のリストの表示, 165
  説明, 185
kadmind デーモン
  Kerberos と, 186
  マスター KDC と, 187
kclient コマンド
  説明, 185
kdb5_ldap_util コマンド
  説明, 185
kdb5_util コマンド
  KDC データベースの作成, 83
  stash ファイルの作成, 89, 91, 91
  説明, 186
kdc.conf ファイル
  FIPS 140-2 用の構成, 77
  説明, 183
  チケットの有効期限と, 190
kdc.log ファイル
  説明, 184
KDC
  2 番目のマスターの構成
    手動, 90
  host 主体の作成, 84
  クロックの同期
    スレーブ KDC, 81, 89
  クロックの同期化
    マスター KDC, 78, 91, 97
  計画, 67
  サーバーへのアクセスの制限, 161
  スレーブ, 67
    定義, 187
  スレーブ KDC の構成
    DSEE を使用した, 104
  スレーブからマスターへの管理ファイルのコピー, 87
  スレーブの構成
    手動, 86
    対話的, 80
  スレーブまたはマスター, 53, 75
  データベースの作成, 83
  データベースの伝播, 68
  デーモンの起動, 90

バックアップと伝播, 148
ポート, 67
マスター
定義, 187
マスターとスレーブのスワップ, 144
マスターの構成
DSEE を使用した, 104
OpenLDAP を使用した, 94
OUD を使用した, 99
自動, 78
手動, 81
対話的, 79
マルチマスターの構成
DSEE を使用した, 104
KDC サーバー
LDAP 上での構成, 93
KDC サーバーへのアクセスの制限, 161
kdcmgr コマンド
サーバーのステータス, 78, 80
スレーブの構成
対話的, 80
説明, 186
マスターの構成
自動, 78
kdestroy コマンド
Kerberos と, 185
例, 180
Kerberos
DTrace, 279
DTrace でのオーセンティケータ情報の使用, 282
DTrace での接続情報の使用, 281
DTrace でのメッセージ情報の使用, 281
DTrace 引数構造体, 281
DTrace プローブ, 279
FIPS 140-2 暗号化タイプ, 58
KDC サーバーの構成, 75
Kerberos V5 プロトコル, 46
LDAP 上での KDC サーバーの構成, 93
LDAP 上での Kerberos の構成, 93
LDAP 上での構成, 93
USDT DTrace プローブ, 279
アカウントのロックアウト, 169
暗号化タイプ
概要, 66
使用, 56
エラーメッセージ, 195
概要
認証システム, 58
認証プロセス, 46
管理, 163
クロックの同期
クライアント, 113, 119
計画, 63
構成の決定, 63
コマンド, 182, 184
コンポーネント, 54
サーバーへのアクセスの取得, 58
参照, 183
使用, 177
デーモン, 186
トラブルシューティング, 195
パスワード管理, 180
パスワード辞書, 161
パスワード辞書の使用, 161
ファイル, 183
用語, 186, 187
リモートアプリケーション, 51
レルム 参照 レルム (Kerberos)
Kerberos クライアント
計画
自動インストール (AI), 69
自動インストール (AI), 69
Kerberos コマンド, 182
Kerberos 認証
と Secure RPC, 274
keyserv デーモン, 275
keytab オプション
SASL と, 219
keytab ファイル
delete_entry コマンドによるホストのサービスの無効化, 174
ktremove コマンドによる主体の削除, 172
ktutil コマンドによる管理, 170
ktutil コマンドによる内容の表示, 172, 172
list コマンドによる鍵リストバッファーの表示, 172, 174
read_kt コマンドによる keytab バッファーへの読み込み, 172, 174
管理, 170

サービス主体の削除, 172
サービス主体の追加, 170, 171
へのマスター KDC のホスト主体の追加, 84
kinit コマンド
Kerberos と, 185
チケットの有効期限, 190
例, 178
klist コマンド
-f オプション, 178
Kerberos と, 185
例, 178
kpasswd コマンド
Kerberos と, 185
passwd コマンドと, 181
kprop コマンド
説明, 185
kpropd.acl ファイル
説明, 183
kpropd デーモン
Kerberos と, 186
kproplog コマンド
説明, 186
krb5.conf ファイル
domain_realm セクション, 65
FIPS 140-2 用の構成, 77
説明, 183
編集, 82
ポートの定義, 67
krb5.keytab ファイル
説明, 183
krb5cc_*UID* ファイル
説明, 184
krb5kdc デーモン
Kerberos と, 186
起動, 90
マスター KDC と, 187
ktadd コマンド
構文, 171
サービス主体の追加, 170, 171
ktkt_warnd デーモン
Kerberos と, 186
ktremove コマンド, 172
ktutil コマンド
delete_entry コマンド, 174
Kerberos と, 185
keytab ファイルの管理, 170
list コマンド, 172, 174
read_kt コマンド, 172, 174
主体のリストの表示, 172, 172
kvno コマンド
Kerberos と, 185

L

LDAP
KDC サーバーの構成, 93
Kerberos と, 93
Kerberos の構成, 93
PAM モジュール, 42
libccid
USB デバイス番号, 234
スマートカードのデバッグレベル, 233
電圧レベル, 234
libccid ライブラリ
スマートカードのサポート, 226
libpcsclite.so モジュール, 232
libusb ライブラリ
スマートカードのサポート, 226
list コマンド, 172, 174
log_level オプション
SASL と, 220

M

max_life 値
説明, 190
max_renewable_life 値
説明, 191
mech_list オプション
SASL と, 219

N

Network Time Protocol 参照 NTP
newkey コマンド
NIS ユーザーに鍵を作成する, 277
NFS サーバー
Kerberos のための構成, 133
NFS ファイルシステム
AUTH_DH による安全なアクセス, 278

認証, 273
NIS ネームサービス
認証, 273
nowarn オプション
エラーメッセージの無効化, 27
NTP
Kerberos クライアント, 113, 119
Kerberos 計画と, 66
スレーブ KDC と, 81, 89
マスター KDC と, 78, 91, 97

O

OCSP レスポンダ
スマートカードの構成, 236
スマートカードのサポート, 226
ocspd デーモン, 236, 238
openca-ocspd
スマートカードライブラリのサポート, 226
レスポンダの構成, 236
OpenLDAP (LDAP)
マスター KDC の構成, 94
OpenSSH およびスマートカード, 231
openssl.conf ファイル, 236
optional 制御フラグ
PAM, 38
OTP Auth Manage All Users 権利プロファイル, 266, 269
OTP 参照 ワンタイムパスワード (OTP)
OTP のオーセンティケータアプリ, 266
OTP の秘密鍵
ユーザーによる設定, 266
OTP のモバイルアプリ, 266
otpadm コマンド, 264
OUD (LDAP)
を使用したマスター KDC の構成, 99
ovsec_adm.*xxxxxx* ファイル
説明, 184

P

pam_pkcs11.conf ファイル, 239
pam_pkcs11 モジュール
スマートカードのサポート, 226
スマートカード用の構成, 239
pam_policy キーワード
使用, 22, 270
pam.d ディレクトリ
構成ファイルの変更, 24
PAM
/etc/syslog.conf ファイル, 32
CACKey のための pam_pkcs11 の構成 , 240
Coolkey のための pam_pkcs11 の構成, 240
Kerberos と, 55
nowarn オプションを使用した, 27
アーキテクチャー, 20
エラーのロギング, 32
概要, 19
計画, 21
検索順序, 35
構成ファイル, 34
Kerberos と, 184
概要, 35
構文, 35, 35, 35
サイト固有の作成, 25
スタック, 36
制御フラグ, 37
サービスモジュール, 41
サイト固有の構成ファイルの作成, 29
スタック
図, 38
説明, 36
例, 40
スマートカードおよび, 239
タスク, 23
トラブルシューティング, 33
フレームワーク, 19
ホームディレクトリの暗号化, 26
モジュールの追加, 27
リファレンス, 34
ワンタイムパスワード (OTP) モジュール, 263
PAM のサポート
スマートカード, 226
PAM モジュール
pam_pkcs11, 239
リスト, 41
passwd コマンド
と kpasswd コマンド, 180
pcscd デーモン, 226
pcsclite ライブラリ

スマートカードのサポート, 226
pkcs11_inspect
スマートカード情報の表示, 240
PKI 認証
スマートカードの使用, 230
plain.so.1 プラグイン
SASL と, 218
plugin_list オプション
SASL と, 219
principal.kadm5.lock ファイル
説明, 184
principal.kadm5 ファイル
説明, 184
principal.ok ファイル
説明, 184
principal.ulog ファイル
説明, 184
principal ファイル
説明, 184
proftpd デーモン
Kerberos と, 186
PS/SC
スマートカードへのドライバの接続, 226
ドライバのスマートカードへの接続, 228
PTP
Kerberos クライアント, 113, 119
スレーブ KDC と, 81, 89
マスター KDC と, 78, 91, 97
publickey マップ
DH 認証 , 274
pwcheck_method オプション
SASL と, 219

R

rcp コマンド
Kerberos と, 185
read_kt コマンド, 172, 174
reauth_timeout オプション
SASL と, 219
required 制御フラグ
PAM, 38
requisite 制御フラグ
PAM, 38
rlogin コマンド
Kerberos と, 185
rlogind デーモン
Kerberos と, 186
root 主体
ホストの keytab への追加, 170
rsh コマンド
Kerberos と, 185
rshd デーモン
Kerberos と, 186
rsyslog.conf エントリ
IP フィルタのための作成, 32

S

SASL
オプション, 219
概要, 217
環境変数, 218
プラグイン, 218
saslauthd_path オプション
SASL と, 219
scp コマンド
Kerberos と, 185
Secure NFS, 273
Secure RPC
説明, 273
と Kerberos, 274
Secure Shell
クライアント
スマートカード用の構成, 246
スマートカード用の構成, 239, 246
ハードウェアのエントリポイント, 222
sftp コマンド
Kerberos と, 185
slave_datatrans_*slave* ファイル
説明, 184
slave_datatrans ファイル
KDC の伝播と, 148
説明, 184
smartcard パッケージ, 225
SMF
キーサーバーの有効化, 275
solaris-desktop パッケージ, 234
ssh コマンド
Kerberos と, 185

sshd デーモン
  Kerberos と, 186
stash ファイル
  作成, 89, 91, 91
  定義, 187
subject_mapping ファイル, 242
sufficient 制御フラグ
  PAM, 38
Sun Ray Software (SRS)
  警告, 221, 231, 251
svcadm コマンド
  キーサービスデーモンを有効にする, 276
svcs コマンド
  キーサーバーサービスの一覧表示, 275
syslog.conf エントリ
  IP フィルタのための作成, 32

T

/tmp/krb5cc_*UID* ファイル
  説明, 184
/tmp/ovsec_adm.*xxxxxx* ファイル
  説明, 184
telnet コマンド
  Kerberos と, 185
telnetd デーモン
  Kerberos と, 186
TGS
  資格の取得, 59
TGT
  Kerberos, 47
TOTP 参照 ワンタイムパスワード (OTP)

U

/usr/bin/ftp コマンド
  Kerberos と, 185
/usr/bin/kdestroy コマンド
  Kerberos と, 185
/usr/bin/kinit コマンド
  Kerberos と, 185
/usr/bin/klist コマンド
  Kerberos と, 185
/usr/bin/kpasswd コマンド
  Kerberos と, 185
/usr/bin/ktutil コマンド
  Kerberos と, 185
/usr/bin/kvno コマンド
  Kerberos と, 185
/usr/bin/rcp コマンド
  Kerberos と, 185
/usr/bin/rlogin コマンド
  Kerberos と, 185
/usr/bin/rsh コマンド
  Kerberos と, 185
/usr/bin/scp コマンド
  Kerberos と, 185
/usr/bin/sftp コマンド
  Kerberos と, 185
/usr/bin/ssh コマンド
  Kerberos と, 185
/usr/bin/telnet コマンド
  Kerberos と, 185
/usr/lib/$ISA/pcsc/drivers/ifd-ccid.bundle/Contents ディレクトリ, 233
/usr/lib/inet/proftpd デーモン
  Kerberos と, 186
/usr/lib/kprop コマンド
  説明, 185
/usr/lib/krb5/kadmind デーモン
  Kerberos と, 186
/usr/lib/krb5/kpropd デーモン
  Kerberos と, 186
/usr/lib/krb5/krb5kdc デーモン
  Kerberos と, 186
/usr/lib/krb5/ktkt_warnd デーモン
  Kerberos と, 186
/usr/lib/libsasl.so ライブラリ
  概要, 217
/usr/lib/ocspd デーモン, 236
/usr/lib/pam_pkcs11/pkcs11_inspect
  スマートカードでの使用, 240
/usr/lib/pcscd デーモン, 233
/usr/lib/ssh/sshd デーモン
  Kerberos と, 186
/usr/sbin/gsscred コマンド
  説明, 185
/usr/sbin/in.rlogind デーモン
  Kerberos と, 186
/usr/sbin/in.rshd デーモン

Kerberos と, 186
/usr/sbin/in.telnetd デーモン
Kerberos と, 186
/usr/sbin/kadmin.local コマンド
説明, 185
/usr/sbin/kadmin コマンド
説明, 185
/usr/sbin/kclient コマンド
説明, 185
/usr/sbin/kdb5_ldap_util コマンド
説明, 185
/usr/sbin/kdb5_util コマンド
説明, 186
/usr/sbin/kgcmgr コマンド
説明, 186
/usr/sbin/kproplog コマンド
説明, 186
USB デバイス番号
libccid, 234
USDT プローブ
Kerberos, 279
use_authid オプション
SASL と, 220
User Management 権利プロファイル, 269

V

/var/krb5/kadmin.log ファイル
説明, 184
/var/krb5/kdc.log ファイル
説明, 184
/var/krb5/principal.kadm5.lock ファイル
説明, 184
/var/krb5/principal.kadm5 ファイル
説明, 184
/var/krb5/principal.ok ファイル
説明, 184
/var/krb5/principal.ulog ファイル
説明, 184
/var/krb5/principal ファイル
説明, 184
/var/krb5/slave_datatrans_*slave* ファイル
説明, 184
/var/krb5/slave_datatrans ファイル
説明, 184
/var/krb5/.k5.レルム ファイル
説明, 184

W

warn.conf ファイル
説明, 183
Web ブラウザ
スマートカードを使用したサイトへの認証, 249
winscard API, 232

X

XDMCP
スマートカード用のデスクトップの構成, 235