Solaris のシステム管理 (セキュリティサービス)

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–2464–10
2003 年 8 月

原典: *System Administration Guide: Security Services*

Part No: 817-0365-10

Revision A

031007@6671

目次

はじめに

『*Solaris* のシステム管理 (セキュリティサービス)』は、Solaris™ のシステム管理マニュアルの一部です。このマニュアルでは、SunOS™ 5.9 オペレーティングシステムがすでにインストールされているものとします。さらに、使用するネットワークソフトウェアが設定されているものとします。SunOS 5.9 オペレーティングシステムは、Solaris 9 プロダクトファミリの一部です。Solaris 9 プロダクトファミリには、Solaris 共通デスクトップ環境 (CDE) をはじめとする多くの機能が含まれています。

注 – Solaris オペレーティング環境は、SPARC® と x86 の 2 種類のハードウェア (プラットフォーム) 上で動作します。また、Solaris オペレーティング環境は、64 ビットと 32 ビットのアドレス空間で動作します。このマニュアルの情報は、両方のプラットフォームと両方のアドレス空間に適用されます。例外がある場合は、特別な章、節、注、箇条書き、図、表、例、またはコード例で、その旨を明記します。

対象読者

このマニュアルは、Solaris 9 のシステム管理者を対象にしています。このマニュアルを利用するにあたっては、UNIX® のシステム管理について 1 〜 2 年の経験が必要です。UNIX システム管理のトレーニングコースに参加することをお勧めします。

内容の紹介

『Solaris のシステム管理』全 7 巻には、主に次に示す内容が記載されています。

マニュアルのタイトル	トピック
『*Solaris* のシステム管理 *(*基本編*)*』	ユーザーアカウントとグループ、サーバーとクライアントの サポート、システムのシャットダウンとブート、リムーバブルメディア、ソウトウェアの管理 (パッケージとパッチ)、ディスクとデバイス、ファイルシステム、およびデータのバックアップと復元
『*Solaris* のシステム管理 *(*上級編*)*』	印刷サービス、端末とモデム、システム資源 (ディスク割り当て、アカウンティング、および crontab)、システムプロセス、 および Solaris ソフトウェアの障害追跡
『*Solaris* のシステム管理 *(IP* サービス*)*』	TCP/IP ネットワーク、IPv4 と IPv6、DHCP、IP セキュリティ、モバイル IP、および IP ネットワークのマルチパス化
『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: DNS*、*NIS*、*LDAP* 編*)*』	DNS、NIS、および LDAP のネーミングとディレクトリサービス
『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: FNS*、*NIS+* 編*)*』	FNS および NIS+ のネーミングとディレクトリサービス
『*Solaris* のシステム管理 *(*資源管理とネットワークサービス*)*』	システム資源、リモートファイルシステム、メールサービス、SLP、および PPP
『*Solaris* のシステム管理 *(*セキュリティサービス*)*』(このマニュアル)	監査、デバイス割り当て、PAM、RBAC、Solaris Secure Shell、および SEAM

関連書籍

以下に、このマニュアルで参照している関連書籍を示します。

- Carasik, Anne 著 『*UNIX Secure Shell*』McGraw Hill 発行、1999
- Cheswick, William R、Steven M. Bellovin 共著『*Firewalls and Internet Security*』Addison Wesley 発行、1994

Sun のオンラインマニュアル

docs.sun.com では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索を行うこともできます。URL は、http://docs.sun.com です。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–1** 表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 system%
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	system% **su** password:
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、rm *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

コード例は次のように表示されます。

- C シェル

```
machine_name% command y|n [filename]
```

- C シェルのスーパーユーザー

```
machine_name# command y|n [filename]
```

- Bourne シェルおよび Korn シェル

```
$ command y|n [filename]
```

- Bourne シェルおよび Korn シェルのスーパーユーザー

```
# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

一般規則

- このマニュアルでは、英語環境での画面イメージを使っています。このため、実際に日本語環境で表示される画面イメージとこのマニュアルで使っている画面イメージが異なる場合があります。本文中で画面イメージを説明する場合には、日本語のメニュー、ボタン名などの項目名と英語の項目名が、適宜併記されています。
- このマニュアルでは、「x86」という用語は、Intel 32 ビット系列のマイクロプロセッサチップ、および AMD が提供する互換マイクロプロセッサチップを意味します。

パート I

セキュリティの概要

このマニュアルでは、Solaris™ オペレーティング環境のセキュリティ支援機能について説明します。セキュリティ機能のシステム管理者とユーザーを対象としています。この章の内容は以下のとおりです。

第 1 章

セキュリティサービス (概要)

セキュリティサービスの概要

コンピュータ環境のセキュリティ強化を支援するために、Solaris オペレーティング環境では、次のような機能を提供しています。

- マシンセキュリティ – ユーザーの事故や侵入者の悪意によってマシンリソースやファイルが変更されるのを防止する機能
- 認証 – 安全にユーザーを識別する機能。ユーザー名とその証明書 (通常はパスワード) を要求する
- セキュリティ保護された通信 – 認証されたユーザーまたはグループが通信するときに、傍受、改ざん、または偽装を防ぐ
- 監査 – ファイルアクセス、セキュリティ関連のシステムコール、および認証の失敗など、セキュリティの変更が発生した場所を識別してシステムに通知する

システムセキュリティ全般については、第 2 章を参照してください。

マシンセキュリティ

マシンセキュリティは、マシンのリソースが正しく使用されることを保証します。ユーザーまたは管理者は、アクセス制御を利用して、システムリソースへのアクセス権を許可するユーザーを制限できます。Solaris オペレーティング環境には、次のようなマシンセキュリティ機能とアクセス制御機能が含まれています。

- ログイン制御 – コンピュータ上のハードウェア、ファイル、プロセスへのアクセス。第 3 章を参照

- UNIX® アクセス権 – ファイルまたはディレクトリの属性。アクセス権を使用すれば、ファイルの読み取り、書き込み、実行、あるいはディレクトリの検索を行えるユーザーおよびグループを制限できる。第 4 章を参照
- 役割によるアクセス制御 (RBAC) – 特殊な制限付きユーザーアカウントを作成するためのアーキテクチャ。特定のセキュリティ関連タスクの実行を許可する。第 5 章を参照
- セキュリティ強化スクリプト – スクリプトを使用することにより、多数のシステムファイルとパラメータを調整し、セキュリティの危険性を減少させる。第 8 章を参照
- デバイス割り当て – フロッピーディスクや CD-ROM ドライブなどのデバイスを使用できるユーザーを制限する機能。デバイス割り当てにより、権限を持つ特定のユーザーだけがデバイスを使用できる。375 ページの「デバイス割り当ての管理 (作業)」を参照
- SunScreen™ 3.2 Secure Net – 組織のネットワークを出入りする情報のフローを選択的に制御するためのファイアウォール。このファイアウォールは、ネットワーク内のセグメント間の情報フローも制御できる。SunScreen 3.2 のマニュアルセットを参照

認証サービス

認証とは、定義済みの条件に基づいてユーザーまたはサービスを識別するメカニズムのことです。認証サービスには、単純な認証システム (名前とパスワードの組み合わせ) から複雑な暗号化認証システム (スマートカード、生体認証など) まで、さまざまな形態があります。強力な認証メカニズムは、ユーザーだけが知っている情報や検証可能な情報を使用します。ユーザー名は、ユーザーが知っている情報の一例です。スマートカードや指紋は、検証可能な情報の一例です。Solaris オペレーティング環境の認証機能は、次の要素で構成されます。

- Secure RPC – Diffie-Hellman 方式に基づいた認証技術。165 ページの「Secure RPC の概要」を参照
- Pluggable Authentication Module (PAM) – システムエントリサービス (`login`、`ftp` など) に変更を与えずに、さまざまな認証技術をプラグイン可能にするフレームワーク。第 10 章を参照
- Sun Enterprise Authentication Mechanism (SEAM) – クライアント/サーバーアーキテクチャの 1 つで、暗号化を使用して認証を行う。第 13 章を参照
- スマートカード –マイクロプロセッサとメモリーが組み込まれたプラスチックのカード。システムにアクセスするときに、カードリーダーを使用する。『*Solaris* スマートカードの管理』を参照
- ログイン管理ツール – ログインまたはセッション終了を管理するためのさまざまなコマンド。第 3 章を参照

セキュリティ保護された通信

セキュリティ保護された通信は、暗号化された認証を基本としています。認証を利用して、送信元と送信先が正しいユーザーまたはグループであることを保証します。通信は、送信元で暗号化され、送信先で復号化されます。暗号化されていれば、侵入者が通信を傍受できたとしても、その内容が解読されることはありません。Solaris オペレーティング環境のセキュリティ保護された通信機能は、次の要素で構成されます。

- Sun Enterprise Authentication Mechanism (SEAM) – クライアント/サーバーアーキテクチャの 1 つで、暗号化を使用して認証を行う。第 13 章を参照
- インターネットプロトコルセキュリティアーキテクチャ (IPsec) – IP データグラムを保護するアーキテクチャ。機密性、強力なデータ完全性、データ認証、部分的なシーケンス完全性を実現する。部分的なシーケンス完全性とは再生保護である。『*Solaris* のシステム管理 *(IP* サービス*)*』の「IPsec (概要)」を参照
- Solaris Secure Shell – データ転送と対話型ユーザーのネットワークセッションを、盗聴、セッションハイジャック、および man-in-the-middle 攻撃から保護するプロトコルの 1 つ。公開鍵暗号化によって、強力な認証を提供する。X Window System などのネットワークサービスは、Secure Shell 接続によって安全にトンネル化することで、セキュリティが向上する。第 11 章を参照

監査とデバイス管理

監査は、システムのセキュリティと保全性に関する基本概念です。監査は、システムの動作とイベントの履歴を検査して、発生した処理を確認するプロセスです。監査では、発生した処理、実行したユーザー、実行日時、影響を受けた処理がログに記録されます。デバイス管理では、フロッピーディスクや CD-ROM などの周辺機器の割り当てを制御します。Solaris の監査とデバイス管理については、第 20 章を参照してください。

パート II　システムセキュリティの管理

第 2 章	ファイルのセキュリティ、マシンのセキュリティ、およびネットワークのセキュリティについて、概要を説明します。
第 3 章	マシンの保護、マシン使用の監視、root アクセスの不当な使用の防止などの手順について説明します。
第 4 章	ファイル情報の表示方法、ファイルの所有権とアクセス権の変更方法、特殊なファイルアクセス権の設定方法の手順について説明します。
第 5 章	役割によるアクセス制御 (RBAC) の概要について説明します。
第 6 章	RBAC を使用する手順について説明します。
第 7 章	RBAC の参照情報を示します。
第 8 章	自動セキュリティ拡張ツール (ASET) の概要について説明します。ASET を対話的または定期的に実行するための手順を説明します。定期的な実行は、`cron` ジョブを使って行います。クライアントの ASET レポートをサーバー上で収集する方法についても説明します。

第 2 章

マシンセキュリティの管理 (概要)

マシンの情報のセキュリティを保つことは、重要なシステム管理作業です。この章では、マシンセキュリティ管理の概要を説明します。

この章の内容は以下のとおりです。

コンピュータシステムへのアクセスを制御する

サイトでは、1 台のサーバーに接続された多数のマシンを、1 つの大規模で多面的なシステムとみなすことができます。システム管理者は、この大規模なシステムのセキュリティ管理に責任があります。システム管理者は、ネットワークの外部からの侵入を防ぐ必要があります。また、ネットワーク内部のマシン上のデータの完全性を確保する必要もあります。

ファイルレベルにおいて、Solaris オペレーティング環境にはいくつかの標準セキュリティ機能が組み込まれており、ファイル、ディレクトリ、およびデバイスを保護するため使用できます。システムレベルとネットワークレベルでは、セキュリティの内容はほぼ同じです。セキュリティ防御の第一線は、システムへのアクセスを制御することです。次の方法でシステムへのアクセスを制御または監視できます。

- 物理的なセキュリティの管理
- ログイン制御の管理
- リソース使用の監視と制限
- ファイルの保護

- ネットワークアクセスの制御
- セキュリティ問題の報告

物理的なセキュリティの管理

システムへのアクセスを制御するには、コンピュータ環境の物理的なセキュリティを管理する必要があります。たとえば、マシンにログインしたままこれを放置することは不当なアクセスを招く原因になります。侵入者がオペレーティングシステムやネットワークにアクセスしないとも限らないからです。コンピュータの周辺環境やコンピュータハードウェアは、不当なアクセスから物理的に保護する必要があります。

システム管理者は、ハードウェア設定に対する不当なアクセスから SPARC マシンを物理的に保護することができます。`eeprom(1M)` コマンドを使って、パスワードがないと PROM にアクセスできないようにしてください。詳細は、59 ページの「ハードウェアアクセスのパスワードを必須にする方法」を参照してください。

ログイン制御の管理

システムやネットワークへの無許可のログインも防止する必要があります。この制限は、パスワード制御とログイン制御によって行うことができます。システム上のすべてのアカウントには、パスワードを設定します。パスワードはシンプルな認証メカニズムです。アカウントにパスワードを設定しないと、ユーザー名を推測できる侵入者であれば誰でもネットワーク全体にアクセスできることになります。力ずくの野蛮な攻撃を許さないためには、強力なパスワードアルゴリズムが必要です。

ユーザーがシステムにログインすると、`login` コマンドは `/etc/nsswitch.conf` ファイル内の情報に従って、該当するデータベースを照会します。このファイルには次のエントリを含めることができます。

- `files` – ローカルマシンの `/etc` ファイルを指定する
- `nis` – NIS マスターサーバーの NIS データベースを指定する
- `nisplus` – NIS+ root サーバーの NIS+ データベースを指定する
- `ldap` – LDAP サーバーの LDAP ディレクトリサービスを指定する

`nsswitch.conf` ファイルの詳細は、`nsswitch.conf(4)` のマニュアルページを参照してください。ネームサービスまたはディレクトリサービスの詳細は、『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: DNS*、*NIS*、*LDAP* 編*)*』または『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: FNS*、*NIS+* 編*)*』を参照してください。

`login` コマンドは、入力されたユーザー名とパスワードを確認します。ユーザー名がパスワードファイルにないと、 `login` コマンドはマシンへのアクセスを拒否します。あるいは、入力されたユーザー名に対するパスワードが正しくないと、`login` コマンドはマシンへのアクセスを拒否します。有効なユーザー名とそれに対応するパスワードが入力されれば、システムはマシンへのアクセスをユーザーに許可します。

Solaris システムには、精巧な認証メカニズムと承認メカニズムが備わっています。ネットワークレベルでの認証メカニズムや承認メカニズムについては、43 ページの「リモートアクセスの認証と承認」を参照してください。

パスワード情報の管理

ユーザーはシステムにログインするときに、ユーザー名とパスワードの両方を入力する必要があります。ログイン名は公開されていますが、パスワードは秘密にしなければなりません。ユーザーは、自分のパスワードを他人に知られてはいけません。また、ユーザーが各自のパスワードを慎重に選択し、頻繁に変更するようにしなければなりません。

パスワードは、最初にユーザーアカウントを設定するときに作成されます。ユーザーアカウントのセキュリティを管理するために、パスワード有効期限を設定し、パスワードを定期的に強制変更することができます。また、ユーザーアカウントを無効にして、パスワードをロックすることもできます。パスワードの管理の詳細については、『*Solaris* のシステム管理 *(*基本編*)*』の「ユーザーアカウントとグループの管理 (概要)」および `passwd(1)` のマニュアルページを参照してください。

ローカルパスワード

ネットワークで `/etc` 内のファイルを使用している場合、パスワード情報はシステムの `/etc/passwd` ファイルと `/etc/shadow` ファイルに保持されます。ユーザー名などの情報は、パスワードファイル `/etc/passwd` に保持されます。暗号化されたパスワードは、`/etc/shadow` という「シャドウファイル」に保持されます。このセキュリティ方式によって、暗号化されたパスワードにアクセスされることを防ぎます。`/etc/passwd` ファイルは、マシンにログインするユーザーであれば誰でも使用できますが、`/etc/shadow` ファイルを読み取ることができるのはスーパーユーザーだけです。`passwd` コマンドを使用すると、ローカルシステム上のユーザーのパスワードを変更できます。

NIS パスワードおよび *NIS+* パスワード

ネットワークで NIS+ を使用している場合、パスワード情報は NIS+ データベースに保持されます。NIS+ データベース内の情報は、アクセス権を許可されたユーザーを制限することによって保護できます。NIS+ データベースに保持されているユーザーのパスワードを変更するには、`passwd` コマンドを使用します。

ネットワークで NIS を使用している場合、パスワード情報は NIS パスワードマップに保持されます。NIS では、パスワードの有効期間を指定できません。NIS パスワードマップに保持されているユーザーのパスワードを変更するには、`passwd` コマンドを使用します。

LDAP パスワード

Solaris の LDAP ネーミングサービスは、パスワード情報とシャドウ情報を LDAP ディレクトリツリーの `ou=people` コンテナに格納します。Solaris LDAP ネーミングサービスクライアントでユーザーのパスワードを変更するには、`passwd -r ldap` コマンドを使用します。LDAP ネーミングサービスは、パスワードを LDAP リポジトリに格納します。

Solaris 9 12/02 リリースでは、Sun™ Open Net Environment (Sun ONE) Directory Server 上のパスワードポリシーが適用されます。つまり、クライアントの `pam_ldap` モジュールは、Sun ONE Directory Server で適用されているパスワードポリシー制御に従います。詳細は、『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: DNS*、*NIS*、*LDAP* 編*)*』の「LDAP ネームサービスのセキュリティモデル」を参照してください。

パスワードの暗号化

パスワードの強力な暗号化は攻撃に対する最初の障壁になります。Solaris 9 12/02 リリースには、4 つのパスワード暗号化モジュールがあります。MD5 モジュール群と Blowfish モジュールでは、UNIX アルゴリズムよりも強固なパスワードの暗号化が行われます。

サイトのアルゴリズム構成は、`/etc/security/policy.conf` ファイルに指定します。`policy.conf` ファイルには、次の表に示す識別子でアルゴリズムを指定します。

表 **2–1** パスワードの暗号化アルゴリズム

識別子	説明	アルゴリズムのマニュアルページ
`1`	BSD システムや Linux システムの MD5 アルゴリズムと互換性のある MD5 アルゴリズム	`crypt_bsdmd5(5)`
`2a`	BSD システムの Blowfish アルゴリズムと互換性のある Blowfish アルゴリズム	`crypt_bsdbf(5)`
`md5`	BSD バージョンや Linux バージョンの MD5 よりも強力とされている Sun MD5 アルゴリズム	`crypt_sunmd5(5)`
`__unix__`	従来の UNIX 暗号化アルゴリズム。`policy.conf` ファイルのデフォルトモジュールはこのアルゴリズム	`crypt_unix(5)`

policy.conf ファイルのアルゴリズム構成

次に、デフォルトの `policy.conf` ファイルを示します。

```
#
# Copyright 1999-2002 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
```

```
#
# /etc/security/policy.conf
#
# security policy configuration for user attributes. see policy.conf(4)
#
#ident  "@(#)policy.conf        1.6     02/06/07 SMI"
#
AUTHS_GRANTED=solaris.device.cdrw
PROFS_GRANTED=Basic Solaris User

# crypt(3c) Algorithms Configuration
#
# CRYPT_ALGORITHMS_ALLOW specifies the algorithms that are allowed to
# be used for new passwords.  This is enforced only in crypt_gensalt(3c).
#
CRYPT_ALGORITHMS_ALLOW=1,2a,md5

# To deprecate use of the traditional unix algorithm, uncomment below
# and change CRYPT_DEFAULT= to another algorithm.  For example,
# CRYPT_DEFAULT=1 for BSD/Linux MD5.
#
#CRYPT_ALGORITHMS_DEPRECATE=__unix__

# The Solaris default is the traditional UNIX algorithm.  This is not
# listed in crypt.conf(4) since it is internal to libc.  The reserved
# name __unix__ is used to refer to it.
#
CRYPT_DEFAULT=__unix__
```

CRYPT_DEFAULT の値を変更すると、新しいユーザーのパスワードは、新しい値に対応するアルゴリズムを使って暗号化されます。現在のユーザーがパスワードを変更したときに新しいパスワードがどのアルゴリズムで暗号化されるかは、古いパスワードがどのように暗号化されているかによって異なります。

たとえば、CRYPT_ALGORITHMS_ALLOW=1,2a,md5 かつ CRYPT_DEFAULT=1 であるとします。次の表は、パスワードの暗号化にどのアルゴリズムが使用されるかを示します。

元のパスワードアルゴリズム	変更後のパスワードアルゴリズム	意味
crypt_bsdmd5	crypt_bsdmd5	crypt_bsdmd5 の識別子は 1、すなわち CRYPT_DEFAULT の値。ユーザーのパスワードは引き続き crypt_bsdmd5 アルゴリズムで暗号化される
crypt_bsdbf	crypt_bsdbf	crypt_bsdbf の識別子は 2a。CRYPT_ALGORITHMS_ALLOW リストには 2a があるため、新しいパスワードは crypt_bsbdf アルゴリズムで暗号化される

元のパスワードアルゴリズム	変更後のパスワードアルゴリズム	意味
crypt_md5	crypt_md5	crypt_md5 の識別子は md5。CRYPT_ALGORITHMS_ALLOW リストには md5 があるため、新しいパスワードは crypt_md5 アルゴリズムで暗号化される
crypt_unix	crypt_bsdmd5	crypt_unix の識別子は __unix__。__unix__ 識別子は CRYPT_ALGORITHMS_ALLOW リストにないため、crypt_unix アルゴリズムを使用することはできない。新しいパスワードは CRYPT_DEFAULT アルゴリズムで暗号化される

アルゴリズムの選択を構成するための構文については、policy.conf(4) のマニュアルページを参照してください。新しいパスワード暗号化アルゴリズムを使用する方法については、54 ページの「パスワード暗号化のデフォルトアルゴリズムを変更する」を参照してください。

特別なシステムログイン

システムにアクセスする一般的な方法としては、通常のユーザーログインを使用するものと、root ログインを使用するものがあります。また、多数の特別な「システム」ログインを使用すると、ユーザーは root アカウントを使用しなくても管理コマンドを実行できます。システム管理者は、これらのログインアカウントにパスワードを割り当てます。

次の表に、システムのログインアカウントとその用途を示します。システムログインは特殊な機能を実行します。それぞれのログインに固有のグループ識別子番号 (GID) が付いています。各ログインには固有のパスワードを設定し、必要のある人だけに知らせるようにしてください。

表 **2–2** システムログイン

ログインアカウント	グループ **ID**	用途
root	0	ほぼ無制限。ほかのすべてのログイン、保護、アクセス権に優先する。root アカウントはシステム全体へのアクセス権を持つ。root ログインのパスワードはきわめて厳密に保護する必要がある。root アカウントはほとんどの Solaris コマンドを所有する
daemon	1	バックグラウンド処理を制御する
bin	2	一部の Solaris コマンドを所有する
sys	3	多数のシステムファイルを所有する
adm	4	特定のシステム管理ファイルを所有する

表 2–2 システムログイン　(続き)

ログインアカウント	グループ ID	用途
lp	71	プリンタ用のオブジェクトデータファイルとスプールデータファイルを所有する
uucp	5	UNIX 間のコピープログラム、UUCP 用のオブジェクトデータファイルとスプールデータファイルを所有する
nuucp	9	システムにログインしてファイル転送を開始するためにリモートシステムで使用される

リモートログイン

侵入者にとって、リモートログインは魅力的な手段です。Solaris オペレーティング環境には、リモートログインを監視、制限、および無効にするコマンドがいくつもあります。手順については、49 ページの「ログインとパスワードのセキュリティ」を参照してください。

デフォルトでは、システムのマウスやキーボード、フレームバッファ、オーディオデバイスなど、一定のシステムデバイスを、リモートログインを通して制御したり読み取ったりすることはできません。詳細は、`logindevperm(4)` のマニュアルページを参照してください。

ダイヤルアップログイン

モデムやダイヤルアップポートを通してアクセスされうるコンピュータには、セキュリティ層をもう 1 つ追加できます。つまり、モデムやダイヤルアップポートを通してシステムにアクセスするユーザーには「ダイヤルアップパスワード」を要求することができます。ダイヤルアップパスワードは、ユーザーがマシンへのアクセス権を取得する前に入力する必要があるパスワードです。

スーパーユーザー以外はダイヤルアップパスワードを作成または変更できません。システムの完全性を確保するために、月に一度はパスワードを変更する必要があります。この機能の最も有効な使用方法は、ゲートウェイシステムへのアクセス権を取得するためのダイヤルアップパスワードを要求することです。ダイヤルアップパスワードの設定方法については、52 ページの「ダイヤルアップパスワードを作成する方法」を参照してください。

ダイヤルアップパスワードの作成には、`/etc/dialups` と `/etc/d_passwd` という 2 つのファイルが必要です。`dialups` ファイルには、ダイヤルアップパスワードを必要とするポートのリストを含みます。`d_passwd` ファイルには、追加のダイヤルアップパスワードとして暗号化パスワードを必要とするシェルプログラムのリストを含みます。これら 2 つのファイルの情報は次のように処理されます。

- `/etc/passwd` 内のユーザーのログインシェルが `/etc/d_passwd` 内のエントリと一致する場合、そのユーザーはダイヤルアップパスワードを入力する必要があります。

- /etc/passwd 内のユーザーのログインシェルが /etc/d_passwd 内で見つからない場合、/usr/bin/sh のパスワードエントリが使用されます。

  /etc/passwd 内のログインシェルフィールドが空の場合は、/usr/bin/sh のパスワードエントリが使用されます。
- /etc/passwd 内のユーザーのログインシェルが /etc/d_passwd 内で見つからない場合、そのユーザーはデフォルトのパスワードを入力する必要があります。デフォルトのパスワードは /usr/bin/sh のエントリです。
- /etc/passwd 内のログインシェルフィールドが空の場合、そのユーザーはデフォルトのパスワードを入力する必要があります。デフォルトのパスワードは /usr/bin/sh のエントリです。
- /etc/d_passwd に /usr/bin/sh のエントリがない場合、/etc/passwd 内のログインシェルフィールドが空のユーザー、または /etc/d_passwd 内のエントリと一致しないユーザーには、ダイヤルアップパスワードの入力を求めるプロンプトは表示されません。
- /etc/d_passwd にエントリ /usr/bin/sh:*: しか入っていない場合、ダイヤルアップログインは使用できません。

マシンリソースへのアクセス制御

システム管理者は、システム活動の制御や監視を行うことができます。システム管理者は、だれがどのリソースを使用できるかを制限したり、リソースの使用状況を記録したり、だれがリソースを使用しているかを監視したりできます。 さらに、システム管理者は、リソースの不適切な使用を最小限に抑えるようにマシンを設定できます。

スーパーユーザーの制限と監視

システムをスーパーユーザーモードにするには、root パスワードが必要です。デフォルトの構成では、ユーザーはリモートのシステムに root としてログインできません。リモートログインするとき、ユーザーは自分のユーザー名でログインしてから、su コマンドを使用して root になる必要があります。セキュリティ上の理由から、su コマンドを使用中のユーザー、特にスーパーユーザーのアクセス権を取得しようとしているユーザーを監視する必要があります。 スーパーユーザーを監視したり、スーパーユーザーの使用を制限する手順については、57 ページの「スーパーユーザーの監視と制限」を参照してください。

役割によるアクセス制御を構成して root を置き換える

役割によるアクセス制御 (RBAC) は、スーパーユーザーの権限を制限できるように設計されています。スーパーユーザーすなわち root ユーザーは、システムのすべてのリソースにアクセスできますが、RBAC を使用すれば、スーパーユーザーの権限を個別の権限からなる役割の集合に置き換えることができます。たとえば、ユーザーアカウントの作成を行う役割やシステムファイルの変更を行う役割を個別に設定できます。特定の機能または機能群を扱う役割を設定したら、root の機能からこれらの機能を取り除くことができます。

個々の役割を引き受けるためには、既存のユーザーが自分のユーザー名とパスワードを使ってログインする必要があります。ログインしたユーザーは、特別な役割パスワードを入力してその役割を引き受けます。これによって、他人が root パスワードを知ったとしても、システムに損傷を与える能力は限定されます。RBAC の詳細は、第 5 章を参照してください。

マシンリソースの意図しない使用の防止

システム管理者やユーザーによって、意図しないエラーが引き起こされるのを防止できます。たとえば、PATH 変数を正しく設定することによって、トロイの木馬の実行を防止できます。あるいは、システムのうち各人の作業に必要な部分だけをユーザーに提供することによって、ユーザーエラーを避けることができます。実際、注意深い設定を行えば、ユーザーに対して、作業を能率的に行うのに必要な部分だけを見せるようにできます。

パス変数の設定

パス変数を正しく設定しないと、他人が持ち込んだプログラムを誤って実行し、データを壊したりシステムを損傷したりするおそれがあります。このようなプログラムはセキュリティ上の危険を招くので、「トロイの木馬」と呼ばれます。たとえば、公開ディレクトリの中に別の su プログラムが置かれていると、システム管理者が気づかずに実行してしまう可能性があります。このようなスクリプトは正規の su コマンドとまったく同じに見えます。このようなスクリプトは実行後に自らを削除してしまうため、トロイの木馬が実際に実行されたという証拠はほとんど残りません。

パス変数はログイン時に自動的に設定されます。パスは、起動ファイル、すなわち .login、.profile、および .cshrc を通して設定されます。現在のディレクトリ (.) への検索パスを最後に指定すれば、このタイプのトロイの木馬を実行するのを防ぐことができます。スーパーユーザーのパス変数には、現在のディレクトリを指定しないでください。

自動セキュリティ拡張ツール (ASET) は、起動ファイルのパス変数が正しく設定されているかどうかを調べます。また、パス変数にドット (.) エントリが含まれていないか確認します。

制限付きシェルの割り当て

標準シェルを使用すると、ユーザーはファイルを開く、コマンドを実行するなどの操作を行うことができます。制限付きシェルは、/usr/lib/rsh コマンドで呼び出されます。制限付きシェルを使用すると、ユーザーによるディレクトリの変更やコマンドの実行を制限できます。制限付きシェルは、リモートシェル (/usr/sbin/rsh) ではありません。標準のシェルと異なる点は次のとおりです。

- ユーザーはホームディレクトリ内に限定されるため、 cd コマンドを使用してディレクトリを変更できない。したがって、システムファイルを閲覧することはできない。
- ユーザーは PATH 変数を変更できないため、システム管理者によって設定された PATH のコマンドしか使用できない。さらに、完全なパス名を使ってコマンドやスクリプトを実行することもできない。
- ユーザーは > または >> を使用して出力をリダイレクトできない。

制限付きシェルでは、ユーザーが使用できるシステムファイルを制限できます。このシェルは、特定のタスクを実行するユーザーのために限られた環境を作成します。ただし、制限付きシェルは完全に安全なわけではありません。このシェルの目的は、あくまでも、経験の少ないユーザーが誤ってシステムファイルを損傷するのを防止することです。

制限付きシェルについては、rsh(1M) のマニュアルページを参照してください。

制限付きシェルよりさらにセキュリティを強化したシェルが Secure Shell、すなわち ssh コマンドです。Secure Shell を使用すると、セキュリティ保護されていないネットワーク上のリモートホストに、安全にアクセスすることができます。Secure Shell 上の使用方法については、第 12 章を参照してください。

ファイル内のデータへのアクセス制限

Solaris オペレーティング環境はマルチユーザー環境なので、ファイルシステムのセキュリティは、システムの最も基本的な問題です。ファイルの保護には、従来の UNIX のファイル保護と、より確実なアクセス制御リスト (ACL) との両方が使用できます。

ログイン制限を設定したあと、マシン上のデータへのアクセスを制御できます。一部のユーザーには特定のファイルの読み取りを許可し、別のユーザーには特定のファイルを変更または削除するアクセス権を与えることができます。誰にも見せたくないデータがある場合もあります。ファイルのアクセス権の設定方法については、第 4 章を参照してください。

setuid 実行可能ファイルの制限

実行可能ファイルがセキュリティリスクとなる場合があります。多くの実行可能プログラムは、スーパーユーザー (root) として実行しなければ適切に動作しません。このようなプログラムはユーザー ID が 0 (つまり、setuid=0) で実行されます。このよ

うなプログラムはだれが実行したとしても root ID で実行されます。root ID で動作するプログラムは、プログラムがセキュリティを念頭に置いて作成されていない限り、セキュリティの問題をはらんでいます。

Sun が setuid ビットを root に設定して出荷する実行可能プログラムを除き、setuid プログラムの使用を許可すべきではありません。setuid プログラムの使用を禁止できない場合は、少なくともその使用を制限すべきです。しっかりした管理を行うためには setuid プログラムの数を少なくする必要があります。

自動セキュリティ拡張ツール (ASET) の使用

ASET セキュリティパッケージには、システムのセキュリティを制御して監視できるように、自動管理ツールが組み込まれています。システム管理者は ASET セキュリティレベルを指定します。ASET には、低、中、高という 3 つのセキュリティレベルがあります。上のレベルほど、ASET のファイル制御機能が増え、ファイルアクセスは減少し、マシンセキュリティが厳しくなります。

詳細は、第 8 章を参照してください。

リソースマネージャの使用

Solaris ソフトウェアには、リソースマネージャと呼ばれる精巧なリソース管理ツールがあります。リソースマネージャは、サービス拒否攻撃を防止する上で有効な場合があります。リソースマネージャでは、特定のプロジェクトに必要なリソースを割り当てたり、スクリプトがマシンのリソースを過度に使用するのを防止したり、プロジェクトが占有できる領域を制限したりすることができます。リソースマネージャの使用方法の説明や詳しい例については、『*Solaris* のシステム管理 *(*資源管理とネットワークサービス*)*』の「Solaris 9 リソースマネージャ (トピック)」を参照してください。

マシンリソースの使用状況の監視

システム管理者は、システムの動作を監視する必要があります。次の点を含め、マシンのあらゆる側面に注意する必要があります。

- 通常の負荷はどの程度か
- 誰がシステムへのアクセス権を持っているか
- 各ユーザーはいつシステムにアクセスするか
- システムでは通常どのようなプログラムを実行するか

このような情報を把握していれば、ツールを使用してマシンの使用状況を監査し、各ユーザーのアクティビティを監視できます。セキュリティ違反が疑われる場合は、監視作業が特に役立ちます。監査モジュールの詳細については、第 20 章を参照してください。

ファイルアクセスの制御

Solaris オペレーティング環境はマルチユーザー環境です。マルチユーザー環境では、マシンにログインしているすべてのユーザーが、ほかのユーザーに属しているファイルを読み取ることができます。さらに、適切なアクセス権をもっているユーザーは、ほかのユーザーに属しているファイルを使用できます。表 2–3 は、ファイルシステムセキュリティのコマンドの一覧です。ファイルのセキュリティの作業手順については第 4 章を参照してください。

ファイルシステムセキュリティのコマンド

次の表は、ファイルとディレクトリの監視およびセキュリティに関するコマンドの一覧です。

表 2–3 ファイルシステムセキュリティのコマンド

コマンド名	説明	マニュアルページ
`ls`	ディレクトリ内のファイルとファイル情報を表示する	`ls(1)`
`chown`	ファイルの所有権を変更する	`chown(1)`
`chgrp`	ファイルのグループ所有権を変更する	`chgrp(1)`
`chmod`	ファイルのアクセス権を変更する。記号モード (文字と記号) または絶対モード (8 進数) を使用して、ファイルのアクセス権を変更できる	`chmod(1)`

ファイルの暗号化

ほかのユーザーがアクセスできないようにすることによって、ファイルを安全に保つことができます。たとえば、アクセス権 `600` の付いたファイルに、所有者やスーパーユーザー以外の人がアクセスすることはできません。アクセス権 `700` の付いたディレクトリも同様です。ただし、ほかのだれかがユーザーパスワードや `root` パスワードを推測して発見すると、そのファイルにアクセスできます。さらに、アクセス不能なはずのファイルも、マシンファイルのバックアップをテープにとるたびに、バックアップテープ上に保存されます。

米国では、Solaris ソフトウェアのすべてのユーザーは、もう 1 つのセキュリティ層として暗号化キットを使用できます。この暗号化キットには `crypt` コマンドが組み込まれており、テキストを変換してデータを暗号化します。詳細は、`crypt(1)` のマニュアルページを参照してください。

アクセス制御リスト (ACL)

ACL (「アクル」と読む) では、ファイルアクセス権の制御をより強化できます。ACL は、Solaris オペレーティング環境の従来の UNIX ファイル保護機能では不十分な場合に追加で使用します。従来の UNIX ファイル保護機能は、所有者、グループ、その他のユーザーという 3 つのユーザークラスに読み取り権、書き込み権、実行権を提供します。 ACL では、ファイルセキュリティを管理するレベルがさらに詳細になります。ACL では、次のファイルアクセス権を定義できます。

- 所有者のファイルアクセス権
- 所有者のグループのファイルアクセス権
- 所有者のグループに属していないユーザーのファイルアクセス権
- 特定ユーザーのファイルアクセス権
- 特定グループのファイルアクセス権
- 以上のカテゴリそれぞれのデフォルトアクセス権

ACL を設定する手順については、77 ページの「アクセス制御リスト (ACL) の使用」を参照してください。

次の表に、ファイルやディレクトリに対して ACL を管理するコマンドを示します。

表 **2–4** アクセス制御リスト (ACL) コマンド

コマンド名	説明	マニュアルページ
`setfacl`	ACL エントリの設定、追加、変更、および削除を行う	`setfacl(1)`
`getfacl`	ACL エントリを表示する	`getfacl(1)`

マシン間でのファイルの共有

ネットワークファイルサーバーは、どのファイルを共有できるかを制御できます。また、共有ファイルにアクセスできるクライアント、およびそれらのクライアントに許可するアクセス権の種類も制御します。一般に、ファイルサーバーは、すべてのクライアントまたは特定のクライアントに、読み取り権と書き込み権、または読み取り専用アクセス権を与えることができます。アクセス制御は、`share` コマンドで資源を利用可能にするときに指定します。

ファイルサーバー上の `/etc/dfs/dfstab` ファイルは、サーバーがネットワーク上のクライアントに提供するファイルシステムを、一覧表示します。ファイルシステムの共有の詳細については、『*Solaris* のシステム管理 *(*資源管理とネットワークサービス*)*』の「ファイルシステムの自動共有」を参照してください。

共有ファイルへの root アクセスの制限

一般的にスーパーユーザーは、ネットワーク上で共有されるファイルシステムには root としてアクセスできません。NFS システムは、要求者のユーザーをユーザー ID 60001 を持つユーザー nobody に変更することによって、マウントされているファイルシステムへの root アクセスを防止します。ユーザー nobody のアクセス権は、公共ユーザーに与えられているアクセス権と同じです。つまり、ユーザー nobody のアクセス権は資格をもたないユーザーのものと同じです。たとえば、ファイルの実行権しか公共に許可していなければ、ユーザー nobody はそのファイルを実行することしかできません。

NFS サーバーは、共有ファイルシステムのスーパーユーザー特権をホスト単位で与えることができます。この特権を与えるには、share コマンドの root=*hostname* オプションを使用します。このオプションは慎重に使用してください。

ネットワークアクセスの制御

コンピュータは通常、複数のコンピュータからなる構成の一部です。この構成を「ネットワーク」と呼びます。ネットワークでは、接続されたコンピュータの間で情報を交換できます。さらに、ネットワークに接続されたコンピュータは、ネットワーク上のほかのコンピュータにあるデータなどのリソースにアクセスできます。ネットワーキングによってコンピュータの処理能力と性能が高まります。しかし、同時に、ネットワーキングによってコンピュータのセキュリティが危険にさらされます。

たとえば、ネットワーク内では、個々のマシンは情報を共有できるように開放されています。また、多数の人々がネットワークにアクセスするので、特にユーザーエラーを通じて、不当なアクセスが発生する可能性も大きくなります。たとえば、パスワードの不適切な扱いも不当なアクセスの原因になりえます。

ネットワークセキュリティ機構

一般にネットワークのセキュリティは、リモートシステムからの操作を制限またはブロックすることを指しています。次の図は、リモート操作に適用できるセキュリティ制限を示します。

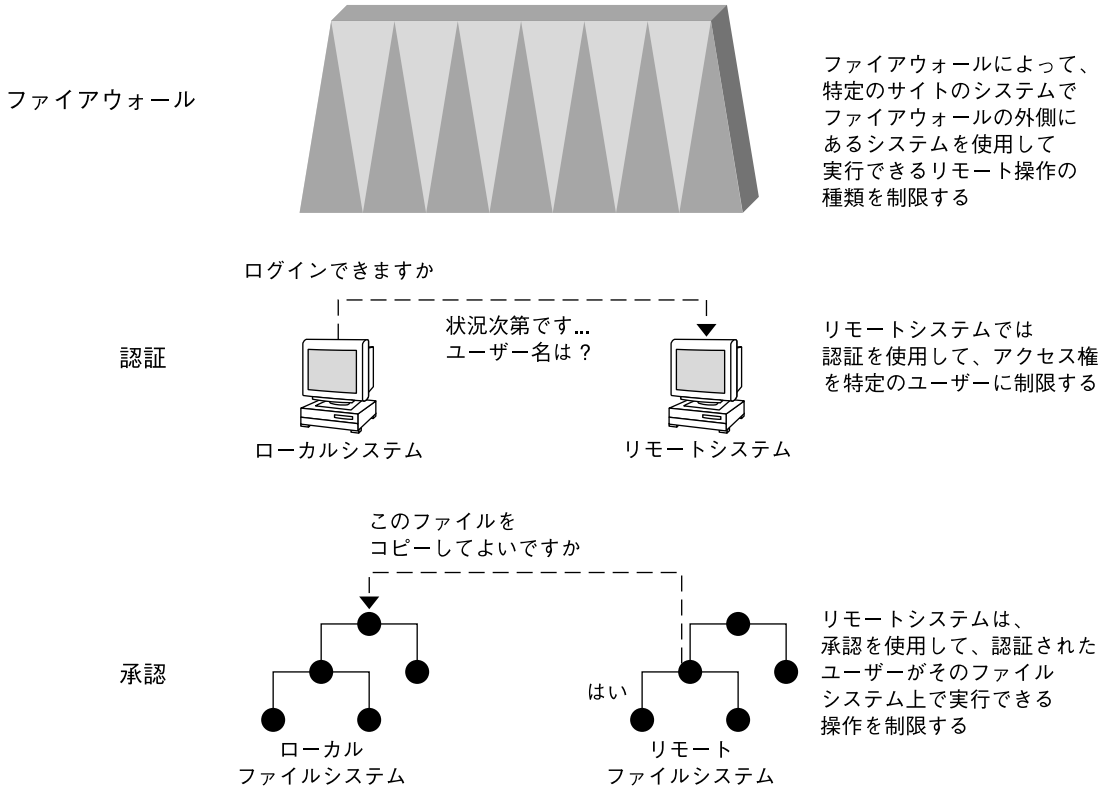


図 **2–1** リモート操作のセキュリティ制限

リモートアクセスの認証と承認

「認証」とは、リモートマシンにアクセスできるユーザーを特定のユーザーに限定する方法です。認証は、マシンレベルでもネットワークレベルでも設定できます。いったんユーザーがリモートマシンにアクセスすると、「承認」という方法でそのユーザーがリモートシステム上で実行できる操作が制限されます。次の表に、ネットワーク上のマシンを許可されていない使い方から保護できる、認証と承認の種類を示します。

表 **2–5** リモートアクセスの認証と承認の種類

形式	説明	参照先
LDAP と NIS+	LDAP ディレクトリサービスと NIS+ ネームサービスは、ネットワークレベルで認証および承認を行う	『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: DNS*、*NIS*、*LDAP* 編*)*』および『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: FNS*、*NIS+* 編*)*』
リモートログインコマンド	リモートログインコマンドを使用すると、ユーザーはネットワーク経由でリモートマシンにログインし、そのリソースを使用できる。リモートログインコマンドには `rlogin`、`rcp`、 `ftp` がある。「信頼される (trusted) ホスト」の場合、認証は自動的に処理される。それ以外の場合は、自分自身を認証するように求められる	『*Solaris* のシステム管理 *(*資源管理とネットワークサービス*)*』の「リモートシステムへのアクセス (手順)」
Secure RPC	Secure RPC を使用すると、リモートマシン上で要求を出したユーザーの認証が行われ、ネットワーク環境のセキュリティが高まる。Secure RPC には、UNIX、DES、または Kerberos 認証システムを使用できる	165 ページの「Secure RPC の概要」
	Secure RPC を使用すると、NFS 環境にセキュリティを追加できる。Secure RPC を備えた NFS 環境を Secure NFS と呼ぶ	165 ページの「NFS サービスと Secure RPC」
DES 暗号化	データ暗号化規格 (DES) 暗号化機能は 56 ビットの鍵を使用して、秘密鍵を暗号化する	166 ページの「DES 暗号化」
Diffie-Hellman 認証	この認証方法は、送信側マシンの、共通鍵を使用して現在の時刻を暗号化する機能を利用する。受信側マシンは共通鍵の復号化を行い、復号化された時刻と受信側マシンの現在の時刻とを比較する	166 ページの「Diffie-Hellman 認証」
Kerberos	Kerberos は DES 暗号化を使用して、システムのログイン時にユーザーを認証する	例については、233 ページの「マスター KDC を構成する方法」を参照

Solaris システム間での特権付きポートの使用

Secure RPC を実行したくない場合は、代わりに Solaris の「特権付きポート」メカニズムを使用できます。特権付きポートには、1024 未満のポート番号が割り当てられます。クライアントシステムは、クライアントの資格を認証したあと、特権付きポートを使用してサーバーへの接続を設定します。次に、サーバーは接続のポート番号を検査してクライアントの資格を確認します。

ただし、Solaris 以外のクライアントは、特権付きポートを使用して通信できないことがあります。クライアントが特権付きポートを使って通信できない場合は、次のようなエラーメッセージが表示されます。

```
"Weak Authentication
NFS request from unprivileged port"
```

ファイアウォールシステム

ファイアウォールシステムを設定すると、ネットワーク内のリソースを外部のアクセスから保護できます。「ファイアウォールシステム」は、内部ネットワークと外部ネットワークの間のバリアとして機能するセキュリティ保護ホストです。個々のネットワークはほかのネットワークを「信頼された状態でない」ものとして扱います。内部ネットワークと、インターネットなどの外部ネットワークとの間に、このような設定を必ず行うようにしてください。

ファイアウォールはゲートウェイとしても機能しますし、バリアとしても機能します。ファイアウォールは、まず、ネットワーク間でデータを渡すゲートウェイとして機能します。さらに、ファイアウォールは、データが勝手にネットワークに出入りしないようにブロックするバリアとして機能します。ファイアウォールは、内部ネットワーク上のユーザーに対して、ファイアウォールシステムにログインしてリモートネットワーク上のホストにアクセスするように要求します。また、外部ネットワーク上のユーザーは、内部ネットワーク上のホストにアクセスする前に、ファイアウォールシステムにログインしなければなりません。

ファイアウォールは、一部の内部ネットワーク間でも有効です。たとえば、ファイアウォール、すなわちセキュリティ保護ゲートウェイコンピュータを設定することによって、パケットの転送を制限できます。ゲートウェイコンピュータは、ゲートウェイ自身をパケットの発信元アドレスまたは着信先アドレスとしないような、2 つのネットワーク間のパケット交換を禁止できます。また、ファイアウォールは、特定のプロトコルについてのみパケットを転送するように設定する必要があります。たとえば、パケットでメールを転送できるが、`telnet` や `rlogin` コマンドのパケットは転送できないようにできます。ASET は、高度なセキュリティを適用して実行すると、インターネットプロトコル (IP) パケットの転送機能を無効にします。

さらに、内部ネットワークから送信されるすべての電子メールは、まずファイアウォールシステムに送信されます。ファイアウォールは、このメールを外部ネットワーク上のホストに転送します。ファイアウォールシステムは、すべての着信電子メールを受信して、内部ネットワーク上のホストに配信します。

注意 – ファイアウォールは、アクセス権のないユーザーが内部ネットワーク上のホストにアクセスする行為を防止します。ファイアウォールに適用される厳密で確実なセキュリティを管理する必要がありますが、ネットワーク上の他のホストのセキュリティはもっと緩やかでもかまいません。ただし、ファイアウォールシステムを突破できる侵入者は、内部ネットワーク上の他のすべてのホストへのアクセスを取得できる可能性があります。

ファイアウォールシステムには、「信頼されるホスト」を配置しないでください。信頼されるホストとは、ユーザーがログインするときに、パスワードを入力する必要がないホストのことです。ファイアウォールシステムでは、ファイルシステムを共有しないでください。また、ほかのサーバーのファイルシステムをマウントしないでください。

ASET を使用すると、マシンをファイアウォールに強固に組み込むことができます。ASET によって、ファイアウォールシステムのセキュリティが高まります (第 8 章を参照)。同様に、IPsec もファイアウォールの保護が可能です。IPsec を使ってネットワークトラフィックを保護する方法については、『*Solaris* のシステム管理 *(IP* サービス*)* 』の「IPsec (概要)」を参照してください。

パケットスマッシング

ほとんどのローカルエリアネットワークでは、データはパケットと呼ばれるブロック単位でコンピュータ間で転送されます。アクセス権のないユーザーが、「パケットスマッシング」という方法により、データを損傷する可能性があります。あるいは、データを破壊する可能性もあります。パケットスマッシングでは、パケットが宛先に到達する前に取り込まれます。侵入者は、その内容に勝手なデータを書き込み、パケットを元のコースに戻します。ローカルエリアネットワーク上では、パケットはサーバーを含むすべてのマシンに同時に到達するので、パケットスマッシングは不可能です。ただし、ゲートウェイ上ではパケットスマッシングが可能なため、ネットワーク上のすべてのゲートウェイを保護する必要があります。

最も危険なのは、データの完全性に影響するような攻撃です。このような攻撃を受けると、パケットの内容が変更されたり、ユーザーが偽装されたりします。盗聴を伴う攻撃では、データの完全性が損なわれることはありません。盗聴者は、会話を記録して、あとで再生します。盗聴者がユーザーを偽装することはありません。盗聴攻撃によってデータの完全性が損なわれることはありませんが、プライバシが侵害されます。ネットワーク上でやりとりされるデータを暗号化すると、重要な情報のプライバシを保護できます。IP データグラムの暗号化方法については、『 *Solaris* のシステム管理 *(IP* サービス*)*』の「インターネットキー交換」を参照してください。

セキュリティ問題の報告

セキュリティの問題が発生した可能性がある場合は、The Computer Emergency Response Team/Coordination Center (CERT/CC) に連絡してください。CERT/CC は、Defense Advanced Research Projects Agency (DARPA) の資金提供を受けたプロジェクトで、カーネギメロン大学の Software Engineering Institute にあります。CERT/CC はセキュリティ問題の解決を支援できます。また、特定のニーズに合った他の Computer Emergency Response Team を紹介することもできます。CERT/CC に連絡するには、24 時間のホットラインに電話する方法と、電子メールを `cert@cert.sei.cmu.edu` に送る方法があります。

第 3 章

マシンのセキュリティの適用 (手順)

この章では、Solaris 環境のマシンのセキュリティを設定する手順について説明します。これらの手順の概要については、次の節を参照してください。

- 47 ページの「マシンのセキュリティの適用 (作業マップ)」

マシンセキュリティの概要については、第 2 章を参照してください。

マシンのセキュリティの適用 (作業マップ)

コンピュータのセキュリティレベルは、コンピュータの最も弱い点によって決まります。次の作業マップは、どのような分野を監視してそのセキュリティを確保すべきかを示しています。

作業	説明	参照先
ユーザーのログイン状態を表示する	`logins` コマンドを使ってユーザーのログイン状態を表示する	49 ページの「ユーザーのログイン状態を表示する方法」
パスワードを所有していないユーザーを発見する	`logins` コマンドを使って、パスワードを必要としないアカウントを持つユーザーだけを発見する	50 ページの「パスワードを持たないユーザーを表示する方法」
ログインを一時的に無効にする	システムシャットダウンや定常的な保守の中でマシンへのユーザーログインを拒否する	50 ページの「ユーザーのログインを一時的に無効にする方法」

作業	説明	参照先
パスワードの強力な暗号化を可能にする	パスワード暗号化のアルゴリズムを指定する	54 ページの「パスワード暗号化のアルゴリズムを指定する方法」
ネームサービスでパスワードの強力な暗号化を提供する	ネームサービスを使用するときのパスワード暗号化のアルゴリズムを指定する	55 ページの「NIS+ ドメイン用の新しいパスワードアルゴリズムを指定する方法」 55 ページの「NIS ドメイン用の新しいパスワードアルゴリズムを指定する方法」 56 ページの「LDAP ドメイン用の新しいパスワードアルゴリズムを指定する方法」
新しいパスワード暗号化モジュールを追加する	サードパーティアルゴリズムを追加する	56 ページの「サードパーティのパスワード暗号化モジュールをインストールする方法」
ログイン失敗操作を保存する	正しいパスワードの入力に 5 回失敗したユーザーのログを作成する	51 ページの「ログイン失敗操作を保存する方法」
ダイヤルアップパスワードを作成する	モデムやダイヤルアップポートを通してリモートログインするユーザーに追加パスワードを要求する	52 ページの「ダイヤルアップパスワードを作成する方法」
ダイヤルアップエントリを一時的に無効にする	ユーザーがモデムやポートを通してリモートからダイヤルできないようにする	53 ページの「ダイヤルアップログインを一時的に無効にする方法」
su コマンドを使用しているユーザーを監視する	sulog ファイルを定期的に読み取る	58 ページの「su コマンドを使用するユーザーを監視する方法」
スーパーユーザーの活動をコンソールに表示する	スーパーユーザーのアクセス操作を監視する	58 ページの「コンソールへのスーパーユーザー (root) アクセス操作を表示する方法」
スーパーユーザーがリモートからコンソールにアクセスできないようにする	自身のユーザー名でログインしてから root になるよう、リモートユーザーに要求する	59 ページの「スーパーユーザー (root) でのリモートログインを防ぐ方法」
ユーザーがマシンパラメータを変更できないようにする	ユーザーが PROM の設定を変更できないようにする	59 ページの「ハードウェアアクセスのパスワードを必須にする方法」
アボートシーケンスを無効にする	ユーザーが PROM にアクセスできないようにする	60 ページの「システムのアボートシーケンスを無効または有効にする方法」

ログインとパスワードのセキュリティ

この節では、ログインの制御と監視について説明します。

▼ ユーザーのログイン状態を表示する方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **logins** コマンドを使用してユーザーのログイン状態を表示します。

```
# logins -x -l username
```

`-x`	ログイン状態情報の拡張セットを表示する
`-l` *username*	指定するユーザーのログイン状態を表示する。*username* はユーザーのログイン名。複数のログイン名は、コンマで区切って指定する

`logins` コマンドは、適切なパスワードファイルを使ってユーザーのログイン状態を表示します。このファイルは、ローカルの `/etc/passwd` ファイルか、ネームサービスのパスワードデータベースです。詳細は、`logins(1M)` のマニュアルページを参照してください。

例 — ユーザーのログイン状態を表示する

次の例では、ユーザー `rimmer` のログイン状態が表示されます。

```
# logins -x -l rimmer
rimmer       500     staff           10   Annalee J. Rimmer
                     /export/home/rimmer
                     /bin/sh
                     PS 010170 10 7 -1
```

`rimmer`	ユーザーのログイン名を示す
`500`	ユーザー ID (UID) を示す
`staff`	ユーザーの一次グループを示す
`10`	グループ ID (GID) を示す
`Annalee J. Rimmer`	コメントを示す
`/export/home/rimmer`	ユーザーのホームディレクトリを示す

/bin/sh	ログインシェルを示す
PS 010170 10 7 -1	次のパスワード有効期限情報を示す ■ パスワードの最終変更日 ■ 次に変更するまでに必要な日数 ■ 変更しないで使用できる日数 ■ 警告期間

▼ パスワードを持たないユーザーを表示する方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **logins** コマンドを使用して、パスワードを持っていないユーザーをすべて表示します。

   ```
   # logins -p
   ```

   -p オプションを指定すると、パスワードを持たないユーザーが表示されます。logins コマンドでは、ローカルマシン上およびネットワーク上のパスワードデータベースを使用できます。このコマンドでは、ローカルの /etc/passwd ファイルを使用できます。このコマンドではまた、ネームサービスのパスワードデータベースを使ってユーザーのログイン状態を表示できます。

例 — パスワードを持たないユーザーを表示する

次の例では、パスワードを持っていないユーザー pmorph が表示されます。

```
# logins -p
pmorph          501     other           1       Polly Morph
#
```

▼ ユーザーのログインを一時的に無効にする方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. エディタを使用して、**/etc/nologin** ファイルを作成します。

   ```
   # vi /etc/nologin
   ```

3. システムの利用に関するメッセージを入力します。
4. ファイルを閉じて、保存します。

   このファイルを作成することによって、システムシャットダウンや定常的な保守の際にユーザーログインを禁止できます。nologin ファイルが存在するシステムにユーザーがログインしようとすると、このファイルの内容が表示されます。そのあとユーザーのログインは中断されます。

スーパーユーザーのログインは影響を受けません。詳細については、nologin(4) のマニュアルページを参照してください。

例 — ユーザーのログインを無効にする

この例は、システムが利用できないことをユーザーに通知する方法を示しています。

```
# vi /etc/nologin
(ここでシステムメッセージを追加する)

# cat /etc/nologin
***ログインは許可されません。***

***システムは AM12:00 まで利用できません。***
```

システムを実行レベル 0、すなわちシングルユーザーモードにすることもできます。シングルユーザーモードに変更する方法の詳細については、『*Solaris* のシステム管理 (基本編)』の「システムのシャットダウン (手順)」を参照してください。

▼ ログイン失敗操作を保存する方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **/var/adm** ディレクトリに **loginlog** ファイルを作成します。

   ```
   # touch /var/adm/loginlog
   ```

3. **loginlog** ファイルに **root** の読み取り権と書き込み権を設定します。

   ```
   # chmod 600 /var/adm/loginlog
   ```

4. **loginlog** ファイルのグループのメンバーシップを **sys** に変更します。

   ```
   # chgrp sys /var/adm/loginlog
   ```

5. 間違ったパスワードを使用してシステムに **5** 回ログインしようとしたログが記録されていることを確認します。次に、**/var/adm/loginlog** ファイルを表示します。

   ```
   # more /var/adm/loginlog
   rimmer:/dev/pts/1:Wed Jan 16 09:22:31 2002
   rimmer:/dev/pts/1:Wed Jan 16 09:22:39 2002
   rimmer:/dev/pts/1:Wed Jan 16 09:22:45 2002
   rimmer:/dev/pts/1:Wed Jan 16 09:22:53 2002
   rimmer:/dev/pts/1:Wed Jan 16 09:23:01 2002
   #
   ```

   loginlog ファイルには、失敗操作ごとに 1 つずつエントリが入っています。各エントリには、ユーザーのログイン名、tty デバイス、操作の失敗回数が入っています。4 回以下の失敗であれば、ログに記録されません。

   loginlog ファイルは急激に大きくなることがあります。このファイルを適切に使用できるように保つためには、その内容をときどき確認して消去する必要があり

ます。loginlog ファイルに多数の操作が記録されている場合は、コンピュータシステムにだれかが侵入しようとした可能性があります。詳細は、loginlog(4) のマニュアルページを参照してください。

▼ ダイヤルアップパスワードを作成する方法

注意 – 最初にダイヤルアップパスワードを設定するときには、少なくとも 1 つのポートにログインしたまま別のポートでパスワードをテストしてください。ログアウトして新しいパスワードをテストすると、元どおりログインできなくなることがあります。まだ別のポートにログインしていれば、元に戻ってミスを訂正できます。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. シリアルデバイスのリストが入った **/etc/dialups** ファイルを作成します。このファイルには、ダイヤルアップパスワードで保護されているすべてのポートを含めてください。

   /etc/dialups ファイルは次のようになります。

   ```
   /dev/term/a
   /dev/term/b
   /dev/term/c
   ```

3. ダイヤルアップパスワードを要求するログインプログラムが入った **/etc/d_passwd** ファイルを作成します。

   uucico、sh、ksh、csh など、ユーザーがログイン時に実行できるシェルプログラムを含めます。/etc/d_passwd ファイルは次のようになります。

   ```
   /usr/lib/uucp/uucico:encrypted-password:
   /usr/bin/csh:encrypted-password:
   /usr/bin/ksh:encrypted-password:
   /usr/bin/sh:encrypted-password:
   ```

   この手順のあとの方で、各ログインプログラムに暗号化されたパスワードを追加することになります。

4. **2** つのファイルの所有権を **root** に設定します。

   ```
   # chown root /etc/dialups /etc/d_passwd
   ```

5. **2** つのファイルのグループの所有権を **root** に設定します。

   ```
   # chgrp root /etc/dialups /etc/d_passwd
   ```

6. **2** つのファイルの **root** の読み取り権と書き込み権を設定します。

   ```
   # chmod 600 /etc/dialups /etc/d_passwd
   ```

7. 暗号化パスワードを作成します。

a. 一時的なユーザーを作成します。

```
# useradd username
```

b. 一時的なユーザーのパスワードを作成します。

```
# passwd username
```

c. 暗号化パスワードを取り出します。

```
# grep username /etc/shadow> username.temp
```

d. *username*`.temp` ファイルを編集します。

暗号化パスワードを除くすべてのフィールドを削除します。2 つ目のフィールドに、暗号化パスワードが入っています。

たとえば、次の行では、暗号化パスワードは `U9gp9SyA/JlSk` です。

```
temp:U9gp9SyA/JlSk:7967:::::7988:
```

e. 一時的なユーザーを削除します。

```
# userdel username
```

8. *username*`.temp` ファイルから `/etc/d_passwd` ファイルに暗号化パスワードをコピーします。

ログインシェルごとに別のパスワードを作成するか、共通のパスワードを使用できます。

9. パスワードをダイヤルアップユーザーに知らせます。

盗聴のおそれがない方法でパスワードを知らせる必要があります。

▼ ダイヤルアップログインを一時的に無効にする方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. `/etc/d_passwd` ファイルのエントリを次の **1** 行だけにします。

```
/usr/bin/sh:*:
```

パスワード暗号化のデフォルトアルゴリズムを変更する

デフォルトでは、ユーザーパスワードは crypt_unix アルゴリズムで暗号化されます。Solaris 9 12/02 リリースでは、デフォルトのパスワード暗号化アルゴリズムを変更することによって、MD5 や Blowfish など、より強力な暗号化アルゴリズムを使用することができます。ユーザーがパスワードを次に変更するときには、パスワードは、指定されているアルゴリズムによって暗号化されます。

注 – 以前の Solaris リリースが動作している環境で以下の各手順を使用することはできません。この機能は、Solaris 9 12/02 以降のリリースの Solaris オペレーティング環境が動作しているマシンでのみ有効です。

▼ パスワード暗号化のアルゴリズムを指定する方法

この手順では、ユーザーがパスワードを変更するときのデフォルト暗号化アルゴリズムとして BSD-Linux バージョンの MD5 アルゴリズムが使用されます。このアルゴリズムは、Solaris、BSD、Linux バージョンの UNIX が混在するマシンネットワークに適しています。パスワード暗号化アルゴリズムとアルゴリズム識別子の一覧は、表 2–1 を参照してください。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. 暗号化アルゴリズムの識別子を、**/etc/security/policy.conf** ファイルの **CRYPT_DEFAULT** 変数の値として指定します。

   選択についての説明をコメントとして入力できます。

```
# vi  /etc/security/policy.conf
...
CRYPT_ALGORITHMS_ALLOW=1,2a,md5
#
# Use the version of MD5 that works with Linux and BSD systems.
# Passwords previously encrypted with __unix__ will be encrypted with MD5
# when users change their passwords.
#
#CRYPT_DEFAULT=__unix__
CRYPT_DEFAULT=1
```

   この例では、アルゴリズム構成を指定することによって、最も弱いアルゴリズムである crypt_unix がパスワードの暗号化に使用されることがないようにします。crypt_unix モジュールで暗号化されているパスワードは、次回のパスワード変更から crypt_bsdmd5 で暗号化されます。

   アルゴリズムの選択を構成するための構文については、policy.conf(4) のマニュアルページを参照してください。

例 — パスワードの暗号化に Blowfish アルゴリズムを使用する

この例では、CRYPT_DEFAULT 変数の値として Blowfish アルゴリズムの識別子、すなわち 2a が指定されています。パスワードの暗号化を制御する policy.conf のエントリは次のようになります。

```
CRYPT_ALGORITHMS_ALLOW=1,2a,md5
#CRYPT_ALGORITHMS_DEPRECATE=__unix__
CRYPT_DEFAULT=2a
```

この構成は、Blowfish アルゴリズムを使用する BSD システムに対応しています。

▼ NIS+ ドメイン用の新しいパスワードアルゴリズムを指定する方法

1. パスワード暗号化アルゴリズムを **NIS+** マスター上の **/etc/security/policy.conf** ファイルに指定します。
2. 混乱をできるだけ少なくするために、**NIS+** マスターの **/etc/security/policy.conf** ファイルを **NIS+** ドメインのすべてのホストにコピーします。

NIS+ ドメインのユーザーがパスワードを変更すると、NIS+ ネームサービスは、NIS+ マスターにある/etc/security/policy.conf ファイルのアルゴリズム構成を調べます。rpc.nispasswd デーモンが動作するこの NIS+ マスターが、暗号化されたパスワードを作成します。

▼ NIS ドメイン用の新しいパスワードアルゴリズムを指定する方法

1. パスワード暗号化アルゴリズムを **NIS** クライアント上の **/etc/security/policy.conf** ファイルに指定します。
2. 変更された **/etc/security/policy.conf** ファイルを **NIS** ドメインのすべてのクライアントマシンにコピーします。
3. 混乱をできるだけ少なくするために、変更された **/etc/security/policy.conf** ファイルを **NIS** ルートサーバーとスレーブサーバーにコピーします。

NIS ドメインのユーザーがパスワードを変更すると、NIS クライアントは、/etc/security/policy.conf ファイルにある自身のローカルアルゴリズム構成を調べ、パスワードを暗号化します。

▼ LDAP ドメイン用の新しいパスワードアルゴリズムを指定する方法

適切に構成された LDAP クライアントでは、新しいパスワードアルゴリズムを使用できます。LDAP クライアントは NIS クライアントと同じように動作します。

1. パスワード暗号化アルゴリズムを **LDAP** クライアント上の **/etc/security/policy.conf** ファイルに指定します。
2. 変更された **policy.conf** ファイルを **LDAP** ドメインのすべてのクライアントマシンにコピーします。
3. クライアントの **/etc/pam.conf** ファイルが **pam_ldap** モジュールを使用していないことを確認します。

   pam_ldap.so.1 を含むエントリの前にコメント記号 (#) があることを確認します。また、pam_authtok_store.so.1 モジュールには新しい server_policy オプションを使用しないでください。

ローカルアルゴリズム構成に基づくパスワードの暗号化や、パスワードの認証は、クライアントの pam.conf ファイルの PAM エントリに従って行われます。

LDAP ドメインのユーザーがパスワードを変更すると、LDAP クライアントは、/etc/security/policy.conf ファイルにある自身のローカルアルゴリズム構成を調べ、パスワードを暗号化します。クライアントは、{crypt} タグ付きの暗号化パスワードをサーバーに送信します。このタグは、パスワードが暗号化済みであることをサーバーに知らせます。パスワードはそのままの形でサーバーに格納されます。認証時に、クライアントはこのパスワードをサーバーから取り出します。クライアントは、このパスワードと、入力されたユーザーのパスワードからクライアントが暗号化したばかりのパスワードとを比較します。

注 – LDAP サーバーでパスワードポリシー制御を使用するには、pam.conf ファイルの pam_authtok_store エントリに server_policy オプションを指定します。これによって、パスワードは、Sun ONE Directory Server の暗号化メカニズムを使って暗号化されます。この手順については、『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: DNS*、*NIS*、*LDAP* 編*)*』の「クライアントの設定 (手順)」を参照してください。

▼ サードパーティのパスワード暗号化モジュールをインストールする方法

サードパーティのパスワード暗号化アルゴリズムは、通常、ソフトウェアパッケージの一部として配布されます。したがって、pkgadd コマンドを実行すると、/etc/security/crypt.conf ファイルはベンダーのスクリプトによって 変更されるはずです。このあとで、/etc/security/policy.conf ファイルに新しいモジュールとその識別子を指定してください。

1. **pkgadd** コマンドを実行してソフトウェアを追加します。

   ソフトウェアの追加方法については、『*Solaris* のシステム管理 (基本編)』の「ソフトウェアパッケージの追加または削除」を参照してください。

2. **/etc/security/crypt.conf** ファイルを開いて、暗号化アルゴリズムのリストに新しいモジュールとモジュール識別子が含まれていることを確認します。

   次の例は、crypt_rot13 アルゴリズムをインストールしたパッケージによって変更された crypt.conf ファイルです。

```
# crypt.conf
#
md5 /usr/lib/security/$ISA/crypt_md5.so
rot13 /usr/lib/security/$ISA/crypt_rot13.so

# For *BSD - Linux compatibility
# 1 is MD5,  2a is Blowfish
1 /usr/lib/security/$ISA/crypt_bsdmd5.so
2a /usr/lib/security/$ISA/crypt_bsdbf.so
```

3. **/etc/security/policy.conf** ファイルに、新たにインストールされたアルゴリズムの識別子を追加します。

   次に、識別子 rot13 を追加する必要がある policy.conf ファイルの一部を示します。

```
# Copyright 1999-2002 Sun Microsystems, Inc.  All rights reserved.
# ...
#ident  "@(#)policy.conf        1.6     02/06/07 SMI"
# ...
# crypt(3c) Algorithms Configuration
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,rot13
#CRYPT_ALGORITHMS_DEPRECATE=__unix__
CRYPT_DEFAULT=md5
```

この例では、現在のパスワードが crypt_rot13 アルゴリズムで暗号化されていれば、rot13 アルゴリズムが使用されます。新しいユーザーパスワードは crypt_sunmd5 アルゴリズムで暗号化されます。このアルゴリズム構成は Solaris だけからなるネットワークで有効です。

スーパーユーザーの監視と制限

スーパーユーザーのアカウントを使用する代わりに、役割によるアクセス制御を設定できます。役割によるアクセス制御を RBAC と呼びます。RBAC の概要については、第 5 章を参照してください。RBAC の設定方法については、第 6 章を参照してください。

▼ su コマンドを使用するユーザーを監視する方法

sulog ファイルには、ユーザーからスーパーユーザーに切り替えたときの su コマンドの使用を含め、すべての su コマンドの使用歴が記録されます。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **/var/adm/sulog** ファイルの内容を定期的に監視します。

```
# more /var/adm/sulog
SU 12/20 16:26 + pts/0 nathan-root
SU 12/21 10:59 + pts/0 nathan-root
SU 01/12 11:11 + pts/0 root-janedoe
SU 01/12 14:56 + pts/0 pmorph-root
SU 01/12 14:57 + pts/0 pmorph-root
```

ここには、次のような情報が表示されます。

- コマンドが入力された日時
- コマンドの成否

  + は成功を表します。- は失敗を表します。
- コマンドが実行されたポート
- ユーザー名と切り替えたユーザー ID

このファイルへの su ログの記録は、デフォルトで、 /etc/default/su ファイルの次のエントリで有効になっています。

```
SULOG=/var/adm/sulog
```

▼ コンソールへのスーパーユーザー (root) アクセス操作を表示する方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **/etc/default/su** ファイルを編集します。
3. 次の行のコメントを解除します。

```
CONSOLE=/dev/console
```

4. **su** コマンドを使ってスーパーユーザー になります。

   システムコンソールにメッセージが出力されるか確認します。

   この方法では、現在作業中のシステムでスーパーユーザーのアクセス権を取得しようとする人をただちに検出できます。

▼ スーパーユーザー (root) でのリモートログインを防ぐ方法

注 – Solaris リリースをインストールすると、デフォルトで、スーパーユーザーログインはコンソールに限定されます。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **/etc/default/login** ファイルを編集します。
3. 次の行のコメントを解除します。

   ```
   CONSOLE=/dev/console
   ```

   スーパーユーザーアクセスをコンソールに限定しておくと、コンソールからしかスーパーユーザーとしてシステムにログインできません。このシステムにリモートログインするユーザーは、まず自分のユーザーログインを使用してログインする必要があります。自分のユーザー名でログインしたあとに、su コマンドを使ってスーパーユーザーになります。
4. このシステムにスーパーユーザーとしてリモートログインして、操作が失敗することを検証してください。

ハードウェアのセキュリティの適用

マシンの起動時にパスワードの入力を必須にすることによって物理マシンを保護することができます。さらに、ユーザーがアボートシーケンスを使ってウィンドウシステムを離れるのを防ぐことによってマシンを保護する方法もあります。

▼ ハードウェアアクセスのパスワードを必須にする方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. 端末から **PROM** セキュリティモードに入ります。次の行を入力します。

   ```
   # eeprom security-mode=command

   Changing PROM password:
       New password: password
       Retype new password: password
   ```

値として command か full を選択します。詳細については、eeprom(1M) のマニュアルページを参照してください。

3. **PROM** パスワードの入力を要求されない場合は、システムがすでに **PROM** パスワードを持っています。**PROM** パスワードを変更する場合は、次のコマンドを実行します。

```
# eeprom security-password=<Return キーを押す>
Changing PROM password:
New password: password
Retype new password: password
```

新しい PROM セキュリティモードとパスワードはただちに有効になりますが、それがわかるのは、ほとんどの場合、次回の起動時です。

注意 – PROM パスワードを忘れないでください。このパスワードがないと、ハードウェアは使用できません。

▼ システムのアボートシーケンスを無効または有効にする方法

マシンのアボートシーケンスを無効にするには、次の手順を使用します。デフォルトのシステム動作では、システムのアボートシーケンスは有効になっています。

一部のサーバーシステムにはキースイッチがあります。このスイッチを安全な位置に設定すると、ソフトウェアキーボードのアボート設定が無効になります。そのため、次の手順で行なった変更が実装されないことがあります。

1. スーパーユーザーになるか、同等の役割を引き受けます。

2. **KEYBOARD_ABORT** の値を **disable** に変更します。

   /etc/default/kbd ファイルの enable 行をコメントにします。次に disable 行を追加します。

```
# vi /etc/default/kbd
...
# KEYBOARD_ABORT affects the default behavior of the keyboard abort
# sequence, see kbd(1) for details.  The default value is "enable".
# The optional value is "disable".  Any other value is ignored.
...
#KEYBOARD_ABORT=enable
KEYBOARD_ABORT=disable
```

3. キーボードのデフォルトを更新します。

```
# kbd -i
```

第 4 章

ファイルのセキュリティの適用 (手順)

この章では、Solaris 環境のファイルのセキュリティを設定する手順について説明します。

この章で説明する手順は次のとおりです。

ファイルのセキュリティに関する機能

この節では、ファイルのセキュリティを構成する機能について説明します。

ユーザークラス

各ファイルには、セキュリティのレベルを指定する 3 つのユーザークラスがあります。

- ユーザー – ファイルまたはディレクトリの所有者。通常はファイルを作成したユーザー。ファイルの所有者は、ファイルの読み取り権、書き込み権 (変更する権利)、または実行権 (コマンドの場合) を与えるユーザーを決定できる
- グループ – グループのメンバー
- その他 –ファイルまたはグループの所有者以外のすべてのユーザー

ファイルのアクセス権を割り当てたり変更したりできるのは、そのファイルの所有者かスーパーユーザーだけです。

ファイルのアクセス権

次の表に、各ユーザークラスに与えることができるファイルのアクセス権を示します。

表 **4–1** ファイルのアクセス権

記号	アクセス権	説明
r	読み取り	ファイルの内容を開いて読み込むことができる
w	書き込み	ファイルに対して書き込み (内容の変更)、追加、または削除を行うことができる
x	実行	ファイル (プログラムまたはシェルスクリプトの場合) を実行できる。または、exec(2) システムコールのいずれかを使用してプログラムを実行できる
-	拒否	ファイルの読み込み、書き込み、または実行を行うことができない

これらのファイルアクセス権は、通常のファイルと同様にデバイス、ソケット、名前付きパイプ (FIFO) などの特殊ファイルにも適用できます。

シンボリックリンクには、そのリンクが指すファイルのアクセス権が適用されます。

ディレクトリのアクセス権

次の表に、各ユーザークラスに与えることができるディレクトリのアクセス権を示します。

表 **4–2** ディレクトリのアクセス権

記号	アクセス権	説明
r	読み取り	ディレクトリ内のファイルを一覧表示できる

表 **4–2** ディレクトリのアクセス権　*(続き)*

記号	アクセス権	説明
w	書き込み	ディレクトリに対してファイルまたはリンクの追加または削除を行うことができる
x	実行	ディレクトリ内のファイルを開いたり、実行したりできる。また、ディレクトリを作成し、その下にサブディレクトリを作成できる

ディレクトリとそのサブディレクトリ内のファイルを保護するには、ファイルのアクセス権を制限して、そのディレクトリに対するアクセスを拒否します。ただし、スーパーユーザーはシステム上のすべてのファイルとディレクトリにアクセスできます。

特殊なファイルアクセス権 (setuid、setgid、スティッキビット)

実行可能ファイルと公開ディレクトリには、3 種類の特殊なアクセス権を設定できます。これらのアクセス権を設定すると、その実行可能ファイルを実行するユーザーは、そのファイルの所有者 (またはグループ) のユーザー ID を持つことができます。

特殊なアクセス権はセキュリティ上の問題を引き起こすため、設定するときは十分な注意が必要です。たとえば、スーパーユーザー権限を取得するには、ユーザー ID (UID) を root に設定してプログラムを実行します。また、すべてのユーザーは、所有するファイルに対して特殊なアクセス権を設定できるため、これもセキュリティ上の問題の原因となります。

setuid や setgid アクセス権を使用して、不正にスーパーユーザー権限が取得されていないかどうかシステムを監視する必要があります。これらのアクセス権を使用しているファイルをすべて検索して表示する方法については、74 ページの「setuid アクセス権が設定されているファイルを検索する方法」を参照してください。このようなプログラムの所有権が、root や bin ではなく、一般ユーザーになっているものが疑わしいと考えられます。

setuid アクセス権

ユーザー ID 設定 (setuid) のアクセス権を実行可能ファイルに設定すると、このファイルを実行するプロセスには、その実行可能ファイルを実行しているユーザーではなく、ファイルの所有者 (通常は root) に基づいてアクセス権が与えられます。この特殊なアクセス権を使用すると、通常は所有者しか利用できないファイルやディレクトリにアクセスできます。たとえば次に示すように、passwd コマンドは root の setuid アクセス権が設定されているので、ユーザーは root ID の権限でパスワードを変更できます。

```
-r-sr-sr-x   3 root     sys        104580 Sep 16 12:02 /usr/bin/passwd
```

この特殊なアクセス権は、プロセスの実行が終了したあとでも、高度な知識のあるユーザーは setuid プロセスによって与えられたアクセス権を維持する手段を見つけることができるため、セキュリティ上の危険が存在します。

注 – プログラムから予約済み UID (0 – 99) で setuid アクセス権を使用しても、実効 UID は正しく設定されない場合があります。シェルスクリプトを代わりに使用するか、setuid アクセス権では予約済み UID を使用しないようにしてください。

---

setgid アクセス権

グループ ID 設定 (setgid) のアクセス権は setuid に似ていますが、プロセスの実効グループ ID (GID) はファイルのグループ所有者に変更され、ユーザーにはそのグループに与えられたアクセス権に基づくアクセス権が与えられます。/usr/bin/mail コマンドには次のように setgid アクセス権が設定されています。

```
-r-x--s--x   1 root     mail       63628 Sep 16 12:01 /usr/bin/mail
```

setgid アクセス権がディレクトリに適用されると、このディレクトリ内で作成されたファイルは、生成するプロセスが所属するグループではなく、ディレクトリが所属するグループに含まれることになります。ディレクトリに対する書き込み権および実行権を持つユーザーは、そのディレクトリにファイルを作成できます。ただし、作成したファイルの所有権は、ユーザーのグループではなく、ディレクトリを所有するグループに割り当てられます。

setuid や setgid アクセス権を使用して、不正にスーパーユーザー権限が取得されていないかどうかシステムを監視する必要があります。これらのアクセス権を使用しているファイルをすべて検索して表示する方法については、74 ページの「setuid アクセス権が設定されているファイルを検索する方法」を参照してください。このようなプログラムのグループ所有権が、root や bin ではなく、一般ユーザーになっているものが疑わしいと考えられます。

スティッキビット

「スティッキビット」は、ディレクトリ内のファイルを保護するアクセス権ビットです。ディレクトリにスティッキビットが設定されている場合、そのファイルを削除できるのはその所有者、ディレクトリの所有者、またはスーパーユーザーだけです。この特殊なアクセス権により、ユーザーは /tmp などの公開ディレクトリから他のユーザーのファイルを削除できなくなります。

```
drwxrwxrwt 7  root  sys   400 Sep  3 13:37 tmp
```

TMPFS ファイルシステム上で公開ディレクトリを設定するときには、スティッキビットを手動で設定してください。

デフォルトの umask 設定

ファイルやディレクトリを作成するときには、デフォルトのアクセス権が設定されます。デフォルトのアクセス権は、/etc/profile ファイル、またはユーザーの .cshrc、 .login ファイル内の umask 設定によって決定されます。デフォルトでは、システムはテキストファイルのアクセス権を 666 に設定してユーザー、グループ、その他のユーザーに読み取り権と書き込み権を与え、ディレクトリまたは実行可能ファイルに対しては 777 を設定します。

umask コマンドによって割り当てられる値は、デフォルトから差し引かれます。この処理には、chmod コマンドでアクセス権を与えるのと同じ方法でアクセス権を拒否する効果があります。たとえば、chmod 022 コマンドはグループとその他のユーザーに書き込み権を与えますが、umask 022 コマンドはグループとその他のユーザーの書き込み権を拒否します。

次の表に、典型的な umask の設定とその設定が実行可能ファイルに与える影響を示します。

表 **4–3** 各セキュリティレベルの umask 設定

セキュリティレベル	**umask** 設定	許可されないアクセス権
緩やか (744)	022	グループとその他のユーザーによる w
中程度 (740)	027	グループによる w、その他のユーザーによる rwx
中程度 (741)	026	グループによる w、その他のユーザーによる rw
厳しい (700)	077	グループとその他のユーザーによる rwx

umask 値の設定の詳細については、umask(1) のマニュアルページを参照してください。

ファイル情報の表示

この節では、ファイルの情報を表示する方法について説明します。

▼ ファイル情報を表示する方法

ls コマンドを使用して、ディレクトリ内のすべてのファイルに関する情報を表示します。

- 次のコマンドを入力すると、現在のディレクトリ内のすべてのファイルの一覧が長形式で表示されます。

```
% ls -la
```

`-l`	ユーザーとグループの所有権およびファイルのアクセス権などを長形式で表示する
`-a`	ドット (.) で始まる隠しファイルを含め、すべてのファイルを表示する

ファイルに関する次の情報が各行に表示されます。

- ファイル形式
  ファイルには 7 つの形式があります。次の表にファイル形式の一覧を示します。

  表 **4–4** ファイル形式

記号	形式
`-`	テキストまたはプログラム
`D`	ドア
`d`	ディレクトリ
`b`	ブロック型特殊ファイル
`c`	文字型特殊ファイル
`p`	名前付きパイプ (FIFO)
`l`	シンボリックリンク
`s`	ソケット

- アクセス権 (37 ページの「パス変数の設定」 と 表 4–2 を参照)
- ハードリンク数
- ファイルの所有者
- ファイルのグループ
- ファイルのバイト数
- ファイルの作成日または最終変更日
- ファイル名

例 — ファイル情報を表示する

次の例では、`/sbin` ディレクトリ内のファイルを部分的に表示しています。

```
% cd /sbin
% ls -la
```

```
total 13456
drwxr-xr-x   2 root     sys          512 Sep  1 14:11 .
drwxr-xr-x  29 root     root        1024 Sep  1 15:40 ..
-r-xr-xr-x   1 root     bin       218188 Aug 18 15:17 autopush
lrwxrwxrwx   1 root     root          21 Sep  1 14:11 bpgetfile -> ...
-r-xr-xr-x   1 root     bin       505556 Aug 20 13:24 dhcpagent
-r-xr-xr-x   1 root     bin       456064 Aug 20 13:25 dhcpinfo
-r-xr-xr-x   1 root     bin       272360 Aug 18 15:19 fdisk
-r-xr-xr-x   1 root     bin       824728 Aug 20 13:29 hostconfig
-r-xr-xr-x   1 root     bin       603528 Aug 20 13:21 ifconfig
-r-xr-xr-x   1 root     sys       556008 Aug 20 13:21 init
-r-xr-xr-x   2 root     root      274020 Aug 18 15:28 jsh
-r-xr-xr-x   1 root     bin       238736 Aug 21 19:46 mount
-r-xr-xr-x   1 root     sys         7696 Aug 18 15:20 mountall
   .
   .
   .
```

ファイルの所有権の変更

この節では、ファイルの所有権とグループ所有権の変更方法について説明します。

デフォルトでは、所有者は chown コマンドを使用して、ファイルやディレクトリの所有者を変更できません。ただし、次の行をシステムの /etc/system ファイルに追加して、システムをリブートすると、所有者は chown コマンドを使用できるようになります。

```
set rstchown = 0
```

詳細は、chown(1) のマニュアルページを参照してください。

またデフォルトでは、所有者は chgrp コマンドを使用しても、ファイルのグループをその所有者が属するグループ以外には変更できません。たとえば、ファイルの所有者が staff と sysadm グループだけに属する場合、所有者は、ファイルのグループを staff または sysadm グループ以外には変更できません。

ただし、次の行をシステムの /etc/system ファイルに追加して、システムをリブートすると、所有者は、ファイルのグループを、所有者が属していないグループにも変更できるようになります。

```
set rstchown = 0
```

詳細は、chgrp(1) のマニュアルページを参照してください。

また、NFS マウントされているファイルシステム上で所有権およびグループを変更するときは、他にも制約があることに注意してください。

▼ ファイルの所有者を変更する方法

次の手順でファイルの所有権を変更します。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **chown** コマンドを使用してファイルの所有者を変更します。

   # **chown** *new-owner filename*

new-owner	ファイルまたはディレクトリの新しい所有者のユーザー名または UID を指定する
filename	ファイルまたはディレクトリを指定する

3. ファイルの所有者が変更されていることを確認します。

   $ **ls -l** *filename*

例 — ファイルの所有者を変更する

次の例では、myfile の所有権をユーザー rimmer に変更します。

```
$ chown rimmer myfile
# ls -l myfile
-rw-r--r--   1 rimmer   scifi   112640 May 24 10:49 myfile
```

▼ ファイルのグループ所有権を変更する方法

次の手順を使用して、ファイルのグループ所有権を変更します。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **chgrp** コマンドを使用して、ファイルのグループ所有者を変更します。

   $ **chgrp** *group filename*

group	ファイルまたはディレクトリの新しいグループ名または GID を指定する
filename	ファイルまたはディレクトリを指定する

   グループの設定については、『*Solaris* のシステム管理 *(*基本編*)*』の「ユーザーアカウントとグループの管理 (概要)」を参照してください。

3. ファイルの所有者が変更されていることを確認します。

   # **ls -l** *filename*

例 — ファイルのグループ所有権を変更する

次の例では、myself の所有権をグループ scifi に変更します。

```
$ chgrp scifi myfile
$ ls -l myfile
-rwxrw-- 1 rimmer scifi 12985 Nov 12 16:28 myfile
```

ファイルのアクセス権の変更

chmod コマンドを使用すると、ファイルのアクセス権を変更できます。ファイルまたはディレクトリの所有者、あるいはスーパーユーザーだけがそのアクセス権を変更できます。

chmod コマンドを使用して、次のどちらかのモードでアクセス権を設定できます。

- 「絶対モード」 – ファイルのアクセス権を表す数値を使用します。これは、アクセス権を設定するときに最も一般的に使用される方法です。絶対モードを使用してアクセス権を変更するときは、3 つ 1 組のアクセス権を 8 進数で表します。
- 「記号モード」 – 英字と記号の組み合わせを使用して、アクセス権を追加または削除します。

次の表に、絶対モードでファイルのアクセス権を設定するための 8 進数値を示します。これらの数字を 3 つ組み合せて、所有者、グループ、その他のユーザーのファイルアクセス権をこの順に設定します。たとえば、値 644 は、所有者に対して読み取り権と書き込み権を設定し、グループとその他のユーザーに対しては読み取り権だけを設定します。

表 **4–5** 絶対モードによるファイルのアクセス権の設定

8 進数値	ファイルのアクセス権	設定されるアクセス権
0	---	なし
1	--x	実行権のみ
2	-w-	書き込み権のみ
3	-wx	書き込み権と実行権
4	r--	読み取り権のみ
5	r-x	読み取り権と実行権
6	rw-	読み取り権と書き込み権
7	rwx	読み取り権、書き込み権、実行権

ファイルには、絶対モードまたは記号モードで特殊なアクセス権を設定できます。ただし、絶対モードを使用してディレクトリの setuid アクセス権を設定または削除することはできません。この場合、記号モードを使用してください。絶対モードでは、3つ1組のアクセス権の左端に新しい8進数値を追加して、特殊なアクセス権を設定します。次の表に、ファイルに特殊なアクセス権を設定する8進数値を示します。

表 **4–6** 絶対モードによる特殊なアクセス権の設定

8 進数値	特殊なアクセス権の設定
1	スティッキビット
2	setgid
4	setuid

次の表に、記号モードでファイルのアクセス権を設定するための記号を示します。記号では、アクセス権を設定または変更できる対象ユーザー、実行される操作、あるいは割り当てるまたは変更するアクセス権を指定できます。

表 **4–7** 記号モードによるファイルのアクセス権の設定

記号	機能	説明
u	対象ユーザー	ユーザー (所有者)
g	対象ユーザー	グループ
o	対象ユーザー	その他のユーザー
a	対象ユーザー	すべてのユーザー
=	操作	割り当て
+	操作	追加
-	操作	削除
r	アクセス権	読み取り
w	アクセス権	書き込み
x	アクセス権	実行
l	アクセス権	強制ロック、setgid ビットはオン、グループ実行ビットはオフ
s	アクセス権	setuid または setgid ビットはオン
S	アクセス権	suid ビットはオン、ユーザー実行ビットはオフ
t	アクセス権	スティッキビットはオン、その他の実行ビットはオン
T	アクセス権	スティッキビットはオン、その他の実行ビットはオフ

機能列に <対象ユーザー> <操作> <アクセス権> の順で、ファイルまたはディレクトリのアクセス権を変更する記号を指定します。

<対象ユーザー>	アクセス権を変更する対象となるユーザーを指定する
<操作>	実行する操作を指定する
<アクセス権>	変更するアクセス権を指定する

▼ アクセス権を絶対モードで変更する方法

次の手順を使用して、アクセス権を絶対モードで変更します。

1. ファイルまたはディレクトリの所有者でない場合は、スーパーユーザーになるか、同等の役割を引き受けます。

   現在の所有者またはスーパーユーザーだけが、chmod コマンドを使用してファイルまたはディレクトリの所有者を変更できます。

2. **chmod** コマンドを使用してアクセス権を絶対モードで変更します。

   ```
   % chmod nnn filename
   ```

nnn	所有者、グループ、その他のユーザーのアクセス権をこの順序で表す 8 進数値を指定する。有効な 8 進数値については、表 4–5 を参照
filename	ファイルまたはディレクトリを指定する

   注 – chmod コマンドを使用して ACL エントリを持つファイルのグループアクセス権を変更する場合、グループアクセス権と ACL マスクの両方が新しいアクセス権に変更されます。新しい ACL マスクのアクセス権は、そのファイル上に ACL エントリを持つ追加ユーザーおよびグループのアクセス権を変更する場合があるので注意が必要です。getfacl コマンドを使用して、すべての ACL エントリに適切なアクセス権が設定されていることを確認してください。詳細については、getfacl(1) のマニュアルページを参照してください。

   ---

3. ファイルのアクセス権が変更されていることを確認します。

   ```
   % ls -l filename
   ```

例 — アクセス権を絶対モードで変更する

次の例では、公開ディレクトリのアクセス権が、744 (読み取り/書き込み/実行; 読み取り専用 ; 読み取り専用) から 755 (読み取り/書き込み/実行; 読み取り/実行; 読み取り/実行) に変更されます。

```
# ls -ld public_dir
drwxr--r--  1 ignatz   staff    6023 Aug  5 12:06 public_dir
# chmod 755 public_dir
# ls -ld public_dir
drwxr-xr-x  1 ignatz   staff    6023 Aug  5 12:06 public_dir
```

次の例では、実行可能シェルスクリプトのアクセス権が、読み取り/書き込みから、読み取り/書き込み/実行に変更されます。

```
% ls -l my_script
-rw------- 1 ignatz   staff    6023 Aug  5 12:06 my_script
% chmod 700 my_script
% ls -l my_script
-rwx------ 1 ignatz   staff    6023 Aug  5 12:06 my_script
```

▼ 特殊なアクセス権を絶対モードで変更する方法

次の手順を使用して、特殊なアクセス権を絶対モードで変更します。

1. ファイルまたはディレクトリの所有者でない場合は、スーパーユーザーになるか、同等の役割を引き受けます。

   現在の所有者またはスーパーユーザーだけが、chmod コマンドを使用してファイルまたはディレクトリの所有者を変更できます。

2. **chmod** コマンドを使用して特殊アクセス権を絶対モードで変更します。

   ```
   % chmod nnnn filename
   ```

nnnn	ファイルまたはディレクトリのアクセス権を変更する 8 進数値を指定する。一番左端の 8 進数値で、ファイルに特殊なアクセス権を設定する。特殊なアクセス権に有効な 8 進数値のリストについては、表 4–6 を参照
filename	ファイルまたはディレクトリを指定する

   注 – chmod コマンドを使用して ACL エントリを持つファイルのグループアクセス権を変更する場合、グループアクセス権と ACL マスクの両方が新しいアクセス権に変更されます。新しい ACL マスクのアクセス権は、そのファイル上に ACL エントリを持つ追加ユーザーおよびグループのアクセス権を変更する場合があるので注意が必要です。getfacl コマンドを使用して、すべての ACL エントリに適切なアクセス権が設定されていることを確認してください。詳細については、getfacl(1) のマニュアルページを参照してください。

   ---

3. ファイルのアクセス権が変更されていることを確認します。

   ```
   % ls -l filename
   ```

例 — 特殊なアクセス権を絶対モードで設定する

次の例は、dbprog ファイルに setuid のアクセス権を設定します。

```
# chmod 4555 dbprog
# ls -l dbprog
-r-sr-xr-x   1 db     staff        12095 May  6 09:29 dbprog
```

次の例では、dbprog2 ファイルに setgid のアクセス権を設定します。

```
# chmod 2551 dbprog2
# ls -l dbprog2
-r-xr-s--x   1 db     staff       24576 May  6 09:30 dbprog2
```

次の例では、public_dir ディレクトリにスティッキビットのアクセス権を設定します。

```
# chmod 1777 public_dir
# ls -ld public_dir
drwxrwxrwt   2 ignatz   staff          512 May 15 15:27 public_dir
```

▼ アクセス権を記号モードで変更する方法

次の手順を使用して、アクセス権を記号モードで変更します。

1. ファイルまたはディレクトリの所有者でない場合は、スーパーユーザーになります。

   現在の所有者またはスーパーユーザーだけが、chmod コマンドを使用してファイルまたはディレクトリの所有者を変更できます。

2. **chmod** コマンドを使用してアクセス権を記号モードで変更します。

   % chmod *who operator permission filename*

who operator permission	*who* では、アクセス権を変更するユーザーを指定し、*operator* では実行する操作を指定し、*permission* では変更後のアクセス権を指定する。有効な記号の一覧については 表 4–7 を参照
filename	ファイルまたはディレクトリを指定する

3. ファイルのアクセス権が変更されていることを確認します。

   % ls -l *filename*

例 — アクセス権を記号モードで変更する

次の例では、その他のユーザーから読み取り権を削除します。

```
% chmod o-r filea
```

次の例では、ユーザー、グループ、およびその他のユーザーに、読み取り権と実行権を追加します。

```
$ chmod a+rx fileb
```

次の例では、グループに読み取り権、書き込み権、および実行権を割り当てます。

```
$ chmod g=rwx filec
```

特殊なアクセス権の検索

プログラムの setuid や setgid アクセス権を使用して、不正にスーパーユーザー権限が取得されていないかどうかシステムを監視する必要があります。このようなプログラムの所有権が、root や bin ではなく、一般ユーザーになっているものが疑わしいと考えられます。

▼ setuid アクセス権が設定されているファイルを検索する方法

次の手順を使用して、setuid アクセス権が設定されているファイルを検索します。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **find** コマンドを使用して **setuid** アクセス権が設定されているファイルを検索します。

   # **find** *directory* **-user root -perm -4000 -exec ls -ldb {} \;>/tmp/***filename*

find *directory*	指定したディレクトリから始めて、マウントされているすべてのパスを検査する。ディレクトリとしてルート (/)、sys、bin、または mail を指定できる
-user root	root が所有するファイルを表示する
-perm -4000	アクセス権が 4000 に設定されているファイルだけを表示する
-exec ls -ldb	find コマンドの出力を ls -ldb 形式で表示する
>/tmp/*filename*	結果がこのファイルに書き込まれる

3. 結果を **/tmp/***filename* に出力する。

   # **more** /tmp/*filename*

   setuid アクセス権の詳細については、63 ページの「setuid アクセス権」を参照してください。

例 — setuid アクセス権が設定されているファイルを検索する

```
# find / -user root -perm -4000 -exec ls -ldb {} \;> /tmp/ckprm
# cat /tmp/ckprm
-r-sr-xr-x 1 root bin 38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x 1 root bin 19812 Aug 10 16:16 /usr/bin/crontab
---s--x--x 1 root sys 46040 Aug 10 15:18 /usr/bin/ct
-r-sr-xr-x 1 root sys 12092 Aug 11 01:29 /usr/lib/mv_dir
-r-sr-sr-x 1 root bin 33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-x 1 root bin 38696 Aug 10 15:55 /usr/lib/lpsched
---s--x--- 1 root rar 45376 Aug 18 15:11 /usr/rar/bin/sh
-r-sr-xr-x 1 root bin 12524 Aug 11 01:27 /usr/bin/df
-rwsr-xr-x 1 root sys 21780 Aug 11 01:27 /usr/bin/newgrp
-r-sr-sr-x 1 root sys 23000 Aug 11 01:27 /usr/bin/passwd
-r-sr-xr-x 1 root sys 23824 Aug 11 01:27 /usr/bin/su
    #
```

この出力は、rar というユーザーが /usr/bin/sh のコピーを作成し、そのアクセス権を root への setuid に設定したことを示しています。この結果、rar は /usr/rar/bin/sh を実行して特権付きユーザーになることができます。この出力を参考のために保存したい場合は、ファイルを /tmp ディレクトリ以外のファイルへ移動してください。

実行可能スタックとセキュリティ

セキュリティのバグの多くは、デフォルトで読み取り権、書き込み権、および実行権が設定された実行可能スタックで発生します。実行可能スタックには実行権が割り当てられていますが、ほとんどのプログラムは実行可能スタックがなくても正しく機能します。

noexec_user_stack 変数を使うと、スタックを実行可能として割り当てるかどうかを指定できます。この変数は、Solaris 2.6 から利用可能です。デフォルトでは、この変数は 0 に設定されるため (64 ビットアプリケーションを除く)、プログラムは ABI に準拠して動作します。この変数が 0 以外に設定された場合、システムはシステム中のすべてのプロセスのスタックに読み取り権と書き込み権のマークを付けますが、実行権のマークは付けません。

この変数が設定されている場合、プログラムがスタック上でコードを実行しようとすると SIGSEGV シグナルが送信されます。通常、このシグナルが送信されると、プログラムはコアダンプして終了します。このようなプログラムは、違反しているプログラム名、プロセス ID、およびプログラムを実行した実ユーザー ID を含む警告メッセージも生成します。次に例を示します。

```
a.out[347] attempt to execute code on stack by uid 555
```

メッセージは、syslog kern 機能が notice レベルに設定されているときに、syslog デーモンによってログに記録されます。このログへの記録は、デフォルトで syslog.conf ファイルに設定されていて、メッセージがコンソールと /var/adm/messages ファイルの両方に送信されることを意味します。詳細は、syslogd(1M) と syslog.conf(4) のマニュアルページを参照してください。

syslog メッセージは、潜在的なセキュリティの問題を調べるときに役立ちます。また、このメッセージは、この変数を設定したために正しく動作しなくなった、実行可能スタックに依存するプログラムを確認するのにも役立ちます。メッセージを記録したくない場合、システム管理者は、/etc/system ファイルで noexec_user_stack_log 変数を 0 に設定します。メッセージが記録されなくなりますが、SIGSEGV シグナルは送られるので、実行プログラムのコアダンプは起きます。

mprotect() 関数を使用すれば、プログラムのスタックが実行可能であることを明示的にマーク付けできます。詳細は、mprotect(2) のマニュアルページを参照してください。

ハードウェアの制限のため、実行可能スタックの問題を捕捉して報告する機能は、sun4m と sun4u のプラットフォームでしか利用できません。

▼ プログラムが実行可能スタックを使用できないようにする方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **/etc/system** ファイルを編集して、次の行を追加します。

   ```
   set noexec_user_stack=1
   ```

3. リブートします。

   ```
   # init 6
   ```

▼ 実行可能スタックのメッセージ記録を無効にする方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **/etc/system** ファイルを編集して、次の行を追加します。

   ```
   set noexec_user_stack_log=0
   ```

3. リブートします。

   ```
   # init 6
   ```

アクセス制御リスト (ACL) の使用

従来の UNIX ファイル保護機能は、所有者、グループ、その他のユーザーという 3 つのユーザークラスに読み取り権、書き込み権、実行権を提供します。 ACL を使用すると、所有者、所有者のグループ、その他のユーザー、特定のユーザーおよびグループのファイルアクセス権を定義でき、これらのカテゴリごとにデフォルトのアクセス権を定義できるため、ファイルのセキュリティを強化できます。

たとえば、グループ内のすべてのユーザーがファイルを読み取れるようにしたい場合は、単にそのファイルにグループの読み取り権を設定します。その場合に、そのグループ内の 1 人のユーザーだけに書き込み権を与えたいとします。標準の UNIX ではファイルセキュリティをこのように設定することはできませんが、ACL では可能です。

ACL エントリはファイルの ACL を定義する手段であり、`setfacl` コマンドにより設定します。ACL エントリは、次のようにコロンで区切ったフィールドで構成されます。

entry-type:[*uid*|*gid*]*:perms*

entry-type	ファイルのアクセス権を設定する ACL エントリの種類。たとえば、*entry-type* は `user` (ファイルの所有者) または `mask` (ACL マスク) に設定できる。ACL エントリの一覧については、表 4–8 と表 4–9 を参照
uid	ユーザー名またはユーザー ID (UID)
gid	グループ名またはグループ ID (GID)
perms	*entry-type* に設定するアクセス権を表す。*perms* は、記号文字 `rwx` または番号 (`chmod` コマンドに使用するのと同じアクセス権番号) で指定できる

次の例に、ユーザー `nathan` の読み取り権および書き込み権を設定する ACL エントリを示します。

```
user:nathan:rw-
```

注意 – ACL などの UFS ファイルシステム属性は UFS ファイルシステムだけでサポートされます。そのため、`/tmp` ディレクトリ (通常は、TMPFS ファイルシステムとしてマウントされている) で ACL エントリを持つファイルを復元またはコピーすると、その ACL エントリは失われます。UFS ファイルを一時的に格納するには、`/var/tmp` ディレクトリを使用してください。

ファイルの ACL エントリ

次の表は、ファイルに ACL を設定するときに使用する有効な ACL エントリの一覧です。最初の 3 つの ACL エントリは、基本的な UNIX のファイル保護機能を提供します。

表 **4–8** ファイルの ACL エントリ

ACL エントリ	説明
`u[ser]::`*perms*	所有者のアクセス権
`g[roup]::`*perms*	グループのアクセス権
`o[ther]:`*perms*	所有者やグループのメンバー以外のユーザーのアクセス権
`m[ask]:`*perms*	ACL マスク。マスクエントリは、ユーザー (所有者以外) とグループに許可される最大アクセス権を示す。マスクは、すべてのユーザーとグループのアクセス権を即時に変更する手段である。 たとえば、`mask:r--` マスクエントリは、ユーザーとグループが書き込み権および実行権を持っていても、読み取り権しか使用できないことを示す
`u[ser]:`*uid:perms*	特定のユーザーのアクセス権。*uid* には、ユーザー名または UID の数値を指定できる
`g[roup]:`*gid:perms*	特定のグループのアクセス権。*gid* には、グループ名または GID の数値を指定できる

ディレクトリの ACL エントリ

表 4–8 に示した ACL エントリの他に、ディレクトリにはデフォルトの ACL エントリも設定できます。デフォルトの ACL エントリを持つディレクトリ内で作成されたファイルまたはディレクトリは、デフォルトの ACL エントリと同じ ACL エントリを持つことになります。表 4–9 は、ディレクトリに使用するデフォルトの ACL エントリの一覧です。

ディレクトリ上の特定のユーザーおよびグループに対してデフォルトの ACL エントリを初めて設定するときは、所有者、グループ、その他のユーザー、および ACL マスクにもデフォルトの ACL エントリを設定する必要があります。これらのエントリは、必ず設定しなければなりません。具体的には、次の表のデフォルト ACL エントリのうち、最初の 4 つを設定する必要があります。

表 **4–9** ディレクトリのデフォルト ACL エントリ

デフォルトの **ACL** エントリ	説明
`d[efault]:u[ser]::`*perms*	所有者のデフォルトアクセス権

表 4–9 ディレクトリのデフォルト ACL エントリ　(続き)

デフォルトの **ACL** エントリ	説明
`d[efault]:g[roup]::`*perms*	グループのデフォルトアクセス権
`d[efault]:o[ther]:`*perms*	所有者やグループのメンバー以外のユーザーのデフォルトアクセス権
`d[efault]:m[ask]:`*perms*	デフォルトの ACL マスク
`d[efault]:u[ser]:`*uid:perms*	特定のユーザーのデフォルトアクセス権。*uid* には、ユーザー名または UID の数値を指定できる
`d[efault]:g[roup]:`*gid:perms*	特定のグループのデフォルトアクセス権。*gid* には、グループ名または GID の数値を指定できる

▼ ファイルの ACL を設定する方法

1. **`setfacl`** コマンドを使用してファイルの **ACL** エントリを設定します。

```
% setfacl -s user::perms,group::perms,other:perms,mask:perms,acl-entry-list  filename ...
```

`-s`	ファイルに対して ACL を設定する。すでに ACL が設定されている場合、新しい ACL に置き換える。このオプションには、少なくとも所有者、グループ、およびその他のユーザーのエントリを指定する必要がある
`user::`*perms*	所有者のアクセス権を指定する
`group::`*perms*	グループのアクセス権を指定する
`other:`*perms*	所有者またはグループのメンバー以外のユーザーのアクセス権を指定する
`mask:`*perms*	ACL マスクのアクセス権を指定する。マスクは、ユーザー (所有者以外) とグループに許される最大アクセス権を示す
acl-entry-list	ファイルまたはディレクトリ上で特定のユーザーとグループに設定する 1 つまたは複数の ACL エントリのリスト。ディレクトリ上でデフォルトの ACL エントリを設定することもできる。有効な ACL エントリについては、表 4–8 と 表 4–9 を参照
filename ...	ACL に設定する 1 つまたは複数のファイルまたはディレクトリを空白で区切って指定する

注意 – すでにファイル上に ACL が存在する場合、`-s` オプションを指定すると、ACL 全体が新しい ACL に置き換えられます。

詳細は、`setfacl(1)` のマニュアルページを参照してください。

2. **ACL (ACL エントリ)** がファイルに設定されたことを確認します。

   `% getfacl` *filename*

   詳細は、81 ページの「ファイルに ACL が設定されているかどうかを検査する方法」を参照してください。

例 — ファイルの ACL を設定する

次の例では、ch1.doc ファイルのアクセス権を設定しています。所有者には読み取り権と書き込み権が設定され、グループには読み取り専用が設定され、その他のユーザーには何も設定されません。また、ユーザー george には、このファイルの読み取り権および書き込み権が与えられ、ACL マスクには読み取り権および書き込み権が設定されます。これは、ユーザーやグループは実行権を持たないことを意味します。

```
% setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:george:rw- ch1.doc
% ls -l
total 124
-rw-r-----+  1 nathan  sysadmin   34816 Nov 11 14:16 ch1.doc
-rw-r--r--   1 nathan  sysadmin   20167 Nov 11 14:16 ch2.doc
-rw-r--r--   1 nathan  sysadmin    8192 Nov 11 14:16 notes
% getfacl ch1.doc
# file: ch1.doc
# owner: nathan
# group: sysadmin
user::rw-
user:george:rw-    #effective:rw-
group::r--         #effective:r--
mask:rw-
other:---
```

次の例では、ch2.doc ファイルのアクセス権を設定します。所有者には読み取り権、書き込み権、および実行権が設定され、グループには読み取り専用が設定され、その他のユーザーには何も設定されません。また、ACL マスクには読み取り権が設定されます。さらに、ユーザー george には読み取り権および書き込み権が与えられます。ただし、ACL マスクの設定により、george のアクセス権は読み取り専用です。

```
% setfacl -s u::7,g::4,o:0,m:4,u:george:7 ch2.doc
% getfacl ch2.doc
# file: ch2.doc
# owner: nathan
# group: sysadmin
user::rwx
user:george:rwx         #effective:r--
group::r--              #effective:r--
mask:r--
other:---
```

▼ ACL をコピーする方法

- **getfacl** の出力先を変更することにより、ファイルの **ACL** を他のファイルへコピーします。

```
% getfacl filename1 | setfacl -f -filename2
```

filename1	ACL のコピー元ファイルを指定する
filename2	ACL のコピー先ファイルを指定する

例 — ACL をコピーする

次の例では、ch2.doc の ACL が ch3.doc にコピーされます。

```
% getfacl ch2.doc | setfacl -f - ch3.doc
```

▼ ファイルに ACL が設定されているかどうかを検査する方法

- **ls** コマンドを使用して、ファイルに **ACL** が設定されているかどうかを検査します。

  ```
  % ls -l filename
  ```

  filename には、ファイルまたはディレクトリを指定します。

出力のモードフィールドの右側にプラス記号 (+) が表示されているときは、そのファイルに ACL が設定されています。

注 – ユーザーやグループの ACL エントリをファイルに追加しない場合、ファイルの ACL は「弱い」とみなされ、「+」は表示されません。

例 — ファイルに ACL が設定されているかどうかを検査する

次の例の ch1.doc ファイルでは、モードフィールドの右側にプラス記号 (+) が表示されているため、ACL が設定されています。

```
% ls -l ch1.doc
-rwxr-----+  1 nathan   sysadmin      167 Nov 11 11:13 ch1.doc
```

▼ ファイルの ACL エントリを変更する方法

1. **setfacl** コマンドを使用してファイルの **ACL** エントリを変更します。

   ```
   % setfacl -m acl-entry-list  filename ...
   ```

`-m`	既存の ACL エントリを変更する
acl-entry-list	ファイルまたはディレクトリで変更する 1 つまたは複数の ACL エントリのリスト。ディレクトリのデフォルト ACL エントリを変更することもできる。有効な ACL エントリについては、表 4–8 と 表 4–9 を参照
filename ...	1 つまたは複数のファイルまたはディレクトリを空白で区切って指定する

2. ファイルの **ACL** エントリが変更されたことを確認するには、**getfacl** コマンドを使用します。

   ```
   % getfacl filename
   ```

例 — ファイルの ACL エントリを変更する

次の例では、ユーザー george のアクセス権を読み取りおよび書き込みに変更します。

```
% setfacl -m user:george:6 ch3.doc
% getfacl ch3.doc
# file: ch3.doc
# owner: nathan
# group: staff
user::rw-
user::george:rw-         #effective:r--
group::r-                #effective:r--
mask:r--
other:r-
```

次の例では、book ディレクトリのデフォルトアクセス権を変更します。グループ staff のデフォルトアクセス権を読み取りに変更し、ACL マスクのデフォルトアクセス権を読み取りおよび書き込みに変更します。

```
% setfacl -m default:group:staff:4,default:mask:6 book
```

▼ ファイルから ACL エントリを削除する方法

1. **`setfacl`** コマンドを使用してファイルから **ACL** エントリを削除します。

   `%` **`setfacl -d`** *acl-entry-list filename ...*

`-d`	指定した ACL エントリを削除する
acl-entry-list	ファイルまたはディレクトリから (アクセス権を指定せずに) 削除する ACL エントリのリスト。特定のユーザーとグループの ACL エントリとデフォルトの ACL エントリ以外は削除できない。有効な ACL エントリについては、表 4–8 と 表 4–9 を参照
filename ...	1 つまたは複数のファイルまたはディレクトリを空白で区切って指定する

   `setfacl -s` を使用してファイルのすべての ACL エントリを削除してから、指定した新しい ACL エントリで置き換えることもできます。

2. ファイルから **ACL** エントリが削除されたことを確認するには、**`getfacl`** コマンドを使用します。

   `%` **`getfacl`** *filename*

例 — ファイルから ACL エントリを削除する

次の例では、`ch4.doc` ファイルからユーザー `george` を削除します。

```
% setfacl -d user:george ch4.doc
```

▼ ファイルの ACL エントリを表示する方法

- **`getfacl`** コマンドを使用してファイルの **ACL** エントリを表示します。

  `%` **`getfacl`** [**`-a`** | **`-d`**] *filename ...*

`-a`	指定したファイルまたはディレクトリのファイル名、所有者、グループ、ACL エントリを表示する
`-d`	指定したディレクトリのファイル名、所有者、グループ、デフォルトの ACL エントリを表示する
filename ...	1 つまたは複数のファイルまたはディレクトリを空白で区切って指定する

  複数のファイル名をコマンド行に指定した場合は、各 ACL エントリの間に空白行が表示されます。

例 — ファイルの ACL エントリを表示する

次の例は、ch1.doc ファイルのすべての ACL エントリを示します。ユーザーエントリとグループエントリの隣の #effective: は、ACL マスクによって変更された後のアクセス権の設定を示します。

```
% getfacl ch1.doc

# file: ch1.doc
# owner: nathan
# group: sysadmin
user::rw-
user:george:r--         #effective:r--
group::rw-              #effective:rw-
mask:rw-
other:---
```

次の例は、book ディレクトリのデフォルトの ACL エントリを示します。

```
% getfacl -d book

# file: book
# owner: nathan
# group: sysadmin
user::rwx
user:george:r-x         #effective:r-x
group::rwx              #effective:rwx
mask:rwx
other:---
default:user::rw-
default:user:george:r--
default:group::rw-
default:mask:rw-
default:other:---
```

第 5 章

役割によるアクセス制御 (概要)

この章では、役割によるアクセス制御 (RBAC) について説明します。RBAC は、セキュリティ機能の 1 つで、通常はスーパーユーザーに制限されているオペレーションのアクセス権を制御します。この章の内容は以下のとおりです。

RBAC の手順については、第 6 章を参照してください。RBAC の要素とツールに関する参照については、第 7 章を参照してください。

RBAC: スーパーユーザーモデルの置き換え

従来の UNIX システムでは、`root` ユーザー (スーパーユーザー) はすべての権限を持ちます。つまり、任意のファイルに対する読み取り権と書き込み権、すべてのプログラムの実行権、および任意のプロセスに終了シグナルを送信する権限があります。スーパーユーザーになるユーザーは、事実上、使用するサイトのファイアウォールを変更したり、監査トレールを変更したり、給与台帳などの機密レコードを読み取ったり、ネットワーク全体をシャットダウンしたりすることができます。

役割によるアクセス制御 (RBAC) は、スーパーユーザーモデルの権限をすべて与えるか、またはまったく与えないかの機能を置き換えます。RBAC では、基本的に最小限の特権以外は許可しません。つまり、そのユーザーに必要な特権だけを許可します。

RBAC を使用すると、組織はスーパーユーザーの機能を分割して、それらを「役割」と呼ばれる特殊なユーザーアカウントに割り当てることができます。役割は、ジョブ要件に応じて、特定のユーザーに割り当てます。

役割は柔軟に設定できるため、さまざまなセキュリティポリシーに対応できます。RBAC には、簡単に設定できる次の 3 つの推奨される役割が用意されています。

- **Primary Administrator –** `root` に相当する強力な役割
- **System Administrator –** Primary Administrator より権限の小さな役割。セキュリティに関係しない管理を行う。この役割のユーザーは、パスワードを設定できない
- **Operator –** 最も権限の小さな役割。バックアップ、復元、プリンタ管理などの操作を行う

これらの役割の実装は、必須ではありません。役割は、組織のセキュリティ要件に応じて設定する機能です。役割は、セキュリティ、ネットワーク、ファイアウォールの管理など、特定の目的の管理に合わせて設定できます。担当するシステムの修正権限だけを持つ管理ユーザーという役割と 1 人の強力な管理者という役割を作成することもできます。

Solaris RBAC の要素

Solaris オペレーティング環境の RBAC モデルでは、通常のユーザーとしてログインし、役割に応じて制限された管理グラフィカルツールとコマンドを実行できる役割を引き受けます。RBAC モデルでは、これらの要素を Solaris オペレーティング環境に導入します。

- 「特権付きアプリケーション」 – システム設定よりも優先して、特定のユーザー ID (UID)、グループ ID (GID)、または承認を確認するアプリケーション (88 ページの「特権付きアプリケーション」 を参照)
- 「役割」 – 割り当てられたユーザーだけが引き受けることができる特権付きアプリケーションを実行するための特殊な ID
- 「承認」 – 通常はセキュリティポリシーによって許可されない操作を実行するために役割またはユーザーに割り当てることができる (権利プロファイルに埋め込むことができる) アクセス権
- 「権利プロファイル」 – 役割またはユーザーに割り当てることができるシステムの設定より優先されるオペレーションの集合。権利プロファイルは、承認、`setuid` または `setgid` アクセス権 (セキュリティ属性と呼ばれる) と共にコマンド、およびその他の権利プロファイルで構成される

次の図では、各 RBAC 要素の動作を示します。

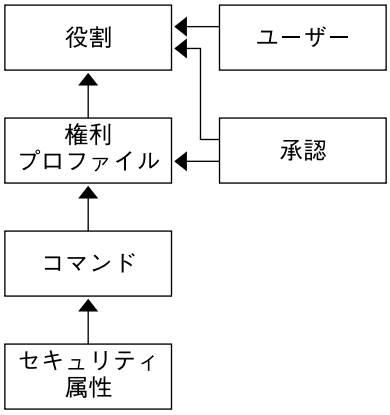


図 **5–1** Solaris RBAC 要素の動作

RBAC では、ユーザーに役割が割り当てられます。役割が使用する機能は、権利プロファイルと承認から取得します。通常は、論理的に関連付けられた権利プロファイルに対して、承認が割り当てられます。ただし、承認を役割に直接割り当てることもできます。

注 **–** 権利プロファイルと承認は、ユーザーに直接割り当てることもできます。ただし、特権が不注意に使用されると、問題が発生することがあるため、できるだけ直接割り当てないでください。

セキュリティ属性 (実 UID 、実 GID、実効 UID、実効 GID) と共にコマンドを権利プロファイルに割り当てることができます。

次の図では、Operator の役割の Printer Management 権利プロファイルを使用して、RBAC 関係の例を示します。

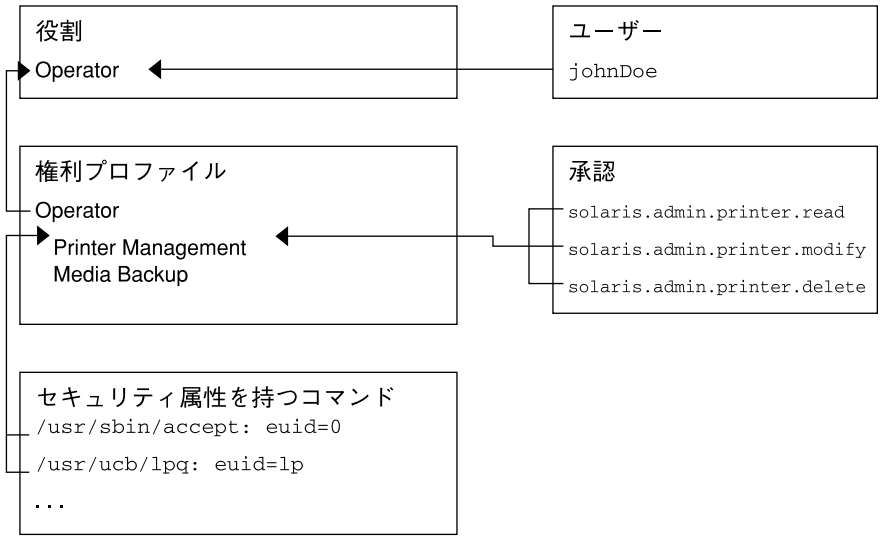


図 **5–2** Solaris RBAC 要素の関係

Operator の役割を使用して、プリンタ管理と媒体のバックアップを実行します。ユーザー `johnDoe` は、Operator の役割に割り当てられているため、「Operator」パスワードを入力してその役割を引き受けることができます。

Operator 権利ファイルは、Operator の役割に割り当てられています。Operator 権利ファイルには、「Printer Management」と「Media Backup」という補助プロファイルが割り当てられています。これらの補助プロファイルは、Operator の役割の主要な作業です。

「Printer Management」権利プロファイルは、プリンタ、印刷デーモン、およびスプーラを管理するプロファイルです。「Printer Management」権利プロファイルには、`solaris.admin.printer.read`、`solaris.admin.printer.delete`、および `solaris.admin.printer.modify` 承認が割り当てられています。これらの承認を使用することで、プリンタキューの情報を操作できます。「Printer Management」プロファイルには、`euid=0` を指定した `/usr/sbin/accept`、`euid=lp` を指定した `/usr/ucb/lpq` のように、セキュリティ属性を指定したコマンドが割り当てられます。

特権付きアプリケーション

「特権付きアプリケーション」とは、システムの設定より優先して指定できるアプリケーションです。

UID と GID を確認するアプリケーション

root またはその他の特定の UID または GID を確認する特権付きアプリケーションは、従来より UNIX のアプリケーションとして存在していました。RBAC 権利プロファイルメカニズムを使用すれば、特定のコマンドに UID または GID を指定することができます。任意のユーザーがアクセスできるコマンドの ID を変更する代わりに、権利プロファイルでセキュリティ属性を指定したコマンドとして実行することができます。その権利プロファイルを持つユーザーまたは役割であれば、root 以外でもプログラムを実行できます。

ID として、実 ID または実効 ID を指定できます。実効 ID を割り当てた場合は、実 ID より優先されます。実効 ID はアクセス権ビットの setuid 機能に相当し、監査上、UID を識別します。ただし、root の実 UID を要求するシェルスクリプトやプログラムのために、実 ID も設定できます。たとえば、pkgadd コマンドは、実効 UID ではなく実 UID を要求します。コマンドを実行するときに、指定した実効 UID では権限が十分でない場合は、「SMC Right Properties」ダイアログボックスの「Set Security Attributes」オプションを使用して特権を実 UID に変更する必要があります。112 ページの「権利プロファイルの作成または変更」 を参照してください。

承認を確認するアプリケーション

RBAC には、承認を確認するコマンドも用意されています。root にはすべての承認が定義されているため、任意のアプリケーションを実行できます。現時点では、次のアプリケーションで、承認が確認されます。

- Solaris 管理コンソールのすべてのツール
- バッチジョブ関連のコマンド – at、atq、batch、crontab
- デバイス向けのコマンド – allocate、deallocate、 list_devices、 cdrw

プロファイルシェル

承認されたユーザーは、Solaris 管理コンソール起動ツールまたは「プロファイルシェル」のコマンド行から特権付きアプリケーションを取得できます。プロファイルシェルは特別な種類のシェルで、プロファイルに割り当てられている特権付きアプリケーションにアクセスできます。プロファイルシェルは、ユーザーが su を実行して役割を引き受けたときに起動されます。プロファイルシェルには、pfsh、 pfcsh、および pfksh があります。これらはそれぞれ、Bourne シェル (sh)、C シェル (csh)、および Korn シェル (ksh) に対応します。

RBAC の役割

「役割」は、特権付きアプリケーションを実行できる特別な種類のユーザーアカウントです。役割は、ユーザーアカウントと同じ方法で作成され、ホームディレクトリ、グループ、パスワードなどを持ちます。役割は、権利プロファイルとそれに割り当てられている承認により機能します。役割には継承はありません。

ユーザーが役割を引き受けると、その役割の属性がすべてのユーザー属性を置き換えます。役割の情報は、`passwd`、`shadow`、`user_attr`、および `audit_user` データベースに格納されます。役割の設定の詳細については、121 ページの「推奨される役割の構成」、106 ページの「役割の作成」、および 110 ページの「役割プロパティの変更」を参照してください。

同じ役割になるすべてのユーザーは、同じ役割のホームディレクトリを持ち、同じ環境で動作し、同じファイルへのアクセス権を持ちます。ユーザーは、コマンド行から `su` を実行し、役割名とパスワードを指定して役割を引き受けることができます。Solaris 管理コンソールツールを開いて、役割を引き受けることもできます。

役割に直接ログインすることはできません。これは、`root` 役割を作成して、匿名による `root` ログインをできないようにするためです。詳細は、102 ページの「`root` を役割にする」を参照してください。ユーザーは、最初に通常のユーザーアカウントにログインする必要があります。ほかの役割から直接役割になることはできません。ユーザーの実 UID は常に監査されます。

Solaris 9 には、事前定義の役割は用意されていません。この章の前述のとおり、3 つの推奨される役割は簡単に構成できます。

RBAC の承認

「承認」は、役割またはユーザーに許可できる個別の権限です。RBAC に準拠したアプリケーションは、ユーザーの承認を確認してから、アプリケーションまたはアプリケーション内の特定の操作に対するアクセス権を許可します。この承認の確認は、従来の UNIX アプリケーションが行なっていた `UID=0` の確認に代わるものです。承認の詳細については、126 ページの「承認」、130 ページの「`auth_attr` データベース」、および 135 ページの「承認を必要とするコマンド」を参照してください。

RBAC の権利プロファイル

「権利プロファイル」は、役割またはユーザーに割り当てることができるシステムの設定より優先されるオペレーションの集合です。権利プロファイルには、実効 UID、実効 GID、実 UID、または実 GID を定義したコマンド、承認、その他の権利プロファイルが含まれます。権利プロファイル情報は、`prof_attr` および `exec_attr` データベースに分割して格納されます。権利プロファイルの詳細は、122 ページの「権利プロファイルの内容」、132 ページの「`prof_attr` データベース」、および 133 ページの「`exec_attr` データベース」を参照してください。

ネームサービスの適用範囲

ネームサービスの適用範囲は、RBAC を理解する上で重要な概念です。役割の適用範囲は、個別のホスト、または ネームサービス (NIS、NIS+、LDAP など) を使用するすべてのホストに適用されます。ローカルな構成ファイルとネームサービスにより配布された構成情報の優先順位は、`/etc/nsswitch.conf` ファイルに指定されています。検索は、最初に一致した時点で停止します。たとえば、プロファイルが 2 つの適用範囲に存在する場合は、最初の適用範囲に含まれるエントリだけが使用されます。

第 6 章

役割によるアクセス制御 (手順)

この章では、RBAC 要素を管理するための作業手順について説明します。次に、この章で説明する作業マップを示しています。RBAC の初期設定の作業手順については、94 ページの「RBAC の構成 (作業マップ)」を参照してください。RBAC 要素の管理全般については、第 5 章を参照してください。

この章の内容は次のとおりです。

RBAC に関連する作業は、Solaris 管理コンソールを使用して実行することをお勧めします。RBAC 要素を管理するためのコンソールツールは、ユーザーツールコレクションに含まれています。

ローカルファイルの操作は、Solaris 管理コンソールやほかのコマンド行インタフェースを使用して行うこともできます。Solaris 管理コンソールコマンドを使用してサーバーに接続するときは、認証が要求されます。このため、Solaris 管理コンソールコマンドは、スクリプトでは使用できません。その他のコマンドは、スーパーユーザーまたは同等の役割になることを必要とし、ネームサービスのデータベースには適用できません。

ヒント – また、コマンド行を使用して RBAC 情報を管理した場合は、編集がただちに有効にならないことがあります。編集を有効にするには、ネームサービスキャッシュデーモン (`nscd(1M)`) を停止して再起動する必要があります。

RBAC の構成 (作業マップ)

作業	説明	参照先
1. RBAC を計画する	RBAC の概念を理解し、お使いのサイトのセキュリティ要件を調査して、RBAC を運用に統合する方法を計画する	94 ページの「RBAC の実装を計画する方法」
2. Solaris 管理コンソールからユーザーツールを起動する	すべての RBAC 操作は、ユーザーツールで実行できる	96 ページの「ユーザーツールコレクションを実行する方法」
3. 必要に応じて初期ユーザーをインストールする	最初の役割に割り当てる 1 人または複数の既存のユーザーが必要である	98 ページの「ユーザーアカウントツールを使用して初期ユーザーを作成する方法」
4. 最初の役割をインストールする	最初の役割 (通常は Primary Administrator) は、`root` ユーザーがインストールする必要がある	96 ページの「ユーザーツールコレクションを実行する方法」
5. (省略可能) `root` を役割にする	匿名の `root` ログインを防ぐために、`root` を役割にできる	103 ページの「`root` を役割にする方法」

RBAC の計画

RBAC は、組織の情報資源を管理するときに、重要な役割を果たします。RBAC を計画する際には、RBAC の機能と組織のセキュリティ要件を十分に理解しておく必要があります。

▼ RBAC の実装を計画する方法

1. **RBAC** の基本概念を理解します。

   第 5 章を参照してください。RBAC を使用したシステム管理は、従来の UNIX を使用した場合と大きく異なります。実装を開始する前に、RBAC の概念を理解する

必要があります。詳細は、第 7 章を参照してください。

2. セキュリティポリシーを調査します。

   組織のセキュリティポリシーには、システムに対する潜在的な脅威を詳細に記述し、各脅威の危険性の分析結果に応じて適切な対応策を定義する必要があります。RBAC を使用したセキュリティ関連の作業とは切り離して行うことをお勧めします。推奨される役割とその構成をそのままインストールすることもできますが、セキュリティポリシーによっては RBAC の構成のカスタマイズが必要になることがあります。

3. 組織に必要な **RBAC** を決定します。

   組織のセキュリティ要件に応じて、さまざまなレベルの RBAC を使用できます。

   - 「RBAC を使用しない」 – すべての作業を `root` ユーザーとして実行できる。この例では、通常のユーザーとしてログインし、コンソールツールを選択するときに、ユーザーとして `root` を入力する
   - 「役割として `root` を使用する」 – この方式では、匿名の `root` ログインを防ぐために、すべてのユーザーが `root` としてログインできないようにする。代わりに、通常のユーザーとしてログインしてから、`root` 役割を引き受ける (102 ページの「`root` を役割にする」を参照)
   - 「役割を 1 つだけ使用する」 – この方式では、Primary Administrator 役割だけを追加する。スーパーユーザーモデルと似た方式である
   - 「推奨される役割」 – 容易に構成可能な 3 つの推奨される役割として、Primary Administrator、System Administrator、および Operator を利用できる。 さまざまな責任レベルの管理者の業務が、推奨される役割と一致している組織の場合は、この方式が適している
   - 「カスタム役割」 – 独自の役割を作成して、組織のセキュリティ要件を満たすことができる。新しい役割は、既存またはカスタマイズした権利プロファイルに基づいて作成できる

4. 組織に適した推奨される役割を決定します。

   推奨される役割とそのデフォルトの権利プロファイルの機能を確認します。推奨される役割を構成するときは、次の 3 つの権利プロファイルを使用できます。

   - 「Primary Administrator」権利プロファイル – すべての管理タスクを実行できる役割用。ほかのユーザーに権限を与えたり、管理役割に関連付けられた権限を編集したりする。この役割のユーザーは、Primary Administrator 役割を割り当てたり、他のユーザーに権利を与えたりすることができる
   - 「System Administrator」権利プロファイル – セキュリティに関係しない管理タスクを実行できる役割用。たとえば、System Administrator は、新しいユーザーアカウントは追加できるが、パスワードを設定したりほかのユーザーに権利を与えたりすることはできない
   - 「Operator」権利プロファイル – バックアップと復元、プリンタ管理など、単純な管理タスクを実行できる役割用

   これらの権利プロファイルを利用すると、システム管理者は 1 つの権利プロファイルを使って推奨される役割を構成することができます。

これらの権利プロファイルを詳細に検証するには、権限ツールを使用して内容を表示します。標準的な権利プロファイルの概要については、122 ページの「権利プロファイルの内容」も参照してください。コンソールツールを使用すると、これらの役割と権利プロファイルをカスタマイズして、組織の要件を満たすことができます。

5. 追加する任意の役割または権利プロファイルが組織に適切であるかどうかを判断します。

   使用するサイトで、アクセスを制限する必要があるアプリケーションを調べます。セキュリティに影響するアプリケーション、サービス拒否が発生する可能性のあるアプリケーション、特別な管理者教育を必要とするアプリケーションには、RBAC を適用することをお勧めします。

   a. 新しい操作に必要なコマンドを決定します。

   b. この操作に適切な権利プロファイルを決定します。

      既存の権利プロファイルがこの操作に割り当てられていないか、または別の権利プロファイルを作成する必要がないかどうかを確認します。

   c. この権利プロファイルに適した役割を決定します。

      この操作の権利プロファイルを既存の役割に割り当てるかどうか、または新しい役割を作成するかどうかを決定します。既存の役割を使用する場合は、この役割を割り当てるユーザーにほかの権利プロファイルが適していないかどうかを確認します。

6. 役割に割り当てるユーザーを決定します。

   必要な権限だけを割り当てるために、ユーザーの信頼レベルに応じて役割を割り当てます。ユーザーが使用しない操作の権限は割り当てないようにすると、問題が発生する可能性が減少します。

ユーザーツールコレクションを初めて使用する

初期ユーザーをインストールして役割を割り当てるには、最初に通常のユーザーとしてログインします。Solaris 管理コンソールにユーザー自身を認証させるときは、`root` ユーザーを指定します。

▼ ユーザーツールコレクションを実行する方法

1. 通常のユーザーとしてログインし、**Solaris** 管理コンソールを起動します。

```
% whoami
johnDoe
% /usr/sadm/bin/smc&
```

2. 「ユーザーツールコレクション **(User Tool Collection)**」に移動して、アイコンをクリックします。

   a. ナビゲーション区画の 「管理ツール **(Management Tool)**」で、「このコンピュータ **(This Computer)**」というラベルのアイコンを見つけます。

   b. その左にある切り替えアイコンをクリックします。

      切り替えアイコンは、レバーに似ています。レバーが水平の場合、フォルダの内容は表示されていません。レバーが垂直の場合、内容が表示されています。切り替えアイコンをクリックすると、フォルダの表示と非表示が切り替わります。

   c. 「**System Configuration**」フォルダの横にある切り替えアイコンをクリックして、その内容を表示します。

   d. 「ユーザー **(User)**」アイコンをクリックして、「ユーザーツールコレクション **(User Tool Collection)**」を開きます。

      ユーザーログインダイアログボックスが表示されます。

3. 「ログイン**:** ユーザー名 **(Login:User Name)**」ダイアログボックスに、**root** と **root** パスワードを入力します。 「了解 **(OK)**」をクリックします。

   通常は、ここでユーザー名を入力して役割を引き受けます。しかし、最初は役割が存在しないため、root ユーザーを入力する必要があります。「ユーザーツールコレクション (User Tool Collection)」が開きます (次の図を参照)。

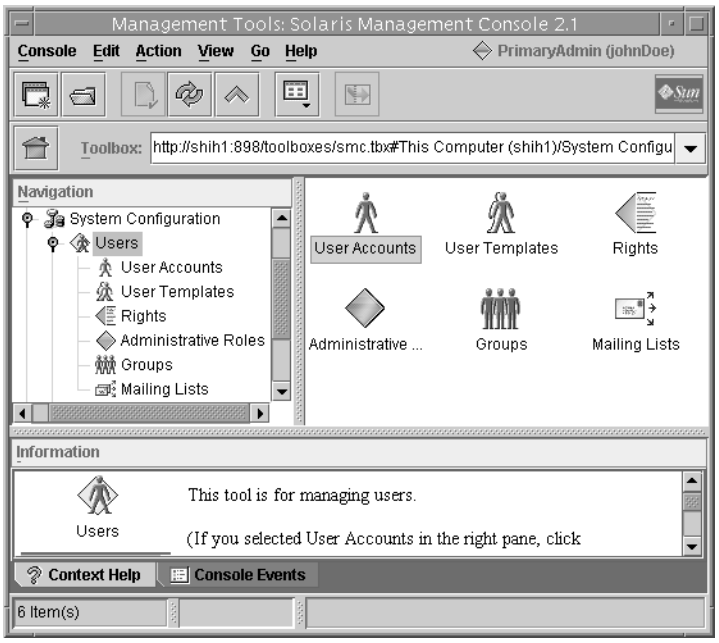


図 **6–1** 「ユーザーツールコレクション (User Tool Collection)」

初期ユーザーの設定

すべてのユーザーが役割に割り当てられ、すでにシステムにインストールされている場合は、この手順を省略して 100 ページの「初期役割の設定」に進みます。

▼ ユーザーアカウントツールを使用して初期ユーザーを作成する方法

1. ナビゲーション区画または「ユーザーツールコレクション **(User Tool Collection)**」の表示区画で、ユーザーアカウントツールのアイコンをクリックします。

   ユーザーアカウントツールが起動します。「アクション (Action)」メニューにこのツールのオプションが表示されます。

2. 「アクション **(Action)**」メニューから「ユーザーを追加 **(Add User)**」→「ウィザードを使用 **(With Wizard)**」を選択します。

「ユーザーを追加 (Add User)」ウィザードが起動します。このウィザードは、ユーザーの構成に必要な情報を要求するダイアログボックスの集合です。「次へ (Next)」および「戻る (Back)」ボタンを使用して、ダイアログボックスを移動します。「次へ (Next)」ボタンは、すべての必須フィールドに入力するまで有効になりません。最後に、入力したデータを確認するダイアログボックスが表示されます。前のダイアログボックスに戻って入力を変更するか、「完了 (Finish)」をクリックして新しい役割を保存します。

次の図は、最初に表示される「手順 1: ユーザー名を入力します。(Step 1:Enter a user name)」ダイアログボックスです。

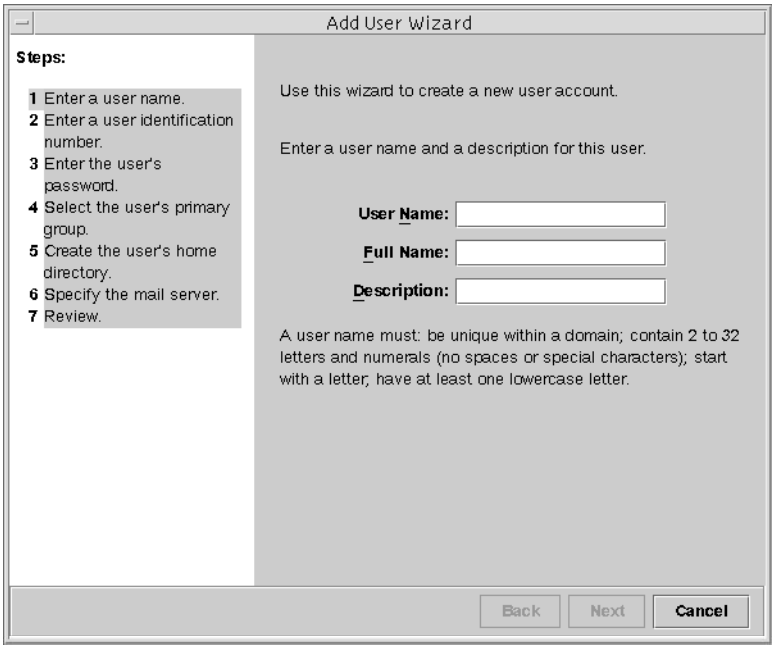


図 **6–2** 「ユーザーを追加 (Add User)」ウィザード

3. 最初のユーザー名とその他の識別情報を入力します。

4. 「手順 **2:** ユーザー識別番号を入力します。**(Step 2: Enter a User Identification Number)**」ダイアログボックスで **UID** を入力します。

   このエントリは、ユーザーの既存の UID と一致している必要があります。

5. 「手順 **3:** ユーザーのパスワードを入力します。**(Step 3: Enter the User's Password)**」ダイアログボックスで、パスワードを設定するかどうかを指定します。

   ユーザー自身のパスワードをこのアカウントに設定する場合は、2 番目のオプションをクリックします。この場合、ユーザー自身のパスワードを入力し、確認を行います。

6. 「手順 **4:** ユーザーの一次グループを選択します。**(Step 4: Select the User's Primary Group)**」ダイアログボックスで、適切なグループを選択します。

7. 「手順 **5:** ユーザーのホームディレクトリを作成します。**(Step 5: Create the User's Home Directory)**」ダイアログボックスで、ホームディレクトリのパスを指定します。

8. 「手順 **6:** メールサーバーを指定します。**(Step 6: Specify the Mail Server)**」ダイアログボックスで、デフォルトのメールサーバーとメールボックスを確認します。

   これらの設定は、「ユーザープロパティ (User Properties)」ダイアログボックスであとで変更することができます。

9. 「確認します。**(Review)**」ダイアログボックスで情報を確認します。保存する場合は、「完了 **(Finish)**」をクリックします。情報を再入力する場合は、「戻る **(Back)**」をクリックします。

   情報が不足していたり、間違っていた場合は、「戻る (Back)」ボタンを繰り返しクリックして、不正な情報を入力したダイアログボックスを表示します。次に、「次へ (Next)」を繰り返しクリックして、「確認します。(Review)」ダイアログボックスに戻ります。

初期役割の設定

最初に、ユーザーと役割を管理する役割を作成します。通常は、Primary Administrator です。まず、ローカルホストにそのユーザーと役割をインストールします。ネームサービススコープのツールボックスを設定した場合は、そのネームサービスに同じユーザーと役割を作成する必要があります。『*Solaris* のシステム管理 *(*基本編*)*』の「ネームサービス環境で Solaris 管理ツールを使用する (作業マップ)」を参照してください。最初の役割を設定して、ユーザー自身に割り当てたあと、`root` になる代わりに、役割を引き受けてコンソールツールを実行することができます。

▼ 管理役割ツールを使用して最初の役割 (Primary Administrator) を作成する方法

最初の役割をインストールするには、通常のユーザーとしてログインする必要があります。Solaris 管理コンソールにユーザー自身を認証させるには、`root` ユーザーを指定します。まず、ローカルホストに役割をインストールします。最初の役割を設定して、ユーザー自身に割り当てたら、`root` としてではなく、設定した役割でコンソールツールを実行することができます。

1. 「ログイン**:** ユーザー名 **(Login:User Name)**」ダイアログボックスに、**`root`** と **`root`** パスワードを入力します。「了解 **(OK)**」をクリックします。

2. ナビゲーション区画または「ユーザーツールコレクション **(User Tool Collection)**」の表示区画で、「管理役割 **(Administrative Roles)**」アイコンをクリックします。

   管理役割ツールが起動します。「アクション (Action)」メニューにこのツールのオプションが表示されます。

3. 「アクション **(Action)**」メニューから「管理役割を追加 **(Add Administrative)**」を選択します。

   「管理役割を追加 (Add Administrative)」ウィザードが起動します。このウィザードは、役割の構成に必要な情報を要求するダイアログボックスの集合です。「次へ (Next)」および「戻る (Back)」ボタンを使用して、ダイアログボックスを移動します。「次へ (Next)」ボタンは、すべての必須フィールドに入力するまで有効になりません。最後に、入力したデータを確認するダイアログボックスが表示されます。前のダイアログボックスに戻って入力を変更するか、「完了 (Finish)」をクリックして新しい役割を保存します。

   次の図は、最初に表示される「手順 1: 役割名を入力します。(Step 1:Enter a Role Name)」ダイアログボックスです。

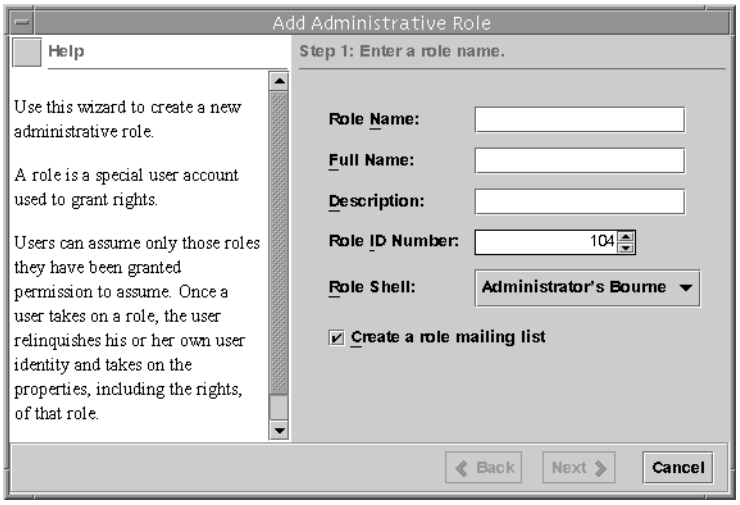


図 **6–3** 「管理役割を追加 (Add Administrative)」ウィザード

4. **`primaryadmin`** または使用している役割名と、その他の識別情報を入力します。

   役割メーリングリストオプションを選択すると、この役割を持つユーザーの別名を作成できます。

5. 「手順 **2:** 役割パスワードを入力します。**(Step 2: Enter a Role Password)**」ダイアログボックスの「役割パスワード **(Role Password)**」フィールドに新しい役割のパスワードを入力し、「パスワードを確認 **(Confirm Password)**」フィールドに再度入力します。

入力を確認することにより、スペルミスのパスワードが保存されるのを防ぐことができます。

6. 「手順 **3:** 役割権利を選択します。**(Step 3: Enter Role Rights)**」ダイアログボックスで、「**Primary Administrator**」の権利プロファイルを選択します。

   「有効な権利 (Available Rights)」列 (左側) の Primary Administrator 権利プロファイルをダブルクリックします。「許可された権利 (Granted Rights)」列 (右側) の権利プロファイルが、この役割に割り当てられた権利プロファイルです。この例では、Primary Administrator 権利プロファイルだけが必要です。

7. 「手順 **4:** ホームディレクトリを選択します。**(Step 4: Select a Home Directory)**」ダイアログボックスで、ホームディレクトリのサーバーとパスを指定します。

8. 「手順 **5:** この役割にユーザーを割り当てます。**(Step 5: Assign Users to This Role)**」ダイアログボックスで、**Primary Administrator** の役割に割り当てるユーザーのログイン名を入力します。

   追加するユーザーは、動作中のネームサービススコープに定義されている必要があります。「手順 1: 役割名を入力します。(Step 1: Enter a Role Name)」ダイアログボックスで役割のメーリングリストオプションを選択した場合、これらのユーザーは Primary Administrator 役割に送信された電子メールを受信します。

9. 「確認します。**(Review)**」ダイアログボックスで情報を確認します。保存する場合は、「完了 **(Finish)**」をクリックします。情報を再入力する場合は、「戻る **(Back)**」をクリックします。

   情報が不足していたり、間違っていた場合は、「戻る (Back)」ボタンを繰り返しクリックして、不正な情報を入力したダイアログボックスを表示します。次に、「次へ (Next)」を繰り返しクリックして、「確認します。(Review)」ダイアログボックスに戻ります。

10. 端末ウィンドウを開いてスーパーユーザーになり、ネームサービスキャッシュデーモンを起動して停止します。

    新しい役割は、ネームサービスキャッシュデーモンを再起動するまで有効になりません。スーパーユーザーで、次のように入力します。

```
# /etc/init.d/nscd stop
# /etc/init.d/nscd start
```

root を役割にする

この手順では、ローカルな有効範囲内の root ユーザーを役割にします。root を役割にすると、そのサーバーに root ユーザーとして直接ログインすることを防ぐことができます。ユーザーはまず、通常のユーザーとしてログインする必要があり、その UID は監査できるようになります。

注意 – root を役割にするときに、root を有効なユーザー、または root に相当する現在の既存の役割に割り当てなかった場合は、すべてのユーザーが root になることができなくなります。

▼ root を役割にする方法

1. 対象のサーバーにログインします。
2. スーパーユーザーになります。
3. **/etc/user_attr** ファイルを編集します。
   次に、標準的な user_attr ファイルの一部を示します。

   ```
   root::::type=normal;auths=solaris.*,solaris.grant;profiles=All
   johnDoe::::type=normal
   ```

4. このファイルにユーザー名があることを確認します。
5. ユーザー自身のレコードに割り当てられている役割に対して、**root** を追加します。
   root の役割を、任意の適用可能なユーザーに割り当てます。primaryadmin を最も強力な役割として使用する場合は、root を任意のユーザーに割り当てる必要はありません。

   ```
   johnDoe::::type=normal;roles=root
   ```

6. このファイルの **root** レコードに移動し、**type=normal** を **type=root** に変更します。

   ```
   root::::type=role;auths=solaris.*,solaris.grant;profiles=All
   ```

7. ファイルを保存します。

RBAC 情報の管理 (作業マップ)

次の作業マップでは、特定の RBAC 操作に必要な情報の入手先を示します。

作業	説明	参照先
特権付きアプリケーションの使用	セキュリティまたはシステム運用に影響するアプリケーションを実行するには、スーパーユーザーになるか、または同等の役割を引き受ける必要がある	105 ページの「コンソールツールで役割を引き受ける方法」 105 ページの「コマンド行で役割を引き受ける方法」
役割の作成	新しい役割 (特権付きアプリケーションを実行するための特殊な ID) を追加する	106 ページの「管理役割ツールを使用して役割を作成する方法」 108 ページの「コマンド行から役割を作成する方法」
役割プロパティの変更	役割 (割り当てられたユーザー、権利プロファイル、および役割に割り当てられた承認) のプロパティを変更する	110 ページの「管理役割ツールを使用して役割を変更する方法」 112 ページの「コマンド行から役割を変更する方法」
権利プロファイルの作成または変更	承認、セキュリティ属性を持つコマンド、および補助権利プロファイルの割り当てなどの権利プロファイルを追加または変更する	112 ページの「権利ツールを使用して権利プロファイルを作成または変更する方法」 116 ページの「コマンド行から権利プロファイルを変更する方法」
ユーザーの RBAC プロパティの変更	ユーザーに割り当てられた役割、権利プロファイル、または承認を変更する	117 ページの「ユーザーアカウントツールを使用してユーザーの RBAC プロパティを変更する方法」 117 ページの「コマンド行からユーザーの RBAC プロパティを変更する方法」
レガシーアプリケーションのセキュリティ保護	レガシーアプリケーションの set ID アクセス権を有効にする。スクリプトでは set ID を有効にしたコマンドを使用できる。必要に応じて、レガシーアプリケーション内で承認を確認できる	118 ページの「レガシーアプリケーションにセキュリティ属性を追加する方法」 118 ページの「スクリプト内のコマンドにセキュリティ属性を追加する方法」 119 ページの「スクリプトまたはプログラム内の承認を確認する方法」

これらの手順を行なって、役割によるアクセス制御 (RBAC) で使用される要素を管理します。ユーザー管理手順については、『*Solaris* のシステム管理 *(*基本編*)*』の「ユーザーアカウントとグループの管理 (手順)」を参照してください。

特権付きアプリケーションの使用

特権付きアプリケーションを実行するには、まずスーパーユーザーになるか、または同等の役割を引き受ける必要があります。特権付きアプリケーションは通常のユーザーとして実行することもできますが、不注意に実行するとエラーが発生します。できるだけ、通常のユーザーとして実行しないでください。

▼ コマンド行で役割を引き受ける方法

1. **su** コマンドを次のように使用します。

   ```
   % su my-role
   Password: my-role-password
   #
   ```

   su だけを入力すると、スーパーユーザーになります。役割名を指定して su と入力すると、その役割を引き受けます (役割が割り当てられている場合)。適切なパスワードを入力してください。役割になると、コマンド行がその役割のプロファイルシェルに切り替わります。その役割の権利プロファイルに割り当てられたセキュリティ属性でコマンドが実行されるように、プロファイルシェルが変更されます。

2. シェルにコマンドを入力します。

   コマンドは、割り当てられたセキュリティ属性と setuid または setgid アクセス権で実行されます。

▼ コンソールツールで役割を引き受ける方法

1. **Solaris** 管理コンソールを起動します。

   次のいずれかの方式を使用します。

   - コマンド行で smc と入力する
   - 「ツール (Tools)」サブパネルの 「Solaris 管理コンソール (Solaris Management Console)」アイコンをクリックする
   - 「アプリケーションマネージャ (Application Manager)」の「Solaris 管理コンソール (Solaris Management Console)」アイコンをダブルクリックする

   すべての Solaris 管理コンソールツールでは、拡張コンテキストヘルプを使用して、各フィールドの説明を表示できます。また、「ヘルプ (Help)」メニューからさまざまなヘルプトピックにアクセスできます。コンソールを起動するときは、root としてログインしても、通常のユーザーとしてログインしてもかまいません。

2. 操作に必要なツールボックスを選択します。

適切な適用範囲のツールまたはコレクションを含むツールボックスに移動し、アイコンをクリックします。スコープには、ファイル (ローカル)、NIS、NIS+、および LDAP があります。適切なツールボックスがナビゲーション区画に表示されていない場合は、「コンソール (Console)」メニューから「ツールボックスを開く (Open Toolbox)」を選択して、関連するツールボックスを読み込みます。

3. ツールを選択します。

   使用するツールまたはコレクションに移動して、アイコンをクリックします。RBAC 要素を管理するツールは、すべて「ユーザーツールコレクション (User Tool Collection)」にあります。

4. 「ログイン: ユーザー名 **(Login: User Name)**」ダイアログボックスでユーザー自身を認証させます。

   次のいずれかを選択します。

   - ユーザー名とパスワードを入力して、通常のユーザーとして役割を引き受ける、または操作する
   - `root` および `root` パスワードを入力して、スーパーユーザーとして操作する

   役割を設定していないか、設定されている役割が必要な操作を実行できない場合は、`root` としてログインする必要があります。ユーザーが、`root` または役割が割り当てられていないユーザーとして認証された場合は、ツールがコンソールに読み込まれます。この場合は、手順 6 に進んでください。

5. 「ログイン: 役割名 **(Login: Role)**」ダイアログボックスでユーザーを認証させます。

   ダイアログボックスの「役割 (Role)」オプションメニューに、割り当てられる役割が表示されます。役割を選択して、役割のパスワードを入力します。通常のユーザーとして操作している場合は、ユーザー自身のユーザー名とパスワードを入力します。

6. 実行するツールに移動して、アイコンをクリックします。

役割の作成

役割を作成するには、Primary Administrator 権利プロファイルが割り当てられている役割になるか、`root` ユーザーとして実行する必要があります。役割の詳細は、90 ページの「RBAC の役割」と 121 ページの「推奨される役割の構成」を参照してください。

▼ 管理役割ツールを使用して役割を作成する方法

1. 管理役割ツールを起動します。

管理役割ツールを実行して、Solaris 管理コンソールを起動します (105 ページの「コンソールツールで役割を引き受ける方法」を参照)。次に、「ユーザーツールコレクション (User Tool Collection)」を開いて、「管理役割 (Administrative Roles)」アイコンをクリックします。

2. 「管理役割を追加 **(Add Administrative)**」ウィザードが起動します。

   「アクション (Action)」メニューから「管理役割を追加 (Add Administrative Role)」を選択して、「管理役割を追加 (Add Administrative)」ウィザードを起動します。

3. 表示されるダイアログボックスのフィールドに入力します。入力が完了したら、「完了 **(Finish)**」をクリックします。

   「次へ (Next)」および「戻る (Back)」ボタンを使用して、ダイアログボックスを移動します。「次へ (Next)」ボタンは、すべての必須フィールドに入力するまで有効になりません。最後に、入力したデータを確認するダイアログボックスが表示されます。前のダイアログボックスに戻って入力を変更するか、「完了 (Finish)」をクリックして新しい役割を保存します。表 6–1 に、ダイアログボックスの要約を示します。

4. 端末ウィンドウを開いてスーパーユーザーになり、ネームサービスキャッシュデーモンを起動して停止します。

   新しい役割は、ネームサービスキャッシュデーモンを再起動するまで有効になりません。スーパーユーザーで、次のように入力します。

   ```
   # /etc/init.d/nscd stop
   # /etc/init.d/nscd start
   ```

表 6–1 管理役割を追加ウィザードのダイアログボックスとフィールド

ダイアログボックス	フィールド	フィールドの説明
手順 1: 役割名を入力します (Step 1: Enter a role name)	役割名 (Role Name)	役割の短縮名
	役割の正式名 (Full Name)	正式名
	備考欄 (Description)	役割の説明
	役割 ID 番号 (Role ID Number)	役割の UID。自動的に増分する
	役割シェル (Role Shell)	役割に使用できるプロファイルシェル: Administrator の C シェル、Administrator の Bourne シェル、または Administrator の Korn シェル
	役割メーリングリストの作成 (Create a role mailing list)	この役割に割り当てられているユーザーのメーリングリストを作成する

表 **6–1** 管理役割を追加ウィザードのダイアログボックスとフィールド　(続き)

ダイアログボックス	フィールド	フィールドの説明
手順 2: 役割パスワードを入力します。(Step 2: Enter a role password)	役割パスワード (Role Password)	********
	パスワードの確認 (Confirm Password)	********
手順 3: 役割権利を選択します。(Step 3: Select role rights)	有効な権利 / 許可された権利 (Available Rights / Granted Rights)	役割の権利プロファイルの割り当てまたは削除を行う 同一のコマンドを複数回入力しても、エラーにはならない。ただし、権利プロファイルでは、同一のコマンドが複数回発生した場合、最初のコマンドに割り当てられた属性が優先され、後続の同一コマンドはすべて無視される。順番を変更するときは、上矢印または下矢印を使用する
手順 4: ホームディレクトリを選択します。(Step 4: Select a home directory)	サーバー (Server)	ホームディレクトリのサーバー
	パス (Path)	ホームディレクトリのパス
手順 5: この役割にユーザーを割り当てます。(Step 5: Assign users to this role)	追加 (Add)	この役割を引き受けるユーザーを追加する。同じスコープ内でユーザーでなければならない
	削除 (Delete)	この役割が割り当てられているユーザーを削除する

▼ コマンド行から役割を作成する方法

1. スーパーユーザーになるか、ほかの役割を作成できる役割を引き受けます。
2. 次のいずれかの役割の作成方法を選択します。
   - ローカルスコープの役割を作成する場合、`roleadd` コマンドを使用して、新しいローカル役割とその属性を指定する
   - また同じくローカルスコープの役割を作成する場合、`user_attr` ファイルを編集して、ユーザーに `type=role` を追加することもできる

     この方法は、入力ミスが発生しやすいため、緊急時以外はできるだけ使用しない
   - ネームサービスの役割を作成する場合は、`smrole` コマンドを使用して、新しい役割とその属性を指定する

このコマンドは、スーパーユーザー、またはその他の役割を作成できる役割による認証を必要とする。smrole コマンドは、すべてのネームサービスに適用でき、Solaris 管理コンソールサーバーのクライアントとして動作する

3. ネームサービスキャッシュデーモンを起動して停止します。

   新しい役割は、ネームサービスキャッシュデーモンを再起動するまで有効になりません。スーパーユーザーで次のように入力します。

   ```
   # /etc/init.d/nscd stop
   # /etc/init.d/nscd start
   ```

例 **6–1** smrole コマンドを使用してカスタムの Operator 役割を作成する

次のコマンドシーケンスは、smrole コマンドを使用して役割を作成します。この例では、新しい Operator 役割が作成され、標準の Operator 権利プロファイルと Media Restore 権利プロファイルが割り当てられます。

```
% su primaryadmin
# /usr/sadm/bin/smrole add -H myHost -- -c "Custom Operator" -n oper2 -a johnDoe \
-d /export/home/oper2 -F "Backup/Restore Operator" -p "Operator" -p "Media Restore"
Authenticating as user: primaryadmin

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::     <primaryadmin パスワードを入力する>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadmin was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::    <oper2 パスワードを入力する>

# /etc/init.d/nscd stop
# /etc/init.d/nscd start
```

新しく作成した役割およびその他の役割を表示するには、次のように smrole コマンドに list サブコマンドを指定します。

```
# /usr/sadm/bin/smrole list --
Authenticating as user: primaryadmin

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::     <primaryadmin パスワードを入力する>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadmin was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.
root                    0               Super-User
primaryadmin            100             Most powerful role
sysadmin                101             Performs non-security admin tasks
oper2                   102             Backup/Restore Operator
```

役割プロパティの変更

役割を変更するには、Primary Administrator 権利プロファイルが割り当てられている役割を引き受ける必要があります。役割が設定されていない場合は、スーパーユーザーとして「ユーザーツールコレクション (User Tool Collection)」を実行する必要があります。

▼ 管理役割ツールを使用して役割を変更する方法

1. 管理役割ツールを起動します。

   管理役割ツールを実行するには、Solaris 管理コンソールを実行する必要があります (105 ページの「コンソールツールで役割を引き受ける方法」を参照)。次に、「ユーザーツールコレクション (User Tool Collection)」を開いて、「管理役割 (Administrative Roles)」アイコンをクリックします。

   管理役割ツールが起動したら、既存の役割のアイコンが表示区画に表示されます。

2. 変更する役割をクリックして、「アクション **(Action)**」メニューから適切な項目を次のように選択します。

   - 役割に割り当てられるユーザーを変更するには、「管理役割を割り当てる (Assign Administrative Role)」を選択する

     「管理役割を割り当てる (Assign Administrative Role)」ダイアログボックスが表示されます。「管理役割を割り当てる (Assign Administrative Role)」ダイアログボックスは、「役割プロパティ (Role Properties)」ダイアログボックスに似ていますが、「ユーザー (Users)」タブだけで構成されます。現在のスコープのユーザーをこの役割に割り当てるときは、「追加 (Add)」フィールドを使用します。ユーザーに割り当てられた役割を削除するときは、「削除 (Delete)」フィールドを使用します。「了解 (OK)」をクリックして、保存します。

   - 役割に割り当てられた権利を変更するときは、「ユーザーへ権利を割り当てる (Assign Rights to Role)」を選択する

     「ユーザーへ権利を割り当てる (Assign Rights to Role)」ダイアログボックスが表示されます。「ユーザーへ権利を割り当てる (Assign Rights to Role)」ダイアログボックスは、「役割プロパティ (Role Properties)」ダイアログボックスに似ていますが、「権利 (Rights)」タブだけで構成されます。「有効な権利 (Available Right)」列と「許可された権利 (Granted Rights)」列を使用して、選択した役割に対して権利プロファイルの追加または削除を行います。「了解 (OK)」をクリックして、保存します。

   - 役割の任意のプロパティを変更するときは、「プロパティ (Properties)」 を選択するか、役割アイコンをダブルクリックする

「役割プロパティ (Role Properties)」ダイアログボックスが表示され、すべての役割プロパティにアクセスすることができます (次の図と表を参照)。タブを使用して変更する情報に移動し、必要な変更を行なってから、「了解 (OK)」をクリックして保存します。

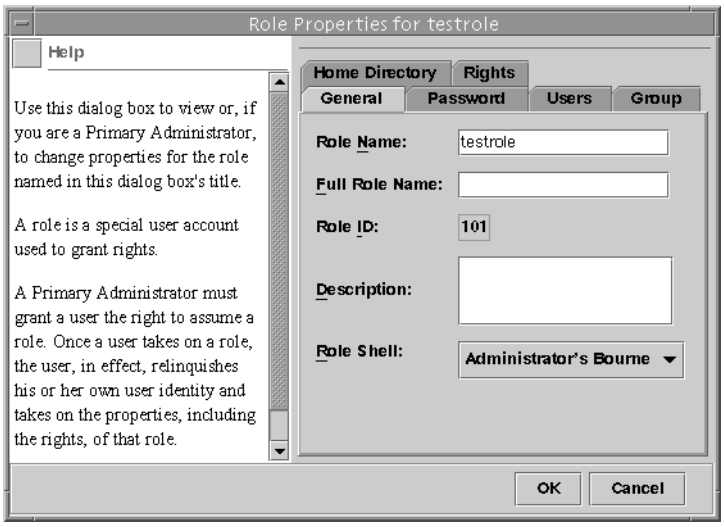


図 **6–4** 「役割プロパティ (Role Properties)」ダイアログボックス

表 **6–2** 「役割プロパティ (Role Properties)」の要約

タブ	タブの説明
基本 (General)	役割 ID 情報とデフォルトのログインシェルを指定する
パスワード (Password)	役割パスワードを指定する
ユーザー (Users)	役割に割り当てるユーザーを指定する
グループ (Group)	役割の一次グループと二次グループを設定する。ファイルとディレクトリのアクセスおよび作成に使用する
ホームディレクトリ (Home Directory)	役割のホームディレクトリ、ホームディレクトリサーバー、自動マウント、およびディレクトリアクセスを指定する
権利 (Rights)	権利プロファイルを特定の役割に割り当てることができるようにする。割り当てられた権利プロファイルの優先度は、ここで変更できる

▼ コマンド行から役割を変更する方法

1. スーパーユーザーになるか、ほかの役割を作成できる役割を引き受けます。
2. 操作に適切な次のコマンドを使用します。
   - ローカルに定義された役割の属性を変更するときは、`rolemod` コマンドを使用する
   - ローカルに定義された役割を削除するときは、`roledel` コマンドを使用する
   - ローカル役割に割り当てられた承認または権利プロファイルを変更するときは、`user_attr` ファイルを編集する

     この方法は、入力ミスが発生しやすいため、緊急時以外はできるだけ使用しない
   - ネームサービスの役割の属性を変更するときは、`smrole` コマンドを使用する

     このコマンドは、スーパーユーザーか、ほかの役割を作成できる役割としての認証を必要とする。`smrole` コマンドは、Solaris 管理コンソールサーバーのクライアントとして動作する

権利プロファイルの作成または変更

権利プロファイルを作成または変更するには、Primary Administrator 権利プロファイルが割り当てられている役割を引き受ける必要があります。役割が設定されていない場合は、スーパーユーザーとしてユーザーツールコレクションを実行する必要があります。権利プロファイルの詳細は、90 ページの「RBAC の役割」と 121 ページの「推奨される役割の構成」を参照してください。

▼ 権利ツールを使用して権利プロファイルを作成または変更する方法

1. 権利ツールを起動します。

   権利ツールを実行するには、Solaris 管理コンソールを起動する必要があります (105 ページの「コンソールツールで役割を引き受ける方法」を参照)。次に、「ユーザーツールコレクション (User Tool Collection)」を開いて、「権利 (Rights)」アイコンをクリックします。

   権利ツールが起動すると、既存の権利プロファイルのアイコンが表示区画に表示されます。
2. 権利プロファイルの作成または変更に適した次の動作を選択します。
   - 新しい権利プロファイルを作成するときは、「アクション (Action)」メニューから「権利を追加 (Add Rights)」を選択する

- 既存の権利プロファイルを変更するときは、その権利プロファイルのアイコンをクリックして、「アクション (Action)」メニューから「プロパティ (Properties)」を選択するか、権利プロファイルのアイコンをダブルクリックする

どちらの場合も 「権利プロパティ (Rights Properties)」に似たダイアログボックスが表示されます。次の図に示す「権利を追加 (Add Right)」ダイアログボックスには、書き込み可能な 「名前 (Name)」フィールドがあります。標準の「権利プロパティ (Rights Properties)」ダイアログボックスの「名前 (Name)」フィールドは、読み取り専用になっています。いったん定義した権利プロファイルは、変更できないためです。

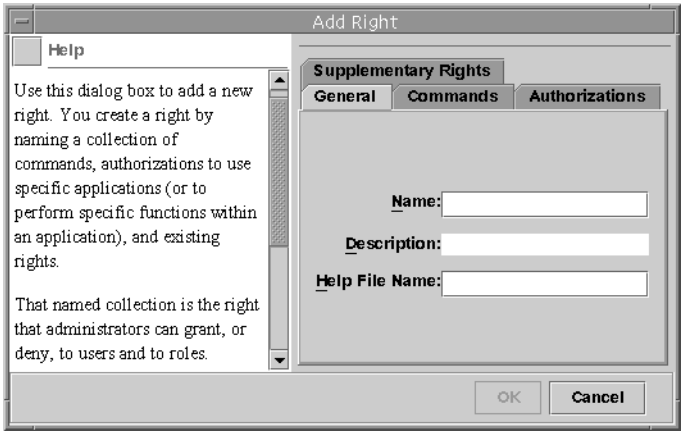


図 **6–5** 「権利を追加 (Add Right)」ダイアログボックス

3. 新しい情報を入力します。「了解 **(OK)**」をクリックして、権利プロファイルを保存します。

次の表は、「権利プロパティ (Right Properties)」ダイアログボックスのタブとフィールドの一覧です。

タブ	フィールド	フィールドの説明
基本 (General)	名前 (Name)	新しい権利プロファイル名
	備考欄 (Description)	新しい権利プロファイルの説明
	ヘルプファイル名 (Help File Name)	新しい権利プロファイルの HTML ヘルプファイル名
コマンド (Commands)	ディレクトリを追加 (Add Directory)	「拒否されたコマンド (Commands Denied)」または「許可されたコマンド (Commands Permitted)」列に存在しないディレクトリを追加するためのダイアログボックスを開く

タブ	フィールド	フィールドの説明
	拒否されたコマンド (Commands Denied) / 許可されたコマンド (Commands Permitted)	権利プロファイルのコマンドの割り当てまたは削除を行う
	セキュリティ属性を設定 (Set Security Attributes)	コマンドのセキュリティ属性 (実 UID または GID、あるいは実効 UID または GID) の割り当てまたは削除を行うダイアログボックスを開く (図 6–6 を参照) 注 – 実 ID ではなく、実効 ID を割り当てることが望ましい。実 ID は、`pkgadd` などのコマンドで必要な場合にだけ使用する
	検索 (Find (command))	2 つのコマンドリストから指定した文字列を検索する
承認 (Authorizations)	含まれない承認 (Authorizations Excluded) / 含まれる承認 (Authorizations Included)	権利プロファイルの承認の割り当てまたは削除を行う
補助権利 (Supplementary Rights)	含まれない権利 (Rights Excluded) / 含まれる権利 (Rights Included)	権利プロファイルの補助権利プロファイルの割り当てまたは削除を行う

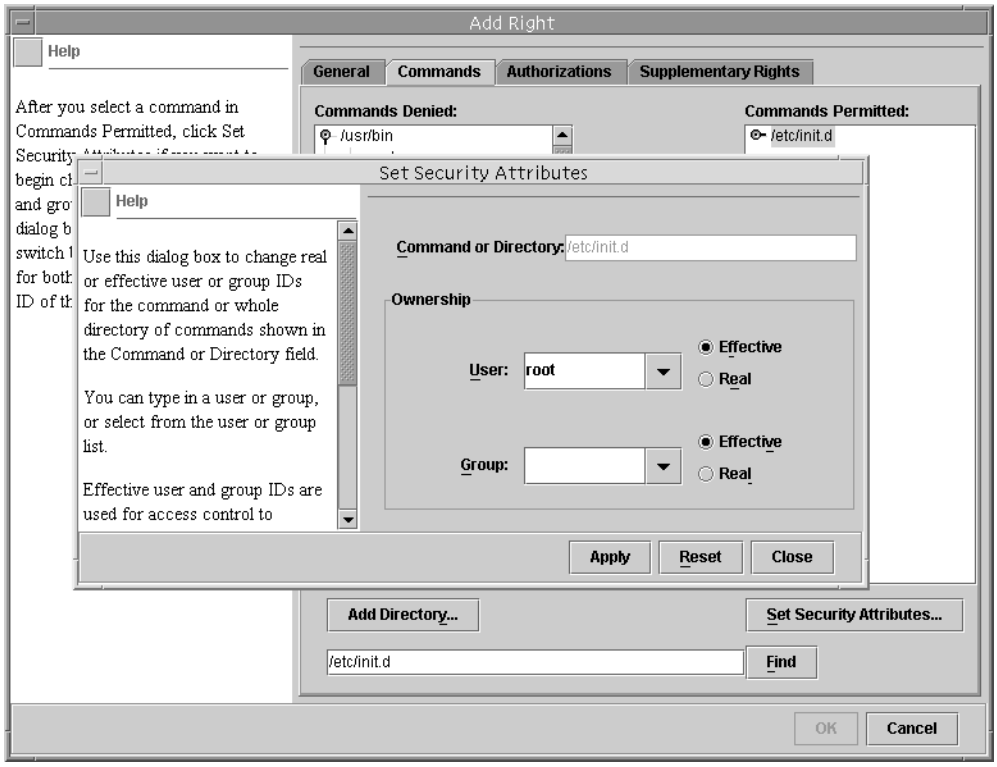


図 **6–6** セキュリティ属性をコマンドに追加する

例 **6–2** 権利ツールを使用して新しい権利プロファイルを作成する

次の表は、「Restart」と呼ばれる仮想権利プロファイルを作成するときのサンプルデータです。この例の権利プロファイル「Restart」には、サブディレクトリ `/etc/init.d` のコマンドが割り当てられています。これらのコマンドの実効 UID は 0 です。この権利プロファイルは、`/etc/init.d` 内のデーモンを停止および起動できるシステム管理者が使用します。

タブ	フィールド	例
基本 (General)	名前 (Name)	Restart
	備考欄 (Description)	`/etc/init.d` 内のデーモンを起動および停止する
	ヘルプファイル名 (Help File Name)	`Restart.html`

例 **6–2** 権利ツールを使用して新しい権利プロファイルを作成する　(続き)

タブ	フィールド	例
コマンド (Commands)	ディレクトリを追加 (Add Directory)	「ディレクトリを追加 (Add Directory)」をクリックし、ダイアログボックスに `/etc/init.d` と入力して、「了解 (OK)」をクリックする
	拒否されたコマンド (Commands Denied) / 許可されたコマンド (Commands Permitted)	`/etc/init.d` を選択し、「追加 (Add)」をクリックして、「許可されたコマンド (Commands Permitted)」列にコマンドを移動する
	セキュリティ属性を設定 (Set Security Attributes)	`/etc/init.d` を選択し、「セキュリティ属性を設定 (Set Security Attributes)」をクリックして、`Effective UID = root` を設定する (図 6–6 を参照)
	検索 (Find (command))	
承認 (Authorizations)	含まれない承認 (Authorizations Excluded) / 含まれる承認 (Authorizations Included)	
補助権利 (Supplementary Rights)	含まれない権利 (Rights Excluded) / 含まれる権利 (Rights Included)	

▼ コマンド行から権利プロファイルを変更する方法

1. スーパーユーザーになるか、**Primary Administrator** 権利プロファイルを持つ役割を引き受けます。
2. 操作に適した **`smprofile`** のサブコマンドを使用します。

   このコマンドは認証を必要とします。このコマンドは、すべてのネームサービスに適用できます。`smprofile` コマンドは、Solaris 管理コンソールサーバーのクライアントとして動作します。

   - 新しいプロファイルを追加するときは、 `smprofile` の `add` サブコマンドを使用する
   - 既存のプロファイルを変更するときは、`smprofile` の `modify` サブコマンドを使用する

ユーザーの RBAC プロパティの変更

ユーザーのプロパティを変更するには、ユーザーツールコレクションをスーパーユーザーとして実行するか、Primary Administrator 権利プロファイルが割り当てられている役割を持つ必要があります。

▼ ユーザーアカウントツールを使用してユーザーの RBAC プロパティを変更する方法

1. ユーザーアカウントツールを起動します。

   ユーザーアカウントツールを実行するには、Solaris 管理コンソールを起動する必要があります (105 ページの「コンソールツールで役割を引き受ける方法」を参照)。次に、「ユーザーツールコレクション (User Tool Collection)」を開いて、「ユーザーアカウント (User Accounts)」アイコンをクリックします。

   ユーザーアカウントツールが起動すると、既存のユーザーアカウントのアイコンが表示区画に表示されます。

2. 変更するユーザーアカウントのアイコンをクリックして、「アクション **(Action)**」メニューから「プロパティ **(Properties)**」を選択するか、ユーザーアカウントのアイコンをダブルクリックします。

3. 変更するプロパティのダイアログボックスで、適切なタブを次のように選択します。

   - ユーザーに割り当てられた役割を変更するときは、「役割 (Role)」タブをクリックして、変更する役割を「有効な役割 (Available Roles)」または「割り当てられた役割 (Assigned Roles)」列に移動します。
   - ユーザーに割り当てられた権利プロファイルを変更するときは、「権利 (Rights)」タブをクリックして、変更する権利プロファイルを「有効な権利 (Available Rights)」または「許可された権利 (Assigned Rights)」列に移動します。

   注 – 権利プロファイルは、できるだけユーザーに直接割り当てないでください。特権付きアプリケーションを実行するときは、ユーザーが役割を引き受けるようにしてください。通常のユーザーが、特権を不正に使用できないようにするためです。

▼ コマンド行からユーザーの RBAC プロパティを変更する方法

1. スーパーユーザーになるか、ユーザーファイルを変更できる役割を引き受けます。

2. 次のように適切なコマンドを使用します。

   - ローカルスコープに定義されたユーザーに割り当てられている承認、役割、または権利プロファイルを変更する場合は、`usermod` コマンドを使用する
   - また同じくローカルスコープに定義されたユーザーに割り当てられている承認、役割、または権利プロファイルを変更する場合は、`user_attr` ファイルを編集することもできる

この方法は、入力ミスが発生しやすいため、緊急時以外はできるだけ使用しない

- ネームサービスに定義されたユーザーに割り当てられている承認、役割、または権利プロファイルを変更するときは、 `smuser` コマンドを使用する

  このコマンドは、スーパーユーザー、またはユーザーファイルを変更できる役割による認証を必要とする。`smuser` コマンドは、すべてのネームサービスに適用でき、Solaris 管理コンソールサーバーのクライアントとして動作する

レガシーアプリケーションのセキュリティ保護

この節では、レガシーアプリケーションのセキュリティを強化する方法について説明します。レガシーアプリケーションを Solaris 管理コンソールに追加するときは、『*Solaris* のシステム管理 *(*基本編*)*』の「Solaris 管理コンソールにツールを追加する」を参照してください。

レガシーアプリケーションにセキュリティ属性を追加する方法

レガシーアプリケーションにセキュリティ属性を追加するときは、コマンドに追加する場合と同じ方法で追加します。「権利プロパティ (Rights property)」ダイアログボックスの「コマンド (Commands)」タブにある「拒否されたコマンド (Commands Denied)」列に、コマンドまたはそのディレクトリを追加する必要があります。次に、そのコマンドを「許可されたコマンド (Commands Permitted)」列に移動します。

スクリプト内のコマンドにセキュリティ属性を追加する方法

スクリプト内のコマンドに setUID ビットを設定して実行する場合は、セキュリティ属性を同じ権利プロファイル内のそのコマンドに追加します。112 ページの「権利ツールを使用して権利プロファイルを作成または変更する方法」を参照してください。

スクリプトまたはプログラム内の承認を確認する方法

承認を必要とするスクリプトを作成するには、auths コマンドに基づいたテストを追加する必要があります (auths(1) のマニュアルページを参照)。たとえば、次の行では、 $1 引数に指定した承認がユーザーに与えられているかどうかをテストします。

```
if [ `/usr/bin/auths|/usr/xpg4/bin/grep $1` ]; then
        echo Auth granted
else
        echo Auth denied
fi
```

さらに詳細にテストするには、ワイルドカードを使用してその他の承認を確認する論理を追加する必要があります。たとえば、solaris.admin.usermgr.write 承認がユーザーに与えられているかどうかをテストするには、solaris.admin.usermgr.write、solaris.admin.usermgr.* 、solaris.admin.*、および solaris.* という文字列を確認する必要があります。

プログラムを作成している場合は、getauthattr() 関数を使用して、承認をテストします。

第 7 章

役割によるアクセス制御 (参照)

この章では、第 5 章 の追加情報を提供します。

この章の内容は次のとおりです。

RBAC タスクについては、第 6 章を参照してください。

RBAC 要素: 参照情報

この節では、役割によるアクセス制御 (RBAC) の要素について詳細に説明します。

推奨される役割の構成

Solaris 9 には、事前定義済みの役割は組み込まれていません。顧客サイトの管理者は、設定する役割の種類を決定する必要があります。ただし、適切な事前定義済みの権利プロファイルを対応する役割に割り当てると、次の 3 つの推奨される役割を簡単に構成できます。

- 「Primary Administrator」権利プロファイル – すべての管理タスクを実行できる役割用。ほかのユーザーに権限を与えたり、管理役割に関連付けられた権限を編集したりする。この役割のユーザーは、Primary Administrator 役割を割り当てたり、他のユーザーに権利を与えたりすることができる
- 「System Administrator」権利プロファイル – セキュリティに関係しない管理タスクを実行できる役割用。たとえば、System Administrator は、新しいユーザーアカウントは追加できるが、パスワードを設定したりほかのユーザーに権利を与えたりすることはできない
- 「Operator」権利プロファイル – バックアップと復元、プリンタ管理など、単純な管理タスクを実行できる役割用

これらの権利プロファイルを利用すると、システム管理者は権利プロファイルを組み合わせたり調整したりしなくても、1 つの権利プロファイルを使って推奨される役割を構成することができます。

役割をカスタマイズするときは、役割に割り当てられた権利プロファイルの順序を詳細に確認する必要があります。同一のコマンドを複数回入力しても、エラーにはなりません。権利プロファイルでは、最初に発生したコマンドに割り当てられる属性が優先され、後続の同一コマンドはすべて無視されます。

注 – `root` は、役割として設定することもできます。`root` を役割に設定すると、ユーザーは `root` として直接ログインできなくなり、通常のユーザーとしてログインする必要があります。102 ページの「`root` を役割にする」を参照してください。

権利プロファイルの内容

この節では、いくつかの標準的な権利プロファイルについて説明します。

- 「All」権利プロファイルでは、セキュリティ属性のないコマンドへアクセスする権利が割り当てられる
- 「Primary Administrator」権利プロファイルは、Primary Administrator 役割用に設計されている。Primary Administrator 権利プロファイルでは、ワイルドカードを使用できる
- 「System Administrator」権利プロファイルは、System Administrator 役割用に設計されている。System Administrator 権利プロファイルでは、複数の補助プロファイルを組み合わせて強力な役割を作成する
- 「Operator」権利プロファイルは、Operator 役割用に設計されている。Operator 権利プロファイルでは、複数の補助プロファイルを組み合わせて単純な役割を作成する
- 「Basic Solaris User」権利プロファイルには、 `policy.conf` ファイルを使用して、セキュリティに関係しないタスクをユーザーに割り当てる
- 「Printer Management」権利プロファイルは、特定の管理領域専用のプロファイルの 1 つ

次の節の表では、コマンド、承認、補助権限、権利プロファイル、関連するヘルプファイルなど、これらの権利プロファイルの目的と内容を示します。

ヘルプファイルは HTML 形式なので、必要に応じて簡単にカスタマイズできます。ヘルプファイルは、/usr/lib/help/auths/locale/C ディレクトリにあります。

Solaris 管理コンソールの権利ツールを使用して、権利プロファイルの内容を検査することもできます。

All 権利プロファイル

All 権利プロファイルは、セキュリティ属性のないコマンドを除いたすべてのコマンドを使用できるようにワイルドカードを使用したプロファイルです。この権利プロファイルは、ほかの権利プロファイルに明示的に割り当てられていないすべてのコマンドにアクセスできる役割です。All 権利プロファイルまたはワイルドカードを使用するその他の権利プロファイルを使用しないと、役割は明示的に割り当てられているコマンド以外にはアクセスできません。これは、あまり実用的ではありません。

権利プロファイルのコマンドは、発生順に解釈されます。このため、ワイルドカードを使用する場合は、最後に指定します。明示的に割り当てた属性が、誤って優先指定されないようにするためです。All 権利プロファイルを使用する場合は、最後に割り当ててください。

表 **7–1** All 権利プロファイルの内容

目的	内容
ユーザーまたは役割として任意のコマンドを実行する	コマンド: * ヘルプファイル: RtAll.html

Primary Administrator 権利プロファイル

Primary Administrator 権利プロファイルには、システム上で最も強力な役割が割り当てられます。実質的に、スーパーユーザーの機能を持つ役割が提供されます。

- solaris.* 承認は、実質的に Solaris ソフトウェアから提供されるすべての承認を割り当てる
- solaris.grant 承認は、任意の権利プロファイル、役割、またはユーザーに任意の承認を割り当てる
- *:uid=0;gid=0 のコマンド割り当ては、UID=0 および GID=0 ですべてのコマンドを実行する

ヘルプファイル RtPriAdmin.html はサイト内では同一であるため、必要に応じて変更できます。ヘルプファイルは、 /usr/lib/help/auths/locale/C ディレクトリに格納されています。

Primary Administrator 権利プロファイルがサイトのセキュリティポリシーと矛盾する場合は、 このプロファイルを変更したり、割り当てないようにしたりすることもできます。ただし、Primary Administrator 権利プロファイルのセキュリティ機能は、ほかの権利プロファイル処理するのに必要となります。

表 **7–2** Primary Administrator 権利プロファイルの内容

目的	内容
すべての管理タスクを実行する	コマンド: *
	承認: solaris.*、solaris.grant
	ヘルプファイル: RtPriAdmin.html

System Administrator 権利プロファイル

System Administrator 権利プロファイルは、System Administrator 役割用に設計されています。System Administrator では、Primary Administrator の強力な権限を持たないため、ワイルドカードは使用できません。代わりに、セキュリティに関係しない個別の管理権利プロファイルが割り当てられます。次の表では、補助権利プロファイルに割り当てられているコマンドは説明していません。

All 権利プロファイルは、補助権利プロファイルのリストの最後にあります。

表 **7–3** System Administrator 権利プロファイルの内容

目的	内容
セキュリティに関係しない管理タスクを実行する	補助権利プロファイル: Audit Review、Printer Management、Cron Management、Device Management、File System Management、Mail Management、Maintenance and Repair、Media Backup、Media Restore、Name Service Management、Network Management、Object Access Management、Process Management、Software Installation、User Management、All
	ヘルプファイル: RtSysAdmin.html

Operator 権利プロファイル

Operator 権利プロファイルは、権限の弱い管理権利プロファイルで、バックアップとプリンタ管理を行います。ファイルの復元は、セキュリティに影響するため、デフォルトではこの権利プロファイルに割り当てられていません。

表 **7–4** Operator 権利プロファイルの内容

目的	内容
単純な管理タスクを実行する	補助権利プロファイル: Printer Management、Media Backup、All ヘルプファイル: `RtOperator.html`

ユーザー用の Basic Solaris User 権利プロファイル

デフォルトでは、Basic Solaris User 権利プロファイルは、`policy.conf` ファイルによってすべてのユーザーに自動的に割り当てられます。この権利プロファイルでは、通常の操作に使用する基本的な承認を与えます。Basic Solaris User 権利プロファイルを使用するときは、サイトのセキュリティ要件を考慮する必要があります。高いセキュリティを必要とするサイトでは、この権利プロファイルを `policy.conf` ファイルから削除することをお勧めします。

表 **7–5** Basic Solaris User 権利プロファイルの内容

目的	内容
すべてのユーザーに自動的に権限を割り当てる	承認: `solaris.profmgr.read`、 `solaris.admin.usermgr.read`、`solaris.admin.logsvc.read`、 `solaris.admin.fsmgr.read`、`solaris.admin.serialmgr.read` 、`solaris.admin.diskmgr.read`、`solaris.admin.procmgr.user` 、`solaris.compsys.read`、`solaris.admin.printer.read` 、`solaris.admin.prodreg.read`、`solaris.admin.dcmgr.read` 補助権利プロファイル: All ヘルプファイル: `RtDefault.html`

Printer Management 権利プロファイル

Printer Management は、特定のタスクを実行する標準権限ファイルです。Printer Management 権利プロファイルには、承認とコマンドが割り当てられます。次の表では、使用できるコマンドの一部を示します。

表 **7–6** Printer Management 権利プロファイルの内容

目的	内容
プリンタ、デーモン、スプール処理を管理する	承認: `solaris.admin.printer.delete`、`solaris.admin.printer.modify`、`solaris.admin.printer.read` コマンド: `/usr/sbin/accept:euid=lp`、`/usr/ucb/lpq:euid=0`、`/etc/init.d/lp:euid=0`、`/usr/bin/lpstat:euid=0`、`/usr/lib/lp/lpsched:uid=0`、`/usr/sbin/lpfilter:euid=lp` ヘルプファイル: `RtPrntMngmnt.html`

承認

「承認」とは、役割またはユーザーに許可できる個別の権限のことです。RBAC に準拠したアプリケーションによって承認が確認されてから、ユーザーはアプリケーションまたはアプリケーションの特定の操作へのアクセス権を取得します。従来の UNIX アプリケーションでは、`UID=0` で承認が確認されていました。

承認の命名規則

承認名には、RBAC の内部およびファイル内で使用される名前 (`solaris.admin.usermgr.pswd` など) と、グラフィカルユーザーインタフェースに表示される短い名前 (`Change Passwords` など) があります。

承認名の書式は、インターネット名と逆の順序になり、サプライヤ、被認証者領域、任意の下位領域、および承認の機能で構成されます。各要素の区切り文字はドット (.) です。たとえば、`com.xyzcorp.device.access` のように指定します。ただし、Sun から許可される承認では、インターネット名の代わりに接頭辞 `solaris` が使用されます。システム管理者は、ドットの右側に任意の文字列を表すワイルドカード (*) を使用して、承認を階層方式で適用することができます。

承認レベルの違いの例

ここでは、承認の使用方法の例を示します。Operator 役割のユーザーは、多くの場合、`solaris.admin.usermgr.read` 承認に制限されます。この承認では、ユーザーの構成ファイルに対する読み取り権は許可されますが、書き込み権は許可されません。System Administrator 役割では、`solaris.admin.usermgr.read` 承認だけでなく、ユーザーのファイルを変更できる `solaris.admin.usermgr.write` 承認も許可されます。ただし、System Administrator には、`solaris.admin.usermgr.pswd` 承認が許可されないため、パスワードは変更できません。Primary Administrator では、これらの 3 つの承認がすべて許可されます。

Solaris 管理コンソールのユーザーツールのパスワードを変更するには、solaris.admin.usermgr.pswd 承認が必要です。この承認は、smuser、smmultiuser、および smrole コマンドのパスワード変更オプションを使用するときにも必要になります。

承認の委託

接尾辞が grant の承認が許可されたユーザーまたは役割は、割り当てられている承認のうち同じ接頭辞を持つ任意の承認を、ほかのユーザーに委託することができます。

たとえば、solaris.admin.usermgr.grant 承認と solaris.admin.usermgr.read 承認を持つ役割は、solaris.admin.usermgr.read 承認をほかのユーザーに委託できます。solaris.admin.usermgr.grant と solaris.admin.usermgr.* を持つ役割は、solaris.admin.usermgr 接頭辞を持つ任意の承認をほかのユーザーに委託できます。

RBAC をサポートするデータベース

次の 4 つのデータベースには、RBAC 要素のデータが格納されます。

- user_attr (拡張ユーザー属性のデータベース) – ユーザーと役割を承認と権利に関連付ける
- auth_attr (承認属性のデータベース) – 承認とその属性を定義し、関連するヘルプファイルを指定する
- prof_attr (権利プロファイル属性のデータベース) – 権利プロファイルを定義し、その権利プロファイルに割り当てられた承認を指定し、関連するヘルプファイルを指定する
- exec_attr (実行属性のデータベース) – 特定の権利プロファイルに割り当てられたセキュリティ属性を持つコマンドを指定する

注 – コマンドには、セキュリティポリシーを指定することもできます。Solaris オペレーティング環境で現在利用できるセキュリティポリシーは、suser (スーパーユーザーの短縮形) だけです。suser ポリシーはデフォルトで使用され、ID 属性と承認が格納されます。Solaris 環境と相互運用できる Trusted Solaris 環境では、tsol というポリシーが使用されます。今後のリリースでは、ポリシーが追加される予定です。

RBAC の実装では、policy.conf データベースも重要です。このデータベースには、すべてのユーザーにデフォルトで適用される承認と権利プロファイルが含まれます。

RBAC データベースの関係

次の図は、RBAC データベースの相互関係を示しています。

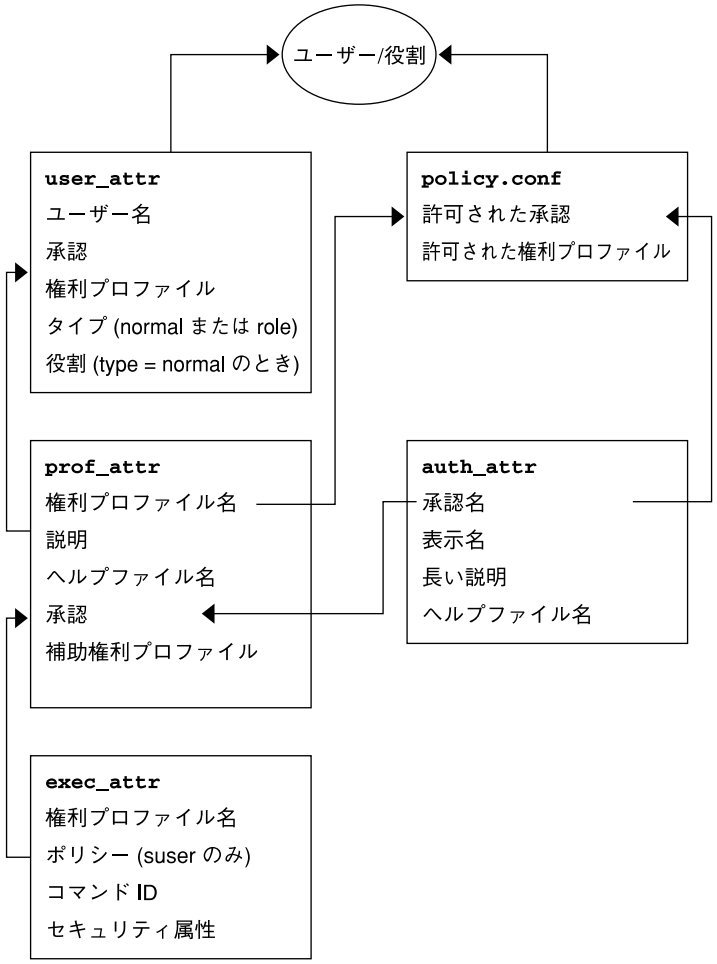


図 **7–1** RBAC データベースの関係

`user_attr` データベースには、ユーザーと役割の基本定義が格納されます。ユーザーと役割は、タイプフィールドで識別します。 `user_attr` データベースには、図に示す属性が格納されます。権利プロファイル名を、コンマで区切って指定します。権利プロファイルは、2 つのデータベースに分けて定義します。`prof_attr` データベースには、権利プロファイルの ID 情報、そのプロファイルに割り当てる承認、および補助プロファイルが格納されます。`exec_attr` データベースには、セキュリティポリシーを識別し、コマンド、およびコマンドに関連付けられたセキュリティ属

性が格納されます。auth_attr データベースには、Sun 管理コンソールツールに渡す承認情報が格納されます。policy.conf データベースには、すべてのユーザーに適用されるデフォルトの承認と権利プロファイルが格納されます。

各データベースには、*key=value* という構文を使用して、値を格納します。この方式は、データベースの拡張に対応するだけでなく、ポリシーが認識できない鍵が検出された場合にも対応できます。

RBAC データベースの適用範囲は、NIS、NIS+、LDAP などのネームサービスを使用している各ホストまたはすべてのホストに適用できます。ローカル構成ファイルと 配布された user_attr データベースの優先順位は、/etc/nsswitch.conf ファイルの passwd エントリに設定します。prof_attr データベースと auth_attr データベースの優先順位は、/etc/nsswitch.conf に個別に設定します。exec_attr データベースには、prof_attr と同じ優先順位が適用されます。たとえば、セキュリティ属性を指定したコマンドを特定のプロファイルに割り当てた場合に、そのプロファイルが 2 つの適用範囲に存在するときは、最初の適用範囲のエントリだけが使用されます。

これらのデータベースは、ローカルシステムに配置するか、NIS、NIS+、LDAP ネームサービスによって管理します。

これらのデータベースは手動編集でき、134 ページの「RBAC を管理するコマンド行アプリケーション」で説明するコマンドを使用して操作できます。

user_attr データベース

user_attr データベースには、ユーザーと役割の情報が格納されます。これらの情報は、passwd および shadow データベースによって利用されます。user_attr データベースには、承認、権利プロファイル、割り当てられた役割など、さまざまなユーザー属性が格納されます。user_attr データベースの各フィールドは次のようにコロンで区切ります。

```
user:qualifier:res1:res2:attr
```

次の表で、これらのフィールドについて説明します。

フィールド名	説明
user	passwd データベースに指定されているユーザー名または役割名
qualifier	将来の使用に予約
res1	将来の使用に予約
res2	将来の使用に予約

フィールド名	説明
attr	セミコロン (;) で区切られた、鍵と値のペアからなるリスト (省略可能)。ユーザーがコマンドを実行したときに適用されるセキュリティ属性を表す。有効な鍵は、type、auths、profiles、roles の 4 つ ■ type 鍵には、アカウントが通常ユーザーの場合は normal、役割の場合は role を設定する ■ auths 鍵には、auth_attr データベースの定義名から選択した承認名をコンマで区切って指定する。承認名には、ワイルドカードとしてアスタリスク (*) を使用できる。たとえば、solaris.device.* はすべての Solaris デバイスの承認を意味する ■ profiles 鍵には、prof_attr データベースに定義されている権利プロファイル名をコンマで区切って指定する。権利プロファイルの順序は、UNIX 検索パスと同様に動作する。実行するコマンドにどの属性が適用されるかは、そのコマンドが含まれている、リストの最初の権利プロファイルによって決まる (属性を使用する場合) ■ roles 鍵には、ユーザーに割り当てる役割名をコンマで区切って指定する。役割も同じ user_attr データベースに定義されることに注意する。役割の場合は、type 値に role が設定される。役割を他の役割に割り当てることはできない

次の例では、Operator 役割を標準的な user_attr データベースに定義して、それをユーザー johnDoe に割り当てる方法を示しています。役割とユーザーは、type キーワードによって識別されます。

```
% grep operator /etc/user_attr
johnDoe::::type=normal;roles=sysadmin,operator
operator::::profiles=Operator;type=role
```

auth_attr データベース

承認はすべて auth_attr データベースに格納されます。承認は、 user_attr データベースのユーザー (または役割) に直接割り当てることができます。承認は、ユーザーに割り当てられている権利プロファイルに割り当てることもできます。

auth_attr データベースのフィールドは次のようにコロンで区切ります。

```
authname:res1:res2:short_desc:long_desc:attr
```

次の表で、これらのフィールドについて説明します。

フィールド名	説明
authname	承認を識別する一意の文字列。書式は *prefix*.[*suffix*] 。Solaris オペレーティング環境では、承認の接頭辞として Solaris を使用する。他のすべての承認には、承認を作成する組織のインターネットドメインを逆にしたもので始まる接頭辞を使用する (たとえば、com.xyzcompany)。接尾辞は、一般には機能領域と操作、およびどのように承認されるかを示す authname が接頭辞と機能領域で構成され、ピリオドで終わるときは、実際の承認としてではなく、GUI 内でアプリケーションによって使用されるヘッダーとして機能する。たとえば、authname が solaris.printmgr. の場合、ヘッダーとして使用される authname が grant で終わるときは、認可承認として機能する。この承認を持つユーザーは、同じ接頭辞と機能領域で構成される承認をほかのユーザーに委託できる。たとえば、solaris.printmgr.grant が authname の場合は、認可承認として使用される。solaris.printmgr.grant が許可されたユーザーは、 solaris.printmgr.admin や solaris.printmgr.nobanner などの承認をほかのユーザーに委託する権利を持つ
res1	将来の使用に予約
res2	将来の使用に予約
short_desc	GUI のスクロールリストの中など、ユーザーインタフェースでの表示に適している承認の簡略名
long_desc	詳しい記述。このフィールドには、承認の目的、承認が使用されるアプリケーション、この使用に関心があるユーザーの種類などを記述する。詳しい記述は、アプリケーションのヘルプテキストに表示できる
attr	承認の属性を記述する鍵と値のペアをセミコロン (;) で区切ったリスト (オプション)。0 または 1 つ以上の鍵を指定できる。 キーワード help には HTML 形式のヘルプファイルを指定する。ヘルプファイルは、/usr/lib/help/auths/locale/C ディレクトリの index.html ファイルからアクセスできる

次の例は、標準的な値がいくつか設定された auth_attr データベースを示します。

```
% grep printer /etc/security/auth_attr
solaris.admin.printer.:::Printer Information::help=AuthPrinterHeader.html
solaris.admin.printer.delete:::Delete Printer Information::help=AuthPrinterDelete.html
solaris.admin.printer.modify:::Update Printer Information::help=AuthPrinterModify.html
solaris.admin.printer.read:::View Printer Information::help=AuthPrinterRead.html
```

solaris.admin.printer. はドット(.) で終わっているため、ヘッダーとして定義されます。ヘッダーは、承認の集合を作成するために、GUI によって使用されます。

prof_attr データベース

prof_attr データベースには、権利プロファイルに割り当てる名前、説明、ヘルプファイルの場所、および承認が格納されます。権利プロファイルに割り当てたコマンドとセキュリティ属性は、exec_attr データベースに格納されます (133 ページの「exec_attr データベース」を参照)。prof_attr データベースのフィールドは次のようにコロンで区切ります。

```
profname:res1:res2:desc:attr
```

次の表で、これらのフィールドについて説明します。

フィールド名	説明
profname	権利プロファイルの名前。権利プロファイル名では大文字と小文字が区別される。この名前は、役割とユーザーに権利プロファイルを指定するために、user_attr データベースでも使用される
res1	将来の使用に予約
res2	将来の使用に予約
desc	詳しい記述。このフィールドでは、この使用に適したユーザーの種類など、権利プロファイルの目的を説明する。詳しい記述は、アプリケーションのヘルプテキストとして適しているものである必要がある
attr	実行時にそのオブジェクトに適用するセキュリティ属性を記述する鍵と値のペアをセミコロン (;) で区切ったリスト (オプション)。0 または 1 つ以上の鍵を指定できる。有効な鍵は、help と auths の 2 つ。 キーワード help には HTML 形式のヘルプファイルを指定する。ヘルプファイルは、/usr/lib/help/auths/locale/C ディレクトリの index.html ファイルからアクセスできる キーワード auths には、auth_attr データベースから選択した承認名をコンマで区切って指定する。承認名には、ワイルドカードとしてアスタリスク (*) を使用できる

次の例では、標準的な prof_attr データベースを示します。Printer Management 権利プロファイルは、Operator 権利プロファイルに割り当てられる補助権利プロファイルです。

```
% grep 'Printer Management' /etc/security/prof_attr
Printer Management:::Manage printers, daemons, spooling:help=RtPrntAdmin.html; \
auths=solaris.admin.printer.read,solaris.admin.printer.modify,solaris.admin.printer.delete
\
Operator:::Can perform simple administrative tasks:profiles=Printer Management,\
Media Backup,All;help=RtOperator.html
...
```

exec_attr データベース

実行属性は、特定の UID または GID に関連付けられるコマンドで、権利プロファイルに割り当てられます。セキュリティ属性を指定したコマンドは、権利プロファイルが割り当てられているユーザーまたは役割が実行できます。

exec_attr データベースには、実行属性の定義が格納されます。

exec_attr データベースのフィールドは次のようにコロンで区切って指定します。

```
name:policy:type:res1:res2:id:attr
```

次の表で、これらのフィールドについて説明します。

フィールド名	説明
name	権利プロファイルの名前。権利プロファイル名では大文字と小文字が区別される。この名前は、prof_attr データベースの権利プロファイルを参照する
policy	このエントリに関連付けるセキュリティポリシー。現時点では、suser (スーパーユーザーポリシーモデル) が唯一の有効なエントリである
type	指定するエンティティの種類。現在は、cmd (コマンド) が唯一の有効なエンティティである
res1	将来の使用に予約
res2	将来の使用に予約
id	エンティティを識別する文字列。コマンドには、完全パスかワイルドカードをもつパスを指定する。引数を指定する場合は、引数をもつスクリプトを作成し、そのスクリプトを id に指定する
attr	実行時にそのエンティティに適用するセキュリティ属性を記述する鍵と値のペアをセミコロン (;) で区切ったリスト (オプション)。0 または 1 つ以上の鍵を指定できる。有効なキーワードのリストは、適用するポリシーによって異なる。有効な鍵は、euid、uid、egid、gid の 4 つである euid および uid キーワードには、1 つのユーザー名またはユーザー ID (UID) の数値が含まれる。euid を使用すると、コマンドは指定された実効 UID で動作する。これは、実行可能ファイルに setuid ビットを設定することと同じである。uid を使用すると、コマンドは指定された実 UID と実効 UID で動作する egid および gid キーワードには、1 つのグループ名またはグループ ID (GID) の数値が含まれる。egid を使用すると、コマンドは指定された実効 GID で動作する。これは、実行可能ファイルに setgid ビットを設定することと同じである。gid を使用すると、コマンドは指定された実 GID と実効 GID で動作する

次の例に、exec_attr データベースの標準的な値をいくつか示します。

```
% grep 'Printer Management' /etc/security/exec_attr
Printer Management:suser:cmd:::/usr/sbin/accept:euid=lp
Printer Management:suser:cmd:::/usr/ucb/lpq:euid=0
Printer Management:suser:cmd:::/etc/init.d/lp:euid=0
```

.
.
.

policy.conf ファイル

policy.conf ファイルには、特定の権利プロファイルと承認をすべてのユーザーに与える方法を定義します。このファイルは、次の 2 つの鍵と値のペアで構成されます。

- AUTHS_GRANTED=*authorizations* – 1 つまたは複数の承認
- PROFS_GRANTED=*right profiles* – 1 つまたは複数の権利プロファイル

次の例では、policy.conf データベースの標準的な値をいくつか示します。

```
# grep AUTHS /etc/security/policy
AUTHS_GRANTED=solaris.device.cdrw

# grep PROFS /etc/security/policy
PROFS_GRANTED=Basic Solaris User
```

RBAC コマンド

この節では、RBAC の管理に使用するコマンドを一覧します。承認を使用してアクセス権を制御できるコマンドについても説明します。

RBAC を管理するコマンド行アプリケーション

RBAC データベースを直接編集するほかに、次のコマンドを使用して RBAC のタスクへのアクセスを管理できます。

表 **7–7** RBAC 管理コマンド

コマンド名	説明
auths(1)	ユーザーに対する承認を表示する
makedbm(1M)	dbm ファイルを作成する
nscd(1M)	ネームサービスキャッシュデーモン。 user_attr、prof_attr、および exec_attr データベースをキャッシュするときに使用する

表 **7–7** RBAC 管理コマンド　(続き)

コマンド名	説明
pam_role_auth(5)	PAM 用の役割アカウント管理モジュール。役割になる承認があるかを検査する
pfexec(1)	プロファイルシェルによって使用される。exec_attr データベースに指定されている属性を使用してコマンドを実行する
policy.conf(4)	セキュリティポリシーの構成ファイル。与えられている承認を一覧表示
profiles(1)	指定したユーザーの権利プロファイルを表示する
roles(1)	ユーザーに与えられている役割を表示する
roleadd(1M)	役割をシステムに追加する
roledel(1M)	役割をシステムから削除する
rolemod(1M)	システム上の役割のプロパティを変更する
smattrpop(1M)	2 つのセキュリティ属性データベースをマージする。ローカルデータベースをネームサービスにマージするとき、および変換スクリプトを使用しないでアップグレードするときに使用する
smexec(1M)	exec_attr データベースのエントリを管理する。認証を必要とする
smmultiuser(1M)	ユーザーアカウントの一括操作を管理する。認証を必要とする
smuser(1M)	ユーザーエントリを管理する。認証を必要とする
smprofile(1M)	prof_attr および exec_attr データベースの権利プロファイルを管理する。認証を必要とする
smrole(1M)	役割アカウントの役割とユーザーを管理する。認証を必要とする
useradd(1M)	ユーザーアカウントをシステムに追加する。ユーザーのアカウントに役割を割り当てるには、-P オプションを使用する
userdel(1M)	ユーザーのログインをシステムから削除する
usermod(1M)	システム上のユーザーのアカウントプロパティを変更する

承認を必要とするコマンド

次の表では、承認を使用して Solaris 環境のコマンドオプションを制限する方法を示します。126 ページの「承認」も参照してください。

表 **7–8** コマンドおよび関連する承認

コマンド	承認の要件
at(1)	solaris.jobs.user がすべてのオプションで必要 (at.allow ファイルおよび at.deny ファイルがない場合)

表 **7–8** コマンドおよび関連する承認　(続き)

コマンド	承認の要件
atq(1)	solaris.jobs.admin がすべてのオプションで必要
crontab(1)	ジョブを送信するオプションの場合は、solaris.jobs.user が必要 (crontab.allow および crontab.deny ファイルがない場合)
	ほかのユーザーの crontab ファイルを表示または変更する場合は、solaris.jobs.admin が必要
allocate(1) (BSM が有効な場合のみ)	デバイスを割り当てる場合は、solaris.device.allocate (または、device_allocate(4) に指定されている承認) が必要
	ほかのユーザーにデバイスを割り当てる場合 (-F オプション) は、solaris.device.revoke (または、device_allocate ファイルに指定されている別承認) が必要
deallocate(1) (BSM が有効な場合のみ)	ほかのユーザーのデバイスの割り当てを解除する場合は、solaris.device.allocate (または device_allocate(1) に指定されている承認) が必要
	指定したデバイス (-F オプション) またはすべてのデバイス (-I オプション) の割り当てを強制的に解除する場合は、solaris.device.revoke (または、device_allocate に指定されている承認) が必要
list_devices(1) (BSM が有効な場合のみ)	ほかのユーザーのデバイスを表示する場合 (-U オプション) は、solaris.device.revoke が必要

第 8 章

自動セキュリティ拡張ツールの使用 (手順)

この章では、自動セキュリティ拡張ツール (ASET) を使用して、システムファイルおよびディレクトリへのアクセスを監視または制限する方法について説明します。

この章で説明する手順は次のとおりです。

自動セキュリティ拡張ツール (ASET)

Solaris 9 には、ASET が組み込まれています。ASET を使用すると、ほかの場合には手作業で実行する作業が自動的に実行され、システムのセキュリティを監視して制御できます。

ASET セキュリティパッケージには、システムのセキュリティを制御して監視できるように、自動管理ツールが組み込まれています。ASET を実行するセキュリティレベルには、低レベル、中レベル、または高レベルを指定できます。上のレベルほど、ASET のファイル制御機能が増え、ファイルアクセスは減少し、システムセキュリティが厳しくなります。

ASET には 7 つのタスクがあり、各タスクがシステムファイルに対して特定の検査と調整を行います。ASET のタスクはファイルのアクセス権を厳しくし、重要なシステムファイルの内容にセキュリティ上の弱点がないかどうかを確認して、重要な領域を監視します。ASET では、ゲートウェイシステムとして機能するシステムにファイアウォールシステムの基本要件を適用し、ネットワークも保護できます (141 ページの「ファイアウォールの設定」を参照)。

ASET は、構成用のマスターファイルを使用します。マスターファイルやレポートなどの ASET ファイルは、`/usr/aset` ディレクトリにあります。これらのファイルは、サイトの特定の要件に合わせて変更できます。

各タスクは、検出されたセキュリティ上の弱点と、タスクがシステムファイルに対して行なった変更を示すレポートを生成します。最上位のセキュリティレベルで実行すると、ASET はシステムセキュリティ上のすべての弱点を変更しようとします。潜在的なセキュリティ問題を解決できない場合、ASET は問題の存在を報告します。

ASET セッションを起動するには、`/usr/aset` コマンドを対話的に使用します。ASET を定期的に実行する場合は、`crontab` ファイルにエントリを指定します。

ASET のタスクはディスクをかなり使用するため、通常の動作の妨げになることがあります。システム性能への影響を最小限度に抑えるために、24 時間ごとまたは 48 時間ごとに深夜など、システムの稼働レベルが最も低いときに ASET を実行するようにスケジュールしてください。

ASET のセキュリティレベル

ASET は、低、中、高の 3 つのセキュリティレベルのいずれかで動作するように設定できます。上のレベルほど、ASET のファイル制御機能が増え、ファイルアクセスが減少し、システムのセキュリティが厳しくなります。これらの機能には、ユーザーによるファイルアクセスを制限せずにシステムセキュリティを監視する最低レベルから、システムが完全にセキュリティ保護される最高レベルまで、アクセス権が段階的に厳しくなります。

次の表で、この 3 つのセキュリティレベルの概要について説明します。

セキュリティレベル	説明
低	ファイルシステムの属性が標準リリース値に設定されることが保証されます。ASET は複数の検査を実行し、セキュリティ上の潜在的な弱点を報告します。低レベルでは、ASET は動作せず、システムサービスは影響を受けません。
中	ほとんどの環境で十分にセキュリティが制御されます。ASET はシステムファイルとパラメータの設定の一部を変更し、システムアクセスを制限し、セキュリティ上の攻撃による危険を減らします。ASET は、セキュリティ上の弱点と、アクセスを制限するために行なった変更を報告します。中レベルでは、ASET はシステムサービスに影響しません。
高	システムに高度なセキュリティが適用されます。ASET は多数のシステムファイルとパラメータの設定を調整して、アクセス権を最小限度に抑えます。ほとんどのシステムアプリケーションとコマンドは、正常に機能します。ただし、高レベルでは、セキュリティがほかのシステムの動作より優先されます。

注 – セキュリティレベルを下げるか、システムを ASET 実行前の設定に意図的に戻さない限り、ASET によってファイルのアクセス権のセキュリティが低くなることはありません。

ASET のタスク

この節では、ASET のタスクについて説明します。レポートを解釈して活用するには、各 ASET のタスク (その目的、実行される操作、および影響を受けるシステム構成要素) を理解しておく必要があります。

ASET のレポートファイルには、各 ASET タスクで検出された問題をできるだけ詳細に記述するメッセージが含まれています。これらのメッセージによって、問題を診断して解決できます。ただし、ASET を活用するには、システム管理とシステム構成要素を全般的に理解していることが前提となります。システム管理者になったばかりのユーザーは、ほかの Solaris システム管理マニュアルと関連するマニュアルページを参照して、ASET の管理の概要を把握してください。

`taskstat` ユーティリティは、完了したタスクとまだ実行中のタスクを識別します。完了したタスクごとにレポートファイルが生成されます。`taskstat` ユーティリティの詳細は、`taskstat(1M)` のマニュアルページを参照してください。

システムファイルのアクセス権の調整

このタスクでは、システムファイルのアクセス権を指定したセキュリティレベルに設定します。このタスクは、システムのインストール時に実行されます。以前に設定したレベルをあとから変更したい場合は、このタスクを再度実行してください。低セキュリティレベルでは、アクセス権は開放型の情報共有環境に適した値に設定されています。中セキュリティレベルでは、アクセス権はほとんどの環境に十分なセキュリティが適用される程度に厳しくなります。高セキュリティレベルでは、アクセスが最も厳しく制限されます。

このタスクによってシステムファイルのアクセス権やパラメータの設定に加えられた変更は、`tune.rpt` ファイル内にレポートされます。ASET がアクセス権を設定するときに調整するファイルの例については、153 ページの「調整ファイル」を参照してください。

システムファイルの確認

このタスクでは、システムファイルが検査され、マスターファイル内に一覧された各ファイルの記述と比較されます。マスターファイルは、ASET がこのタスクを実行するときに初めて作成されます。マスターファイルには、指定したセキュリティレベルの `checklist` によって適用されるシステムファイル設定が含まれています。

ファイルが確認されるディレクトリのリストは、セキュリティレベルごとに定義されます。デフォルトのリストを使用するか、レベルごとに異なるディレクトリを指定して変更できます。

ファイルごとに次の条件が確認されます。

- 所有者とグループ
- アクセス権ビット
- サイズとチェックサム
- リンク数
- 最終変更時刻

矛盾が見つかると、`cklist.rpt` ファイル内にレポートされます。このファイルには、システムファイルのサイズ、アクセス権、およびチェックサムの値について、マスターファイルと比較した結果が入っています。

ユーザーとグループの確認

このタスクでは、`passwd` ファイルと `group` ファイル内で定義されているユーザーアカウントとグループの整合性と完全性が確認されます。ローカルパスワードファイルと、NIS または NIS+ パスワードファイルが検査されます。NIS+ パスワードファイルの問題はレポートされますが、解決はされません。このタスクでは、次の違反が検査されます

- 重複名または重複 ID
- 不正な形式のエントリ
- パスワードが付いていないアカウント
- 無効なログインディレクトリ
- アカウント `nobody`
- 空のグループパスワード
- NIS (または NIS+) サーバー上の `/etc/passwd` ファイル内のプラス記号 (+)

矛盾は `usrgrp.rpt` ファイル内にレポートされます。

システム構成ファイルの確認

このタスクでは、ASET はあらゆるシステムテーブルを検査します。テーブルのほとんどは `/etc` ディレクトリに入っており、次のシステム構成ファイルが検査されます。

- `/etc/default/login`
- `/etc/hosts.equiv`
- `/etc/inetd.conf`
- `/etc/aliases`
- `/var/adm/utmpx`
- `/.rhosts`
- `/etc/vfstab`

- /etc/dfs/dfstab
- /etc/ftpd/ftpusers

ASET は、これらのファイルに関して各種の検査と変更を実行し、すべての問題を sysconf.rpt ファイル内にレポートします。

環境変数の確認

このタスクでは、スーパーユーザー用とその他のユーザー用の PATH 環境変数と UMASK 環境変数が /.profile、/.login、/.cshrc ファイル内でどのように設定されているかを検査します。

環境のセキュリティ状況を検査した結果は、env.rpt ファイル内にレポートされます。

eeprom の確認

このタスクでは、eeprom セキュリティパラメータの値が検査され、適切なセキュリティレベルに設定されていることを確認します。eeprom セキュリティパラメータは、none、command、または full に設定できます。

ASET はこの設定を変更しませんが、推奨値を eeprom.rpt ファイル内にレポートします。

ファイアウォールの設定

このタスクでは、システムをネットワークリレーとして安全に使用できることが保証されます。45 ページの「ファイアウォールシステム」で説明しているように、このタスクでは、ファイアウォール専用システムが設定され、内部ネットワークが外部の公共ネットワークから保護されます。このファイアウォールシステムでは、ネットワークが 2 つに分割されます。このとき、分割された各ネットワークは、互いに信頼されないネットワークとして通信します。ファイアウォールの設定タスクによって、インターネットプロトコル (IP) パケットを転送できなくなり、ルーティング情報は外部ネットワークから隠されます。

ファイアウォールのタスクはすべてのセキュリティレベルで実行されますが、ファイアウォールとしての本来の機能は最上位レベルでのみ動作します。ASET を高セキュリティレベルで実行したいときでも、システムにはファイアウォール保護が不要であることがわかった場合は、asetenv ファイルを編集してファイアウォールタスクをはずすことができます。

行われた変更はすべて firewall.rpt ファイル内にレポートされます。

ASET 実行ログ

ASET を対話形式またはバックグラウンドで実行すると、実行ログが生成されます。デフォルトでは、ASET はログファイルを標準出力に生成します。実行ログは、ASET が指定された時刻に実行されたことを確認するもので、実行エラーメッセージも含まれています。`aset -n` コマンドを使用すると、ログを指定したユーザーに電子メールで配信できます。ASET オプションの一覧については、`aset(1M)` のマニュアルページを参照してください。

ASET 実行ログファイルの例

```
ASET running at security level low

Machine=example; Current time = 0325_08:00


aset: Using /usr/aset as working directory

Executing task list...
        firewall
        env
        sysconfig
        usrgrp
        tune
        cklist
        eeprom
All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
     $/usr/aset/util/taskstat     aset_dir
Where aset_dir is ASET's operating directory, currently=/usr/aset

When the tasks complete, the reports can be found in:
     /usr/aset/reports/latest/*.rpt
You can view them by:
more /usr/aset/reports/latest/*.rpt
```

実行ログはまず、ASET が実行されたシステムと時刻を示します。次に、開始したタスクの一覧を表示します。

139 ページの「ASET のタスク」で説明しているように、ASET はこれらのタスクごとにバックグラウンド処理を呼び出します。タスクは開始されると実行ログに一覧表示されます。この一覧は、タスクの完了を示しているわけではありません。バックグラウンドタスクの状態を確認するには、`taskstat` コマンドを使用します。

ASET レポート

ASET タスクから生成されたすべてのレポートファイルは、ディレクトリ /usr/aset/reports の下のサブディレクトリに入っています。この節では、/usr/aset/reports ディレクトリの構造と、レポートファイルを管理するためのガイドラインについて説明します。

ASET は、指定されたサブディレクトリにレポートファイルを格納し、レポートの生成日時を反映させます。この規則によって、ASET を実行するたびに変わるシステムの状態が記録されたレコードを順番に追跡できます。これらのレポートを監視し、比較して、システムセキュリティの状況を判断できます。

次の図に reports ディレクトリ構造の例を示します。

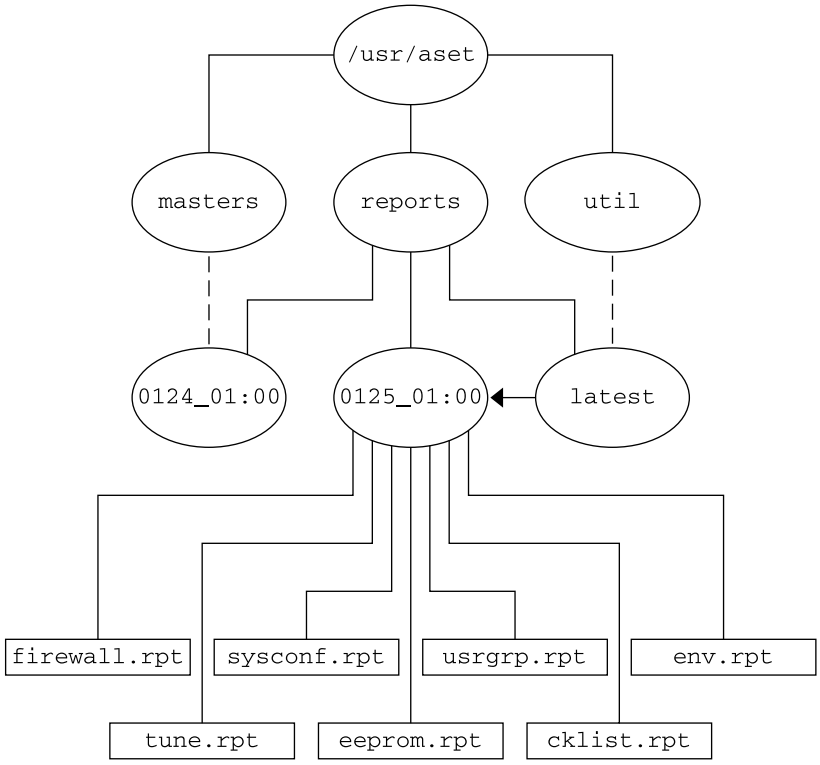


図 **8–1** ASET reports ディレクトリの構造

この例では、2 つのレポートサブディレクトリを示しています。

- 0124_01:00
- 0125_01:00

サブディレクトリ名は、レポートの生成日時を示します。各レポートサブディレクトリ名の形式は次のとおりです。

monthdate_hour:minute

この場合、*month*、*date*、*hour*、*minute* は、いずれも 2 桁の数値です。たとえば、`0125_01:00` は 1 月 25 日の午前 1 時を表します。

2 つのレポートサブディレクトリには、それぞれ ASET を 1 度実行して、生成されたレポートの集合が含まれています。

`latest` ディレクトリは、常に最新レポートが入っているサブディレクトリを指すシンボリックリンクです。したがって、ディレクトリ `/usr/aset/reports/latest` に移動すると、ASET で生成された最新レポートを見ることができます。このディレクトリには、最後に ASET を実行した各タスクのレポートファイルが入っています。

ASET レポートファイルの形式

各レポートファイルは、それを生成したタスクから取った名前が付けられます。 次の表にタスクとそのレポートのリストを示します。

表 8–1 ASET のタスクと生成されるレポート

タスク	レポート
システムファイルのアクセス権の調整 (`tune`)	`tune.rpt`
システムファイルの確認 (`cklist`)	`cklist.rpt`
ユーザーとグループの確認 (`usrgrp`)	`usrgrp.rpt`
システム構成ファイルの確認 (`sysconf`)	`sysconf.rpt`
環境変数の確認 (`env`)	`env.rpt`
`eeprom` の確認 (`eeprom`)	`eeprom.rpt`
ファイアウォールの設定 (`firewall`)	`firewall.rpt`

各レポートファイル内で、メッセージの前後はバナー行で囲まれています。ASET の構成要素を誤って削除したり損傷したりした場合など、タスクが途中で終了することがあります。 このような場合、通常はレポートファイルの末尾の方に、途中で終了した理由を示すメッセージが記録されています。

次にレポートファイル `usrgrp.rpt` の例を示します。

```
*** Begin User and Group Checking ***

Checking /etc/passwd ...
Warning! Password file, line 10, no passwd
:sync::1:1::/:/bin/sync
..end user check; starting group check ...
```

```
Checking /etc/group...
*** End User And group Checking ***
```

ASET レポートファイルの検査

ASET を最初に実行したときまたは再構成したときは、レポートファイルを詳細に検査する必要があります。再構成には、`asetenv` ファイル、`masters` サブディレクトリのマスターファイルを変更したり、ASET が動作するセキュリティレベルを変更したりすることが含まれます。

このレポートには、ASET の再構成によって発生したエラーがすべて記録されます。レポートを詳しく確認すると、問題が発生した時点で対処して解決できます。

ASET レポートファイルの比較

構成上の変更やシステム更新がない期間中にレポートファイルを監視すると、レポートの内容が安定状態になり、予期しない情報は、あってもわずかであることがわかります。`diff` ユーティリティを使用して、レポートを比較できます。

ASET マスターファイル

ASET のマスターファイル `tune.high`、`tune.low`、`tune.med`、および `uid_aliases` は、ディレクトリ `/usr/aset/masters` に入っています。ASET は、マスターファイルを使用してセキュリティレベルを定義します。

調整ファイル

`tune.low`、`tune.med`、`tune.high` マスターファイルでは、利用できる ASET セキュリティレベルが定義されます。各ファイルでは、各レベルのシステムファイルの属性が指定され、比較と参照に使用されます。

`uid_aliases` ファイル

`uid_aliases` ファイルには、同じユーザー ID (UID) を共有する複数のユーザーアカウントの一覧が入っています。このような複数のユーザーアカウントがあると責任の所在があいまいになるため、通常は ASET が警告を出します。`uid_aliases` ファイル内で例外を一覧すると、この規則に例外を設けることができます。重複する UID を持つ `passwd` ファイル内のエントリを `uid_aliases` ファイル内で指定しておくと、これらのエントリは ASET でレポートされません。

複数のユーザーアカウント (パスワードエントリ) に同じ UID を共有しないでください。他の方法で目的を達成することを検討する必要があります。たとえば、複数のユーザーにアクセス権一式を共有させたい場合は、グループアカウントを作成できます。UID の共有は最後の手段であり、どうしても必要で、他の方法では目的を達成できない場合にだけに使用します。

環境変数 UID_ALIASES を使用すると、別の別名ファイルを指定できます。 デフォルトファイルは /usr/aset/masters/uid_aliases です。

確認リストファイル

システムファイルの確認に使用されるマスターファイルは、初めて ASET を実行するときか、セキュリティレベルの変更後に ASET を実行するときに生成されます。

次の環境変数には、このタスクで確認するファイルを定義します

- CKLISTPATH_LOW
- CKLISTPATH_MED
- CKLISTPATH_HIGH

ASET 環境ファイル (asetenv)

環境ファイル asetenv には、ASET タスクに影響する環境変数の一覧が入っています。一部の環境変数を変更すると、ASET の動作を修正することができます。

ASET の構成

この節では、ASET を構成する方法とその操作の基礎となる環境について説明します。

ASET の管理と構成は最小限ですみ、ほとんどの場合はデフォルト値で実行できます。ただし、ASET の処理や動作に影響する一部のパラメータを調整して、その特長を最大限に発揮させることができます。デフォルト値を変更する前に、ASET の機能と、システムの構成要素に及ぼす影響を理解しておく必要があります。

ASET は、次の 4 つの構成ファイルに依存してタスクの動作を制御します。

- /usr/aset/asetenv
- /usr/aset/masters/tune.low
- /usr/aset/masters/tune.med
- /usr/aset/masters/tune.high

環境ファイルの変更 (asetenv)

/usr/aset/asetenv ファイルは、次の 2 つの主要セクションに分かれています

- ユーザーが構成可能な環境変数セクション
- 内部環境変数セクション

ユーザーが構成可能なパラメータセクションは変更できます。ただし、内部環境変数セクションの設定は内部使用だけに限られ、変更できません。

ユーザーが構成可能なセクションのエントリを編集して、次の操作を行うことができます。

- 実行するタスクを選択する
- システムファイルの確認タスク用のディレクトリを指定する
- ASET の実行スケジュールを指定する
- UID 別名ファイルを指定する
- 確認対象を NIS+ テーブルまで拡張する

実行するタスクの選択: TASKS

ASET が実行する各タスクでは、システムセキュリティの特定の領域が監視されます。ほとんどのシステム環境では、すべてのタスクでバランスがとれたセキュリティ範囲を提供する必要があります。ただし、1 つまたは複数のタスクを除外してもかまいません。

たとえば、ファイアウォールタスクはすべてのセキュリティレベルで実行されますが、本来の機能は最上位レベルでのみ動作します。ASET を高セキュリティレベルで実行したい場合でも、ファイアウォール保護は不要なときがあります。

asetenv ファイル内で環境変数の TASKS の一覧を編集すると、ファイアウォール機能を使用しないで高セキュリティレベルで実行するように ASET を設定できます。デフォルトでは、TASKS の一覧にはすべての ASET タスクが含まれています。特定のタスクを削除するには、このファイルからそのタスクに関連する変数を削除します。この場合は、一覧から firewall 環境変数を削除することになります。次の一覧に ASET を実行すると、除外したタスクは実行されません。

次の例では、すべての ASET タスクが含まれる TASKS の一覧が表示されます。

```
TASKS=”env sysconfig usrgrp tune cklist eeprom firewall”
```

システムファイル確認タスクのディレクトリの指定: CKLISTPATH

システムファイル確認では、選択したシステムディレクトリ内のファイルの属性が確認されます。次の確認リストパスの環境変数を使用して、どのディレクトリを確認するかを定義できます。

CKLISTPATH_LOW 変数は、低セキュリティレベルで確認されるディレクトリを定義します。CKLISTPATH_MED と CKLISTPATH_HIGH 環境変数は、それぞれ中セキュリティレベルと高セキュリティレベルで同じように機能します。

セキュリティレベルの低い環境変数を定義したディレクトリの一覧は、次にセキュリティレベルの高い環境変数を定義したディレクトリの一覧のサブセットである必要があります。たとえば、CKLISTPATH_LOW に定義したすべてのディレクトリを CKLISTPATH_MED に含め、CKLISTPATH_MED に指定したすべてのディレクトリを CKLISTPATH_HIGH に含めます。

これらのディレクトリに対して実行される確認は再帰的ではありません。ASET では、環境変数に明示的に指定されているディレクトリだけが確認されます。ASETでは、そのサブディレクトリは確認されません。

これらの環境変数の定義を編集して、ASET に確認させたいディレクトリを追加または削除できます。これらの確認リストは、通常は毎日変更がないシステムファイルにのみ便利なことに注意してください。たとえば、一般にユーザーのホームディレクトリは動的な変化が大きすぎるので、確認リストの候補にはなりません。

ASET の実行スケジュールの指定: PERIODIC_SCHEDULE

ASET を起動する場合、対話形式で起動する方法と、-p オプションを使用して ASET タスクをスケジュール指定した時刻に実行する方法があります。ASET は、システム需要が少ないときに定期的に実行できます。たとえば、ASET は PERIODIC_SCHEDULE を照会して、ASET タスクの実行頻度と実行時刻を判断します。ASET を定期的に実行するように設定する方法については、156 ページの「ASET を定期的に実行する方法」を参照してください。

PERIODIC_SCHEDULE の形式は、crontab エントリの形式と同じです。詳細は、crontab(1) のマニュアルページを参照してください。

別名ファイルの指定: UID_ALIASES

UID_ALIASES 変数は、共有 UID が一覧される別名ファイルを指定します。 デフォルトファイルは /usr/aset/masters/uid_aliases です。

確認範囲を NIS+ テーブルまで拡張する: YPCHECK

YPCHECK 環境変数は、ASET でシステム構成ファイルテーブルも確認するかどうかを指定します。YPCHECK はブール型変数なので、true または false しか指定できません。デフォルト値は false で、NIS+ テーブルの確認は無効になっています。

この環境変数の機能を理解するために、passwd ファイルに与える影響を考えてみてください。false に設定すると、ASET はローカルの passwd ファイルを確認します。true に設定すると、NIS+ の passwd ファイル内でシステムのドメインも確認されます。

注 – ASET ではローカルテーブルが自動的に修復されますが、NIS+ テーブル内の潜在的な問題はレポートされるだけです。ASET ではそれらの問題は変更しません。

調整ファイルの変更

ASET は、3 つのマスター調整ファイル、tune.low、tune.med、tune.high を使用して、重要なシステムファイルへのアクセス制限を緩めたり厳しくしたりします。この 3 つのマスターファイルは /usr/aset/masters ディレクトリにあり、環境に合わせて調整できます。詳細は、153 ページの「調整ファイル」を参照してください。

tune.low ファイルは、アクセス権をデフォルトのシステム設定に適した値に設定します。tune.med ファイルは、これらのアクセス権をさらに制限し、tune.low に含まれていないエントリを追加します。tune.high ファイルは、アクセス権をさらに厳しく制限します。

注 – 調整ファイル内の設定を変更するには、ファイルのエントリを追加または削除します。アクセス権を現在の設定よりも制限が緩やかになるような値に設定しても意味がありません。システムセキュリティを下位レベルに下げない限り、ASET がアクセス権の制限を緩和したことになりません。

ASET で変更されたシステムファイルの復元

ASET を初めて実行すると、元のシステムファイルが保存され保管されます。aset.restore ユーティリティは、これらのファイルを復元します。また、ASET を定期的に実行するようにスケジュール指定している場合は、そのスケジュールを解除します。aset.restore コマンドは、ASET の操作ディレクトリ /usr/aset に入っています。

システムファイルに適用した変更は、aset.restore コマンドを実行すると失われます。

aset.restore コマンドは、次のような場合に使用します。

- ASET の変更を削除して元のシステムを復元したい場合。ASET を永久に無効にしたい場合は、以前にスーパーユーザーの crontab に aset コマンドが追加されていれば、cron スケジュールからそれを削除できます。cron を使用して自動実行を削除する方法については、157 ページの「ASET の定期的な実行を停止する方法」を参照してください。
- ASET を短期間実行したあとに、元のシステム状態を復元する場合
- 一部の主要なシステム機能が正常に動作せず、ASET が原因だと思われる場合

NFS システムを使用するネットワーク操作

通常、ネットワークの一部となっているシステム上でも、ASET はスタンドアロンモードで使用されます。スタンドアロンシステムのシステム管理者は、システムのセキュリティとシステムを保護する ASET の実行と管理を担当することになります。

また、ASET は NFS 分散環境でも使用できます。ネットワーク管理者は、すべてのクライアントの各種管理タスクのインストール、実行、管理を担当します。複数のクライアントシステム間で ASET を管理しやすくするために、構成変更を行ってすべてのクライアントに一括して適用すると、各システムにログインしてプロセスを繰り返す必要がなくなります。

ネットワークシステム上で ASET の設定方法を決めるときには、ユーザーに各自のシステム上でセキュリティをどのように制御させるかと、セキュリティ制御に関する責任をどの程度集中させるかを検討する必要があります。

各セキュリティレベルの一括構成の提供

複数のネットワーク構成を設定したい場合があります。たとえば、低セキュリティレベルに指定したクライアント用に 1 つ、中レベルのクライアント用に 1 つ、さらに高レベルのクライアント用に 1 つのネットワーク構成を設定できます。

セキュリティレベルごとに別の ASET ネットワーク構成を作成したい場合は、サーバー上でレベルごとに 1 つずつ合計 3 つの ASET 構成を作成できます。各構成を該当するセキュリティレベルのクライアントにエクスポートします。3 つの構成すべてに共通の ASET 構成要素は、リンクを使用して共有できます。

ASET レポートの収集

スーパーユーザー特権を持っているかどうかにかかわらず、クライアントにアクセスされるサーバー上に ASET 構成要素を集中できるほか、サーバー上に集中ディレクトリを設定して、各種クライアント上で実行中のタスクによって生成されるすべてのレポートを収集できます。収集メカニズムを設定する方法については、157 ページの「サーバー上で ASET レポートを収集する方法」を参照してください。

サーバー上でレポートを収集するように設定すると、すべてのクライアントに関するレポートを 1 箇所で検討できます。この方法は、クライアントがスーパーユーザー特権を持っているかどうかに関係なく使用できます。また、ユーザーに各自の ASET レポートを監視させたい場合は、ローカルシステム上にレポートディレクトリを残しておくこともできます。

ASET 環境変数

次の表に ASET 環境変数と各変数で指定する値を示します。

表 **8–2** ASET 環境変数とその意味

環境変数	指定する値
ASETDIR	ASET の作業ディレクトリ
ASETSECLEVEL	セキュリティレベル
PERIOD_SCHEDULE	定期的なスケジュール
TASKS	実行するタスク
UID_ALIASES	別名ファイル
YPCHECK	NIS マップと NIS+ テーブルを確認するかどうか
CKLISTPATH_LOW	低セキュリティ用のディレクトリリスト
CKLISTPATH_MED	中セキュリティ用のディレクトリリスト
CKLISTPATH_HIGH	高セキュリティ用のディレクトリリスト

以下の節で示す環境変数は、/usr/aset/asetenv ファイルにあります。ASETDIR 変数と ASETSECLEVEL 変数はオプションで、aset コマンドを使用してシェルからでなければ設定できません。他の環境変数は、ファイルを編集して設定できます。

ASETDIR 環境変数

ASETDIR は、ASET の作業ディレクトリを指定します。

C シェルからは、次のように入力します。

```
% setenv ASETDIR pathname
```

Bourne シェルまたは Korn シェルからは、次のように入力します。

```
$ ASETDIR=pathname
$ export ASETDIR
```

pathname には ASET 作業ディレクトリの完全パス名を設定します。

ASETSECLEVEL 環境変数

ASETSECLEVEL は、ASET タスクが実行されるセキュリティレベルを指定します。

C シェルからは、次のように入力します。

```
% setenv ASETSECLEVEL level
```

Bourne シェルまたは Korn シェルからは、次のように入力します。

```
$ ASETSECLEVEL=level
export ASETSECLEVEL
```

上記のコマンドで、*level* を次のいずれかに設定できます。

`low`	低セキュリティレベル
`med`	中セキュリティレベル
`high`	高セキュリティレベル

PERIODIC_SCHEDULE 環境変数

`PERIODIC_SCHEDULE` の値の形式は、`crontab` ファイルと同じです。変数の値は二重引用符で囲んだ 5 つのフィールドからなる文字列として指定します。各フィールドは次のように 1 つの空白文字で区切ります

"*minutes hours day-of-month month day-of-week*"

表 8–3 `Periodic_Schedule` 変数の値

変数	値
minutes hours	開始時刻を分 (0–59) と時間 (0–23) で指定する
day-of-month	ASET を実行する日付を 1–31 の値で指定する
month	ASET を実行する月を 1–12 の値で指定する
day-of-week	ASET を実行する曜日を 0–6 の値で指定する。日曜日が 0 になる

次の規則が適用されます。

- どのフィールドも、値のリストをコンマで区切って指定できる
- 値を数値または範囲として指定できる。範囲は、1 対の数値をハイフンで結合して指定する。範囲に含まれるすべての時刻に ASET タスクを実行することを示す
- どのフィールドも、値としてアスタリスク (*) を指定できる。アスタリスクを指定すると、そのフィールドで使用できるすべての値を指定したことになる

`PERIODIC_SCHEDULE` 変数のデフォルトエントリでは、ASET が毎日深夜 12:00 に実行されます。

```
PERIODIC_SCHEDULE=”0 0 * * *”
```

TASKS 環境変数

`TASKS` 変数は、ASET で実行されるタスクを一覧します。デフォルトでは、7 つのタスクが次のようにすべて一覧されます

```
TASKS=”env sysconfig usrgrp tune cklist eeprom firewall”
```

UID_ALIASES 環境変数

UID_ALIASES 変数は、別名ファイルを指定します。別名ファイルがあると、ASET は使用可能な複数の別名の一覧をこのファイル内で照会します。書式は UID_ALIASES=*pathname* です。*pathname* は、別名ファイルのフルパス名です。

デフォルトは次のとおりです。

```
UID_ALIASES=${ASETDIR}/masters/uid_aliases
```

YPCHECK 環境変数

YPCHECK 変数は、システムテーブルを確認するタスクを拡張して NIS または NIS+ テーブルを含めます。この変数はブール変数なので、true または false に設定されます。

デフォルトは false で、ローカルシステムテーブルが確認されます。

```
YPCHECK=false
```

CKLISTPATH_level 環境変数

3 つの確認リストパス変数は、システムファイルの確認リストタスクで確認されるディレクトリを一覧します。次の変数の定義は、デフォルトで設定されます。さまざまなレベルの変数の関係を定義しています。

```
CKLISTPATH_LOW=${ASETDIR}/tasks:${ASETDIR}/util:${ASETDIR}/masters:
/etc
CKLISTPATH_MED=${CKLISTPATH_LOW}:/usr/bin:/usr/ucb
CKLISTPATH_HIGH=${CKLISTPATH_MED}:/usr/lib:/sbin:/usr/sbin:/usr/ucblib
```

確認リストパス環境変数の値は、シェルパス変数の値と似ています。つまり、複数のディレクトリ名を指定するときは、コロンで区切ります。等号 (=) を使用すると、変数名にその値を設定できます。

ASET ファイルの例

この節では、調整ファイルや別名ファイルなどの ASET ファイルの例を示します。

調整ファイル

ASET は 3 つの調整ファイルを管理します。次の表では、3 つのすべての調整ファイルのエントリの書式を示します。

表 **8–4** 調整ファイルのエントリの書式

フィールド名	説明
pathname	ファイルのフルパス名
mode	アクセス権の設定を表す 5 桁の数値
owner	ファイルの所有者
group	ファイルのグループ
type	ファイルの形式

調整ファイルを編集するときは、次の規則が適用されます。

- パス名には、アスタリスク (*) や疑問符 (?) など、通常のシェルのワイルドカード文字を使用して、複数のエントリを指定できます。詳細は、`sh(1)` のマニュアルページを参照してください。
- *mode* は、最も制限が緩やかな値を表します。現在の設定が指定した値よりもすでに厳しく制限されている場合、ASET はアクセス権の設定を緩和しません。たとえば、指定した値が `00777` の場合、`00777` は常に現在の設定よりも緩やかな制限を表すため、アクセス権は変更されません。

  セキュリティレベルを下げるか、ASET を削除しない限り、ASET ではこの方法でモード設定を行います。セキュリティレベルを前回の実行時よりも下げるときや、システムファイルを ASET を最初に実行する前の状態に復元したいときには、ASET は操作の内容を認識して保護レベルを下げます。
- *owner* と *group* には、数値 ID ではなく名前を使用する必要があります。
- *owner*、*group*、*type* の代わりに疑問符 (?) を使用すると、ASET がこれらのパラメータの既存の値を変更しないようにします。
- *type* には、`symlink` (シンボリックリンク)、ディレクトリ、またはファイルなどすべての種類を指定できます。
- セキュリティレベルが高くなるほど、調整ファイルは下位レベルよりも緩やかなファイルアクセス権にリセットされます。また、上位セキュリティレベルになるほど、一覧に多数のファイルが追加されます。
- 1 つのファイルで複数の調整ファイルエントリを照合できます。たとえば、`etc/passwd` は `etc/pass*` エントリと `/etc/*` エントリに一致します。
- 2 つのエントリのアクセス権が異なる場合は、ファイルアクセス権は最も厳しいアクセス権を表す値に設定されます。次の例では、`/etc/passwd` のアクセス権は `00755` に設定されますが、これは `00755` は `00770` よりも厳密な制限であることを表します。

  ```
  /etc/pass* 00755 ? ? file
  /etc/* 00770 ? ? file
  ```
- 2 つのエントリの *owner* 指定または *group* 指定が異なる場合は、最後のエントリが優先されます。次の例では、`/usr/sbin/chroot` の所有者が `root` に設定されます。

```
/usr/sbin/chroot 00555 bin bin file
/usr/sbin/chroot 00555 root bin file
```

別名ファイル

別名ファイルには、同じユーザー ID を共有する別名の一覧が含まれています。

各エントリの書式は次のとおりです。

uid=alias1=alias2=alias3=. . .

uid	共有 UID
aliasn	UID を共有するユーザーアカウント

たとえば、次のエントリでは、`sysadm` および `root` アカウントによって共有されている UID `0` を表示しています。

```
0=root=sysadm
```

ASET の実行

この節では、ASET を対話的にまたは定期的に実行する方法について説明します。

▼ ASET を対話的に実行する方法

1. スーパーユーザーになります。
2. **`aset`** コマンドを使用して **ASET** を対話的に実行します。

   `# /usr/aset/aset -l` *level* `-d` *pathname*

level	セキュリティレベルを指定する。有効な値は `low`、`medium`、または `high`。デフォルト設定は `low`。セキュリティレベルについては、138 ページの「ASET のセキュリティレベル」を参照
pathname	ASET の作業ディレクトリを指定する。デフォルトは `/usr/aset`

3. 画面に表示される **ASET** 実行ログを見て、**ASET** が動作していることを確認します。

実行ログメッセージは、動作しているタスクを示します。

例 — ASET を対話的に実行する

次の例では、デフォルトの作業ディレクトリを使用して低セキュリティレベルで ASET を実行します。

```
# /usr/aset/aset -l low
======= ASET Execution Log =======

ASET running at security level low

Machine = jupiter; Current time = 0111_09:26

aset: Using /usr/aset as working directory

Executing task list ...
    firewall
    env
    sysconf
    usrgrp
    tune
    cklist
    eeprom

All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
 /usr/aset/util/taskstat [aset_dir]

where aset_dir is ASET's operating
 directory,currently=/usr/aset.

When the tasks complete, the reports can be found in:
 /usr/aset/reports/latest/*.rpt

You can view them by:
 more /usr/aset/reports/latest/*.rpt
```

▼ ASET を定期的に実行する方法

1. スーパーユーザーになります。
2. 必要であれば、**ASET** を定期的に実行したい時刻を設定します。

   システム需要が少ないときに ASET を実行します。/usr/aset/asetenv ファイル内の PERIODIC_SCHEDULE 環境変数を使用して、ASET を定期的に実行する時刻を設定します。デフォルトでは、時刻は深夜に設定されます。

   別の時刻を設定したい場合は、/usr/aset/asetenv ファイル内の PERIODIC_SCHEDULE 変数を編集します。PERIODIC_SCHEDULE 変数の設定の詳細は、152 ページの「PERIODIC_SCHEDULE 環境変数」を参照してください。

3. **aset** コマンドを使用してエントリを **crontab** ファイルに追加します。

   ```
   # /usr/aset/aset -p
   ```

   -p オプションは、決めた時刻に ASET の実行を開始するように /usr/aset/asetenv ファイル内の PERIODIC_SCHEDULE 環境変数に設定した行を crontab ファイルに挿入します。

4. 次のコマンドを実行すると **crontab** エントリが表示され、**ASET** の実行スケジュールを確認できます。

   ```
   # crontab -l root
   ```

▼ ASET の定期的な実行を停止する方法

1. スーパーユーザーになります。
2. **crontab** ファイルを編集します。

   ```
   # crontab -e root
   ```

3. **ASET** エントリを削除します。
4. 変更を保存して終了します。
5. **crontab** エントリを表示して、**ASET** エントリが削除されていることを確認します。

   ```
   # crontab -l root
   ```

▼ サーバー上で ASET レポートを収集する方法

1. スーパーユーザーになります。
2. サーバー上でディレクトリを設定します。

   a. **/usr/aset** ディレクトリに移動します。

      ```
      mars# cd /usr/aset
      ```

   b. *rptdir* ディレクトリを作成します。

      ```
      mars# mkdir rptdir
      ```

   c. *rptdir* ディレクトリに移動して、*client_rpt* ディレクトリを作成します。
      これにより、クライアント用のサブディレクトリ (*client_rpt*) が作成されます。レポートを収集したいクライアントごとに、この手順を繰り返します。

      ```
      mars# cd rptdir
      mars# mkdir client_rpt
      ```

次の例では、ディレクトリ all_reports、およびサブディレクトリ pluto_rpt と neptune_rpt が作成されます。

```
mars# cd /usr/aset
mars# mkdir all_reports
mars# cd all_reports
mars# mkdir pluto_rpt
mars# mkdir neptune_rpt
```

3. *client_rpt* ディレクトリを **/etc/dfs/dfstab** ファイルに追加します。

   このディレクトリには、読み取り権と書き込み権があります。

   たとえば、dfstab ファイル内の次のエントリは、読み取り権と書き込み権によって共有されます。

```
share -F nfs -o rw=pluto /usr/aset/all_reports/pluto_rpt
share -F nfs -o rw=neptune /usr/aset/all_reports/neptune_rpt
```

4. **dfstab** ファイル内のリソースをクライアントが利用できるようにします。

```
# shareall
```

5. 各クライアント上でクライアントのサブディレクトリを、マウントポイント **/usr/aset/masters/reports** にサーバーからマウントします。

```
# mount server:/usr/aset/client_rpt /usr/aset/masters/reports
```

6. **/etc/vfstab** ファイルを編集して、ブート時にディレクトリを自動的にマウントするようにします。

   neptune 上の /etc/vfstab 内の次のエントリ例には、mars からマウントされるディレクトリ /usr/aset/all_reports/neptune_rpt と、neptune 上のマウントポイント /usr/aset/reports が一覧されています。ブート時には、vfstab 内に一覧されたディレクトリが自動的にマウントされます。

```
mars:/usr/aset/all_reports/neptune.rpt /usr/aset/reports nfs - yes hard
```

ASET の問題の障害追跡

この章では、ASET によって生成されるエラーメッセージについて説明します。

ASET のエラーメッセージ

ASET failed:no mail program found.
原因: ASET は実行ログをユーザーに送るように指示されましたが、メールプログラムが見つからない。

対処方法: メールプログラムをインストールしてください。

```
Usage: aset [-n user[@host]] in /bin/mail or /usr/ucb/mail.
```

```
Cannot decide current and previous security levels.
```

原因: ASET は、今回と前回の呼び出しのセキュリティレベルを判別できない。

対処方法: 現在のセキュリティレベルがコマンド行オプションまたは ASETSECLEVEL 環境変数によって設定されているかどうかを確認してください。また、ASETDIR/archives/asetseclevel.arch の最終行に、以前のセキュリティレベルが正しく反映されているかどうかを確認してください。これらの値が設定されていないか正しくない場合は、訂正してください。

```
ASET working directory undefined.
```

```
To specify, set ASETDIR environment variable or use command line
```

```
option -d.
```

```
ASET startup unsuccessful.
```

原因: ASET の作業 (操作) ディレクトリが定義されていないか、正しく定義されていない。

対処方法: ASETDIR 環境変数または -d コマンド行オプションを使用して訂正してから、ASET を再起動してください。

```
ASET working directory $ASETDIR missing.
```

```
ASET startup unsuccessful.
```

原因: ASET の作業 (操作) ディレクトリが定義されていないか、正しく定義されていない。ASETDIR 変数または -d コマンド行オプションによって、存在しないディレクトリが参照されている可能性があります。

対処方法: 正しいディレクトリ、つまり ASET ディレクトリ階層が入っているディレクトリが正しく参照されているかどうかを確認してください。

```
Cannot expand $ASETDIR to full pathname.
```

原因: ASET が ASETDIR 変数または -d コマンド行オプションで指定されたディレクトリ名を完全パス名に展開できない。

対処方法: ディレクトリ名を正しく指定したかどうかと、ユーザーがアクセス権を持っている既存のディレクトリを参照しているかどうかを確認してください。

```
aset: invalid/undefined security level.
```

```
To specify, set ASETSECLEVEL environment variable or use command
```

```
line option -l, with argument= low/med/high.
```

原因: セキュリティレベルが定義されていないか、正しく定義されていない。low、med、または high の値以外は定義できない。

対処方法: ASETSECLEVEL 変数または -l コマンド行オプションを使用して、low、med、または high のいずれかの値を指定してください。

`ASET environment file asetenv not found in $ASETDIR.`

`ASET startup unsuccessful.`

原因: ASET は `asetenv` ファイルを作業ディレクトリ内で見つけることができない。

対処方法: ASET の作業ディレクトリ内に `asetenv` ファイルが入っているかどうかを確認してください。このファイルについては、`asetenv(4)` のマニュアルページを参照してください。

`filename doesn't exist or is not readable.`

原因: *filename* で指定されたファイルが存在しないか、読み取れない。この問題は、`-u` オプションを使用して、確認したいユーザーを含むファイルを指定したときに発生することがある。

対処方法: `-u` オプションの引数が存在し、読み取れるかどうかを確認してください。

`ASET task list TASKLIST undefined.`

原因: `asetenv` ファイル内で定義されているはずの ASET タスクリストが定義されていない。`asetenv` ファイルが無効である可能性がある。

対処方法: `asetenv` ファイルを検査してください。タスクリストが `User Configurable` セクションで定義されているかどうかを確認します。また、ファイルの他の部分をチェックして、ファイルが変更されていないことを確認します。正常な `asetenv` ファイルの内容については、`asetenv(4)` のマニュアルページを参照してください。

`ASET task list $TASKLIST missing.`

`ASET startup unsuccessful.`

原因: `asetenv` ファイル内で定義されているはずの ASET タスクリストが定義されていない。`asetenv` ファイルが無効である可能性がある。

対処方法: `asetenv` ファイルを検査してください。タスクリストが `User Configurable` セクションで定義されているかどうかを確認します。また、ファイルの他の部分をチェックして、ファイルが変更されていないことを確認します。正常な `asetenv` ファイルの内容については、`asetenv(4)` のマニュアルページを参照してください。

`Schedule undefined for periodic invocation.`

`No tasks executed or scheduled. Check asetenv file.`

原因: `-p` オプションを使用して ASET のスケジュール指定が要求されたが、環境変数 `PERIODIC_SCHEDULE` が `asetenv` ファイル内で定義されていない。

対処方法: `asetenv` ファイルの `User Configurable` セクションをチェックして、変数が定義されていて、正しい書式になっているかどうかを確認してください。

```
Warning! Duplicate ASET execution scheduled.
Check crontab file.
```

原因: ASET のスケジュールが複数回指定されている。つまり、ASET スケジュールがまだ有効な間に別のスケジュールを指定するように要求されている。複数のスケジュールが必要な場合は、このメッセージはエラーを示すものではなく、警告メッセージとなります。複数のスケジュールが必要な場合は、`crontab` コマンドを使用して、正しいスケジュール書式を使用する必要があります。詳細は、`crontab(1M)` のマニュアルページを参照してください。

対処方法: `crontab` コマンドを使用して、正しいスケジュールが有効になっていることを検証してください。ASET に関して不要な `crontab` エントリがないかどうかを確認してください。

パート III　認証サービスと安全な通信

第 9 章	Diffie-Hellman 認証について説明します。
第 10 章	Pluggable Authentication Module (PAM) フレームワークについて説明します。
第 11 章	Solaris Secure Shell の概要と詳細な手順について説明します。
第 12 章	Solaris Secure Shell の構成に使用するファイルについて説明します。
第 13 章	Sun Enterprise Authentication Mechanism (SEAM) の概要について説明します。
第 14 章	SEAM の構成に必要な情報と、構成する前に解決しなければならない問題の一覧を示します。
第 15 章	SEAM を構成する手順について説明します。
第 16 章	SEAM のエラーメッセージ、そのメッセージを生成した条件の修正方法、およびいくつかのエラー条件に対する障害追跡について説明します。
第 17 章	`gkadmin` GUI とコマンド行を使用して SEAM の主体とポリシーを管理する手順について説明します。
第 18 章	SEAM のユーザー操作手順について説明します。
第 19 章	SEAM の参照情報を示します。

第 9 章

認証サービスの使用 (手順)

この章では、Secure RPC で使用できる Diffie-Hellman 認証メカニズムについて説明します。

この章で説明する手順は次のとおりです。

Secure RPC の概要

Secure RPC は、サービスを要求するホストとユーザーを認証するための認証方式です。Secure RPC では、Diffie-Hellman 認証メカニズムを使用しています。この認証メカニズムは 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+ ネームサービスがあります。

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上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。クライアントマシン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、クライアントマシン上のユーザーには、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 の最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 -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。AUTH_DH 認証メカニズムは、Diffie-Hellman 認証で DES 暗号化を使用し、認証されたアクセスを保障します。AUTH_DH メカニズムは、AUTH_DES とも呼びます。

- Secure NFS の設定と管理については、『*Solaris* のシステム管理 *(*資源管理とネットワークサービス*)*』の「Secure NFS システムの管理」を参照してください。
- NIS+ テーブルの設定と cred テーブルへの名前の入力については、『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: DNS*、*NIS*、*LDAP* 編*)*』を参照してください。
- RPC 認証手順の概要については、167 ページの「Diffie-Hellman 認証の実装」を参照してください。

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、すなわちプリンシパルが同じ 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 チップは暗号化をより高速にします。ただし、チップがなくても、ソフトウェアで代用できます。

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システムは鍵を頻繁に変更します。

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。Kerberos は DES 暗号を使用します。Kerberos V4 は、Secure RPC ではサポートされていません。クライアント側には、RPCSEC_GSS を使用する Kerberos V5 がこのリリースに実装されています。詳細は、第 13 章を参照してください。

Diffie-Hellman 認証

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、相互に合意した暗号化機能 (DES など) と共通鍵を使って相互に通信します。この方式は、以前の Solaris リリースの DES 認証と同じです。

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刻を暗号化する機能を利用します。受信側のシステムは、その現在の時刻を復号し、自分の時刻と照合します。クライアントとサーバーで時刻が同期していることを確認してください。

公開鍵と非公開鍵は、NIS または NIS+ のデータベースに格納されます。NIS では、これらの鍵を publickey マップに格納します。NIS+ では、cred テーブルに格納します。これらのファイルには、すべてのユーザーの公開鍵と非公開鍵が入っています。

システム管理者は、NIS マップまたは NIS+ のテーブルを設定して、ユーザーごとに公開鍵と非公開鍵を生成する必要があります。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。

Diffie-Hellman 認証の実装

この節では、DH 認証 (AUTH_DH) を使用するクライアントサーバーセッションにおける一連のトランザクションを説明します。

公開鍵と秘密鍵の生成

システム管理者は、認証を開始する前に、 newkey または nisaddcred コマンドを実行して公開鍵と秘密鍵を生成します。ユーザーごとに一意の公開鍵と秘密鍵が与えられます。公開鍵は、公開データベースに格納されます。秘密鍵は、暗号化された形式で、同じデータベースに格納されます。公開鍵と秘密鍵のペアを変更するには、chkey コマンドを使用します。

keylogin コマンドの実行

通常、ログインパスワードは Secure RPC パスワードと同じです。この場合、keylogin コマンドは必要ありません。ただし、これらのパスワードが異なる場合は、ユーザーはログインするときに keylogin コマンドを明示的に実行する必要があります。

keylogin コマンドを入力すると、Secure RPC パスワードの入力を求めるプロンプトが表示されます。コマンドは、このパスワードを使って秘密鍵を復号化します。次に keylogin コマンドは、復号化された秘密鍵をキーサーバーと呼ばれるプログラムに渡します。キーサーバーは 、各コンピュータ上でローカルインスタンスを伴う RPC サービスです。キーサーバーは、復号化された秘密鍵を格納し、ユーザーとサーバーが Secure RPC トランザクションを開始するのを待機します。

ログインパスワードと RPC パスワードが一致している場合は、ログインプロセスは秘密鍵をキーサーバーに渡します。これらのパスワードが異なり、ユーザーが常に keylogin コマンドを実行する必要がある場合は、keylogin コマンドをユーザーの環境構成ファイル (~/.login、 ~/.cshrc、 ~/.profile ファイルなど) に設定することができます。この場合、ユーザーがログインしたときに、keylogin コマンドが自動的に実行されます。

対話鍵の生成

ユーザーがサーバーとトランザクションを開始すると、次の処理が行われます。

1. キーサーバーはランダムに対話鍵を生成します。
2. カーネルはこの対話鍵を使用して、クライアントのタイムスタンプの暗号化などを行います。
3. キーサーバーは、公開鍵データベースからサーバーの公開鍵を検索します。詳細は `publickey(4)` のマニュアルページを参照してください。
4. キーサーバーはクライアントの秘密鍵とサーバーの公開鍵を使用して、共通鍵を作成します。
5. キーサーバーは共通鍵を使用して対話鍵を暗号化します。

サーバーとの最初の接触

次に、暗号化したタイムスタンプと暗号化した対話鍵を含む伝送データがサーバーに送信されます。伝送データには資格とベリファイアが含まれます。資格は、次の 3 つの構成要素を持ちます。

- クライアントのネット名
- 共通鍵で暗号化された対話鍵
- 対話鍵で暗号化された「ウィンドウ」

この場合の「ウィンドウ」とは、サーバーの時刻とクライアントのタイムスタンプとの間で許容される時間差のことで、クライアントが指定します。サーバーの時刻とクライアントのタイムスタンプとの間の差がウィンドウより大きい場合、サーバーはクライアントの要求を拒否します。通常の状態では、クライアントは RPC セッションを開始する前にサーバーと同期を取るため、クライアントの要求は拒否されません。

クライアントベリファイアは、次の要素で構成されます。

- 暗号化されたタイムスタンプ
- 指定したウィンドウの暗号化されたベリファイアから 1 を引いた値

ウィンドウベリファイアは、他人がユーザーになりすますのを防ぐために使用されます。なりすましを試みる人は、資格やベリファイアの暗号化された各フィールドに正しい情報の代わりにランダムなビットを記入するプログラムを作成します。サーバーはこの対話鍵を任意のランダム鍵に復号化し、それを使用してウィンドウとタイムスタンプを復号化しようと試みます。結果は、乱数が生成されるだけです。しかし、数千回の試行を重ねるうちには、このランダムなウィンドウとタイムスタンプのペアが認証システムを通過することが十分ありえます。ウィンドウベリファイアは、正しい資格の解読をより困難にします。

対話鍵の復号化

サーバーがクライアントから伝送データを受信すると、次の処理が行われます。

1. キーサーバーは、公開鍵データベース内でクライアントの公開鍵を検索します。

2. キーサーバーはクライアントの公開鍵とサーバーの秘密鍵を使用して、共通鍵を計算します。この共通鍵はクライアントが計算したものと同じです。共通鍵の計算は、秘密鍵を知っている必要があるため、そのサーバーとクライアント以外は計算できません。
3. カーネルは共通鍵を使用して、対話鍵を復号化します。
4. カーネルはキーサーバーを呼び出して、復号化された対話鍵によりクライアントのタイムスタンプを復号化します。

サーバーへの格納情報

サーバーは、クライアントのタイムスタンプを復号化したあと、次の 4 つの情報を資格テーブルに格納します。

- クライアントのコンピュータ名
- 対話鍵
- ウィンドウ
- クライアントのタイムスタンプ

サーバーは、最初の 3 つの情報を将来の使用のために格納します。サーバーはタイムスタンプを格納して、同じタイムスタンプが再度使用できないようにします。サーバーは、最後に参照したタイムスタンプよりも時間的に後のタイムスタンプだけを受け付けるため、同じタイムスタンプのトランザクションはすべて拒否されることが保証されます。

注 – この手順において暗黙的に仮定されているのは呼び出し側の名前であり、何らかの方法でこの名前を認証する必要があります。キーサーバーは、呼び出し側を認証するときに、DES 認証を使用できません。DES 認証を使用すると、デッドロックが発生するためです。キーサーバーは、 ユーザー ID (UID) ごとに秘密鍵を格納し、ローカルのルートプロセスへの要求だけを許可することによってこの問題を解決します。

ベリファイアをクライントに返す

サーバーは、ベリファイアをクライアントに返します。ベリファイアには、次の構成要素が含まれます。

- サーバーが自分の資格キャッシュに記録するインデックス ID
- クライアントのタイムスタンプから 1 を引いた値。対話鍵によって暗号化される

タイムスタンプから 1 を引くのは、タイムスタンプを無効化するためです。これによって、タイムスタンプをクライアントのベリファイアとして再利用できなくなります。

クライアントによるサーバーの認証

クライアントがベリファイアを受信し、そのサーバーを認証します。クライアントは、このベリファイアを送信できるのはサーバーだけであることを知っています。その理由は、クライアントが送信したタイムスタンプの内容を知っているのはサーバーだけだからです。

追加のトランザクション

一番目以降のすべてのトランザクションごとに、クライアントは 2 番目のトランザクションでインデックス ID をサーバーに返し、もう 1 つの暗号化されたタイムスタンプを送信します。サーバーは、クライアントのタイムスタンプから 1 を引いた値を対話鍵で暗号化して、返信します。

Diffie-Hellman 認証の管理

システム管理者は、ネットワークを安全にするためのポリシーをネットワーク上に実装できます。必要なセキュリティのレベルはサイトによって異なります。この節では、ネットワークセキュリティに関連するいくつかの作業手順を説明します。

▼ キーサーバーを再起動する方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **keyserv** デーモンが動作しているか確認します。

```
# ps -ef | grep keyserv
root   100      1  16   Apr 11    ?         0:00 /usr/sbin/keyserv
root  2215   2211   5   09:57:28  pts/0     0:00 grep keyserv
```

3. プロセスが動作していない場合は、キーサーバーを起動します。

```
# /usr/sbin/keyserv
```

▼ Diffie-Hellman 認証のために NIS+ の資格に root 鍵を設定する方法

NIS+ セキュリティの詳細については、『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: FNS*、*NIS+* 編*)*』を参照してください。

1. スーパーユーザーになるか、同等の役割を引き受けます。

2. **/etc/nsswitch.conf** ファイルを編集して、次の行を追加します。

   ```
   publickey: nisplus
   ```

3. **NIS+** クライアントを起動します。

   ```
   # nisinit -cH hostname
   ```

   hostname は、そのテーブルにクライアントマシン用のエントリを持つ、信頼されている NIS+ サーバー名です。

4. 次のコマンドを入力して、クライアントを **cred** テーブルに追加します。

   ```
   # nisaddcred local
   # nisaddcred des
   ```

5. **keylogin** コマンドを使用して、設定を確認します。
   パスワードを求めるプロンプトが出たら、この手順は正常に終了しています。

例 — NIS+ クライアント上で root の新しい鍵を設定する

次の例は、ホスト pluto を使用して、earth を NIS+ クライアントとして設定しています。警告は無視できます。keylogin コマンドが受け付けられて、earth が Secure NIS+ 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います。

```
# nisinit -cH pluto
NIS Server/Client setup utility.
This machine is in the North.Abc.COM. directory.
Setting up NIS+ client ...
All done.
# nisaddcred local
# nisaddcred des
DES principal name : unix.earth@North.Abc.COM
Adding new key for unix.earth@North.Abc.Com (earth.North.Abc.COM.)

Network password: xxx      Press Return
Warning, password differs from login password.
Retype password: xxx      Press Return

# keylogin
Password:
#
```

▼ Diffie-Hellman 認証のために NIS+ の資格を使用する新しいユーザー鍵を設定する方法

1. 次のコマンドを入力して、ユーザーをルートマスターサーバー上の **cred** テーブルに追加します。

   ```
   # nisaddcred -p unix.UID@domain-name -P username.domain-name. des
   ```

この場合、*username.domain-name* の終わりにピリオド (.) を付けてください。

2. クライアントとしてログインし、**keylogin** コマンドを入力して、設定を確認します。

例 — NIS+ ユーザー用の新しい鍵を設定する

次の例は、george という名前のユーザーに DES 承認がどのように与えられるかを示しています。

```
# nisaddcred -p unix.1234@North.Abc.com -P george.North.Abc.COM. des
DES principal name : unix.1234@North.Abc.COM
Adding new key for unix.1234@North.Abc.COM (george.North.Abc.COM.)

Password:
Retype password:

# rlogin rootmaster -l george
# keylogin
Password:
#
```

▼ Diffie-Hellman 認証と NIS の資格を使用して root 鍵を設定する方法

1. クライアント上でスーパーユーザーになるか、同等の役割を引き受けます。
2. **/etc/nsswitch.conf** ファイルを編集して、次の行を追加します。

   ```
   publickey: nis
   ```

3. **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 — 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、earth を Secure NIS クライアントとして設定します。

```
# newkey -h earth
Adding new key for unix.earth@North.Abc.COM
New Password:
Retype password:
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

▼ Diffie-Hellman 認証と NIS の資格を使用する新しいユーザー鍵を設定する方法

1. スーパーユーザーとして **NIS** マスターサーバーにログインするか、同等の役割を引き受けます。

   NIS マスターサーバーにログインしたときにユーザーの新しい鍵を作成できるのは、システム管理者だけです。

2. 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。

   ```
   # newkey -u george
   Adding new key for unix.12345@Abc.North.Acme.COM
   New Password:
   Retype password:
   Please wait for the database to get updated...
   Your new key has been successfully stored away.
   #
   ```

3. ログインして **chkey -p** コマンドを入力するように、ユーザーに伝えます。

   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。

   ```
   earth% chkey -p
   Updating nis publickey database.
   Reencrypting key for unix.12345@Abc.North.Acme.COM
   Please enter the Secure-RPC password for george:
   Please enter the login password for george:
   Sending key change request to pluto...
   #
   ```

注 – 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。

▼ Diffie-Hellman 認証でファイルを共有およびマウントする方法

前提条件

Diffie-Hellman の publickey 認証がネットワークで有効である必要があります。170 ページの「Diffie-Hellman 認証のために NIS+ の資格に root 鍵を設定する方法」および 172 ページの「Diffie-Hellman 認証と NIS の資格を使用して root 鍵を設定する方法」を参照してください。

Diffie-Hellman 認証でファイルシステムを共有する

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **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*

Diffie-Hellman 認証でファイルシステムをマウントする

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **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:resource mountpoint*

   `-o sec=dh` オプションは、`AUTH_DH` 認証でファイルシステムをマウントします。

第 10 章

PAM の使用

この章では、Pluggable Authentication Module (PAM) フレームワークについて説明します。PAM は、認証サービスを「プラグイン」する方法を提供して、複数の認証サービスを使用できるようにします。

PAM (概要)

Pluggable Authentication Module (PAM) フレームワークを使用すると、`login`、`ftp`、`telnet` などのシステムエントリサービスを変更しなくても、新しい認証技術を「プラグイン」できるようになります。また、PAM を使用すると、UNIX ログインを Kerberos のような他のセキュリティメカニズムと統合できます。また、アカウント、セッション、およびパスワードの管理メカニズムも「プラグイン」できます。

PAM を使用する利点

PAM を使用すると、システムエントリサービス (`ftp`、`login`、`telnet`、`rsh` など) のユーザー認証を構成できます。PAM には次の利点があります。

- 柔軟な構成ポリシー
  - アプリケーションごとの認証ポリシー
  - デフォルトの認証メカニズムを選択する機能
  - 高度なセキュリティシステムにおける複数のパスワード
- 一般ユーザーにも使いやすい

- メカニズムが異なってもパスワードが同じであれば、パスワードを再入力する必要がない
- ユーザーが複数のコマンドを入力しなくても、複数の認証方式のパスワードを求めるプロンプトを表示できる

- オプションパラメータをユーザー認証サービスに渡す機能

PAM の構成要素

PAM ソフトウェアは、ライブラリ、いくつかのモジュール、および構成ファイルで構成されます。いくつかのコマンドまたはデーモンの新しいバージョンは、PAM インタフェースを利用できます。

次の図は、アプリケーション、PAM ライブラリ、pam.conf ファイル、および PAM モジュール間の関係を示します。

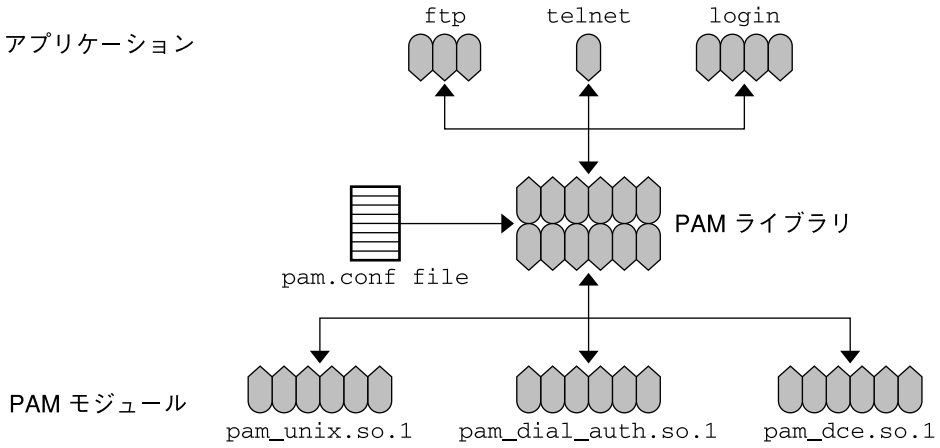


図 **10–1** PAM の動作

アプリケーション (ftp、telnet、および login) は、PAM ライブラリを使用して構成ポリシーを呼び出します。pam.conf ファイルは、使用するモジュールを定義して、各アプリケーションがモジュールを使用する順番を定義します。モジュールからの応答は、ライブラリ経由でアプリケーションに戻されます。

次の節では、PAM の構成要素とアプリケーションの関係について説明します。

PAM ライブラリ

PAM ライブラリは、適切なモジュールをロードし、スタッキング処理を管理するためのフレームワークを提供します。また、すべてのモジュールがプラグインできる汎用構造になっています。詳細は、pam(3PAM) のマニュアルページを参照してください。

パスワードマッピング機能

スタッキング機能では、ユーザーが複数のパスワードを覚えておく必要があります。「パスワードマッピング機能」を使用すると、基本パスワードから他のパスワードを復号できるので、ユーザーが複数のパスワードを覚えたり入力したりする必要はありません。各認証メカニズム間でパスワードの同期を取るためのオプションもあります。ただし、スタック内で使用される最も安全性の低いパスワード方式によってメカニズムのセキュリティが制限されてしまうため、この方法ではセキュリティの危険性が増すおそれがあります。

Solaris 9 リリースにおける PAM への変更

Solaris 9 では、PAM サービスがいくつかの点で拡張されています。ここでは、最も重要な変更点について説明します。

- スタッキングを適切に行うために、`pam_unix` モジュールが個々の単一サービスモジュールに分割されました。これらのモジュールの機能は、`pam_unix` モジュールと同等です。 これらの機能は、次の各モジュールで提供されます。

  - `pam_authtok_get`
  - `pam_authtok_check`
  - `pam_authtok_store`
  - `pam_unix_auth`
  - `pam_dhkeys`
  - `pam_passwd_auth`

  新しいモジュールについては、181 ページの「PAM モジュール」を参照してください。
- `ssh` サービス名が追加されました。PAM サービスについては、184 ページの「PAM で有効なサービス名」を参照してください。
- PAM 構成ファイルが更新されました。構成ファイルについては、187 ページの「一般的な `pam.conf` ファイル」を参照してください。

Solaris 9 Update 2 リリースにおける PAM への変更

Update 2 には、新しいバインディング制御フラグが含まれています。このフラグの導入によって、追加の認証をスキップすることが可能になります。ただし、スキップするためには、そのサービスモジュールが正常に終わるだけでなく、その前に実行されたすべての必須モジュールが正常に終わっていなければなりません。この制御フラグについては、`pam.conf(4)` のマニュアルページと 185 ページの「PAM 制御フラグ」を参照してください。

PAM (手順)

この節では、PAM のフレームワークを完全に機能させるために必要な作業について説明します。特に、PAM 構成ファイルに関連するセキュリティのいくつかの問題について注意する必要があります。

PAM (作業マップ)

作業	説明	参照先
PAM のインストールを計画する	ソフトウェア構成処理を開始する前に、構成を検討および決定する	178 ページの「PAM の計画」
新しい PAM モジュールを追加する	必要に応じて、サイト固有のモジュールを作成およびインストールし、汎用ソフトウェアにない要件に対応する。この手順はインストール処理を含む	179 ページの「PAM モジュールを追加する方法」
`~/.rhosts` によるアクセスを拒否する	`~/.rhosts` によるアクセスを拒否して、セキュリティを強化する手順	180 ページの「PAM を使用して、リモートシステムからの非承認アクセスを防ぐ方法」
エラーレポートを開始する	`syslog` を使用して PAM エラーメッセージのレポートを開始する	180 ページの「PAM のエラーレポートを開始する方法」

PAM の計画

ユーザーの環境に最適な PAM の使用方法を決定するために、次の問題から始めます。

- 何が必要か、特にどのモジュールを選択するかを決定する
- 特別な注意が必要なサービスを確認する。適宜、`OTHER` を使用する
- モジュールを実行する順番を決定する
- 各モジュールに対する制御フラグを選択する
- 各モジュールに必要なオプションを選択する

ここで、PAM 構成ファイルを変更する前に次のことを考慮することをお勧めします。

- すべてのアプリケーションを指定しなくてもいいように、モジュールタイプごとに `OTHER` エントリを使用する
- `sufficient` 制御フラグと `optional` 制御フラグのセキュリティへの影響を確認する

- モジュールに関連するマニュアルページを参照する。これらのマニュアルページには、各モジュールの機能、使用可能なオプション、スタック内のモジュール間の相互作用などの説明がある

注意 – PAM 構成ファイルの構成を間違えたり壊したりすると、スーパーユーザーでもログインできなくなる可能性があります。この場合、`sulogin` コマンドは PAM を使用しないので、スーパーユーザーは、マシンをシングルユーザーモードでブートして問題を解決する必要があります。

`/etc/pam.conf` ファイルを変更したあと、スーパーユーザーとしてログインしている間にできる限り調査します。変更によって影響を受ける可能性があるコマンドは、すべてテストしてください。たとえば、新しいモジュールを `telnet` サービスに追加した場合、`telnet` コマンドを使用して、行なった変更が期待どおりに動作しているかどうかを検証します。

▼ PAM モジュールを追加する方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. 使用する制御フラグとオプションを決定します。
   モジュールについては、181 ページの「PAM モジュール」を参照してください。
3. 新しいモジュールを **`/usr/lib/security/sparcv9`** にコピーします。
   Solaris 8 の場合は、`/usr/lib/security` にコピーします。
4. モジュールファイルの所有者が **`root`** で、そのアクセス権が **`555`** になるように、アクセス権を設定します。
5. **PAM** 構成ファイル **`/etc/pam.conf`** を編集して、このモジュールを適切なサービスに追加します。

検証

構成ファイルが間違って構成されているおそれもあるので、システムをリブートする前にテストを行う必要があります。システムをリブートする前に、`rlogin`、`su`、および `telnet` を実行してください。サービスは、システムをブートしたときに 1 度だけ生成されるデーモンである場合があります。その場合には、システムをリブートしてから、モジュールが追加されていることを確認する必要があります。

PAM を使用して、リモートシステムからの非承認アクセスを防ぐ方法

PAM 構成ファイルから「rlogin auth rhosts_auth.so.1」エントリを削除します。この手順によって、rlogin セッション中、 ~/.rhosts ファイルは読み取られなくなり、ローカルシステムに認証されていないリモートシステムからのアクセスを防止できます。~/.rhosts ファイルまたは /etc/hosts.equiv ファイルの存在またはその内容にかかわらず、すべての rlogin アクセスにはパスワードが必要になります。

注 – ~/.rhosts ファイルへのその他の非承認アクセスを防ぐには、rsh サービスも無効にする必要があります。サービスを無効にする最良の方法は、/etc/inetd.conf ファイルからサービスエントリを削除することです。PAM 構成ファイルを変更しても、サービスを無効にはできません。

▼ PAM のエラーレポートを開始する方法

1. **/etc/syslog.conf** ファイルを編集して、**PAM** エラーレポートに次のエントリを必要に応じて追加します。
   - auth.alert — すぐに修正する必要がある状態についてのメッセージ
   - auth.crit – 致命的なメッセージ
   - auth.err – エラーメッセージ
   - auth.info – 情報提供用メッセージ
   - auth.debug – デバッグメッセージ
2. **syslog** デーモンを再起動するか、**SIGHUP** シグナルを **syslog** デーモンに送信して、**PAM** のエラー報告を有効にします。

例 — PAM のエラーレポートを開始する

次の例では、すべての警告メッセージを画面に表示します。致命的なメッセージはスーパーユーザーに電子メールで送信されます。情報メッセージとデバッグ用メッセージは、/var/log/pamlog ファイルに追加されます。

```
auth.alert    /dev/console
auth.crit    'root'
auth.info;auth.debug    /var/log/pamlog
```

ログ内の各行は、タイムスタンプ、メッセージを生成したシステム名、およびメッセージ本体で構成されます。pamlog ファイルには、大量の情報が記録される可能性があります。

PAM (参照)

PAM は、実行時に取り外しが可能なモジュールを使用して、システムエントリサービスの認証を提供します。スタッキング機能を利用すると、複数のサービスを使用してユーザーを認証できます。また、パスワードマッピング機能を利用すると、ユーザーは複数のパスワードを覚える必要がなくなります。

PAM モジュール

各 PAM モジュールは、特定のメカニズムを実装します。PAM 認証を設定するときは、モジュールとモジュールタイプの両方を指定する必要があります。モジュールタイプは、モジュールが実行する処理を定義します。複数のモジュールタイプ (認証、アカウント管理、セッション管理、またはパスワード管理) を各モジュールに関連付けることができます。

次の表では、各 PAM モジュールについて、モジュール名、モジュールファイル名、および説明を示します。各モジュールのパスは、インストールされている Solaris で使用できる命令によって異なります。モジュールのデフォルトのパスは、`/usr/lib/security/` *$ISA* です。*$ISA* の値は、`sparc` または `i386` です。詳細は、`isalist(5)` のマニュアルページを参照してください。

表 **10–1** PAM モジュール

モジュール名とモジュールファイル名	説明
`authtok_check` `pam_authtok_check.so.1`	パスワード管理をサポートする。このモジュールは、パスワードのさまざまなチェックを行う。たとえば、パスワードの長さをチェックしたり、ログイン名を循環的にシフトしたものとパスワードを比較してその複雑さを検証したり、あるいは新しいパスワードと古いパスワードを比較して変更された文字数をチェックしたりして、ユーザーが単純なパスワードを使用しないように支援する。詳細は、`pam_authtok_check(5)` のマニュアルページを参照
`authtok_get` `pam_authtok_get.so.1`	認証およびパスワード用にパスワードプロンプト機能を提供する。詳細は、`pam_authtok_get(5)` のマニュアルページを参照
`authtok_store` `pam_authtok_store.so.1`	認証だけをサポートする。このモジュールは、ユーザーの認証トークンを更新する。正常に更新したトークンは、指定されたレポジトリまたはデフォルトのレポジトリに格納する。詳細は、`pam_authtok_store(5)` のマニュアルページを参照
`dhkeys` `pam_dhkeys.so.1`	認証で使用する Diffie-Hellman 鍵の管理をサポートする。このモジュールは、Secure RPC 認証と Secure RPC 認証トークンの管理をサポートする。詳細は、`pam_dhkeys(5)` のマニュアルページを参照

表 **10–1** PAM モジュール　(続き)

モジュール名とモジュールファイル名	説明
dial_auth pam_dial_auth.so.1	認証だけで使用する。このモジュールは、/etc/dialups ファイルと /etc/d_passwd ファイルに格納されたデータを、認証用として使用する。主に login コマンドで使用される。詳細は、pam_dial_auth(5) のマニュアルページを参照
krb5 pam_krb5_auth.so.1	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サポートする。認証には Kerberos 資格が使用される。詳細は、pam_krb5(5) のマニュアルページを参照
ldap pam_ldap.so.1	認証とパスワード管理をサポートする。LDAP サーバーのデータの認証に使用される。詳細は、pam_ldap(5) のマニュアルページを参照
projects pam_projects.so.1	アカウント管理をサポートする。詳細は、pam_projects(5) のマニュアルページを参照
rhosts_auth pam_rhosts_auth.so.1	認証だけで使用する。このモジュールは、ruserok() ルーチンによって ~/.rhosts および /etc/host.equiv ファイルに格納されたデータを使用する。主に rlogin と rsh コマンドで使用する。詳細は、pam_rhosts_auth(5) のマニュアルページを参照
roles pam_roles.so.1	アカウント管理だけをサポートする。RBAC の user_attr データベースが、ユーザーが引き受けることができる役割を決定する。詳細は、pam_roles(5) のマニュアルページを参照
sample pam_sample.so.1	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サポートする。テストに使用する。詳細は、pam_sample(5) のマニュアルページを参照
smartcard pam_smartcard.so.1	認証だけをサポートする。詳細は、pam_smartcard(5) のマニュアルページを参照
unix pam_unix.so.1	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サポートする。このモジュールでは、4 種類すべてのモジュールタイプを定義できる 。UNIX パスワードを認証に使用する。 Solaris 環境では、パスワードを取得するための適切なネームサービスの選択は、/etc/nsswitch.conf ファイルで制御する。詳細は、pam_unix(5) のマニュアルページを参照
unix_account pam_unix_account.so.1	アカウント管理をサポートする。このモジュールは、nsswitch.conf ファイルに指定されたレジストリのパスワード有効期限情報を取得して、パスワードおよびユーザーアカウントの期限が切れていないことを確認する。詳細は、pam_unix_account(5) のマニュアルページを参照
unix_auth pam_unix_auth.so.1	認証をサポートする。このモジュールは、PAM ハンドルに含まれているパスワードを検証する。このモジュールはまた、ユーザーのパスワードが、指定されたレポジトリまたはデフォルトのレポジトリに格納されているパスワードと一致しているか確認する。詳細は、pam_unix_auth(5) のマニュアルページを参照

表 **10–1** PAM モジュール　(続き)

モジュール名とモジュールファイル名	説明
unix_session pam_unix_session.so.1	セッション管理をサポートする。このモジュールは、セッション管理を開始するために、/var/adm/lastlog ファイルを更新する。詳細は、pam_unix_session(5) のマニュアルページを参照

セキュリティ上の理由から、これらのモジュールファイルの所有者は root である必要があり、また、group と other に書き込み権があってはなりません。ファイルの所有者が root でない場合、PAM はモジュールをロードしません。

PAM モジュールのタイプ

モジュールタイプはモジュールのインタフェースを定義するため、PAM モジュールのタイプを理解する必要があります。実行時 PAM モジュールには、次の 4 つのタイプがあります。

- 「認証モジュール」はユーザーの認証を行う。さらに、このモジュールでは、資格を設定、更新、または削除できる。そのほか、認証モジュールは、ユーザーを識別するための管理ツールを提供する
- 「アカウントモジュール」は、パスワードの有効期限、アカウントの有効期限、およびアクセス時間制限を確認する。アカウントモジュールは、認証モジュールでユーザーを識別したあと、そのユーザーにアクセス権を与えるべきかどうかを決定する。
- 「セッションモジュール」は、認証セッションの開閉を管理する。セッションモジュールは、動作を記録したり、セッション終了後のクリーンアップを実行したりできる。
- 「パスワードモジュール」によって、実際のパスワードを変更する。

PAM 構成ファイル

PAM 構成ファイル /etc/pam.conf は、使用する認証サービスとその使用順序を決定します。このファイルを編集すると、システムエントリアプリケーションごとに認証メカニズムを選択できます。

PAM 構成ファイルの構文

PAM 構成ファイルは、次の構文のエントリで構成されます。

service_name module_type control_flag module_path module_options

service_name	サービス名 (`ftp`、`login`、`telnet` など)
module_type	サービスのモジュールタイプ。詳細は、183 ページの「PAM モジュールのタイプ」を参照
control_flag	モジュールの継続または失敗の動作を決定する
module_path	サービスを実装するライブラリオブジェクトへのパスを指定する
module_options	サービスモジュールに渡すオプションを指定する

`pam.conf` ファイルにコメントを追加するには、コメント行の最初に # (ポンド記号) を入力します。フィールドを区切るには、空白またはタブを使用します。

注 – 行のフィールド数が 4 つ未満の場合、*module_type* または *control_flag* に無効な値が指定されている場合、または指定したモジュールが存在しない場合は、PAM 構成ファイル内のエントリは無視されます。

PAM で有効なサービス名

下記の表に、次の項目を示します。

- 有効なサービス名
- そのサービスで使用できるモジュールタイプ
- そのサービス名に関連するデーモンまたはコマンド

サービスによっては適用できないモジュールタイプがあります。たとえば、`passsrd` モジュールタイプは、`passwd` コマンドだけに適用できます。また、`passwd` コマンドは認証と関連がないため、このサービスに関連付けられた `auth` モジュールタイプはありません。

表 **10–2** `/etc/pam.conf` ファイルの有効なサービス名

サービス名	デーモンまたはコマンド	適用可能なモジュールタイプ
`cron`	`/usr/sbin/cron`	`auth、account`
`dtlogin`	`/usr/dt/bin/dtlogin`	`auth、account、session`
`dtsession`	`/usr/dt/bin/dtsession`	`auth`

表 **10–2** /etc/pam.conf ファイルの有効なサービス名　(続き)

サービス名	デーモンまたはコマンド	適用可能なモジュールタイプ
ftp	/usr/sbin/in.ftpd	auth、account、session
init	/usr/sbin/init	session
login	/usr/bin/login	auth、account、session
passwd	/usr/bin/passwd	password
ppp	/usr/bin/ppp	auth、account、session
rexd	/usr/sbin/rpc.rexd	account、session
rlogin	/usr/sbin/in.rlogind	auth、account、session
rsh	/usr/sbin/in.rshd	auth、account、session
sac	/usr/lib/saf/sac	session
ssh	/usr/bin/ssh	auth、account、session
su	/usr/bin/su	auth、account
telnet	/usr/sbin/in.telnetd	auth、account、session
ttymon	/usr/lib/saf/ttymon	session
uucp	/usr/sbin/in.uucpd	auth、account、session

PAM 制御フラグ

モジュールの続行動作または失敗動作を決定するには、PAM 構成ファイル /etc/pam.conf のエントリごとに「制御フラグ」を選択する必要があります。スタック内の各モジュールは、要求の成功や失敗を決定できます。

続行動作では、後続のモジュールをチェックするかどうかを定義します。特定のモジュールの応答によっては、追加モジュールをスキップできます。

失敗動作では、エラーメッセージのログや報告をどのように行うかを定義します。失敗には任意 (optional) と必須 (required) があります。必須の場合は、他のモジュールが正常であっても要求は必ず失敗に終わります。任意の場合は、要求が必ず失敗に終わるとは限りません。

これらのフラグはすべてのモジュールタイプに適用されますが、次の説明では、これらのフラグは認証モジュールで使用されていると仮定します。制御フラグは、次のとおりです。

- binding – この制御フラグが適用されたモジュールが成功し、かつそれ以前の必須モジュールの中に失敗したものがない場合は、残りのモジュールをスキップします。失敗が返された場合は、必須の失敗を記録し、スタックの処理を続けます。

binding 制御フラグは、そのモジュールが成功した場合にそれ以後のモジュールのチェックを行わないことを除けば、 required 制御フラグに似ています。このフラグが適用されたモジュールで失敗が 1 つでもあると、ほかのモジュールからの応答がどうであれ、要求は成功とはみなされません。このフラグが適用されたモジュールが成功すると、それ以前の必須モジュールの中に失敗したものがなければ、要求は成功とみなされます。

- required – この制御フラグが適用されたモジュールが成功した場合は、必須の成功を記録し、後続モジュールのチェックを続けます。モジュールが失敗した場合は、これが最初の必須の失敗であれば、エラーメッセージを保存してからスタックのチェックを続けます。これが最初の失敗でなければ、スタックのチェックを続けます。

  要求が成功するために特定のモジュールの成功が欠かせない場合は、required 制御フラグを使用すべきです。このフラグが適用されたモジュールで失敗が 1 つでもあると、ほかのモジュールからの応答がどうであれ、要求は成功とはみなされません。このフラグが適用されたモジュールの 1 つが成功しても、要求の成功を意味するわけではありません。要求が成功するためには、スタックのすべての必須モジュールからの応答が成功でなければなりません。

- requisite – この制御フラグが適用されたモジュールが成功した場合は、必須の成功を記録し、後続のモジュールのチェックを続けます。モジュールが失敗した場合は、必須の失敗を記録し、最初の必須の失敗であるというエラーメッセージを返し、それ以後のチェックをスキップします。

  requisite 制御フラグは、モジュールが失敗した場合にそれ以後のモジュールのチェックを行わないことを除けば、 required 制御フラグに似ています。このフラグが適用されたモジュールで失敗が 1 つでもあると、ほかのモジュールからの応答がどうであれ、要求は成功とはみなされません。このフラグが適用されたモジュールの 1 つが成功しても、要求の成功を意味するわけではありません。要求が成功するためには、スタックのすべての必須モジュールからの応答が成功でなければなりません。

- optional – この制御フラグが適用されたモジュールが成功した場合は、任意の成功を記録し、スタックのチェックを続けます。モジュールが失敗した場合は、任意の失敗を記録し、スタックのチェックを続けます。

  optional 制御フラグは、ユーザーを認証するにはスタック内の他のモジュールのどれかが成功すれば十分であるという場合に使用します。このフラグは特定のモジュールが成功するということがそれほど重要でない場合に使用します。要求の成功または失敗は、ほかの必須の失敗または成功によって決まります。

  ユーザーが作業を行ううえでアクセス権を持つ必要があるモジュールは、optional フラグを付けないでください。

- sufficient – この制御フラグが適用されたモジュールが成功し、かつそれ以前の必須モジュールの中に失敗したものがない場合は、残りのモジュールをスキップします。モジュールが失敗した場合は、任意の失敗を記録し、スタックのチェックを続けます。

  sufficient 制御フラグは、モジュールが成功した場合にそれ以後のモジュールのチェックを行わないことを除けば、 optional 制御フラグに似ています。このフラグが適用されたモジュールが成功すると、それ以前の必須モジュールの中に失敗したものがなければ、要求は成功とみなされます。このフラグが適用されたモ

ジュールが失敗すると、成功したモジュールが以前になければ、要求は失敗とみなされます。

これらの制御フラグの詳細については、次の節を参照してください。次の節では、デフォルトの /etc/pam.conf ファイルについて説明します。

一般的な pam.conf ファイル

一般的な /etc/pam.conf ファイルには、次の動作が指定されています。

1. login コマンドを実行したときは、pam_authtok_get、pam_dhkeys、pam_auth_unix、および pam_dial_auth モジュールの認証が成功する必要があります。
2. rlogin コマンドを実行したときは、pam_rhost_auth の認証に失敗した場合、pam_authtok_get、pam_dhkeys、および pam_auth_unix モジュールの認証が成功する必要があります。
3. sufficient 制御フラグは、 rlogin コマンドを実行したとき、pam_rhost_auth モジュールによる認証が成功すれば十分であることを意味します。次のエントリは無視されます。
4. 認証を必要とするその他のコマンドが実行されたときは、ほとんどの場合、pam_authtok_get、 pam_dhkeys、および pam_auth_unix モジュールの認証が成功する必要があります。
5. rsh コマンドが実行されたときは、pam_rhost_auth モジュールの認証には sufficient フラグが適用されます。pam_rhost_auth モジュールの認証が成功した場合は、それ以外の認証は必要ありません。

OTHER サービス名を使用すると、認証を必要とするがこのファイルには含まれていない他のコマンドに対するデフォルトとして設定できます。OTHER オプションを使用すると、同じモジュールを使用する多数のコマンドを 1 つのエントリだけでカバーできるため、ファイルの管理が簡単になります。また、OTHER サービス名を「すべてを捕捉する」と言う意味で使用すると、1 つのモジュールですべてのアクセスをカバーできます。通常、OTHER エントリは、各モジュールタイプのセクションの最後に指定します。

通常、*module_path* のエントリには「ルートからのパス名」を指定します。*module_path* に入力したファイル名がスラッシュ (/) で始まらない場合、そのファイル名の前にパス /usr/lib/security/$ISA が付きます。モジュールが他のディレクトリにある場合は、フルパスを使用する必要があります。*module_options* の値は、そのモジュールのマニュアルページに記載されています。たとえば、UNIX モジュールは、pam_unix(5) のマニュアルページに記載されています。

```
login   auth required          pam_authtok_get.so.1
login   auth required          pam_dhkeys.so.1
login   auth required          pam_unix_auth.so.1
login   auth required          pam_dial_auth.so.1
```

この例の login サービスでは、4 つの認証モジュールの認証がすべて必須になっています。login コマンドは、いずれかのモジュールからエラーが返されると失敗します。

第 11 章

Solaris Secure Shell の使用 (手順)

Solaris Secure Shell を使用すると、セキュリティ保護されていないネットワーク上のリモートホストに、安全にアクセスすることができます。Solaris Secure Shell には、リモートログインおよびリモートファイル転送を行うコマンドが組み込まれています。この章の内容は次のとおりです。

Solaris Secure Shell の概要

Solaris Secure Shell の認証は、パスワードまたは公開鍵、あるいはその両方を使用して行われます。すべてのネットワークトラフィックは暗号化されます。このため、Solaris Secure Shell では、悪意を持つ侵入者が傍受した通信を読んだり、偽装したりすることはできません。

Solaris Secure Shell は、オンデマンドタイプの仮想プライベートネットワーク (VPN) として使用することもできます。VPN では、暗号化されたネットワークを介して、ローカルマシンとリモートマシン間で、 X Window System のトラフィックを転送したり個々のポート番号を接続したりできます。

Solaris Secure Shell では、次の操作を行うことができます。

- セキュリティ保護されていないネットワークを介して、別のホストに安全にログインする
- 2 つのホスト間でファイルを安全にコピーする
- リモートホスト上でコマンドを安全に実行する

Solaris Secure Shell では、2 つのバージョンの Secure Shell プロトコルを利用できます。バージョン 1 は、Secure Shell プロトコルのオリジナルバージョンです。バージョン 2 は、安全性が向上し、バージョン 1 の基本的なセキュリティ設計上の欠点が修正されています。バージョン 1 は、バージョン 2 へ移行するユーザーを支援する目的だけに提供しています。バージョン 1 は、できるだけ使用しないでください。

注 – このマニュアルでは、v1 はバージョン 1、v2 はバージョン 2 を表しています。

Solaris Secure Shell 認証の要件は、次のとおりです。

- ユーザー認証 – ユーザーは、次のいずれかによって認証されます。
  - パスワード – ユーザーは、ログインプロセスアカウントのパスワードを入力します。
  - 公開鍵 – ユーザーは、公開鍵と非公開鍵のペアを作成します。これらは、ローカルホストに格納されます。リモートホストには公開鍵が渡されます。公開鍵は、認証を完了するために必要です。

    非公開鍵を管理するホストから、認証を行うホストに必要な公開鍵が渡されます。公開鍵認証は、パスワード認証よりも強力な認証メカニズムです。これは、非公開鍵がネットワーク上を移動しないためです。公開鍵と非公開鍵のペアは、ユーザーのホームディレクトリの `.ssh` サブディレクトリの下に格納されます。次の表に、公開鍵と非公開鍵を格納する ID ファイルのデフォルト名を示します。

    表 **11–1** SSH ID ファイルの命名規則

非公開鍵、公開鍵	暗号化方式とプロトコルのバージョン
`identity`、`identity.pub`	RSA v1
`id_rsa`、`id_rsa.pub`	RSA v2
`id_dsa`、`id_dsa.pub`	DSA v2

- ホスト認証 – ホスト認証では、ローカルホストの公開鍵に対するアクセス権をリモートホストに与える必要があります。ローカルホストの公開鍵のコピーは、リモートホストの `$HOME/.ssh/known_hosts` に格納されます。

次の表は、認証方式、互換性のあるプロトコルのバージョン、ローカルホストとリモートホストの要件、およびセキュリティレベルの一覧です。デフォルトの認証方式は、パスワードベースの認証です。

表 **11–2** Solaris Secure Shell の認証方式

認証方式 (プロトコルのバージョン)	ローカルホストの要件	リモートホストの要件	セキュリティレベル
パスワードベース (v1 または v2)	ユーザーアカウント	ユーザーアカウント	中
RSA/DSA 公開鍵 (v2)	ユーザーアカウント `$HOME/.ssh/id_rsa` または `$HOME/.ssh/id_dsa` に非公開鍵 `$HOME/.ssh/id_rsa.pub` または `$HOME/.ssh/id_dsa.pub` に公開鍵	ユーザーアカウント `$HOME/.ssh/authorized_keys` にユーザーの公開鍵 (`id_rsa.pub` または `id_dsa.pub`)	強
RSA 公開鍵 (v1)	ユーザーアカウント `$HOME/.ssh/identity` に非公開鍵 `$HOME/.ssh/identity.pub` に公開鍵	ユーザーアカウント `$HOME/.ssh/authorized_keys` にユーザーの公開鍵 (`identity.pub`)	強
`.rhosts` と RSA (v1)	ユーザーアカウント	ユーザーアカウント `/etc/hosts.equiv`、`/etc/shosts.equiv`、`$HOME/.rhosts`、または `$HOME/.shosts` にローカルホスト名 `$HOME/.ssh/known_hosts` または `/etc/ssh/ssh_known_hosts` にローカルホスト公開鍵	中
`.rhosts` のみ (v1 または v2)	ユーザーアカウント	ユーザーアカウント `/etc/hosts.equiv`、`/etc/shosts.equiv`、`$HOME/.rhosts`、または `$HOME/.shosts` にローカルホスト名	弱

Solaris Secure Shell の使用 (作業マップ)

作業	説明	参照先
公開鍵と非公開鍵のペアを作成する	公開鍵と非公開鍵のペアを使用することは、ユーザー自身を証明し、通信を暗号化するのに望ましい方法である	192 ページの「公開鍵と非公開鍵のペアを作成する方法」

作業	説明	参照先
Solaris Secure Shell を使用してログインする	暗号化された Secure Shell 通信を有効にするには、rsh を使用する場合と同様の方法で、リモートログインする	193 ページの「Solaris Secure Shell を使用して別のホストにログインする方法」
パスワードを使用せずに Solaris Secure Shell を使用してログインする	ssh-agent を使用してパスワードを入力しなくても、Secure Shell を使用してログインできる。ssh-agent コマンドは、手動でまたは起動時スクリプトから実行できる	194 ページの「パスワードを使用せずに ssh-agent コマンドを使用してログインする方法」 196 ページの「ssh-agent コマンドが自動的に動作するように設定する方法」
Solaris Secure Shell にポート転送する	TCP 経由の Secure Shell 接続で使用するローカルポートまたはリモートポートを指定できる	196 ページの「Solaris Secure Shell のポート転送を使用する方法」
Solaris Secure Shell を使用してファイルをコピーする	リモートファイルを安全にコピーできる	198 ページの「Solaris Secure Shell を使用してファイルをコピーする方法」
Solaris Secure Shell を使用してファイルを転送する	ftp と同様に、Secure Shell が稼動するリモートホストに転送用コマンドを使用してログインできる	198 ページの「sftp コマンドを使用したファイルのリモート転送」
ファイアウォールの内部のホストから外部のホストに接続する	Secure Shell には、HTTP または SOCKS5 と互換性のあるコマンドが組み込まれている。これらのコマンドは構成ファイル内やコマンド行で指定できる	199 ページの「ファイアウォール外部のホストにデフォルト接続を設定する方法」 201 ページの「例 — コマンド行からファイアウォール外部のホストに接続する」

Solaris Secure Shell の使用

▼ 公開鍵と非公開鍵のペアを作成する方法

Solaris Secure Shell の公開鍵と非公開鍵のペアを作成するときの、標準的な手順について説明します。追加のオプションについては、ssh-keygen(1) のマニュアルページを参照してください。

1. 鍵の生成プログラムを起動します。

```
myLocalHost% ssh-keygen
Generating public/private rsa key pair.
...
```

2. 鍵が格納されるファイルのパスを入力します。

デフォルトでは、RSA v2 の鍵を表すファイル名 id_rsa がカッコ内に表示されます。このファイルを選択するときは、Return キーを押します。また、別のファイル名を入力することもできます。

```
Enter file in which to save the key(/home/johndoe/.ssh/id_rsa):
<Return キーを押す>
```

公開鍵の名前が自動的に作成されます。公開鍵名の末尾に文字列 .pub が追加されます。

3. 鍵に使用するパスフレーズを入力します。

   このパスフレーズは、非公開鍵を暗号化するときに使用されます。パスフレーズは、10 ～ 30 文字の英数字を混在させて指定してください。単純な英語の文や英語の名前の使用は避けてください。空文字入力は、パスフレーズを使用しないことを意味します。空文字入力は、ユーザーアカウントには極力避けてください。入力したパスフレーズは表示されません。

```
Enter passphrase(empty for no passphrase):      <パスフレーズを入力>
```

4. 確認のためにパスフレーズを再入力します。

```
Enter same passphrase again:     <パスフレーズを入力>
Your identification has been saved in /home/jdohnoe/.ssh/id_rsa.
Your public key has been saved in /home/johndoe/.ssh/id_rsa.pub.
The key fingerprint is:
0e:fb:3d:57:71:73:bf:58:b8:eb:f3:a3:aa:df:e0:d1 johndoe@myLocalHost
```

5. 結果を確認します。

   鍵のフィンガープリント (コロンで区切られた 2 桁の 16 進数値列) が表示されます。鍵へのパスが正しいことを確認します。この例では、パスは /home/johndoe/.ssh/id_rsa.pub です。この時点で公開鍵と非公開鍵のペアが作成されました。

6. 通信先のホスト上で **authorized_keys** ファイルを設定します。

   a. **id_rsa.pub** ファイルを通信先ホストにコピーします。次のコマンドを入力します **(**ただし、バックスラッシュなしで **1** 行に入力**)**。

```
myLocalHost% cat $HOME/.ssh/id_rsa.pub | ssh myRemoteHost \
 'cat>> .ssh/authorized_keys && echo "Key uploaded successfully."'
```

   b. プロンプトが表示されたら、ログインパスワードを入力します。

      ファイルのコピーが完了すると、“Key uploaded successfully.” というメッセージが表示されます。

▼ Solaris Secure Shell を使用して別のホストにログインする方法

1. **ssh** コマンドを使用して、リモートホストの名前を指定します。

```
myLocalHost% ssh myRemoteHost
```

ssh コマンドを初めて実行した場合、次のように、リモートホストの認証を求めるプロンプトが表示されます。

```
The authenticity of host 'myRemoteHost' can't be established.
RSA key fingerprint in md5 is: 04:9f:bd:fc:3d:3e:d2:e7:49:fd:6e:18:4f:9c:26
Are you sure you want to continue connecting(yes/no)?
```

このプロンプトは正常です。**yes** を入力して処理を続行します。このリモートホストで ssh を使用したことがあるのにこのプロンプトが表示される場合は、正常ではありません。セキュリティ違反が発生していないか確認する必要があります。

2. **Solaris Secure Shell** のパスフレーズとアカウントのパスワードを要求するプロンプトが表示されたら、これらを入力します。

```
Enter passphrase for key '/home/johndoe/.ssh/id_rsa': <Return キーを押す>
johndoe@myRemoteHost's password: <Return キーを押す>
Last login: Fri Jul 20 14:24:10 2001 from myLocalHost
myRemoteHost%
```

リモートホストでトランザクションを実行します。ユーザーが送信するコマンドはすべて暗号化されます。ユーザーが受信する応答はすべて暗号化されます。

注 – パスフレーズを後で変更する場合は、 ssh-keygen コマンドに -p オプションを指定して使用します。

---

3. リモートセッションが完了したら、**exit** を入力するか、通常の方法でシェルを終了します。

```
myRemoteHost% exit
myRemoteHost% logout
Connection to myRemoteHost closed
myLocalHost%
```

▼ パスワードを使用せずに ssh-agent コマンドを使用してログインする方法

Solaris Secure Shell を使用するときに、パスフレーズとパスワードを省略する場合は、エージェントデーモンを使用します。セッションを開始するときに、ssh-agent コマンドを使用してください。次に、エージェントを使用して非公開鍵を格納するために、ssh-add コマンドを使用します。ホストごとにアカウントが異なる場合は、セッションで使用する非公開鍵を追加します。

エージェントの起動は、次の手順で説明するように、必要に応じて手動で行うことができます。各セッションを開始するときに、エージェントが自動的に動作するように設定することもできます (196 ページの「ssh-agent コマンドが自動的に動作するように設定する方法」を参照)。

1. エージェントデーモンを起動します。

ssh-agent コマンドがエージェントデーモンを起動し、そのプロセス ID が表示されます。

```
myLocalHost% eval `ssh-agent`
Agent pid 9892
myLocalHost%
```

2. 使用する非公開鍵をエージェントデーモンに追加します。

   ssh-add コマンドがエージェントデーモンに非公開鍵を追加します。このため、後続の Secure Shell の操作では、パスフレーズを要求するプロンプトは表示されません。

```
myLocalHost% ssh-add
Enter passphrase for /home/johndoe/.ssh/id_rsa:
Identity added: /home/johndoe/.ssh/id_rsa(/home/johndoe/.ssh/id_rsa)
myLocalHost%
```

3. **Solaris Secure Shell** セッションを起動します。

```
myLocalHost% ssh myRemoteHost
```

例 — ssh-add オプションを使用する

ssh-add を使用して、別の鍵をデーモンに追加することもできます。たとえば、DSA v2、RSA v2、および RSA v1 の鍵を同時に使用したい場合があります。デーモンに格納されているすべての鍵を表示するには、-l オプションを使用します。デーモンから 1 つの鍵を削除するには、-d オプションを使用します。すべての鍵を削除するには、-D オプションを使用します。

```
myLocalHost% eval `ssh-agent`
Agent pid 3347
myLocalHost% ssh-add
Enter passphrase for /home/johndoe/.ssh/id_rsa:
Identity added: /home/johndoe/.ssh/id_rsa(/home/johndoe/.ssh/id_rsa)
myLocalHost% ssh-add /home/johndoe/.ssh/id_dsa
Enter passphrase for /home/johndoe/.ssh/id_dsa:     <パスフレーズを入力>
Identity added:
/home/johndoe/.ssh/id_dsa(/home/johndoe/.ssh/id_dsa)
myLocalHost% ssh-add -l
md5 1024 0e:fb:3d:53:71:77:bf:57:b8:eb:f7:a7:aa:df:e0:d1
/home/johndoe/.ssh/id_rsa(RSA)
md5 1024 c1:d3:21:5e:40:60:c5:73:d8:87:09:3a:fa:5f:32:53
/home/johndoe/.ssh/id_dsa(DSA)
myLocalHost% ssh-add -d
Identity removed:
/home/johndoe/.ssh/id_rsa(/home/johndoe/.ssh/id_rsa.pub)
/home/johndoe/.ssh/id_dsa(DSA)
```

▼ ssh-agent コマンドが自動的に動作するように設定する方法

Secure Shell を使用するときにパスフレーズとパスワードを入力しないようにするには、エージェントデーモン ssh-agent を起動します。このエージェントデーモンは、.dtprofile スクリプトから起動できます。

1. エージェントデーモンを自動的に起動するには、**$HOME/.dtprofile** スクリプトの最後に次の行を追加します。

```
if [ "$SSH_AUTH_SOCK" = "" -a -x /usr/bin/ssh-agent ]; then
                eval `/usr/bin/ssh-agent`
fi
```

2. **CDE** セッションを終了するときに **Secure Shell** エージェントデーモンを終了するには、**$HOME/.dt/sessions/sessionexit** スクリプトに次の行を追加します。

```
if [ "$SSH_AGENT_PID" != "" -a -x /usr/bin/ssh-agent ]; then
                /usr/bin/ssh-agent -k
fi
```

   このエントリにより、CDE セッションが終了したあとで、Secure Shell エージェントは使用できなくなります。

3. **Solaris Secure Shell** セッションを起動します。

```
myLocalHost% ssh myRemoteHost
```

   パスフレーズのプロンプトは表示されません。

▼ Solaris Secure Shell のポート転送を使用する方法

リモートホストに転送されるローカルポートを指定することができます。指定すると、ソケットはローカル側で、そのポートを待機します。このポートからリモートホストへの接続は、セキュリティ保護されたチャネルを介して行われます。たとえば、IMAP4 で電子メールを安全にリモート受信するためにポート 143 を指定します。また、リモート側のポートを指定することもできます。

注 – Secure Shell ポート転送では TCP 接続を使用する必要があります。Secure Shell は UDP 接続をサポートしていません。

- ローカルポートを転送元として設定するには、**2** つのポートを指定します。待機するローカルポートを指定し、転送先のリモートホストとポートを指定します。

```
myLocalHost% ssh -L localPort:remoteHost:remotePort
```

- リモートポートをセキュリティ保護された接続の受信元として設定するには、**2** つのポートを指定します。待機するリモートポートを指定し、転送先のローカルホストとポートを指定します。

  ```
  myLocalHost% ssh -R remotePort:localHost:localPort
  ```

例 — ローカルポート転送を使用してメールを受信する

次の例は、ローカルポート転送を使用して、リモートサーバーからのメールを安全に受信する方法を示しています。

```
myLocalHost% ssh -L 9143:myRemoteHost:143 myRemoteHost
```

このコマンドは、`myLocalHost` のポート `9143` を `myRemoteHost` のポート `143` (IMAP v2 のサーバーポート) に接続を転送します。ユーザーがメールアプリケーションを起動するときは、ローカルポート番号を指定する必要があります。`dtmail` コマンドの使用例を 図 11–1 に示します。

この例と 197 ページの「例 — リモートポート転送を使用してファイアウォールの外部と通信する」で使用されている `localhost` は、ユーザーのローカルホストを指定するキーワードです。`localhost` キーワードと `myLocalHost` を混同しないでください。`myLocalHost` 変数は、この章の例で使用するローカルホストを表す仮のホスト名です。

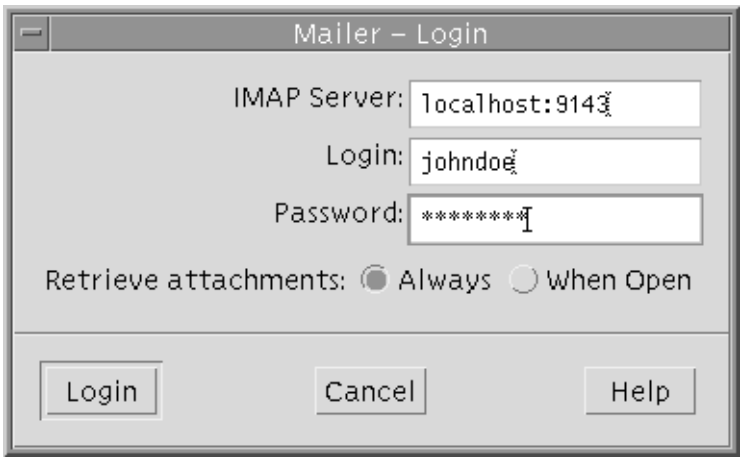


図 11–1 電子メールに使用するポート転送を指定する

例 — リモートポート転送を使用してファイアウォールの外部と通信する

この例では、エンタープライズ環境のユーザーが、外部ネットワーク上のホストから企業のファイアウォール内部のホストに接続を転送する方法を示しています。

```
myLocalHost% ssh -R 9022:myLocalHost:22 myOutsideHost
```

このコマンドは、myOutsideHost 上のポート 9022 への接続をローカルホストのポート 22 (sshd サーバー) に転送します。

```
myOutsideHost% ssh -p 9022 localhost
myLocalHost%
```

このコマンドを使用すると、リモート転送接続が確立されたあとで、ssh コマンドを使用してリモートホストから安全に接続できます。

▼ Solaris Secure Shell を使用してファイルをコピーする方法

scp コマンドを使用して、暗号化されたファイルをホスト間でコピーします。暗号化されたファイルは、ローカルホストとリモートホストとの間、または 2 つのリモートホスト間でコピーできます。scp コマンドは、rcp コマンドと同様に動作しますが、パスワードを要求するプロンプトを表示する点で異なります。詳細は、scp(1) のマニュアルページを参照してください。

1. セキュリティ保護されたコピープログラムを起動します。

   ソースファイル、リモートコピー先のユーザー名、およびコピー先ディレクトリを指定します。

   ```
   myLocalHost% scp myfile.1 johndoe@myRemoteHost:~
   ```

2. **Solaris Secure Shell** パスフレーズを要求するプロンプト表示で、パスフレーズを入力します。

   ```
   Enter passphrase for key '/home/johndoe/.ssh/id_rsa':
   <Return キーを押す>
   myfile.1       25% |*******                       |    640 KB  0:20 ETA
   myfile.1
   ```

   パスフレーズを入力すると、進行状況インジケータが表示されます。上記の出力の 2 行目が進行状況インジケータです。進行状況インジケータには、次の項目が表示されます。

   - ファイル名
   - その時点で転送が完了しているファイルの割合 (%)
   - 転送が完了した割合 (%) に対応したアスタリスク (*)
   - 転送が完了したデータの量
   - ファイル全体が転送されるまでの推定時間 (ETA)。推定残り時間

sftp コマンドを使用したファイルのリモート転送

sftp コマンドの動作は、ftp と類似していますが、使用するサブコマンドセットが異なります。次の表に、代表的なサブコマンドを示します。

表 11–3 対話型 sftp のサブコマンド

カテゴリ	サブコマンド	説明
移動	cd *path*	リモートディレクトリを *path* に変更する
	lcd *path*	ローカルディレクトリを *path* に変更する
所有権	chgrp *group file*	*file* のグループを *group* (数値 GID) に変更する
	chmod *mode file*	*file* のアクセス権を変更する
ファイルのコピー	get *remote_file* [*local-path*]	リモートファイルを取得し、それをローカルホストに格納する
	put *local_file* [*remote_path*]	ローカルファイルをリモートホストに格納する
	rename *old_filenew_file*	ローカルファイルの名前を変更する
ディレクトリのリスト表示	ls [*path*]	リモートディレクトリの内容を一覧表示する
ディレクトリの作成	mkdir *path*	リモートディレクトリを作成する
その他	exit、quit	sftp コマンドを終了する

▼ ファイアウォール外部のホストにデフォルト接続を設定する方法

Solaris Secure Shell を使用して、ファイアウォール内部のホストからファイアウォール外部のホストに接続することができます。接続するには、構成ファイル内またはコマンド行オプションに ssh のプロキシコマンドを指定します。詳細は、201 ページの「例 — コマンド行からファイアウォール外部のホストに接続する」を参照してください。

通常は、個人用構成ファイル $HOME/.ssh/config または管理構成ファイル/etc/ssh/ssh_config を使用して、ssh の対話操作をカスタマイズできます (ssh_config(4) のマニュアルページを参照)。プロキシコマンドには2 種類あり、一方が HTTP 接続用、もう一方が SOCKS5 接続用です。

1. 構成ファイルにプロキシコマンドとホストを指定します。

   次の構文を使用して、必要なプロキシコマンドとホストの数に応じて行を追加します。

   ```
   [Host outside_host]
   ProxyCommand proxy_command [-h proxy_server] \
   [-p proxy_port] outside_host|%h outside_port|%p
   ```

   次に、各引数について説明します。

`Host` *outside_host*	コマンド行でリモートホスト名を指定した場合、プロキシコマンド指定をインスタンスに限定する。*outside_host* でワイルドカードを使用した場合、一連のホストに対して指定が適用される
proxy_command	プロキシコマンドを指定する。次のいずれかを指定できる。 ■ HTTP 接続の場合は、`/usr/lib/ssh/ssh-http-proxy-connect` ■ SOCKS5 接続の場合は、`/usr/lib/ssh/ssh-socks5-proxy-connect`
`-h` *proxy_server* と `-p` *proxy_port*	これらのオプションは、プロキシサーバーとプロキシポートをそれぞれ指定する。これらのオプションは、`HTTPPROXY`、`HTTPPROXYPORT`、`SOCKS5_PORT`、`SOCKS5_SERVER`、`http_proxy` などの、プロキシサーバーとプロキシポートを指定するどのような環境変数よりも優先される。`http_proxy` 変数は URL を指定する。これらのオプションを指定しない場合、適切な環境変数を設定する必要がある。`ssh-socks5-proxy-connect(1)` と `ssh-http-proxy-connect(1)` のマニュアルページを参照
outside_host	接続先のホストを指定する。`%h` を使うとコマンド行からホストを指定できる。
outside_port	接続先のポートを指定する。`%p` を使うとコマンド行からポートを指定できる。`Host` *outside_host* オプションを使わずに `%h` と `%p` を指定した場合、`ssh` コマンドが呼び出されるたびに、引数に指定されたホストにプロキシコマンドが適用されます。

2. 外部のホストを指定して、**Solaris Secure Shell** を実行します。
   たとえば、次のように入力します。

   ```
   myLocalHost% ssh myOutsideHost
   ```

   このコマンドは、個人用構成ファイル内で `myOutsideHost` のプロキシコマンド指定を検索します。指定が検出されない場合、このコマンドは、システム全体の構成ファイル `ssh_config` から検索します。プロキシコマンドが `ssh` に置き換わります。

例 — コマンド行からファイアウォール外部のホストに接続する

ssh コマンドの -o オプションには、ssh 構成ファイル内で使用できる任意の行を入力できます。ここでは、前述の例のプロキシコマンド指定を使用します。

1. 構成ファイルにプロキシコマンドとホストを指定します。
2. ssh コマンドを実行します。-o オプションにプロキシコマンドを指定してください。たとえば、次のように入力します。

   ```
   % ssh -o'Proxycommand=/usr/lib/ssh/ssh-http-proxy-connect \
   -h myProxyServer -p 8080 myOutsideHost 22' myOutsideHost
   ```

   このコマンドは、ssh を HTTP プロキシコマンドに置き換え、ポート 8080 と myProxyServer をプロキシサーバーとして使用し、 myOutsideHost のポート 22 に接続します。

第 12 章

Solaris Secure Shell の管理 (参照)

この章では、システム管理者から見た Solaris Secure Shell の機能とその構成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。

標準的な Solaris Secure Shell セッション

Solaris Secure Shell デーモン (sshd) は通常、/etc/init.d/sshd スクリプトによりブート時に起動されます。デーモンは、クライアントからの接続を待機します。Solaris Secure Shell セッションは、ssh、scp、または sftp コマンドが実行されると開始します。接続を受信するたびに、新しい sshd デーモンがフォークされます。フォークされたデーモンは、鍵の交換、暗号化、認証、コマンドの実行、およびクライアントとのデータ交換を行います。Secure Shell セッションの特性は、クライアント構成ファイルとサーバー構成ファイルによって決定されます。また、コマンド行パラメータを使用することもできます。クライアントとサーバーは、相互に認証する必要があります。認証に成功したあと、ユーザーはコマンドをリモートで実行でき、ホスト間でデータをコピーできます。

セッションの特性

サーバー側の sshd デーモンの動作は、/etc/ssh/sshd_config ファイルのキーワードを設定することで変更できます。sshd が起動しているときに、コマンド行オプションを使用することもできます。たとえば、sshd_config により、サーバーにアクセスするときに使用する認証タイプを変更できます。

クライアント側の動作は、Solaris Secure Shell のパラメータで変更できます。パラメータの優先順位は次のとおりです。

- コマンド行オプション
- ユーザーの構成ファイル (`$HOME/.ssh/config`)
- システム全体の構成ファイル (`/etc/ssh/ssh_config`)

たとえば、`Cipher` として `blowfish` に設定されているシステム全体の構成を変更するには、コマンド行に `-c 3des` を指定します。

認証

Solaris Secure Shell の認証は、次の手順で行われます。

1. ユーザーが、`ssh`、`scp`、または `sftp` コマンドを実行します。
2. クライアントとサーバーは、共有セッション鍵に合意します。

   v1 のリモートホストは、ホスト (RSA) 鍵とサーバー (RSA) 鍵をクライアントに送信します。サーバー鍵は通常、1 時間ごとに生成され、メモリーだけに格納されます。クライアントは、リモートホスト鍵がローカルホストの `$HOME/.ssh/known_hosts` ファイルに格納されていることを確認します。次にクライアントは、256 ビットの乱数を生成して、リモートホストのホスト鍵とサーバー鍵を暗号化します。暗号化された乱数は、セッション内での後続のすべての通信を暗号化するときに、セッション鍵として使用されます。

   v2 のリモートホストは、ホスト鍵の DSA を使用し、サーバー鍵を生成しません。代わりに共有セッション鍵を、Diffie-Hellman 方式で合意したときに生成します。
3. ローカルホストとリモートホストは、相互に認証します。

   v1 のクライアントでは、`.rhosts`、`rhosts` とRSA、RAS challenge-response、またはパスワードベースの認証を使用できます。v2 では、`.rhosts`、DSA、およびパスワードベースの認証だけを使用できます。

コマンドの実行とデータの転送

認証が完了したら、ユーザーは通常、シェルまたはコマンド実行を要求して Solaris Secure Shell を使用します。`ssh` のオプションを使用すれば、擬似端末を配置したり、X11 あるいは TCP/IP の接続を転送したり、セキュリティ保護された接続上で `ssh-agent` を有効にしたりすることができます。ユーザーセッションの基本手順は次のとおりです。

1. ユーザーがシェルまたはコマンドの実行を要求し、セッションモードを開始します。

   セッションモードでは、データは端末を通してクライアント側に送受信され、シェルまたはコマンドを介してサーバー側に送受信されます。
2. ユーザープログラムを終了します。

3. すべての X11 および TCP/IP 接続の転送を停止します。ただし、既存の X11 と TCP/IP 接続は、すべて開いたままです。
4. サーバーはコマンド終了をクライアントに送信し、両サイドの接続が終了します。

Solaris Secure Shell を構成する

Solaris Secure Shell セッションの特性は、構成ファイルで変更できます。コマンド行のオプションを使用することで、このデフォルト設定を変更できます。

Solaris Secure Shell クライアントの構成

ほとんどの場合、Solaris Secure Shell セッションのクライアント側の特性は、システムの構成ファイル (`/etc/ssh/ssh_config`) によって決定されます。このファイルは、システム管理者が設定します。`$HOME/.ssh/config` 内のユーザーの構成は、システムの構成ファイル内の設定より優先されます。さらに、コマンド行での指定は、これらの構成ファイルより優先されます。

コマンド行オプションはクライアント側の要求ですが、 サーバー側の `/etc/ssh/sshd_config` ファイルによって許可または拒否されます (`ssh_config(4)` のマニュアルページを参照)。構成ファイルのキーワードとコマンドオプションについては、次の節で説明します。詳細は、`ssh(1)`、`scp(1)`、`sftp(1)`、および `ssh_config(4)` のマニュアルページを参照してください。2 つのユーザー構成ファイルの中で `Host` キーワードには、ホストまたはワイルドカード表現を指定します。このキーワードは、次の `Host` キーワードが出現するまで、後続のすべてのキーワードに適用されます。

ホスト固有のパラメータ

ローカルホストごとに異なる Solaris Secure Shell 特性を使用する場合、システム管理者は `/etc/ssh/ssh_config` ファイルにホストまたはその正規表現形式とともにいくつかのパラメータセットを定義します。ファイル内のエントリを、 `Host` キーワードでグループ化してください。`Host` キーワードを使用しない場合、クライアント構成ファイル内のエントリは、ユーザーが使用しているローカルホストに適用されます。

クライアント側の認証パラメータ

認証方式を決定するには、次のキーワードのいずれかに「yes」を設定します。

- `DSAAuthentication`
- `PasswordAuthentication`

- RhostsAuthentication
- RhostsRSAAuthentication

キーワード UseRsh は、rlogin と rsh コマンドを使用することを指定します。このキーワードは、多くの場合、Secure Shell がサポートされないときに使用します。

キーワード Protocol には、Solaris Secure Shell プロトコルのバージョン (v1 または v2) を設定します。両方のバージョンを指定する場合は、コンマで区切ります。最初のバージョンの使用に失敗した場合は、2 番目のバージョンが使用されます。

キーワード IdentityFile には、ユーザーの非公開鍵を格納する代替ファイル名を指定します。

キーワード Cipher には、v1 の暗号化アルゴリズム (blowfish または 3des) を指定します。キーワード Ciphers には、v2 の暗号化アルゴリズム (3des-cbc、blowfish-cbc、および aes128-cbc) の優先順序を指定します。ssh および scp コマンドの -c オプションは、コマンド行で暗号化アルゴリズムを指定するときに使用します。

既知のホストファイルパラメータ

既知のホストファイル (/etc/ssh/ssh_known_hosts および $HOME/.ssh/known_hosts) には、すべてのホストの公開鍵が含まれています。この公開鍵は、クライアントが Solaris Secure Shell を使用してホストと通信するときに使用されます。GlobalKnownHostsFile キーワードには、/etc/ssh/ssh_known_hosts の代替ファイルを指定します。UserKnownHostsFile キーワードには、$HOME/.ssh/known_hosts の代替ファイルを指定します。

StrictHostKeyChecking キーワードを指定した場合は、新しいホストを既知のホストファイルに追加するときは手動で行う必要があります。また、公開鍵が変更されたり、公開鍵が既知のホストファイルに存在しないホストは拒否されます。キーワード CheckHostIP を指定すると、DNS のスプーフィングによって鍵が変更された場合に、既知のホストファイルに指定されているホストの IP アドレスが確認されます。

クライアント側の X11 転送とポート転送パラメータ

LocalForward キーワードには、リモートホスト上の特定のポートに転送するローカル TCP/IP ポートを指定します。セキュリティ保護されたチャネルが使用されます。GatewayPorts キーワードを指定すると、ローカル転送されたポートにリモートホストが接続することを可能にします。

ssh コマンドでポート転送するときは、次のオプションを使用できます。

- -L - リモートホスト上の特定のポートに転送するローカルポートを指定する
- -R - ローカルホストおよび特定のポートに転送するリモートポートを指定する

ForwardX11 キーワードを指定すると、 DISPLAY 環境変数が設定されたリモートホストに X11 接続がリダイレクトされます。XAuthLocation キーワードには、xauth プログラムの場所を指定します。

クライアント側の接続とその他のパラメータ

NumberOfPasswordPrompts キーワードには、パスワードを要求する回数を指定します。指定した回数が終了すると、Solaris Secure Shell が終了します。ConnectionAttempts キーワードには、接続試行回数 (1 秒間に 1 回試行) を指定します。この回数が終了すると、Solaris Secure Shell が終了します。ただし、FallBackToRsh キーワードが設定されている場合は、rsh に戻ります。

Compression キーワードを指定すると、転送データが圧縮されます。CompressionLevel キーワードには、圧縮レベル (1 - 9) を設定します。圧縮率とそれを行う時間にはトレードオフの関係があります。

User には、代替ユーザー名を指定します。Hostname には、リモートホストの代替名を指定します。ProxyCommand には、Solaris Secure Shell を起動する代替コマンド名を指定します。プロキシサーバーに接続できるコマンドはすべてここに指定できます。指定するコマンドは、標準入力から読み込んで標準出力に書き込む必要があります。

Batchmode を指定すると、パスワードプロンプトが無効になります。パスワードプロンプトは、スクリプトなどのバッチジョブに使用します。

KeepAlive を指定すると、ホストがクラッシュしたときに、ネットワークの問題が発生したことを示すメッセージが出力されます。LogLevel には、ssh メッセージの冗長レベルを設定します。

EscapeChar には、特殊文字をプレーンテキストとして表示するときに、接頭辞として使用する単一文字を定義します。

Solaris Secure Shell サーバーの構成

サーバー側の Solaris Secure Shell セッションの特性は、/etc/ssh/sshd_config ファイルによって管理されます。このファイルは、システム管理者が設定します。

サーバー側の認証パラメータ

使用できる認証方式のキーワードは、以下のとおりです。

- DSAAuthentication
- PasswordAuthentication
- RhostsAuthentication
- RhostsRSAAuthentication

- RSAAuthentication

HostKey および HostDSAKey には、デフォルトのファイル名を使用しないときに、ホスト公開鍵を格納するファイルを指定します。KeyRegenerationInterval には、サーバー鍵を再生成する頻度を指定します。

Protocol には、バージョンを指定します。Ciphers には、v2 の暗号化アルゴリズムを定義します。ServerKeyBits には、サーバーの鍵のビット数を定義します。

ポートと転送パラメータ

AllowTCPForwarding には、TCP 転送を許可するかどうかを指定します。

GatewayPorts を指定すると、クライアントに転送されたポートにリモートホストが接続されます。Port には、sshd が待機するポート番号を指定します。ListenAddress には、sshd が待機するローカルアドレスを指定します。ListenAddress を指定しない場合、 sshd はデフォルトですべてのアドレスで待機します。

X11Forwarding を指定すると、X11 転送が有効になります。X11DisplayOffset には、転送に使用できる最初の表示番号を指定します。このキーワードを指定すると、sshd が実際の X11 サーバーに干渉しなくなります。XAuthLocation には、xauth プログラムの場所を指定します。

セッション制御パラメータ

KeepAlive を指定すると、接続が切断したときおよびホストがクラッシュしたときに、メッセージが表示されます。LogLevel には、sshd メッセージの冗長レベルを設定します。SyslogFacility には、ログに記録する sshd メッセージの機能コードを指定します。

サーバー接続とその他のパラメータ

AllowGroups、AllowUsers、DenyGroups、および DenyUsers キーワードを使用して、ssh を使用できるユーザーと使用できないユーザーを制御します。

LoginGraceTime、MaxStartups、PermitRootLogin、および PermitEmptyPasswords キーワードを使用して、ログインするユーザーを制御します。StrictModes を指定すると、sshd は、ユーザーがログインする前にファイルのモードと所有権およびホームディレクトリを確認します。UseLogin には、対話型ログインセッションで login を使用するかどうかを指定します。このキーワードは有効にする必要はありません。Solaris 環境ではできるだけ使用しないでください。

Subsystem を指定すると、sftp に使用するファイル転送デーモンが構成されます。

サイト全体で既知のホストを管理する

ホスト間の対話を安全に行うには、ローカルホストの /etc/ssh/ssh_known_hosts ファイルにサーバーの公開鍵を格納する必要があります。 /etc/ssh/ssh_known_hosts ファイルを更新するときに、スクリプトを使用すると簡単に行うことができますが、セキュリティが大幅に低下するため、できるだけ使用しないでください。

/etc/ssh/ssh_known_hosts ファイルを配布するときは、次のようなセキュリティ保護されたメカニズムで行う必要があります。

- Solaris Secure Shell、IPsec、または Kerberos を使用した ftp などのセキュリティ保護された接続を使用して、既知の信頼できるマシンから配布する
- システムインストール時に配布する

known_hosts ファイルに偽の公開鍵を挿入してアクセス権を取得しようとする侵入者をできる限り阻止するには、 ssh_known_hosts ファイルの既知の信頼できるソースとして、JumpStart サーバーを使用します。ssh_known_hosts ファイルは、インストール中に配布できるほか、各ホストで定期的にスクリプト (scp を使用して最新バージョンを取り込む) を実行して配布することもできます。この方法は、JumpStart サーバーの公開鍵がすでに各ホストに保管されているため安全です。

Solaris Secure Shell ファイル

次の表は、重要な Solaris Secure Shell ファイルと推奨される UNIX アクセス権を示します。

表 **12–1** Solaris Secure Shell ファイル

ファイル名	説明	推奨アクセス権と所有者
/etc/ssh/sshd_config	sshd (Secure Shell デーモン) の構成データを含む	-rw-r--r-- root
/etc/ssh/ssh_host_key	ホスト非公開鍵を含む	-rw------- root
/etc/ssh_host_key.pub	ホスト公開鍵を含む。ホスト鍵をローカル known_hosts ファイルにコピーするときに使用する	-rw-r--r-- root

表 **12–1** Solaris Secure Shell ファイル　(続き)

ファイル名	説明	推奨アクセス権と所有者
/var/run/sshd.pid	Secure Shell デーモン sshd のプロセス ID を含む。このデーモンは接続を待機する (複数のデーモンが存在する場合は、起動された最後のデーモンを含む)	rw-r--r-- root
$HOME/.ssh/authorized_keys	v1 の場合は、ユーザーアカウントにログインするときに使用できる RSA 鍵を指定する。v2 の場合は、使用できる DSA と RSA 鍵を指定する	-rw-rw-r-- johndoe
/etc/ssh/ssh_known_hosts	すべてのホストのホスト公開鍵を含む。クライアントはこれらの公開鍵を使用して、セキュリティ保護された通信を行う。このファイルはシステム管理者が用意する	-rw-r--r-- root
$HOME/.ssh/known_hosts	すべてのホストのホスト公開鍵を含む。クライアントはこれらの公開鍵を使用して、セキュリティ保護された通信を行う。このファイルは自動的に管理される。ユーザーが未知のホストに接続すると、リモートホスト鍵がファイルに追加される	-rw-r--r-- johndoe
/etc/nologin	このファイルが存在する場合、sshd は root ログイン以外のユーザーを拒否する。ファイルの内容は、ログインしようとするユーザーに表示される	-rw-r--r-- root
$HOME/.rhosts	ホスト名とユーザー名のペアを含む。ユーザーは、対応するホストにパスワードを使用しないでログインできる。このファイルは、Secure Shell、rlogind、および rshd デーモンで使用される	-rw-r—r-- johndoe
$HOME/.shosts	ホスト名とユーザー名のペアを含む。ユーザーは、対応するホストにパスワードを使用せずに Secure Shell だけを使用してログインできる	-rw-r—r-- johndoe
/etc/hosts.equiv	.rhosts 認証と Secure Shell 認証で使用されるホストを含む	-rw-r--r-- root
/etc/ssh/shosts.equiv	Secure Shell 認証で使用されるホストを含む	-rw-r--r-- root
$HOME/.ssh/environment	ログイン時に割り当てを行う初期化に使用する	-rw------- johndoe
$HOME/.ssh/rc	ユーザーのシェルを起動前に実行する初期化ルーチン	-rw------- johndoe

表 **12–1** Solaris Secure Shell ファイル　(続き)

ファイル名	説明	推奨アクセス権と所有者
/etc/ssh/sshrc	システム管理者がすべてのユーザー用に用意したホスト固有の初期化ルーチン	-rw-r--r-- root

次の表は、主要な Solaris Secure Shell コマンドの要約です。

表 **12–2** Solaris Secure Shell コマンド

コマンド名	説明
ssh	リモートマシンにログインし、リモートマシン上でコマンドを実行するプログラム。このコマンドは、 rlogin と rsh に代わるコマンドである。セキュリティ保護されていないネットワークを介して信頼できないホスト間でセキュリティ保護された暗号化通信を行うときに使用する。X11 接続と任意の TCP/IP ポートも、セキュリティ保護されたチャネルを介して転送される
sshd	Secure Shell が待機するデーモン。このデーモンは、クライアントからの接続を待機する。セキュリティ保護されていないネットワークを介して 2 つの信頼できないホスト間でセキュリティ保護された暗号化通信を行うときに使用する。
ssh-keygen	ssh の認証鍵を生成および管理する
ssh-agent	公開鍵認証時に使用される非公開鍵を格納するプログラム。ssh-agent は、X セッションまたはログインセッションの開始時に起動する。ほかのすべてのウィンドウまたはプログラムは、ssh-agent プログラムのクライアントとして起動する。環境変数を使用すれば、ユーザーが ssh を使用してほかのマシンにログインするときに、エージェントを検出して認証に自動的に使用することができる
ssh-add	RSA または DSA ID (鍵) を認証エージェント ssh-agent に追加する
scp	ssh を使用してデータ転送するときに、ネットワーク上のホスト間でファイルを安全にコピーする。rcp と異なり、scp はパスワードまたはパスフレーズを要求する (認証に必要な場合)
sftp	対話型ファイル転送プログラム。ftp と同様に、暗号化された ssh トランスポートを介してすべての操作を実行する。sftp は、指定したホスト名に接続してログインし、対話型コマンドモードに入る

第 13 章

SEAM について

この章では、Sun Enterprise Authentication Mechanism (SEAM) について説明します。

SEAM とは

SEAM は、ネットワークを介してセキュリティ保護されたトランザクションを提供するクライアントおよびサーバーのアーキテクチャです。SEAM では、強力なユーザー認証とともに、データの完全性とデータの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。さらに SEAM を使用して、送受信するデータの完全性が検査され (「完全性」)、伝送時にデータが暗号化されます (「プライバシ」) 。SEAM を使用して、他のマシン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。SEAM は承認サービスも提供するため、システム管理者はサービスやマシンへのアクセスを制限できます。また、SEAM 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。

SEAM は「シングルサインオン」システムです。つまり、SEAM からセッションについて一度だけ認証を受ければ、そのセッションでは、それ以後のすべてのトランザクションが自動的に認証されます。いったん SEAM から認証されたユーザーは、`ftp`、`rsh` などの SEAM ベースのコマンドを使用したり、NFS ファイルシステム上のデータにアクセスしたりするたびに認証が要求されることはありません。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。

SEAM 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 を使用したことがあるユーザーは、SEAM にはすぐ慣れるはずです。Kerberos V5 はネットワークセキュリティの事実上の業界標準であるため、SEAM はほかのシステムとの相互運用性に優れています。つまり、SEAM は Kerberos V5 を使用するシステムと協調して動作するため、異機種システム混在のネットワークであってもトランザクションのセキュリティが保護されます。さらに SEAM 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の機能を使用できます。

注 – SEAM は、Kerberos V5 に準拠しており、Kerberos V5 との相互運用を実現するために設計されています。このマニュアルでは、「Kerberos」と「SEAM」という用語は、「Kerberos のレルム」や「SEAM ベースのユーティリティ」など、区別しないで使用されることがあります。また、「Kerberos」と「Kerberos V5」も区別しないで使用されています。 必要に応じて「Kerberos」と「Kerberos V5」を区別しています。

SEAM には、Solaris アプリケーションを実行するための柔軟性が備わっています。SEAM は、NFS サービス、`telnet`、および `ftp` などのネットワークサービスを SEAM ベースと SEAM ベースでない両方式の要求に対応できるように構成できます。このため、SEAM がインストールされていないシステムで動作する現在の Solaris アプリケーションも正しく動作します。もちろん、SEAM ベースのネットワーク要求だけを許可するように SEAM を設定することもできます。

さらに、他のセキュリティメカニズムが開発された場合には、アプリケーションで使用するセキュリティメカニズムを SEAM に限定しておく必要はありません。SEAM は、Generic Security Service (GSS) API にモジュールとして統合でき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、必要に応じたセキュリティメカニズムを使用できます。

SEAM の動作

この節では SEAM 認証システムの概要について説明します。詳細は、334 ページの「認証システムの動作方法」を参照してください。

ユーザーの観点からいえば、SEAM は、SEAM セッションが起動されたあとはほとんど意識しません。`rsh` や `ftp` などのコマンドは、通常の方式で適切に動作します。SEAM 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。

SEAM 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 ID や NFS などのサービス ID とな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。SEAM ベースのトランザクションを実行する (ほかのマシンへの

rlogin など) と、鍵配布センター (KDC) に対してチケットの要求が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そのユーザーを認証し、そのマシンへのアクセス権を許可するチケットを返します。「透過的」とは、明示的にチケットを要求する必要がないことを意味します。この要求は rlogin コマンドの中で行われます。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、別のクライアントが識別情報を仮定して rlogin を使用することはできません。

チケットには一定の属性が与えられています。たとえば、チケットには「転送可能」(新しい認証処理を行わなくても別のマシンで使用できる) や「遅延」(指定の日付まで有効にならない) の属性があります。どのユーザーがどの種類のチケットを取得できるかなど、チケットをどのように使用するかは、SEAM のインストールや管理の際に決める「方針」によって設定されます。

注 – 「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。この違いについては、334 ページの「SEAM によるサービスへのアクセス」で詳しく説明します。

次の節では、SEAM 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、後続の認証を準備する初期認証と、後続の認証の 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。

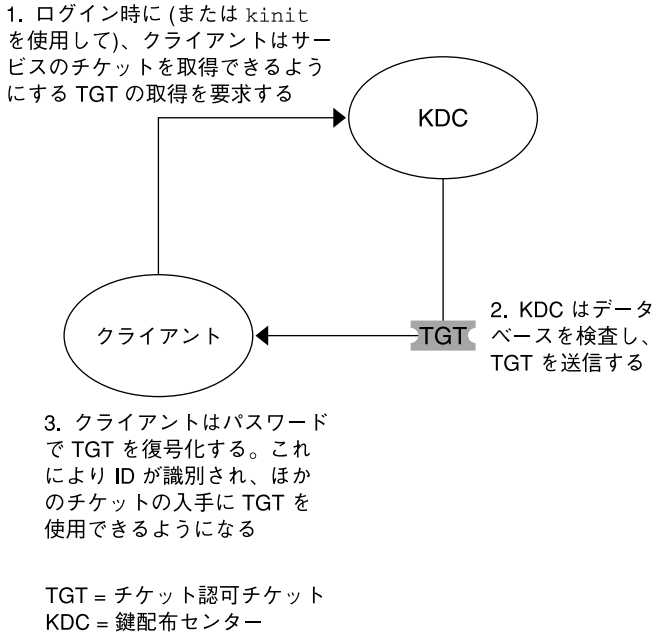


図 **13–1** SEAM セッションの初期認証

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 SEAM 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。

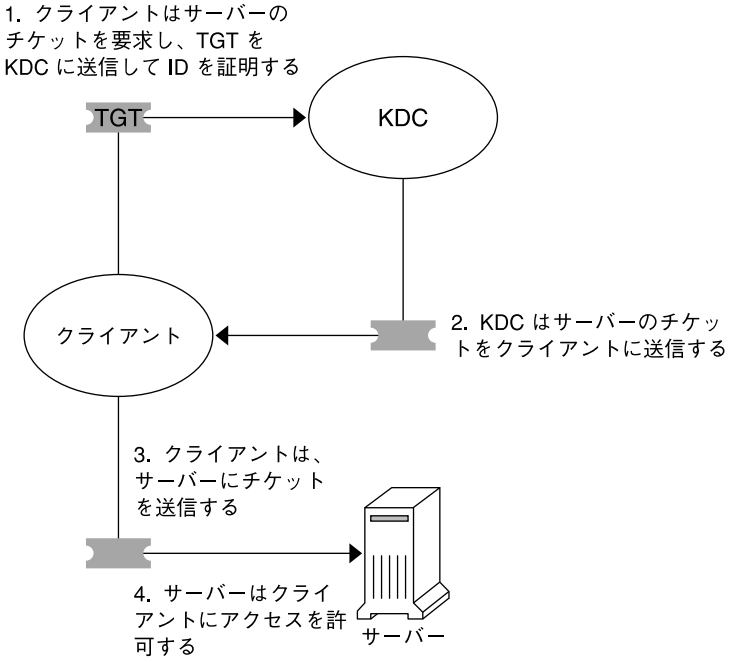
   TGT は、ほかの特定のサービスのチケットを取得するために必要です。TGT は、パスポートに似ています。パスポートと同様に、TGT はユーザーを識別して、さまざまなビザの取得をユーザーに許可します。ここでいうビザ (チケット) は、外国に入国するためのものではなく、リモートマシンやネットワークサービスにアクセスするためのものです。パスポートやビザと同様に、TGT などのチケットには有効期限があります。ただし、Kerberos コマンドは、ユーザーがパスポートを所有していることを通知し、ユーザーに代わってビザを取得します。ユーザー自身がトランザクションを実行する必要はありません。

   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。ユーザーはこのパスを任意のスキー場で提示して (期限切れでない場合)、そのスキー場の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。翌日別のスキー場に行った場合は、またパスを提示して、そのスキー場のリフトチケットを入手します。ただし、SEAM ベースのコマンドは、週末スキーパスを所有していることをユーザーに通知し、ユーザーに代わってリフトチケットを入手します。ユーザー自身がこのトランザクションを実行する必要はありません。
2. KDC は TGT を作成し、それを暗号化してクライアントに送信します。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. クライアントは、有効な TGT を入手したので、TGT が期限切れになるまで、`rlogin`、`telnet` などあらゆる種類のネットワーク操作チケットを要求できます。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。

後続の認証

クライアントが初期認証を受け取ると、各認証は次の図のように実行されます。



TGT = チケット認可チケット
KDC = 鍵配布センター

図 **13–2** サービスへのアクセスを取得する

1. クライアントは、特定のサービス (別のマシンに `rlogin` するなど) のチケットを KDC に要求するために、識別情報の証拠として自身の TGT を KDC に送信します。
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。

   たとえば、ユーザー `joe` が、必要とする krb5 認証と共有関係にある NFS ファイルシステムにアクセスするとします。このユーザーはすでに認証されている (すでに TGT を持っている) ため、そのファイルにアクセスを試みると、NFS クライア

ントシステムは NFS サービスのチケットを KDC から自動的および透過的に取得します。

たとえば、ユーザー joe がサーバー boston 上で rlogin を使用するとします。このユーザーはすでに認証されている (つまり、すでにチケット認可チケットを持っている) ため、rlogin コマンドの一部として自動的かつ透過的にチケットを取得します。このチケットが期限切れになるまで、このユーザーは必要に応じて boston に rlogin できます。 joe がマシン denver に rlogin する場合は、手順 1 の方法で別のチケットを入手します。

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、サーバーと KDC 間の通信は発生していないように見えます。しかし、サーバーは KDC と通信していて、最初のクライアントと同様に、KDC に自身を登録しています。わかりやすくするために、その部分は省略しています。

SEAM リモートアプリケーション

ユーザーは、次の SEAM ベースの (Kerberos 化された) コマンドを使用できます。

- ftp
- rcp
- rlogin
- rsh
- telnet

これらのアプリケーションは、同じ名前の 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 主体を使用し、Kerberos ベースのセキュリティを提供します。主体の詳細は、218 ページの「主体」を参照してください。

主体

SEAM 内のクライアントはその「主体 (プリンシパル)」で識別されます。主体とは、KDC がチケットを割り当てることができる一意の識別情報です。主体には、joe などのユーザー、または nfs、telnet などのサービスがあります。

主体名は慣習により「一次」、「インスタンス」、「レルム」という 3 つの部分からなります。joe/admin@ENG.EXAMPLE.COM は一般的な SEAM 主体の例です。各文字列は次の意味を持ちます。

- joe が一次です。一次には、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、host を指定することもできます。host を指定すると、ftp、rcp、rlogin などのさまざまなネットワークサービスを提供する、サービス主体として設定されます。

- 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 joe が必要に応じてシステム管理者の権限を使用する場合は、joe/admin と通常のユーザー ID を使い分けることができます。同じように、joe が 2 つのホストにアカウントを持っている場合、異なるインスタンスで 2 つの主体名を使用することができます (たとえば、joe/denver.example.com と joe/boston.example.com)。SEAM では、joe と joe/admin は全く別のプリンシパルとして扱われます。

  サービス主体では、インスタンスは完全修飾されたホスト名です。bigmachine.eng.example.com はこのようなインスタンスの例です。したがって、一次とインスタンスは、たとえば、ftp/bigmachine.eng.example.com や host/bigmachine.eng.example.com と表します。

- ENG.EXAMPLE.COM は SEAM レルムです。レルムについては、219 ページの「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。

- joe
- joe/admin
- joe/admin@ENG.EXAMPLE.COM
- ftp/host.eng.example.com@ENG.EXAMPLE.COM
- host/eng.example.com@ENG.EXAMPLE.COM

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 13–3 では、レルム間の関係を示します。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 つのレルム間の割り当てを定義する必要があります。SEAM では、レルム間で共通の認証が可能です。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。この機能は、レルム間認証と呼ばれます。

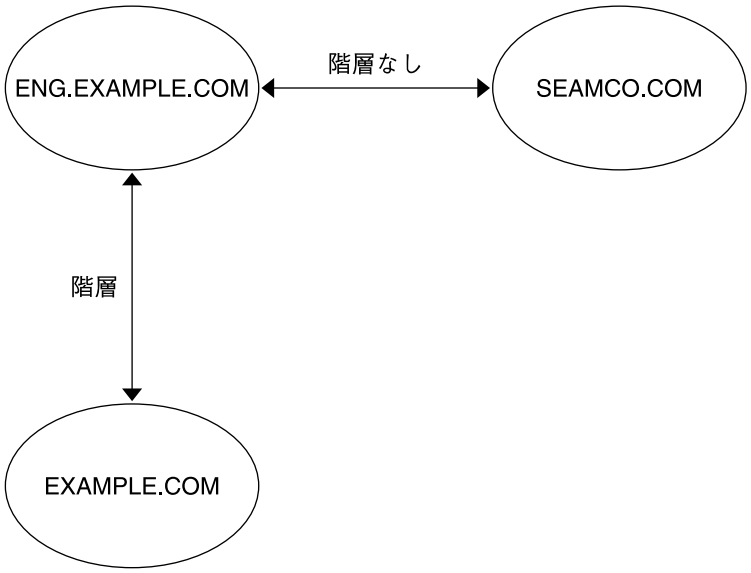


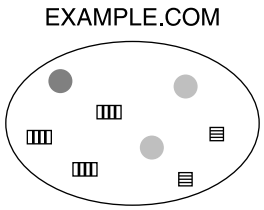
図 **13–3** レルム

レルムとサーバー

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。また各レルムには、主体データベースの重複コピーを保持する「スレーブ KDC サーバー」が少なくとも 1 つ必要です。マスター KDC サーバーおよびスレーブ KDC サーバーは、認証の確立に使用されるチケットを作成します。

レルムはさらに 2 種類の SEAM サーバーを持つことができます。SEAM ネットワークアプリケーションサーバーは、Kerberos 対応のアプリケーション (`ftp`、`telnet`、`rsh` など) へのアクセスを提供するサーバーです。レルムは、NFS サーバーも持つことができます。NFS サーバーは Kerberos 認証を使用する NFS サービスを提供します。SEAM 1.0 または SEAM 1.0.1 をインストールすると、レルムには、Kerberos アプリケーション (`ftp`、`telnet`、`rsh` など) からアクセスされる SEAM ネットワークのアプリケーションサーバーがインストールされます。

次の図では、レルムの構成例を示します。



マスター KDC

クライアント

スレーブ KDC

アプリケーションサーバー

図 **13–4** 一般的なレルム

SEAM セキュリティサービス

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

- 「完全性」 – 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、完全性は、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。完全性の確認は、データの暗号チェックサムによって行われます。完全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、セキュリティがさらに向上します。プライバシは、伝送データの完全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。プライバシにも認証が含まれます。

SEAM に組み込まれている Kerberos アプリケーションの中で実行時にセキュリティサービスを変更できるのは、`ftp` コマンドだけです。開発者は、RPCSEC_GSS プログラミングインタフェースを使用することにより、セキュリティサービスを選択可能な RPC ベースのアプリケーションを設計できます。

SEAM のリリース

SEAM 製品の構成要素は、4 つのリリースに組み込まれています。次の表は、各リリースに組み込まれている構成要素の一覧です。次の節では、すべての構成要素について説明します。

表 **13–1** SEAM リリースの内容

リリース名	内容
Solaris Easy Access Server (SEAS) 3.0 の SEAM 1.0	Solaris 2.6 および Solaris 7 用の SEAM の完全リリース
Solaris 8 の SEAM	SEAM クライアントソフトウェアのみ
Solaris 8 Admin Pack の SEAM 1.0.1	Solaris 8 用の SEAM KDC とリモートアプリケーション
Solaris 9 の SEAM	SEAM KDC とクライアントソフトウェアのみ
SEAM 1.0.2	Solaris 9 用の SEAM リモートアプリケーション

SEAM 1.0 の構成要素

MIT から提供される Kerberos V5 と同様に、SEAM には次の構成要素が含まれます。

- 鍵配布センター (KDC) (マスター)
  - 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`
- スレーブ KDC
- データベース管理プログラム – `kadmin`、`kadmin.local`
- データベース伝播ソフトウェア – `kprop`
- チケットの取得、表示、破棄を行うユーザープログラム – `kinit`、`klist`、`kdestroy` と SEAM パスワードを変更するユーザープログラム – `kpasswd`
- アプリケーション – `ftp`、`rcp`、`rlogin`、`rsh`、`telnet` およびこれらのアプリケーションのデーモン – `ftpd`、`rlogind`、`rshd`、`telnetd`
- 管理ユーティリティ – `ktutil`、`kdb5_util`
- いくつかのライブラリ

さらに、SEAM には次の構成要素が含まれています。

- SEAM 管理ツール (`gkadmin`) – KDC を管理する。システム管理者は、この Java™ テクノロジベースの GUI を使用して、通常は `kadmin` コマンドで実行する作業を実行できる
- Pluggable Authentication Module (PAM) – PAM により、アプリケーションはさまざまな認証メカニズムを使用できる。PAM を使用すると、ログインとログアウトをユーザーが意識する必要がなくなる
- ユーティリティ (`gsscred`) とデーモン (`gssd`) – これらのプログラムは、UNIX ユーザー ID (UID) と主体名の割り当てに使用する。これらのプログラムが必要なのは、SEAM NFS サーバーがユーザーを識別するときに、主体名ではなく UNIX UID を使用しており、主体名と UNIX UID は形式が異なっているためである

- Generic Security Service Application Programming Interface (GSS-API) – アプリケーションは、この API を利用して、複数のセキュリティメカニズムを使用できる。新しいメカニズムを追加するたびに、アプリケーションをコンパイルし直す必要がない。GSS-API はマシンに依存しないため、インターネット上のアプリケーションに適している。GSS-API を使用すると、認証サービスだけでなく、完全性およびプライバシセキュリティサービスをアプリケーションに組み込むことができる
- RPCSEC_GSS Application Programming Interface (API) – NFS サービスが Kerberos 認証を使用することができる。RPCSEC_GSS は、使用しているメカニズムに依存しないセキュリティサービスを提供する新しいセキュリティフレーバである。RPCSEC_GSS は、GSS-API 層の「最上位」に位置している。GSS_API ベースのセキュリティメカニズムは、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る
- 事前構成手順 – SEAM のインストールおよび構成パラメータを設定し、SEAM インストールを自動化できる。この手順は、特に複数のインストールを行うときに適している
- カーネルの変更 – パフォーマンスを向上させることができる

Solaris 8 の SEAM 構成要素

Solaris 8 に含まれている SEAM はクライアント側の部分だけで、SEAM の構成要素の多くは含まれていません。Solaris 8 が動作するシステムであれば、SEAM を別にインストールしなくても SEAM クライアントとしては動作します。これらの SEAM クライアント機能を使用するには、SEAS 3.0 または Solaris 8 Admin Pack、MIT ディストリビューション、あるいは Windows 2000 の KDC をインストールする必要があります。クライアント側の構成要素は、構成済み KDC がないとチケットを配布できません。Solaris 8 には、次の構成要素が含まれます。

- チケットを取得、表示、破棄するユーザープログラム – `kinit`、`klist`、`kdestroy`
- SEAM パスワードを変更するユーザープログラム – `kpasswd`
- 鍵テーブル管理ユーティリティ – `ktutil`
- PAM の拡張 – アプリケーションはさまざまな認証メカニズムを使用できる。PAM を使用すると、ログインとログアウトをユーザーが意識する必要がなくなる
- GSS_API プラグイン – Kerberos プロトコルおよび暗号サポートを提供する
- NFS クライアントおよびサーバーのサポート

SEAM 1.0.1 の構成要素

SEAM 1.0.1 には、Solaris 8 に含まれていない SEAM 1.0 の構成要素がすべて含まれています。次の構成要素が含まれています。

- 鍵配布センター (KDC) (マスター)

    - 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`
- スレーブ KDC
- データベース管理プログラム – `kadmin`、`kadmin.local`
- データベース伝播ソフトウェア – `kprop`
- アプリケーション – `ftp`、`rcp`、`rlogin`、`rsh`、`telnet` および これらのアプリケーションのデーモン – `ftpd`、`rlogind`、`rshd`、`telnetd`
- 管理ユーティリティ – `kdb5_util`
- SEAM 管理ツール (`gkadmin`) – KDC を管理する。システム管理者は、この Java テクノロジベースの GUI を使用して、通常は `kadmin` コマンドで実行する操作を実行できる
- 事前構成手順 – SEAM のインストールおよび構成パラメータを設定し、SEAM インストールを自動化できる。この手順は、特に複数のインストールを行うときに適している
- いくつかのライブラリ

Solaris 9 の SEAM 構成要素

Solaris 9 には、リモートアプリケーションと事前構成手順を除いて、SEAM 1.0 の構成要素がすべて含まれています。

SEAM 1.0.2 の構成要素

SEAM 1.0.2 には、リモートアプリケーションが含まれています。SEAM 1.0 の構成要素のうちで、Solaris 9 リリースに組み込まれていないのはこれらのアプリケーションだけです。リモートアプリケーションの構成要素は次のとおりです。

- クライアントアプリケーション – `ftp`、`rcp`、 `rlogin`、`rsh`、および `telnet`
- サーバーデーモン – `ftpd`、 `rlogind`、`rshd`、および `telnetd`

第 14 章

SEAM の計画

この章は、SEAM のインストールとシステム管理を行うシステム管理者を対象としています。この章では、SEAM をインストールまたは構成する前に、システム管理者が解決しておく必要があるインストールと構成の問題について説明します。

システム管理者やテクニカルサポート担当者が解決する必要がある問題は次のとおりです。

SEAM を計画する理由

SEAM 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。初期インストール後に構成を変更することもできますが、新しいクライアントがシステムに追加されるにつれて、変更が困難になります。また、変更によっては、すべてのシステムを再インストールしなければならないことがあります。このため、SEAM の構成を計画するときは、長期的な目標を考慮することをお勧めします。

レルム

レルムは、ドメインに似た論理ネットワークです。レルムには、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 ドメイン名を設定する場合と同様に、レルム名、レルムの数、および各レルムの大きさは、SEAM を構成する前に解決する必要があります。また、レルム間認証を行う場合は、レルム間の関係も定義する必要があります。

レルム名

レルム名には、任意の ASCII 文字列を使用できます。レルム名には通常、DNS ドメイン名と同じ名前を大文字で指定します。この命名規則を利用すると、すでに使い慣れている名前を使用しながら、SEAM のレルム名と DNS 名前空間のドメイン名を区別することができます。DNS を使用しない場合、または別の文字列を使用する場合は、任意の文字列を使用できます。ただし、構成プロセスがより複雑になります。レルム名を付けるときは、標準のインターネット命名構造に準拠することをお勧めします。

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る。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る
  - 各クライアントが生成する SEAM 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ない
- クライアントとの距離。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
- KDC としてインストールできるホスト数。各レルムには、複数の KDC サーバー (マスターとスレーブ) が必要である

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。このとき、階層チェーン内のすべてのレルムが適切に構成されてい

る必要があります。自動パスを利用すると、管理負荷を軽減することができます。ただし、ドメインのレベルが多い場合は、多くのトランザクションが発生するため、デフォルトのパスは使用しないことをお勧めします。

ドメイン間を直接接続することもできます。直接接続は、2 つの階層ドメイン間にレベルが多すぎる場合または階層関係が設定されていない場合に、使用します。直接接続は、使用するすべてのホストの /etc/krb5/krb5.conf ファイルに接続を定義する必要があります。このため、追加作業が必要になります。概要については、219 ページの「レルム」を参照してください。複数のレルムを構成する手順については、240 ページの「レルム間認証の構成」を参照してください。

ホスト名のレルムへの割り当て

ホスト名のレルム名への割り当ては、krb5.conf ファイルの domain_realm セクションに定義します。これらの割り当ては、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。詳細は、krb5.conf(4) のマニュアルページを参照してください。

クライアントとサービス主体の名前

SEAM を使用する場合、DNS サービスを事前に構成して、すべてのホスト上ですでに実行していることを強くお勧めします。DNS を使用する場合は、システム全体で有効または無効にする必要があります。DNS を有効にした場合、主体名には、各ホストの完全指定ホスト名 (FQDN) を含める必要があります。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 であった場合、ホストの主体名は host/boston.example.com@EXAMPLE.COM にする必要があります。このマニュアルの例では、各ホストの FQDN を使用しています。

ホストの FQDN を含む主体名は、/etc/resolv.conf ファイルの DNS ドメイン名を表す文字列と一致していることが重要です。SEAM では、主体に FQDN を入力するときに、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 ドメイン名には、example.com や Example.COM などの形式が使用できますが、ホストの主体名は、host/boston.example.com@EXAMPLE.COM でなければなりません。

SEAM は DNS サービスがなくても動作しますが、その場合、一部の主要な機能 (レルム間通信など) は動作しません。DNS を構成しない場合は、単純なホスト名がインスタンス名として使用することができます。この場合、主体名は host/boston@EXAMPLE.COM となります。DNS をあとで有効にした場合は、KDC データベースのすべてのホスト主体を削除して置換する必要があります。

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することもできます。ただし、ポート番号を変更する場合は、各クライアントの /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。また、各 KDC の /etc/krb5/kdc.conf ファイルも更新する必要があります。

スレーブ KDC

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。各レルムには、1 つ以上のスレーブ KDC が必要です。次の要因により、スレーブ KDC を追加する必要がある場合が考えられます。

- 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、各セグメントが少なくとも最小限の機能だけは動作するように、ネットワークを設定する必要がある。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る。この場合、KDC はマスターまたはスレーブのどちらでもよい
- 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る

スレーブ KDC の数に制限はありません。ただし、KDC データベースは、各サーバーに伝播する必要があります。このため、インストールした KDC サーバーが多くなるにつれて、レルム全体のデータ更新時間が長くなります。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、セキュリティ違反の危険性が高くなります。

1 つまたは複数のスレーブ KDC をマスター KDC とスワップするように構成することができます。このように 1 つ以上のスレーブ KDC をシステムに事前に構成しておくと、マスター KDC になんらかの理由で障害が発生した場合でも、マスター KDC と簡単にスワップすることができます。 スワップ可能なスレーブ KDC の構成方法については、253 ページの「マスター KDC とスレーブ KDC のスワップ」 を参照してください。

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。最初に解決しなければならない問題の 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に対する最新情報の伝播頻度を決定するときは、更新の完了に必要な時間を考慮する必要があります。データベースの伝播の詳細は、257 ページの「Kerberos データベースの管理」を参照してください。

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。この方法を利用すると、データの更新時間は少なくなりますが、レルムの管理は複雑になります。

クロックの同期

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で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」と呼ばれるこの機能も、Kerberos セキュリティ検査の 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、252 ページの「KDC と SEAM クライアントのクロックの同期化」を参照してください。クロックの同期化には、ほかにも方法があり、NTP は必須ではありません。任意の同期化形式を使用して、クロックスキューによるアクセス障害を回避してください。

オンラインヘルプ URL

SEAM のシステム管理ツールでは、オンラインヘルプ URL が使用されるため、「Help Contents」メニューが機能するように URL を適切に定義する必要があります。このマニュアルの HTML 版は、任意のサーバーにインストールできます。また、 `http://docs.sun.com` から任意のマニュアルを使用することもできます。

URL には、このマニュアルの「主体とポリシーの管理」の「SEAM 管理ツール」を指定してください。必要に応じて、別の HTML ページを選択することもできます。

第 15 章

SEAM の構成 (手順)

この章では、ネットワークアプリケーションサーバーの構成と手順およびインストール手順について説明します。

SEAM の構成 (作業マップ)

構成手順は、その個々の手順がほかの手順に依存するため、規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、 SEAM 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。次の作業マップで、推奨する SEAM のインストール順序を示します。

表 **15–1** 第一段階: SEAM の構成順序

作業	説明	参照先
1. SEAM インストールの計画	ソフトウェア構成処理を開始する前に、構成についての問題を解決する。事前に計画することで、結果的に時間やその他のリソースを節約できる	第 14 章

表 **15–1** 第一段階: SEAM の構成順序　(続き)

作業	説明	参照先
2. (省略可能) NTP のインストール	NTP ソフトウェアなどのクロック同期プロトコルを構成する。SEAM が正常に動作するには、レルムに含まれるすべてのシステムのクロックを同期化する必要がある	252 ページの「KDC と SEAM クライアントのクロックの同期化」
3. マスター KDC の構成	レルムに対してマスター KDC とデータベースを構成および構築する	233 ページの「マスター KDC を構成する方法」
4. (省略可能) スレーブ KDC の構成	レルムに対してスレーブ KDC を構成および構築する	237 ページの「スレーブ KDC を構成する方法」
5. (省略可能) KDC のセキュリティ強化	KDC サーバーに対するセキュリティ違反を回避する	263 ページの「KDC サーバーへのアクセスを制限する方法」
6. (省略可能) スワップ可能な KDC の構成	マスター KDC とスレーブ KDC を簡単にスワップできるようにする	254 ページの「スワップ可能なスレーブ KDC を構成する方法」

必要な手順が完了したあと、必要に応じて次の手順を行います。

表 **15–2** 次の段階: 追加の SEAM 作業

作業	説明	参照先
レルム間認証の構成	レルム間の通信を使用可能にする	240 ページの「レルム間認証の構成」
SEAM クライアントの構成	クライアントが SEAM サービスを使用できるようにする	248 ページの「SEAM クライアントの構成」
SEAM NFS サーバーの構成	Kerberos 認証を必要とするファイルシステムを、サーバーが 共有できるようにする	243 ページの「SEAM NFS サーバーの構成」

KDC サーバーの構成

SEAM ソフトウェアをインストールしたあと、KDC サーバーを構成する必要があります。資格を発行するには、1 つのマスター KDC と 1 つ以上のスレーブ KDC を構成する必要があります。KDC が発行する資格は、SEAM の基本要素であるため、KDC をインストールしないと、ほかの処理を行うことはできません。

マスター KDC とスレーブ KDC の最も大きな違いは、マスター KDC だけがデータベース管理要求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。この機能は、マスター KDC が応答できない場合に、冗長性を確保します。

▼ マスター KDC 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`
- マスター KDC = `kdc1.example.com`
- スレーブ KDC = `kdc2.example.com`
- admin 主体 = `kws/admin`
- オンラインヘルプ URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

注 – この URL は、「SEAM 管理ツール」の節を指すように調整してください (229 ページの「オンラインヘルプ URL」を参照)。

---

1. マスター **KDC** を構成するための前提条件を完了します。
   この手順を実行するには、DNS が動作している必要があります。マスター KDC をスワップ可能にする場合の命名手順については、253 ページの「マスター KDC とスレーブ KDC のスワップ」を参照してください。
2. マスター **KDC** 上でスーパーユーザーになります。
3. **Kerberos** 構成ファイル **(`krb5.conf`)** を編集します。
   レルム名とサーバー名を変更する必要があります。このファイルの詳細は、`krb5.conf(4)` のマニュアルページを参照してください。

```
kdc1 # cat /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
#
# ドメイン名とレルム名が同じ場合、
# このエントリは必要ない
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log
```

```
[appdefaults]
    gkadmin = {

 help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
        }
```

この例では、domain_realm、 kdc、admin_server、およびすべての domain_realm エントリの行を変更しました。また、help_url を定義する行を編集しました。

4. **KDC 構成ファイル (kdc.conf)** を編集します。

   レルム名を変更する必要があります。KDC 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 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

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

   この例では、realms セクションのレルム名定義を変更しました。

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 は、KDC データベースを作成するコマンドです。-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。

```
kdc1 # /usr/sbin/kdb5_util create -r EXAMPLE.COM -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:     <鍵を入力する>
Re-enter KDC database master key to verify:     <鍵を再度入力する>
```

   レルム名がサーバーの名前空間のドメイン名と同じ場合は、レルム名を指定する -r オプションは必要はありません。

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

   /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。最初のエントリは、次のようになります。

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

   このエントリにより、 EXAMPLE.COM レルム内の kws/admin 主体に対して、KDC 内の主体またはポリシーを変更する機能が与えられます。デフォルトのイン

ストールでは、アスタリスク (*) が指定され、すべての admin 主体に変更権限が与えられます。このデフォルトの指定では、セキュリティが低下する可能性があります。そのため、admin 主体をすべてリストに含めると、セキュリティが向上します。詳細は、kadm5.acl(4) のマニュアルページを参照してください。

7. **kadmin.local** コマンドを起動します。

   次の手順では、SEAM で使用される主体を作成します。

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local:
```

   a. データベースに管理主体を追加します。

      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、適切な主体名を置き換えてください。

```
kadmin.local: addprinc kws/admin
Enter password for principal kws/admin@EXAMPLE.COM:     <パスワードを入力する>
Re-enter password for principal kws/admin@EXAMPLE.COM:     <パスワードを再度入力する>
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

   b. **kadmind** サービスのキータブファイルを作成します。

      このコマンドシーケンスは、kadmin および changepw 主体のエントリを保持するキータブを作成します。これらの主体は、kadmind サービスに必要です。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかにかかわらず、FQDN は小文字で入力する必要があります。

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES-CBC-CRC
          added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc1.example.com
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES-CBC-CRC
          added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

   c. **kadmin.local** を終了します。

      次の手順で必要になる主体をすべて追加しました。

```
kadmin.local: quit
```

8. **Kerberos** デーモンを起動します。

```
kdc1 # /etc/init.d/kdc start
kdc1 # /etc/init.d/kdc.master start
```

9. **kadmin** を起動します。

   この時点で、SEAM 管理ツールを使用して、主体を追加します。追加するには、上記の手順で作成した admin 主体名を使用してログインする必要があります。た

だし、以下のコマンド行の例では、簡略化されています。

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <kws/admin パスワードを入力する>
kadmin:
```

a. マスター **KDC** のホスト主体を作成します。

ホスト主体は、`klist` や `kprop` などの Kerberos アプリケーションで使用されます。主体のインスタンスがホスト名のときは、`/etc/resolv.conf` ファイル内のドメイン名が大文字であるか小文字であるかにかかわらず、FQDN は小文字で入力する必要があります。

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

b. **(**省略可能**)** マスター **KDC** の **`root`** 主体を作成します。

この主体は、認証済みの NFS マウントで使用されます。そのため、この主体は、マスター KDC 上にある必要はありません。主体のインスタンスがホスト名のときは、`/etc/resolv.conf` ファイル内のドメイン名が大文字であるか小文字であるかにかかわらず、FQDN は小文字で入力する必要があります。

```
kadmin: addprinc root/kdc1.example.com
Enter password for principal root/kdc1.example.com@EXAMPLE.COM:     <パスワードを入力する>
Re-enter password for principal root/kdc1.example.com@EXAMPLE.COM:     <パスワードを再度入力する>
Principal "root/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

c. マスター **KDC** のキータブファイルにマスター **KDC** のホスト主体を追加します。

キータブファイルに追加したホスト主体が、自動的に使用されます。

```
kadmin: ktadd host/kdc1.example.com
kadmin: Entry for principal host/kdc1.example.com with
  kvno 3, encryption type DES-CBC-CRC added to keytab
  WRFILE:/etc/krb5/krb5.keytab
kadmin:
```

d. **`kadmin`** を終了します。

```
kadmin: quit
```

10. 各 **KDC** のエントリを伝播構成ファイル **(`kpropd.acl`)** に追加します。

このファイルの詳細は、`kprop(1M)` のマニュアルページを参照してください。

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

11. **(**省略可能**) NTP** などのクロック同期メカニズムを使用して、マスター **KDC** のクロックを同期化します。

NTP のインストールと使用は、必須ではありません。ただし、認証が正常終了するには、krb5.conf ファイルの libdefaults セクションに定義されているデフォルト時間内にすべてのクロックが収まるようにする必要があります。NTP については、252 ページの「KDC と SEAM クライアントのクロックの同期化」を参照してください。

▼ スレーブ KDC を構成する方法

この手順では、kdc3 という名前の新しいスレーブ KDC 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
- スレーブ KDC = kdc2.example.com と kdc3.example.com
- admin 主体 = kws/admin
- オンラインヘルプ URL = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956

注 – この URL は、「SEAM 管理ツール」の節を指すように調整してください (229 ページの「オンラインヘルプ URL」を参照)。

---

1. スレーブ **KDC** を構成するための前提条件を完了します。

   マスター KDC が構成済みである必要があります。スレーブ KDC をスワップ可能にする場合の手順については、253 ページの「マスター KDC とスレーブ KDC のスワップ」を参照してください。

2. マスター **KDC** 上でスーパーユーザーになります。

3. マスター **KDC** 上で **kadmin** を起動します。

   マスター KDC を構成するときに作成した admin 主体名を使用して、ログインする必要があります。

   ```
   kdc1 # /usr/sbin/kadmin -p kws/admin
   Enter password:      <kws/admin パスワードを入力する>
   kadmin:
   ```

   a. マスター **KDC** のデータベースにスレーブホスト主体が存在しない場合は、追加します。

      スレーブが機能するには、ホスト主体が必要です。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で入力する必要があります。

      ```
      kadmin: addprinc -randkey host/kdc3.example.com
      Principal "host/kdc3@EXAMPLE.COM" created.
      ```

```
kadmin:
```

b. **(省略可能) マスター KDC 上で、スレーブ KDC の `root` 主体を作成します。**

この主体が必要なのは、認証済みのファイルシステムをスレーブが NFS マウントする場合だけです。主体のインスタンスがホスト名のときは、`/etc/resolv.conf` ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で入力する必要があります。

```
kadmin: addprinc root/kdc3.example.com
Enter password for principal root/kdc3.example.com@EXAMPLE.COM:     <パスワードを入力する>
Re-enter password for principal root/kdc3.example.com@EXAMPLE.COM:     <パスワードを再度入力する>

Principal "root/kdc3.example.com@EXAMPLE.COM" created.
kadmin:
```

c. **`kadmin` を終了します。**

```
kadmin: quit
```

4. マスター **KDC** 上で、**Kerberos** 構成ファイル **(`krb5.conf`)** を編集します。

各スレーブのエントリを追加する必要があります。このファイルの詳細は、`krb5.conf(4)` の マニュアルページを参照してください。

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

[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                kdc = kdc3.example.com
                admin_server = kdc1.example.com
        }

[domain_realm]
        .example.com = EXAMPLE.COM
#
# ドメイン名とレルム名が同じ場合
# このエントリは不要
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {

 help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

5. マスター **KDC** 上で、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。

データベース伝播構成ファイルの詳細は、kprop(1M) のマニュアルページを参照してください。

```
kdc1 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
```

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

   この手順は、マスター KDC サーバーが、各 KDC サーバーに必要な情報を更新したため、すべてのスレーブ KDC 上で実行する必要があります。ftp など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
   - /etc/krb5/kpropd.acl

7. 新しいスレーブ上で **kadmin** を使用して、スレーブのホスト主体をスレーブのキータブファイルに追加します。

   マスター KDC を構成するときに作成したadmin 主体名を使用してログインする必要があります。ログインすると、kprop などの Kerberos アプリケーションが機能します。主体のインスタンスがホスト名のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で入力する必要があります。

```
kdc3 # /usr/sbin/kadmin -p kws/admin
Enter password:     <kws/admin パスワードを入力する>
kadmin: ktadd host/kdc3.example.com
kadmin: Entry for principal host/kdc3.example.com with
  kvno 3, encryption type DES-CBC-CRC added to keytab
  WRFILE:/etc/krb5/krb5.keytab
kadmin: quit
```

8. マスター **KDC** 上で、スレーブ **KDC** 名を **cron** ジョブに追加します。このジョブは、**crontab -e** を実行して、自動的にバックアップを実行します。

   kprop_script 行の末尾に、各スレーブ KDC のサーバー名を追加します。

```
10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com kdc3.example.com
```

   バックアップの時刻を変更することもできます。この構成では、バックアップ処理を毎日午前 3 時 10 分に開始します。

9. マスター **KDC** 上のデータベースを、 **kprop_script** を使用してバックアップし、伝播します。

   データベースのバックアップコピーがすでに作成されている場合は、ここでバックアップを作成する必要はありません。詳細は、260 ページの「Kerberos データベースをスレーブ KDC に手動で伝播する方法」を参照してください。

```
kdc1 # /usr/lib/krb5/kprop_script kdc3.example.com
Database propagation to kdc3.example.com: SUCCEEDED
```

10. 新しいスレーブ上で、**kdb5_util** を使用して **stash** ファイルを作成します。

```
kdc3 # /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:     <鍵を入力する>
```

11. **(**省略可能**)** 新しいスレーブ **KDC** 上で、**NTP** などのクロック同期メカニズムを使用して、マスター **KDC** のクロックと同期化します。

    ネットワーク時間プロトコル (NTP) のインストールと使用は、必須ではありません。ただし、認証が正常終了するには、krb5.conf ファイルの libdefaults セクションに定義されているデフォルト時間内にすべてのクロックが収まるようにする必要があります。NTP については、252 ページの「KDC と SEAM クライアントのクロックの同期化」を参照してください。

12. 新しいスレーブ上で、**KDC** デーモン **(krb5kdc)** を起動します。

```
kdc3 # /etc/init.d/kdc start
```

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。いくつかの方法がありますが、通常は、秘密鍵を作成し、2 つのレルム間で共有します。レルム間の関係には、階層関係または直接接続があります (226 ページの「レルムの階層」を参照)。

▼ 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、 ENG.EAST.EXAMPLE.COM レルムと EAST.EXAMPLE.COM 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。

1. 階層関係のレルム間認証の前提条件を完了します。

   マスター KDC の各レルムが構成済みである必要があります。認証処理を十分にテストするには、複数のクライアントまたはスレーブ KDCをインストールしている必要があります。

2. 最初のマスター **KDC** 上でスーパーユーザーになります。

3. **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:     <kws/admin パスワードを入力する>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
Enter password for principal krgtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM:     <パスワードを
                                                                               入力する>
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal krgtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM:     <パスワードを
                                                                               入力する>
kadmin: quit
```

注 – 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/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM のパスワードは、2 つのレルムで同じである必要があります。

---

4. **Kerberos** 構成ファイル **(krb5.conf)** 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。

```
# cat /etc/krb5/krb5.conf
[libdefaults]
 .
 .
[domain_realm]
        .eng.east.example.com = ENG.EAST.EXAMPLE.COM
        .east.example.com = EAST.EXAMPLE.COM
```

この例では、ENG.EAST.EXAMPLE.COM レルムと EAST.EXAMPLE.COM レルムのドメイン名を定義しています。Kerberos 構成ファイルは先頭から末尾方向に検索されるため、サブドメインは最初に定義してください。

5. **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。

レルム間認証が動作するには、すべてのシステム (スレーブ KDC などのサーバーを含む) に新しいバージョンの Kerberos 構成ファイル (/etc/krb5/krb5.conf) がインストールされている必要があります。

6. もう一方のレルムで上記の手順を繰り返します。

▼ 直接接続のレルム間認証を確立する方法

この手順では、ENG.EAST.EXAMPLE.COM レルムと SALES.WEST.EXAMPLE.COM 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。

1. 直接接続のレルム間認証の前提条件を完了します。

マスター KDC の各レルムが構成済みである必要があります。認証処理を十分にテストするには、複数のクライアントまたはスレーブ KDCをインストールしている必要があります。

2. いずれかのマスター **KDC** サーバー上でスーパーユーザーになります。

3. **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:      <kws/admin パスワードを入力する>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM
Enter password for principal
  krgtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM:      <パスワードを入力する>
kadmin: addprinc krbtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal
  krgtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM:      <パスワードを入力する>
kadmin: quit
```

注 – 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/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM のパスワードは、2 つのレルムで同じである必要があります。

---

4. **Kerberos** 構成ファイル **(krb5.conf)** 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、ENG.EAST.EXAMPLE.COM レルム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するには、レルム名をスワップする必要があります。

```
# cat /etc/krb5/krb5.conf
[libdefaults]
 .
 .
[capaths]
    ENG.EAST.EXAMPLE.COM = {
        SALES.WEST.EXAMPLE.COM = .
    }

    SALES.WEST.EXAMPLE.COM = {
         ENG.EAST.EXAMPLE.COM = .
    }
```

5. **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。
   レルム間認証が動作するには、すべてのシステム (スレーブ KDC などのサーバーを含む) に新しいバージョンの Kerberos 構成ファイル (krb5.conf) がインストールされている必要があります。

6. もう一方のレルムで上記の手順を繰り返します。

SEAM 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ますが、主体を直接使用することはできません。そのため、主体を UID に対応付けるために、ユーザー主体を UNIX UID に割り当てる資格テーブルを作成する必要があります。この節では、SEAM 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 セキュリティモードを有効にする手順を中心に説明します。次の表は、この節で説明する作業の一覧です。

表 **15–3** SEAM NFS サーバーの構成 (作業マップ)

作業	説明	参照先
SEAM NFS サーバーを構成する	Kerberos 認証を必要とするファイルシステムを、サーバーが 共有できるようにする	243 ページの「SEAM NFS サーバーを構成する方法」
資格テーブルを作成する	資格テーブルを生成する	245 ページの「資格テーブルを作成する方法」
ユーザー主体を UNIX UID に割り当てる資格テーブルを変更する	資格テーブルの情報を更新する	245 ページの「資格テーブルに 1 つのエントリを追加する方法」
Kerberos 認証を使用してファイルシステムを共有する	セキュリティモードを使用してファイルシステムを共有し、Kerberos 認証を常に行う	246 ページの「複数の Kerberos セキュリティモードで安全な NFS 環境を設定する方法」

▼ SEAM 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`

1. **SEAM NFS** サーバーを構成するための前提条件を完了します。

   マスター KDC が構成されている必要があります。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。

2. **(**省略可能**) NTP** クライアントなどのクロック同期メカニズムをインストールします。

   NTP のインストールと使用は、必須ではありません。ただし、認証が正常終了するには、`krb5.conf` ファイルの `libdefaults` セクションに定義されているデフォルト時間内に、すべてのクロックを調整する必要があります。NTP については、252 ページの「KDC と SEAM クライアントのクロックの同期化」を参照して

ください。

3. **kadmin** を起動します。

   SEAM 管理ツールを使って主体を追加する方法については、 290 ページの「新しい主体を作成する方法」を参照してください。追加するときは、マスター KDC を構成するときに作成した admin 主体名を使用してログインする必要があります。ただし、次の例では、コマンド行を使用して、必要な主体を追加しています。

   ```
   denver # /usr/sbin/kadmin -p kws/admin
   Enter password:     <kws/admin パスワードを入力する>
   kadmin:
   ```

   a. サーバーの **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のときは、/etc/resolv.conf ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で入力する必要があります。

      ```
      kadmin: addprinc -randkey nfs/denver.example.com
      Principal "nfs/denver.example.com" created.
      kadmin:
      ```

   b. (省略可能) **NFS** サーバーの **root** 主体を作成します。

```
kadmin: addprinc root/denver.example.com
Enter password for principal root/denver.example.com@EXAMPLE.COM:     <パスワードを入力する>
Re-enter password for principal root/denver.example.com@EXAMPLE.COM:     <パスワードを再度入力する>

Principal "root/denver.example.com@EXAMPLE.COM" created.
kadmin:
```

   c. 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ます。

      ```
      kadmin: ktadd nfs/denver.example.com
      kadmin: Entry for principal nfs/denver.example.com with
        kvno 3, encryption type DES-CBC-CRC added to keytab
        WRFILE:/etc/krb5/krb5.keytab
      kadmin:
      ```

   d. **kadmin** を終了します。

      ```
      kadmin: quit
      ```

4. **gsscred** テーブルを作成します。

   詳細は、245 ページの「資格テーブルを作成する方法」を参照してください。

5. **NFS** ファイルシステムを **Kerberos** セキュリティモードで共有します。

   詳細は、246 ページの「複数の Kerberos セキュリティモードで安全な NFS 環境を設定する方法」を参照してください。

6. クライアントごとに、ユーザー主体と **root** 主体を認証します。

▼ 資格テーブルを作成する方法

gsscred 資格テーブルは、SEAM 主体を UID に割り当てるために NFS サーバーが使用します。NFS クライアントが Kerberos 認証を使用して NFS サーバーからファイルシステムをマウントするには、このテーブルを作成して使用可能にする必要があります。

1. **/etc/gss/gsscred.conf** を編集してこのメカニズムを変更します。
   このメカニズムを files に変更します。
2. **gsscred** 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、 /etc/nsswitch.conf ファイル内の passwd エントリに指定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入れたくない場合は、files エントリを一時的に削除しなければならないことがあります。詳細は、gsscred(1M) のマニュアルページを参照してください。

▼ 資格テーブルに 1 つのエントリを追加する方法

この手順を行うには、gsscred テーブルがすでに NFS サーバーに作成済みである必要があります。

1. **NFS** サーバーでスーパーユーザーになります。
2. **gsscred** を使用してエントリをテーブルに追加します。

   ```
   # gsscred -m mech [ -n name [ -u uid ]] -a
   ```

mech	使用するセキュリティメカニズムを定義する
name	KDC に定義されている、ユーザーの主体名を定義する
uid	パスワードデータベースに定義されている、ユーザーの UID を定義する
-a	UID を主体名の割り当てに追加する

例 — 資格テーブルに 1 つのエントリを追加する

次の例では、sandy という名前のユーザーエントリを追加し、UID 3736 に割り当てます。UID をコマンド行に指定しない場合は、パスワードファイルの UID が使用されます。

```
# gsscred -m kerberos_v5 -n sandy -u 3736 -a
```

▼ 複数の Kerberos セキュリティモードで安全な NFS 環境を設定する方法

1. **NFS** サーバー上でスーパーユーザーになります。
2. キータブファイルに **NFS** サービス主体が存在することを確認します。

   `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、243 ページの「SEAM 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. **`/etc/nfssec.conf`** ファイル内の **Kerberos** セキュリティモードを有効にします。

   `/etc/nfssec.conf` ファイルを編集して、Kerberos セキュリティモードの先頭にある “#” を削除します。

   ```
   # cat /etc/nfssec.conf
    .
    .
   #
   # NFS の Kerberos V5 を使用するために次の行をコメントからはずす
   #
   krb5            390003  kerberos_v5     default -               #
    RPCSEC_GSS
   krb5i           390004  kerberos_v5     default integrity       #
    RPCSEC_GSS
   krb5p           390005  kerberos_v5     default privacy         #
    RPCSEC_GSS
   ```

4. **`/etc/dfs/dfstab`** ファイルを編集します。必要なセキュリティモードを **`sec=`** オプションに指定して、適切なエントリに追加します。

   `share -F nfs -o sec=`*mode file-system*

mode	共有するときに使用するセキュリティモードを指定する。複数のセキュリティ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オートマウンタによって使用される
file-system	共有するファイルシステムへのパスを定義する

   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 クライアント上のユーザー主体と `root` 主体が両方とも認証される必要があります。

5. **NFS** サービスがサーバー上で動作していることを確認します。

   `share` コマンドまたは `share` コマンドセットを初めて実行する場合、NFS デーモンが動作していないことがあります。次のコマンドでデーモンを終了し、再起動してください。

   ```
   # /etc/init.d/nfs.server stop
   # /etc/init.d/nfs.server start
   ```

6. **(**省略可能**)** オートマウンタを使用する場合は、**`auto_master`** データベースを編集して、デフォルト以外のセキュリティモードを選択してください。

   ファイルシステムのアクセスにオートマウンタを使用しない場合やデフォルトの選択をセキュリティモードとして使用する場合は、この手順を行う必要はありません。

   file-system `auto_home -nosuid,sec=`*mode*

7. **(**省略可能**)** 手動で **`mount`** コマンドを実行し、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。

   この代わりに、`mount` コマンドにセキュリティモードを指定できますが、オートマウンタは利用できません。

   ```
   # mount -F nfs -o sec=mode file-system
   ```

例 — 1 つの Kerberos セキュリティモードでファイルシステムを共有する

この例の `dfstab` ファイルの行は、NFS サービスを使用してファイルにアクセスするには、Kerberos 認証が正常終了する必要があることを示しています。

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5 /export/home
```

例 — 複数の Kerberos セキュリティ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 セキュリティモードがすべて選択されています。マウント要求にセキュリティモードが指定されていない場合は、NFS V3 のすべてのクライアントに対して、最初のモードが使用されます (この場合は `krb5`)。詳細は、`nfssec.conf(4)` のマニュアルページを参照してください。

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5:krb5i:krb5p /export/home
```

SEAM クライアントの構成

SEAM クライアントは、SEAM サービスを使用する同じネットワーク上のすべてのホスト (KDC サーバーを除く) です。この節では、SEAM クライアントのインストール手順と、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。

▼ SEAM クライアントを構成する方法

この手順では、次の構成パラメータを使用します。

- 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
- 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://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956

注 – この URL は、「SEAM 管理ツール」の節を指すように調整してください (229 ページの「オンラインヘルプ URL」を参照)。

---

1. スーパーユーザーになります。
2. **Kerberos** 構成ファイル **(krb5.conf)** を編集します。
   このファイルを SEAM のデフォルト版から変更するには、レルム名とサーバー名を変更する必要があります。gkadmin のヘルプファイルへのパスも指定する必要があります。

```
kdc1 # cat /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

[appdefaults]
    gkadmin = {

 help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

3. **(**省略可能**) NTP** などのクロック同期メカニズムを使用して、クライアントのクロックをマスター **KDC** のクロックと同期化します。

   NTP のインストールと使用は、必須ではありません。ただし、認証が正常終了するには、`krb5.conf` ファイルの `libdefaults` セクションに定義されているデフォルト時間内に、すべてのクロックを調整する必要があります。NTP については、252 ページの「KDC と SEAM クライアントのクロックの同期化」を参照してください。

4. **(**省略可能**)** 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だけ、ユーザー主体を作成します。SEAM 管理ツールの使用方法については、290 ページの「新しい主体を作成する方法」 を参照してください。以下は、コマンド行の例です。

```
client1 # /usr/sbin/kadmin -p kws/admin
Enter password:     <kws/admin パスワードを入力する>
kadmin: addprinc mre
Enter password for principal mre@EXAMPLE.COM:     <パスワードを入力する>
Re-enter password for principal mre@EXAMPLE.COM:     <パスワードを再度入力する>

kadmin:
```

5. **`root`** 主体を作成します。

   主体のインスタンスがホスト名のときは、`/etc/resolv.conf` ファイル内のドメイン名が大文字であるか小文字であるかに関係なく、FQDN は小文字で入力する必要があります。

```
kadmin: addprinc root/client1.example.com
Enter password for principal root/client1.example.com@EXAMPLE.COM:     <パスワードを入力する>
Re-enter password for principal root/client1.example.com@EXAMPLE.COM:     <パスワードを再度入力する>

kadmin: quit
```

6. **(**省略可能**) NFS** で **Kerberos** を使用するには、 **`/etc/nfssec.conf`** ファイル内の **Kerberos** セキュリティモードを有効にします。

/etc/nfssec.conf ファイルを編集して、Kerberos セキュリティモードの先頭にある “#” を削除します。

```
# cat /etc/nfssec.conf
 .
 .
#
# NFS の Kerberos V5 を使用するために次の行をコメントからはずす
#
krb5            390003  kerberos_v5     default -               #
 RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       #
 RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         #
 RPCSEC_GSS
```

7. **(**省略可能**) SEAM** クライアント上のユーザーが **Kerberos NFS** ファイルシステムを自動的にマウントして **Kerberos** 認証を使用する場合は、**root** ユーザーを認証する必要があります。

   この処理を最も安全に実行するには、kinit コマンドを使用します。ただし、Kerberos によって保護されているファイルシステムをマウントするときは、root として kinit を使用する必要があります。代わりに、キータブファイルを使用することもできます。キータブファイルの要件の詳細については、251 ページの「NFS ファイルシステムをマウントするように root 認証を設定する」を参照してください。

   ```
   client1 # /usr/bin/kinit root/client1.example.com
   Password for root/client1.example.com@EXAMPLE.COM:     <パスワードを入力する>
   ```

   キータブファイルを使用するには、kadmin を使用して root 主体をクライアントのキータブに追加します。

   ```
   client1 # /usr/sbin/kadmin -p kws/admin
   Enter password:     <kws/admin パスワードを入力する>
   kadmin: ktadd root/client1.example.com
   kadmin: Entry for principal root/client.example.com with
     kvno 3, encryption type DES-CBC-CRC added to keytab
     WRFILE:/etc/krb5/krb5.keytab
   kadmin: quit
   ```

8. クライアントから **Kerberos** チケットの有効期限切れをユーザーに警告する場合は、**/etc/krb5/warn.conf** ファイル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を参照してください。

例 - SEAM 以外の KDC を使用するように SEAM クライアントを設定する

SEAM 以外の KDC を使用するように SEAM クライアントを設定することができます。この場合、/etc/krb5/krb5.conf ファイルの realms セクションに、1 行を追加する必要があります。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。この行の書式は次のとおりです。

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

NFS ファイルシステムをマウントするように root 認証を設定する

Kerberos 以外の NFS ファイルシステムにアクセスする場合は、root として NFS ファイルシステムをマウントするか、ユーザーがファイルシステムにアクセスしたときにオートマウンタを介して自動的にアクセスできます。後者の場合、root アクセス権は必要ありません。

Kerberos NFS ファイルシステムをマウントする場合もほとんど同じですが、操作が複雑になります。Kerberos NFS ファイルシステムをマウントするには、root として kinit コマンドを使用し、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。クライアントの root 主体は通常、クライアントのキータブに登録されていないためです。この手順は、オートマウンタが設定されているときでも必要です。また、すべてのユーザーがシステムの root パスワードと root 主体のパスワードを知っている必要があります。

この手順を省略するには、クライアントの root 主体をクライアントのキータブファイルに追加し、root の資格を自動的に与えるようにします。この方法を利用すると、ユーザーは kinit コマンドを実行しなくても NFS ファイルシステムをマウントでき、使いやすさが向上しますが、セキュリティが低下します。たとえば、キータブ内の root 主体を使用してシステムへのアクセス権を取得した場合、root の資格を取得できます。このため、セキュリティ対策を適切に行う必要があります。詳細は、309 ページの「キータブファイルの管理」を参照してください。

KDC と SEAM 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で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」) 。同時に、Kerberos セキュリティを検査することにもなります。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。

アプリケーションサーバーが再実行要求を認識し拒否する目的で、すべての Kerberos プロトコルメッセージをどのくらいの間追跡管理する必要があるかも、クロックスキューで決まります。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 – セキュリティ上の理由から、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 SEAM クライアント間で同期化したクロックを管理することは重要であるため、NTP ソフトウェアを使用して同期化します。デラウェア大学が作成した NTP パブリックドメインソフトウェアが Solaris 2.6 以降の Solaris ソフトウェアに含まれています。

注 – クロックを同期化するときは、`rdate` コマンドと `cron` ジョブを使用することもできます。この方法は、NTP より簡単に使用できます。ただし、ここでは NTP を中心に説明します。ネットワークを使用してクロックを同期化する場合は、クロック同期化プロトコル自体も安全である必要があります。

NTP を使用すると、正確な時間とネットワーククロック同期をネットワーク環境で管理できます。NTP は基本的にはクライアントサーバー実装の状態をとります。1 つのシステムをマスタークロック (NTP サーバー) として指定します。次に、その他のすべてのシステム (NTP クライアント) をマスタークロックと同期するように設定します。

クロックを同期化するために、NTP は `xntpd` デーモンを使用して、インターネット標準時サーバーに合わせて UNIX システムの時刻を設定および管理します。次の図は、NTP のクライアントサーバー実装の例です。

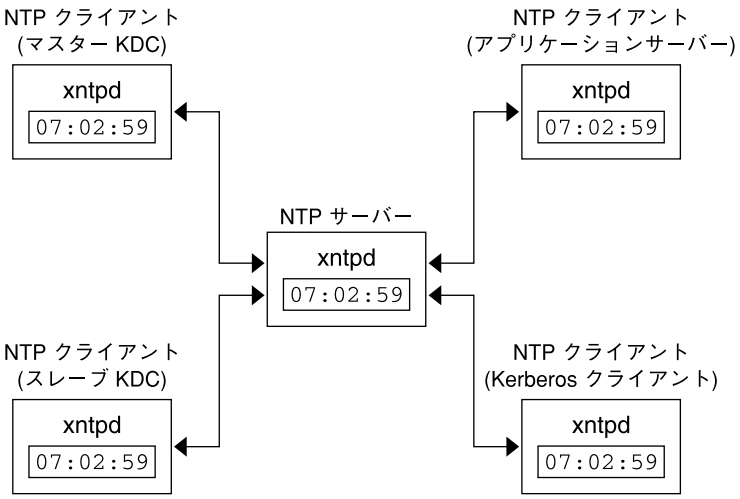


図 **15–1** NTP を使用したクロック同期

KDC および SEAM 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. ネットワークに NTP サーバーを設定します。NTP サーバーは、マスター KDC 以外であればどのシステムでも設定できます。NTP サーバーの作業については、『*Solaris* のシステム管理 *(*資源管理とネットワークサービス*)*』の「NTP の管理 (作業)」を参照してください。
2. ネットワークの KDC と SEAM クライアントを構成するときに、それらを NTP サーバーの NTP クライアントとして設定します。NTP クライアントの作業については、『*Solaris* のシステム管理 *(*資源管理とネットワークサービス*)*』の「NTP の管理 (作業)」を参照してください。

マスター KDC とスレーブ KDC のスワップ

マスター KDC とスレーブ KDC をスワップするときは、この節で説明する手順を行います。マスター KDC とスレーブ KDC のスワップは、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。

▼ スワップ可能なスレーブ KDC を構成する方法

この手順は、マスター KDC にスワップ可能なスレーブ KDC に対して実行します。

1. **KDC** をインストールするときに、マスター **KDC** およびスワップ可能なスレーブ **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。`/etc/krb5/krb5.conf` ファイルにホストを定義するときも、別名を使用します。

2. 手順に従って、スレーブ **KDC** をインストールします。

   スワップするサーバーは、レルム内でスレーブ KDC として動作している必要があります。手順については、237 ページの「スレーブ KDC を構成する方法」を参照してください。

3. マスター **KDC** コマンドを移動します。

   このスレーブ KDC からマスター KDC コマンドが実行されるのを回避するために、`kprop`、`kadmind`、および `kadmin.local` コマンドを予約した場所に移動します。

```
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
```

4. **root** の **crontab** ファイルの **kprop** 行をコメントにします。

   スレーブ KDC が KDC データベースのコピーを伝播しなくなります。

```
kdc4 # crontab -e
#ident  "@(#)root       1.20    01/11/06 SMI"
#
# root の crontab はアカウントデータ収集を実行するために使用する
#
# rtc コマンドは、夏時間を変更する場合に実時間クロックを
# 調整するために実行する
#
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/krb5kprop_script kdc1.example.sun.com #SUNWkr5ma
```

▼ マスター KDC とスレーブ KDC をスワップする方法

この手順を実行するには、スレーブ KDC がスワップ可能なスレーブとして設定されている必要があります (254 ページの「スワップ可能なスレーブ KDC を構成する方法」を参照)。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 となるスレーブ KDC の名前は、`kdc4` です。

1. 旧マスター **KDC** 上で、**kadmind** プロセスを終了します。

```
kdc1 # /etc/init.d/kdc.master stop
```

kadmind プロセスを終了するときは、旧 KDC データベースに対する変更は行わないでください。

2. 旧マスター **KDC** 上で、**root** の **crontab** ファイルの **kprop** 行をコメントにします。

```
kdc1 # crontab -e
#ident  "@(#)root       1.20    01/11/06 SMI"
#
# root の crontab はアカウントデータ収集を実行するために使用する
#
# rtc コマンドは、夏時間を変更する場合に実時間クロックを
# 調整するために実行する
#
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.sun.com #SUNWkr5ma
```

旧マスター KDC が KDC データベースのコピーを伝播しなくなります。

3. 旧マスター **KDC** 上で、**kprop_script** を実行してデータベースをバックアップし、伝播します。

```
kdc1 # /usr/lib/krb5/kprop_script kdc4.example.com
Database propagation to kdc4.example.com: SUCCEEDED
```

4. 旧マスター **KDC** 上で、マスター **KDC** コマンドを移動します。

マスター KDC コマンドが実行されるのを回避するために、kprop、kadmind、および kadmin.local コマンドを予約した場所に移動します。

```
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
kdc4 # mv /etc/krb5/kadm5.acl /etc/krb5/kadm5.acl.save
```

5. **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
```

6. **DNS** サーバー上で、インターネットドメインネームサーバーを再起動します。

両方のサーバー上で次のコマンドを実行して、新しい別名情報を取得します。

```
# pkill -1 in.named
```

7. 新しいマスター **KDC** 上で、マスター **KDC** コマンドを移動します。

```
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
```

8. 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)** を編集します。

   /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。最初のエントリは、次のようになります。

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

   このエントリにより、 EXAMPLE.COM レルム内の kws/admin 主体に対して、KDC 内の主体またはポリシーを変更する機能が与えられます。デフォルトのインストールでは、アスタリスク (*) が指定され、すべての admin 主体に変更権限が与えられます。このデフォルトの指定では、セキュリティが低下する可能性があります。そのため、admin 主体をすべてリストに含めると、セキュリティが向上します。詳細は、kadm5.acl(4) のマニュアルページを参照してください。

9. 新しいマスター **KDC** 上で、**kadmin.local** を使用して **kadmin** のキータブファイルを作成します。

   このコマンドシーケンスは、admin および changepw 主体のエントリを格納するためのキータブを作成します。これらの主体は、kadmind サービスに必要です。

```
kdc4 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc4.example.com
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type DES-CBC-CRC
          added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc4.example.com
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type DES-CBC-CRC
          added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

10. 新しいマスター **KDC** 上で、**kadmind** を起動します。

    ```
    kdc4 # /etc/init.d/kdc.master start
    ```

11. **root** の **crontab** ファイル内の **kprop** 行を有効にします。

    ```
    kdc4 # crontab -e
    #ident  "@(#)root       1.19    98/07/06 SMI"
    #
    # root の crontab はアカウントデータ収集を実行するために使用する
    #
    # rtc コマンドは、夏時間を変更する場合に実時間クロックを
    # 調整するために実行する
    #
    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 kdc1.example.sun.com #SUNWkr5ma
    ```

Kerberos データベースの管理

Kerberos データベースは、Kerberos の最も重要な構成要素であるため、適切に管理する必要があります。この節では、データベースのバックアップと復元、並列伝播の設定、stash ファイルの管理など、Kerberos データベースの管理についていくつかの手順を説明します。データベースを初期設定する手順については、233 ページの「マスター KDC を構成する方法」 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、構成処理の中で最も重要なものの 1 つです。伝播の頻度が少ないと、マスター KDC とスレーブ KDC が同期しなくなります。マスター KDC に障害が発生した場合、スレーブ KDC は最新のデータベース情報を取得できません。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。このため、Kerberos データベースの変更頻度に基づいて、伝播頻度を適切に設定する必要があります。

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。スレーブ KDC のデータが壊れた場合でも、次回のデータベース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、影響が発生しないこともあります。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、壊れたデータがバックアップされると、マスター KDC 上の壊れていない前回のバックアップファイルが上書きされます。

この場合、安全なバックアップコピーが存在しないため、`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、`kdb5_util` の `dump` コマンドを使用して別のバックアップコピーを作成することも必要です。これにより、データベースが壊れても、`kdb5_util` の `load` コマンドを使用して、マスター KDC の最新のバックアップを復元することができます。

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` にだけ与えられます。許可されないアクセスから保護するには、 `kprop` コマンドだけを使用して、データベースダンプファイルを伝播します。この場合、転送するデータが暗号化されます。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。

注意 – Kerberos データベースが伝播されたあとに更新され、次の伝播の前にデータベースが壊れた場合は、スレーブ KDC には更新が反映されません。この更新は失われます。このため、定期的に実行される伝播の前に重要な更新を追加した場合は、データの損失を回避するために手動でデータベースを伝播する必要があります。

kpropd.acl ファイル

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、このマニュアルの SEAM のインストールおよびインストール後の構成手順では、マスター KDC とスレーブ KDC に対して同じ kpropd.acl ファイルを追加するように説明しています。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。また、すべての KDC に同一のコピーを保持すると、構成の管理が容易になります。

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 には、ホスト名のリストを引数として指定します。区切り文字は空白です。指定したホスト名は、伝播先の KDC になります。

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 データベースをバックアップする方法

1. マスター **KDC** 上でスーパーユーザーになります。
2. **kdb5_util** の **dump** コマンドを使用して、**Kerberos** データベースをバックアップします。

   # **/usr/sbin/kdb5_util dump** [**-verbose**] [**-d** *dbname*] [*filename* [*principals*...]]

`-verbose`	バックアップする各主体とポリシー名を出力する
dbname	バックアップするデータベース名を定義する。指定したデータベース名には、`.db` が付加される。ファイルの絶対パスを指定できる。`-d` オプションを指定しない場合、デフォルトのデータベース名は `/var/krb5/principal` となる。実際には、`/var/krb5/principal.db` という名前に変換される
filename	データベースのバックアップに使用するファイルを定義する。ファイルの絶対パスを指定できる。ファイルを指定しなかった場合、データベースは標準出力にダンプされる
principal	バックアップする主体を 1 つ以上定義する (区切り文字は空白)。主体名は完全指定形式にする必要がある。主体を指定しなかった場合、データベース全体がバックアップされる

例 — Kerberos データベースのバックアップ

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。

```
# kdb5_util dump -verbose dumpfile
kadmin/kdc1.eng.example.com@ENG.EXAMPLE.COM
krbtgt/eng.example.com@ENG.EXAMPLE.COM
kadmin/history@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
changepw/kdc1.eng.example.com@ENG.EXAMPLE.COM
```

次の例では、`pak` および `pak/admin` 主体が Kerberos データベースからバックアップされます。

```
# kdb5_util dump -verbose dumpfile pak/admin@ENG.EXAMPLE.COM pak@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
```

▼ Kerberos データベースを復元する方法

1. マスター **KDC** 上でスーパーユーザーになります。
2. **kdb_util** の **load** コマンドを使用して、**Kerberos** データベースを復元します。

   # **/usr/sbin/kdb5_util load** [**-verbose**] [**-d** *dbname*] [**-update**] [*filename*]

`-verbose`	復元する各主体とポリシー名を出力する

dbname	復元するデータベース名を定義する。指定したデータベース名には、.db が付加される。ファイルの絶対パスを指定できる。-d オプションを指定しない場合、デフォルトのデータベース名は /var/krb5/principal となる。実際には、/var/krb5/principal.db という名前に変換される
-update	既存のデータベースを更新する。指定しない場合、新しいデータベースが作成されるか、既存のデータベースが上書きされる
filename	データベースの復元に使用するファイルを定義する。ファイルの絶対パスを指定できる。

例 — Kerberos データベースの復元

次の例では、database1.db 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、復元によって新しいデータベースが作成されます。

```
# kdb5_util load -d database1 dumpfile
```

▼ Kerberos データベースをスレーブ KDC に手動で伝播する方法

この手順では、kprop コマンドを使用して、Kerberos データベースを伝播します。定期的に実行する cron ジョブ以外に、スレーブ KDC とマスター KDC を同期化する必要がある場合は、この手順を行います。また、kprop_script と異なり、kprop を使用した場合は、現在のデータベースバックアップだけを伝播できます。伝播する前に、Kerberos データベースの新しいバックアップは作成されません。

1. マスター **KDC** 上でスーパーユーザーになります。
2. **(**省略可能**) kdb5_util** コマンドを使用して、データベースをバックアップします。

   ```
   # /usr/sbin/kdb5_util dump /var/krb5/slave_datatrans
   ```

3. **kprop** コマンドを使用して、データベースをスレーブ **KDC** に伝播します。

   ```
   # /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave_KDC
   ```

定期的に実行する cron ジョブ以外に、データベースをバックアップし、そのファイルをスレーブ KDC に伝播する場合は、次のように kprop_script コマンドを使用することもできます。

```
# /usr/lib/krb5/kprop_script slave_KDC
```

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。

並列伝播を利用すると、複数のスレーブ KDC 間でマスター KDC の伝播処理を分散できます。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、使用するサイトに 1 つのマスター KDC と 6 個のスレーブ KDC があるとします (図 15–2 参照)。 `slave-1` から `slave-3` で 1 つの論理グループを構成し、`slave-4` から `slave-6` で別の論理グループを構成しています。並列伝播を設定するには、マスター KDC がデータベースを `slave-1` と `slave-4` に伝播し、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。

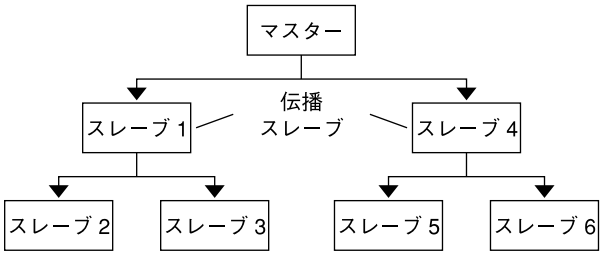


図 **15–2** 並列伝播の構成例

並列伝播を設定する方法

ここでは、並列伝播の詳細な手順は説明しませんが、並列伝播を有効にする構成手順の概要を示します。

1. マスター KDC 上で、`cron` ジョブ内の `kprop_script` エントリを変更して、次の伝播先のスレーブ KDC (伝播スレーブ) だけを引数に指定します。
2. 伝播スレーブごとに、`kprop_script` エントリをその `cron` ジョブに追加し、伝播先のスレーブを引数に指定します。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 – 伝播スレーブにかかる伝播時間は、ネットワークの帯域幅や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. スレーブ KDC ごとに、伝播に必要なアクセス権を設定します。伝播元の KDC のホスト主体名を各スレーブ KDC の kpropd.acl ファイルに追加します。

例 — 並列伝播の設定

図 15–2 のマスター 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
```

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 ユーティリティの stash コマンドを使用して、置き換えることができます。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ときは、stash ファイルも削除する必要があります。データベースを削除しても、stash ファイルは自動的に削除されないため、クリーンアップを完了するには、このファイルを削除する必要があります。

▼ stash ファイルを削除する方法

1. **stash** ファイルが配置されている **KDC** の上でスーパーユーザーになります。
2. **stash** ファイルを削除します。

   ```
   # 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。

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。

セキュリティの強化

SEAM アプリケーションサーバーと KDC サーバーのセキュリティを強化するには、次の手順に従ってください。

▼ KDC サーバーへのアクセスを制限する方法

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、SEAM 全体のセキュリティにとって重要です。

1. **/etc/inetd.conf** ファイルでリモートサービスを無効にします。

   KDC サーバーをセキュリティ保護するために、 /etc/inetd.conf ファイル内でリモートサービスを起動するエントリをコメントにして、不要なネットワークサービスをすべて無効にします。ほとんどの環境では、time と krdb5_kprop サービス以外は、無効にしてかまいません。ループバック TLI を使用するサービス (ticlts、ticotsord、および ticots) は、有効にしておくことができます。編集後のファイルは、次のようになります (簡単に示すために、ほとんどのコメントは削除されている)。

```
kdc1 # cat /etc/inetd.conf
#
#ident  "@(#)inetd.conf 1.33    98/06/02 SMI"   /* SVr4.0 1.5   */
  .
  .
#name     dgram   udp     wait    root    /usr/sbin/in.tnamed     in.tnamed
#
#shell    stream  tcp     nowait  root    /usr/sbin/in.rshd       in.rshd
#login    stream  tcp     nowait  root    /usr/sbin/in.rlogind    in.rlogind
#exec     stream  tcp     nowait  root    /usr/sbin/in.rexecd     in.rexecd
#comsat   dgram   udp     wait    root    /usr/sbin/in.comsat     in.comsat
#talk     dgram   udp     wait    root    /usr/sbin/in.talkd      in.talkd
#
#uucp     stream  tcp     nowait  root    /usr/sbin/in.uucpd      in.uucpd
#
#finger   stream  tcp     nowait  nobody  /usr/sbin/in.fingerd    in.fingerd
#
# Time サービスはクロック同期で使用される
#
time      stream  tcp     nowait  root    internal
time      dgram   udp     wait    root    internal
#
  .
  .
#
100234/1  tli rpc/ticotsord wait  root    /usr/lib/gss/gssd     gssd
#dtspc    stream  tcp     nowait  root    /usr/dt/bin/dtspcd      /usr/dt/bin/dtspcd
```

```
#100068/2-5 dgram rpc/udp wait    root    /usr/dt/bin/rpc.cmsd    rpc.cmsd
100134/1 tli rpc/ticotsord wait   root    /usr/lib/ktkt_warnd kwarnd
krb5_prop stream  tcp     nowait  root    /usr/lib/krb5/kpropd  kpropd
```

変更が完了したら、KDCをリブートします。

2. **KDC** をサポートするハードウェアに対するアクセスを制限します。

   物理的なアクセスを制限するために、KDC とそのモニターは安全な場所に設置します。このサーバーへのアクセスを完全に制限することが目的です。

3. **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。これらのファイルをローカルファイルシステムに格納する場合は、できるだけほかのシステムと共有しないでください。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。

第 16 章

SEAM エラーメッセージと障害追跡

この章では、SEAM を使用するときに発生するエラーメッセージの解決策と、さまざま問題を解決するためのヒントについて説明します。次に、この章で説明するエラーメッセージと障害追跡方法の一覧を示します。

SEAM のエラーメッセージ

この節では、SEAM 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。

SEAM 管理ツールのエラーメッセージ

プリンシパルまたはポリシーのリストにアクセスできません; 「名前」フィールドを使用してください (`Unable to view the list of principals or policies; use the Name field.`)

原因: ログインに使用した `admin` 主体には、Kerberos ACL ファイル (`kadm5.acl`) のリスト特権 (`l`) がありません。このため、主体リストまたはポリシーリストを表示できません。

対処方法: 主体名およびポリシー名を「名前 (Name)」フィールドに入力するか、適切な特権を持つ主体を使用してログインする必要があります。

`JNI: Java array creation failed`

`JNI: Java class lookup failed`

`JNI: Java field lookup failed`

`JNI: Java method lookup failed`

`JNI: Java object lookup failed`

`JNI: Java object field lookup failed`

`JNI: Java string access failed`

`JNI: Java string creation failed`

原因: SEAM 管理ツール (`gkadmin`) で使用される Java Native Interface で重大な問題が発生しました。

対処方法: `gkadmin` を終了して再起動してください。それでも問題が解決しない場合は、バグを報告してください。

SEAM 共通エラーメッセージ (A - M)

この節では、SEAM コマンド、SEAM 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。

major_error minor_error 名前をインポート中に `gssapi` エラー (*major_error minor_error* `gssapi error importing name`)

原因: サービス名をインポートしているときに、エラーが発生しました。

対処方法: サービス主体がホストのキータブファイル内に存在することを確認してください。

`kadmin` インタフェースを初期化中に、`krb5 admin` サーバーホスト名が無効です。(`Bad krb5 admin server hostname while initializing kadmin interface`)

原因: `krb5.conf` ファイルの `admin_server` に、無効なホスト名が設定されています。

対処方法: `krb5.conf` ファイルの `admin_server` 行に、マスター KDC の正しいホスト名を指定してください。

要求されたレルムの `KDC` に接続できません。(`Cannot contact any KDC for requested realm`)

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。

ホスト用のレルムを決定できません。(Cannot determine realm for host)
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) にドメイン名の割り当てを設定してください。

要求されたレルムの KDC が見つかりません。(Cannot find KDC for requested realm)
原因: 要求されたレルムに KDC が見つかりません。

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。

レルム *realm_name* を初期化できません。(cannot initialize realm *realm_name*)
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。krb5kdc を起動するには、/etc/init.d/kdc スクリプトを実行するのが最も簡単です。

要求されたレルムの KDC を解決できません。(Cannot resolve KDC for requested realm)
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。

パスワードは再利用できません。(Cannot reuse password)
原因: 入力したパスワードは、以前にこの主体によって使用されています。

対処方法: 以前に使用されたことのないパスワードを選択してください。KDC データベースに主体ごとに保持されているパスワード番号は、選択しないでください。このポリシーは、主体のポリシーによって適用されます。

転送された資格を取得できません。(Can't get forwarded credentials)
原因: 資格の転送ができません。

対処方法: この主体に転送可能な資格を設定してください。

Kerberos 構成ファイルを開けません / 見つかりません。(Can't open/find Kerberos configuration file)
原因: Kerberos 構成ファイル (krb5.conf) を使用できません。

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。

初期チケット要求でクライアント / サーバーレルムが一致していません。(Client/server realm mismatch in initial ticket request)

原因: 初期チケット要求で、クライアントとサーバーのレルムが一致していません。

対処方法: 通信しているサーバーがクライアントと同じレルムに配置されていること、またはレルム構成が正しいことを確認してください。

クライアントまたはサーバーの鍵が null です。(Client or server has a null key)

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。

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。(Communication failure with server while initializing kadmin interface)

原因: 管理サーバーとして入力したホスト (マスター KDC) 上で、kadmind デーモンが動作していません。

対処方法: マスター KDC に正しいホスト名が指定されていることを確認してください。ホスト名が正しい場合は、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。

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。(Credentials cache file permissions incorrect)

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。

資格キャッシュ入出力操作が失敗しました。*XXX* (Credentials cache I/O operation failed *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し、df コマンドを使用してデバイスの空き領域を確認してください。

復号化で整合性チェックが失敗しました。(Decrypt integrity check failed)

原因: チケットが無効である可能性があります。

対処方法:

1. 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てください。

2. 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。対象ホスト上で `klist -k` を使用して、鍵バージョン番号がその番号であることを確認します。

`df`：ファイルシステムを `statvfs` できません：引数が正しくありません。(`df: cannot statvfs` *filesystem*`: Invalid argument`)

原因: 適切な `root` 資格がないため、`df` コマンドを実行しても、現在マウントされている Kerberos NFS ファイルシステムにアクセスできません。マウントされている Kerberos ファイルシステムの資格を破棄しても、ファイルシステムは自動的にマウント解除されません。

対処方法: Kerberos ファイルシステムにアクセスするには、新しい `root` 資格を作成する必要があります。この Kerberos ファイルシステムにアクセスする必要がない場合は、ファイルのマウントを解除してください。

資格キャッシュを取得できませんでした。(`failed to obtain credentials cache`)

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` 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。

この実装ではフィールドが長すぎます。(`Field is too long for this implementation`)

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。Kerberos が処理できる最大メッセージ長は、65,535 バイトです。また、Kerberos 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: Kerberos アプリケーションから送信されたメッセージサイズが有効範囲であることを確認してください。

`GSS-API（または Kerberos）エラー（GSS-API（or Kerberos）error）`

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: `/etc/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 GSS-API 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。

ホスト名を展開できません。(`Hostname cannot be canonicalized`)

原因: Kerberos がホスト名を完全指定できません。

対処方法: このホスト名が DNS に定義され、ホスト名とアドレス間の双方向の割り当てについて整合性を確認してください。

レルム間のチケットが無効です。(Illegal cross-realm ticket)
原因: 送信されたチケットのレルム間関係が正しくありません。レルム間に正しい信頼関係が設定されていない可能性があります。

対処方法: 使用しているレルム間の信頼関係が正しいことを確認してください。

Kerberos 構成ファイルのフォーマットが不適切です。(Improper format of Kerberos configuration file)
原因: Kerberos 構成ファイル (krb5.conf) 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カッコで囲まれていることも確認してください。

メッセージのチェックサムのタイプが不適切です。(Inappropriate type of checksum in message)
原因: このメッセージに無効なチェックサムタイプが含まれています。

対処方法: krb5.conf および kdc.conf ファイルに指定されているチェックサムタイプが有効であることを確認してください。

ネットアドレスが間違っています。(Incorrect net address)
原因: ネットワークアドレスが一致しません。転送されたチケット内のネットワークアドレスが、チケットが処理されたときのネットワークアドレスと一致しません。このメッセージは、チケットの転送時に発生します。

対処方法: ネットワークアドレスが正しい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。

ファイルロックモードのフラグが無効です。(Invalid flag for file lock mode)
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。

符号化に対し無効なメッセージタイプが指定されました。(Invalid message type specified for encoding)
原因: Kerberos アプリケーションから送信されたメッセージ形式を、Kerberos が認識できません。

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。

文字クラス数が正しくありません。(Invalid number of character classes)
原因: 主体に入力したパスワードに、主体のポリシーによって適用された数のパスワードクラスが含まれていません。

対処方法: ポリシーに指定されている最小パスワードクラス数を使用して、パスワードを入力してください。

KADM エラー: メモリー割り当ての失敗です。(KADM err: Memory allocation failure)
原因: kadmin の実行に必要なメモリーが不足しています。

対処方法: メモリーを解放してから、kadmin を再実行してください。

KDC は要求したオプションを処理できません。(KDC can't fulfill requested option)
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 ポリシーは要求を拒否します。(KDC policy rejects request)
原因: KDC ポリシーが要求を許可しませんでした。たとえば、KDC に対する要求に IP アドレスが含まれていなかったり、要求された転送を KDC が許可しなかった可能性があります。

対処方法: 正しいオプションを指定して kinit を実行していることを確認してください。必要に応じて、主体に関連付けられたポリシーを変更するか、要求が許可されるように主体の属性を変更します。ポリシーまたは主体を変更するには、kadmin を使用します。

KDC 応答は予期したものと一致しませんでした。(KDC reply did not match expectations)
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 RFC1510 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、または KDC が有効であることを確認してください。

鍵テーブルエントリが見つかりません。(Key table entry not found)
原因: ネットワークアプリケーションサーバーのキータブファイルに、サービス主体のエントリがありません。

対処方法: サーバーのキータブファイルに適切なサービス主体を追加して、Kerberos サービスを提供できるようにしてください。

鍵テーブルのプリンシパルの鍵バージョン番号が正しくありません。(Key version number for principal in key table is incorrect)
原因: キータブファイルと Kerberos データベース内の主体の鍵バージョンが異なります。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、旧サービスチケットを使用しているため、鍵が古い可能性があります。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。

login: load_modules: /usr/lib/security/pam_krb5.so.1 モジュールを開けません。(login: load_modules: can not open module /usr/lib/security/pam_krb5.so.1)

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、/etc/pam.conf ファイルに pam_krb5.so.1 への正しいパスが指定されていることも確認してください。

krb5_get_in_tkt 内部でループが検出されました。(Looping detected inside krb5_get_in_tkt)

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。

マスター鍵がデータベースと一致しません。(Master key does not match database)

原因: 読み込まれたデータベースのダンプが、マスター鍵 (/var/krb5/.k5.*REALM* に配置されている) を含むデータベースから作成されませんでした。

対処方法: 読み込まれたデータベースダンプ内のマスター鍵が、/var/krb5/.k5.*REALM* に配置されているマスター鍵と一致していることを確認してください。

一致する資格が見つかりません。(Matching credential not found)

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てください。

メッセージの順序が違います。(Message out of order)

原因: 順次送信されたメッセージが順不同で着信しました。一部のメッセージが転送中に失われました。

対処方法: Kerberos セッションを再度初期化してください。

メッセージストリームが変更されました。(Message stream modified)
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でした。転送中のメッセージが変更された可能性があります。セキュリティ違反が発生している可能性があります。

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。このメッセージが送信中に改変された可能性もあるため、 kdestroy を使用してチケットを破棄し、使用している Kerberos サービスを再度初期化してください。

SEAM 共通エラーメッセージ (N - Z)

この節では、SEAM コマンド、SEAM 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。

資格キャッシュファイルが見つかりません。(No credentials cache file found)
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。存在しない場合は、kinit を再度実行します。

操作には "*privilege*" 特権が必要です。(Operation requires "*privilege*" privilege)
原因: 使用された admin 主体に対して、kadm5.acl ファイルに設定されている適切な特権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、kadm5.acl ファイルを変更して、使用した主体に適切な特権を割り当てます。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: Kerberos V5 認証に失敗しました: パスワードが正しくありません。(PAM-KRB5: Kerberos V5 authentication failed: password incorrect)
原因: UNIX パスワードと Kerberos パスワードが一致していません。ほとんどの Kerberos 以外のコマンド (login など) では 、PAM を介して Kerberos に自動的に認証されるように、UNIX パスワードに指定したパスワードが使用されます。パスワードが異なる場合、Kerberos 認証は失敗します。

対処方法: パスワードを要求されたら、Kerberos パスワードを入力します。

パスワードはパスワード辞書にあります。(Password is in the password dictionary)
原因: 入力したパスワードがパスワードディレクトリにすでに存在し、使用されています。選択したパスワードが適切ではありません。

対処方法: パスワードクラスを組み合わせたパスワードを選択してください。

リプレイキャッシュコードでアクセス権がありません。(Permission denied in replay cache code)
原因: システムの再実行キャッシュを開けませんでした。このサーバーは、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 (/usr/tmp/rc_ *service_name*)。現在の再実行キャッシュのアクセス権を変更する代わりに、再実行キャッシュを削除してから、Kerberos サーバーアプリケーションを別のユーザー ID で実行することもできます。

プロトコルバージョンが一致していません。(Protocol version mismatch)
原因: Kerberos V4 要求が KDC に送信された可能性があります。SEAM 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。

要求は再送です。(Request is a replay)
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。

要求したプリンシパルとチケットは一致しません。(Requested principal and ticket don't match)
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。

要求したプロトコルバージョンはサポートされていません。(Requested protocol version not supported)
原因: Kerberos V4 要求が KDC に送信された可能性があります。SEAM 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。

kadmin インタフェースを初期化中に、必須のパラメータが krb5.conf にありません。(Required parameters in krb5.conf missing while initializing kadmin interface)
原因: krb5.conf ファイル内のパラメータ (admin_server パラメータなど) が存在しません。

対処方法: 存在しないパラメータを判断して、krb5.conf ファイルに追加します。

サーバーが認証を拒否しました（sendauth 交換で）。(Server rejected authentication (during sendauth exchange))

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。

GSS サービス nfs@<host> の設定が失敗しました。NFS サービス資格を確認してください。(Set gss service nfs@<host> failed. Check nfs service credential)

原因: このメッセージは、share コマンドが「無効な引数」メッセージを表示して失敗したあとに、syslog により生成されます。キータブファイルが存在しないか、キータブファイル内に NFS サービス主体が存在しない可能性があります。

対処方法: 問題を特定するために、 klist -k を実行してキータブファイルが存在することを確認し、キータブファイル内にホストの NFS サービス主体が存在することを確認してください。

チケットはわれわれのものではありません。(The ticket isn't for us)

チケット / オーセンティケータが一致しません。(Ticket/authenticator don't match)

原因: チケットとオーセンティケータ (authenticator) が一致しません。要求内の主体名が、サービス主体の名前と一致していない可能性があります。送信されたチケットの主体が FQDN 名で、サービスが予期した主体が FQDN 以外の名前の場合 (またはこの逆の場合) に、この問題が発生します。

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。

チケットの有効期限が切れました。(Ticket expired)

原因: チケットが期限切れになっています。

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てください。

チケットには遅延処理の資格がありません。(Ticket is ineligible for postdating)

原因: この主体は、チケットの遅延を許可していません。

対処方法: kadmin を使用して主体を変更し、遅延を許可してください。

チケットはまだ有効ではありません。(Ticket not yet valid)

原因: 遅延チケットはまだ有効でありません。

対処方法: 正しい日付で新しいチケットを作成するか、現在のチケットが有効になるまで待ちます。

不完全な入力ファイルを検出しました。(`Truncated input file detected`)
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。

要求したプリンシパルは正しくありません。(`Wrong principal in request`)
原因: チケットの主体名が無効です。DNS または FQDN の問題が発生している可能性があります。

対処方法: サービスの主体とチケットの主体が一致していることを確認してください。

SEAM の障害追跡

この節では、SEAM ソフトウェアの障害追跡について説明します。

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は、NFS サーバーに `/var/tmp/rc_nfs` 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 サーバーの `inetd.conf` ファイル内に `gssd` のエントリが存在す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` のエラーメッセージが表示された場合は、NFS ファイルシステムをマウントするときに完全指定形式の DNS 名を使用していない可能性があります。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。SEAM 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
```

1 Ken Hornstein, “Kerberos FAQ,” [http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html], accessed 11 December 1998.

```
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 の認証の問題

使用するシステムのスーパーユーザーになるときの認証に失敗し、ホストのキータブファイルに root 主体がすでに追加されている場合は、2 つの問題を確認する必要があります。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。

第 17 章

主体とポリシーの管理 (手順)

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。

この章は、主体とポリシーを管理する必要のあるユーザーを対象にしています。主体とポリシーを計画するときの考慮事項など、主体とポリシーについて理解している必要があります。第 13 章および第 14 章を参照してください。

この章で説明する情報は次のとおりです。

主体とポリシーの管理方法

マスター KDC の Kerberos データベースには、使用するレルムの 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。主体を作成または削除したり、主体の属性を変更したりするには、`kadmin` または `gkadmin` 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。`kadmin` コマンドには、次の 2 つの種類があります。

- `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る
- `kadmin.local` - マスター KDC 上で直接実行する必要がある

Kerberos を使用してユーザーを認証する点を除いて、2 つの kadmin の機能は同じです。kadmin に必要なデータベースを設定するときは、kadmin.local を使用します。

SEAM には、SEAM 管理ツール (gkadmin) も用意されています。このツールは対話型のグラフィカルユーザーインタフェース (GUI) で、基本的に kadmin コマンドと同じ機能を持ちます。詳細は、280 ページの「SEAM 管理ツール」 を参照してください。

SEAM 管理ツール

SEAM 管理ツールは、対話型グラフィカルユーザーインタフェース (GUI) で、Kerberos 主体とポリシーを管理することができます。このツールは、kadmin コマンドと同じ機能を持ちます。ただし、キータブファイルの管理はサポートしません。キータブファイルを管理するには、kadmin コマンドを使用する必要があります(309 ページの「キータブファイルの管理」を参照)。

kadmin コマンドと同様に、SEAM ツールは、Kerberos 認証と暗号化された RPC 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することができます。SEAM ツールでは、次の操作を行うことができます。

- デフォルト値または既存の主体をベースに新しい主体を作成する
- 既存のポリシーをベースに新しいポリシーを作成する
- 主体のコメントを追加する
- 新しい主体を作成するときのデフォルト値を設定する
- ツールを終了しないで別の主体としてログインする
- 主体一覧とポリシー一覧を印刷または保存する
- 主体一覧とポリシー一覧を表示および検索する

SEAM ツールでは、コンテキストヘルプと一般的なオンラインヘルプも利用できます。

SEAM ツールを使用して実行できる操作について、次の作業マップで説明します。

- 285 ページの「主体の管理 (作業マップ)」
- 297 ページの「ポリシーの管理 (作業マップ)」

また、SEAM ツールで指定または表示できる主体属性とポリシー属性については、305 ページの「SEAM ツールパネルの説明」を参照してください。

SEAM ツールに対応するコマンド行

この節では、SEAM ツールと同じ機能を提供する kadmin コマンドを示します。これらのコマンドは、X Window System で実行しなくても使用できます。この章のほとんどの手順では、SEAM ツールを使用します。ただし、多くの手順では、対応するコマンド行の使用例も挙げています。

表 **17–1** SEAM ツールに対応するコマンド行

SEAM ツールの手順	対応する **kadmin** コマンド
主体の一覧の表示	`list_principals` または `get_principals`
主体の属性の表示	`get_principal`
新しい主体の作成	`add_principal`
主体の複製	対応するコマンド行なし
主体の変更	`modify_principal` または `change_password`
主体の削除	`delete_principal`
新しい主体を作成するときのデフォルトの設定	対応するコマンド行なし
ポリシーの一覧の表示	`list_policies` または `get_policies`
ポリシーの属性の表示	`get_policy`
新しいポリシーの作成	`add_policy`
ポリシーの複製	対応するコマンド行なし
ポリシーの変更	`modify_policy`
ポリシーの削除	`delete_policy`

SEAM ツールによって変更されるファイル

SEAM ツールが変更するファイルは、`$HOME/.gkadmin` ファイルだけです。このファイルには、新しい主体を作成するときのデフォルト値が含まれます。このファイルを更新するには、「Edit」メニューから「Properties」を選択します。

SEAM ツールの印刷機能とオンラインヘルプ機能

SEAM ツールには、印刷機能とオンラインヘルプ機能が用意されています。「Print」メニューから、次の要素をプリンタまたはファイルに送信できます。

- 指定したマスター KDC で使用できる主体の一覧

- 指定したマスター KDC で使用できるポリシーの一覧
- 現在選択されている主体または読み込まれている主体
- 現在選択されているポリシーまたは読み込まれているポリシー

「Help」メニューから、コンテキストヘルプと通常のヘルプを使用できます。「Help」メニューから「Context-Sensitive Help」を選択すると、「Context-Sensitive Help」ウィンドウが表示され、ツールがヘルプモードに切り替わります。ヘルプモードのウィンドウで、任意のフィールド、ラベル、またはボタンをクリックすると、「Help」ウィンドウにその項目のヘルプが表示されます。ツールの通常モードに戻るには、「Help」ウィンドウで「Dismiss」をクリックします。

「Help Contents」を選択すると、HTML ブラウザが開き、この章で説明している概要や操作情報が表示されます。

SEAM ツールで大規模な一覧を使用する

登録した主体とポリシーが増加するにつれて、SEAM ツールが主体とポリシーを読み込んでそれらの一覧を表示する時間が長くなります。このため、ツールによる作業効率が低下します。この問題には、いくつかの対応方法があります。

まず、一覧を読み込む時間を完全になくすために、SEAM ツールに一覧を読み込まないようにします。この方法を設定するには、「Edit」メニューから「Properties」を選択し、「Show Lists」フィールドのチェックマークをはずします。一覧を読み込まない場合、一覧は表示されないため、一覧を使用して主体またはポリシーを選択できなくなります。代わりに、表示された新しい「Name」フィールドに主体またはポリシー名を入力し、その主体またはポリシーに適用する操作を選択する必要があります。名前を入力する操作は、一覧から項目を選択する操作と同じ効果を持ちます。

大規模な一覧を使用するときは、キャッシュを利用することもできます。SEAM ツールのデフォルトの動作として、一定量の一覧がキャッシュに格納されるように設定されています。SEAM ツールは、最初に一覧をキャッシュに読み込む必要がありますが、そのあとは一覧を再度読み込まずにキャッシュを使用できます。この方法では、サーバーから時間をかけて何回も一覧を読み込む必要がありません。

一覧がキャッシュに格納されるように設定するには、「Edit」メニューから「Properties」を選択します。キャッシュの設定には、次の 2 つの方法があります。一覧をキャッシュに永続的に格納するか、制限時間を指定します。制限時間を指定した場合は、その時間が経過すると、ツールはサーバーの一覧をキャッシュに再度読み込みます。

一覧をキャッシュに格納しても、一覧から主体とポリシーを選択することができます。このため、一覧を読み込まない最初の方法と異なり、SEAM ツールの利用には影響しません。また、キャッシュを利用した場合でも、ほかの主体とポリシーの変更を確認できなくなることがあります。ただし、使用している主体とポリシーを変更したときは最新の一覧が表示されます。主体とポリシーを変更すると、サーバーと

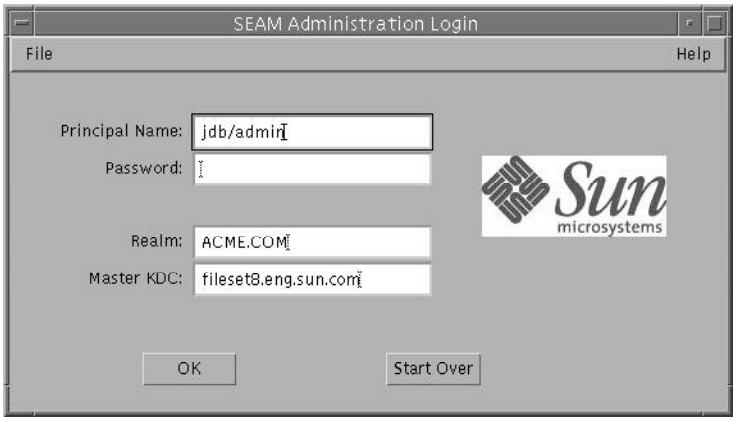
キャッシュの一覧が更新されるためです。キャッシュを更新して、ほかの主体とポリシーの変更を確認し、最新の一覧を取得するには、任意のタイミングで「Refresh」メニューを使用します。サーバーから一覧が読み込まれ、キャッシュを更新することができます。

▼ SEAM ツールを起動する方法

1. **gkadmin** コマンドを使用して **SEAM** ツールを起動します。

   ```
   $ /usr/sbin/gkadmin
   ```

   「SEAM Administration Login」ウィンドウが表示されます。



2. デフォルト値を使用しない場合は、新しいデフォルト値を指定します。

   ウィンドウには、デフォルト値が自動的に表示されます。デフォルトの主体名は、`USER` 環境変数の現在の ID に `/admin` が付加されて作成されます (*username*`/admin`)。デフォルトの「Realm」フィールドおよび「Master KDC」フィールドは、`/etc/krb5/krb5.conf` ファイルから選択されます。デフォルト値を再度取得する場合は、「Start Over」をクリックします。

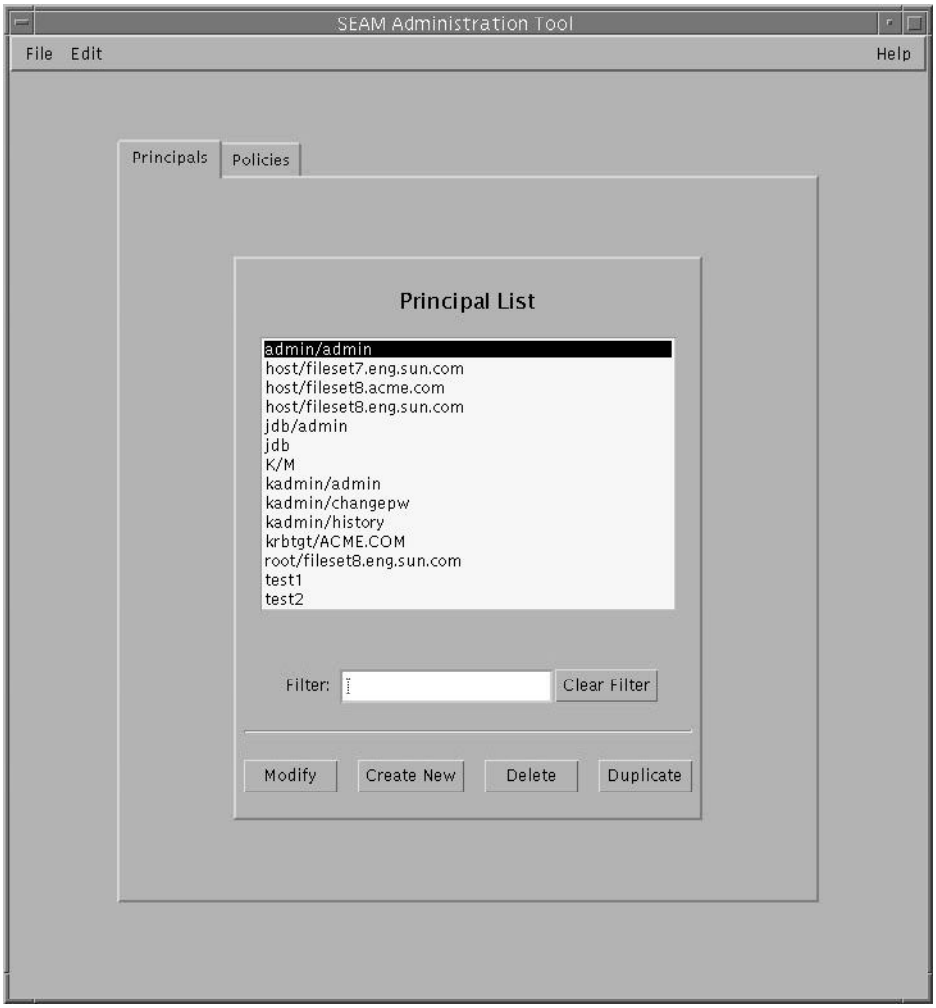
   注 – 各「Principal Name」が実行できる管理操作は、Kerberos ACL ファイルの `/etc/krb5/kadm5.acl` で規定されます。権限の制限については、308 ページの「Kerberos 管理権限を制限して SEAM ツールを使用する」を参照してください。

   ---

3. 指定した主体名のパスワードを入力します。

4. 「了解 **(OK)**」をクリックします。

   次のウィンドウが表示されます。



主体の管理

この節では、SEAM ツールを使用して主体を管理する手順について説明します。また、対応するコマンド行がある場合は、その例も示します。

主体の管理 (作業マップ)

作業	説明	参照先
主体の一覧の表示	「Principals」タブをクリックして、主体の一覧を表示する	286 ページの「主体の一覧を表示する方法」
主体の属性の表示	「Principal List」の「Principal」を選択し、「Modify」ボタンをクリックして、主体の属性を表示する	288 ページの「主体の属性を表示する方法」
新しい主体の作成	「Principal List」パネルの「Create New」ボタンをクリックして、新しい主体を作成する	290 ページの「新しい主体を作成する方法」
主体の複製	「Principal List」から複製する主体を選択し、「Duplicate」ボタンをクリックして、主体を複製する	292 ページの「主体を複製する方法」
主体の変更	「Principal List」から変更する主体を選択し、「Modify」ボタンをクリックして、主体を変更する 主体名は変更できない。主体名を変更するときは、主体を複製し、新しい名前を指定して保存してから、古い主体を削除する必要がある	292 ページの「主体を変更する方法」
主体の削除	「Principal List」から削除する主体を選択し、「Delete」ボタンをクリックして、主体を削除する	293 ページの「主体を削除する方法」
新しい主体を作成するときのデフォルトの設定	「Edit」メニューから「Properties」を選択して、新しい主体を作成するときのデフォルトを設定する	294 ページの「新しい主体を作成するときのデフォルトを設定する方法」
Kerberos 管理権限の変更 (`kadm5.acl` ファイル)	コマンド行のみ。 Kerberos 管理権限により、主体が Kerberos データベースに対して実行できる操作 (追加、変更など) が決定される。各主体の Kerberos 管理権限を変更するときは、`/etc/krb5/kadm5.acl` ファイルを編集する必要がある	295 ページの「Kerberos 管理権限を変更する方法」

新しい主体の自動作成

SEAM ツールは簡単に使用できますが、新しい主体を自動作成することができません。10 個または 100 個などの新しい主体を短時間で作成する場合は、自動作成を利用すると便利です。Bourne シェルスクリプトで `kadmin.local` コマンドを使用すると、主体を自動作成できます。

次のシェルスクリプト行は、新しい主体を自動作成する方法の例を示します。

```
sed -e 's/^\(.*\)$/ank +needchange -pw \1 \1/' < princnames |
        time /usr/sbin/kadmin.local> /dev/null
```

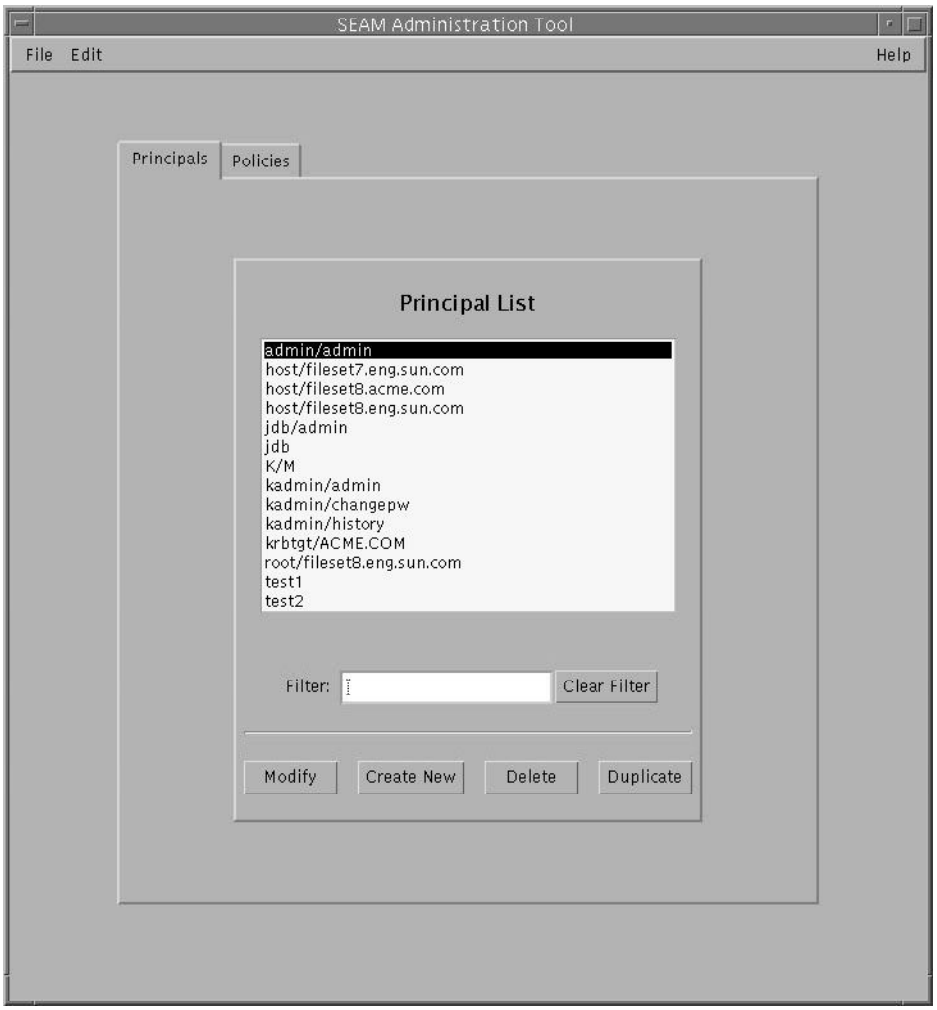
この例は、見やすいように 2 行に分割しています。このスクリプトは、princnames というファイルを読み込んで、そこに含まれている主体名とそのパスワードを Kerberos データベースに追加します。princnames ファイルをあらかじめ作成する必要があります。このファイルの各行には、主体とそのパスワードを 1 つ以上の空白で区切って指定します。主体に +needchange オプションを指定すると、ユーザーがその主体を使用して初めてログインしたときに、新しいパスワードを要求するプロンプトが表示されます。この方法を使用すると、princnames ファイル内のパスワードのセキュリティが向上します。

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。

▼ 主体の一覧を表示する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Principals**」タブをクリックします。
   主体の一覧が表示されます。



3. 特定の主体を表示するか、主体の部分リストを表示します。

   「Filter」フィールドにフィルタ文字列を入力して、`Return` キーを押します。フィルタが正常終了すると、フィルタに一致する主体の一覧が表示されます。

   フィルタ文字列は、1 文字以上の文字列である必要があります。フィルタメカニズムでは大文字と小文字が区別されるため、大文字と小文字を正しく指定する必要があります。たとえば、フィルタ文字列に `ge` と入力すると、主体名に文字列 `ge` を含む主体 (`george`、`edge` など) だけが表示されます。

   すべての主体を表示するには、「Clear Filter」をクリックします。

例 — 主体の一覧の表示 (コマンド行)

次の例では、kadmin の list_principals コマンドを使用して、 test* と一致するすべての主体を表示します。list_principals コマンドでは、ワイルドカードを使用できます。

```
kadmin: list_principals test*
test1@EXAMPLE.COM
test2@EXAMPLE.COM
kadmin: quit
```

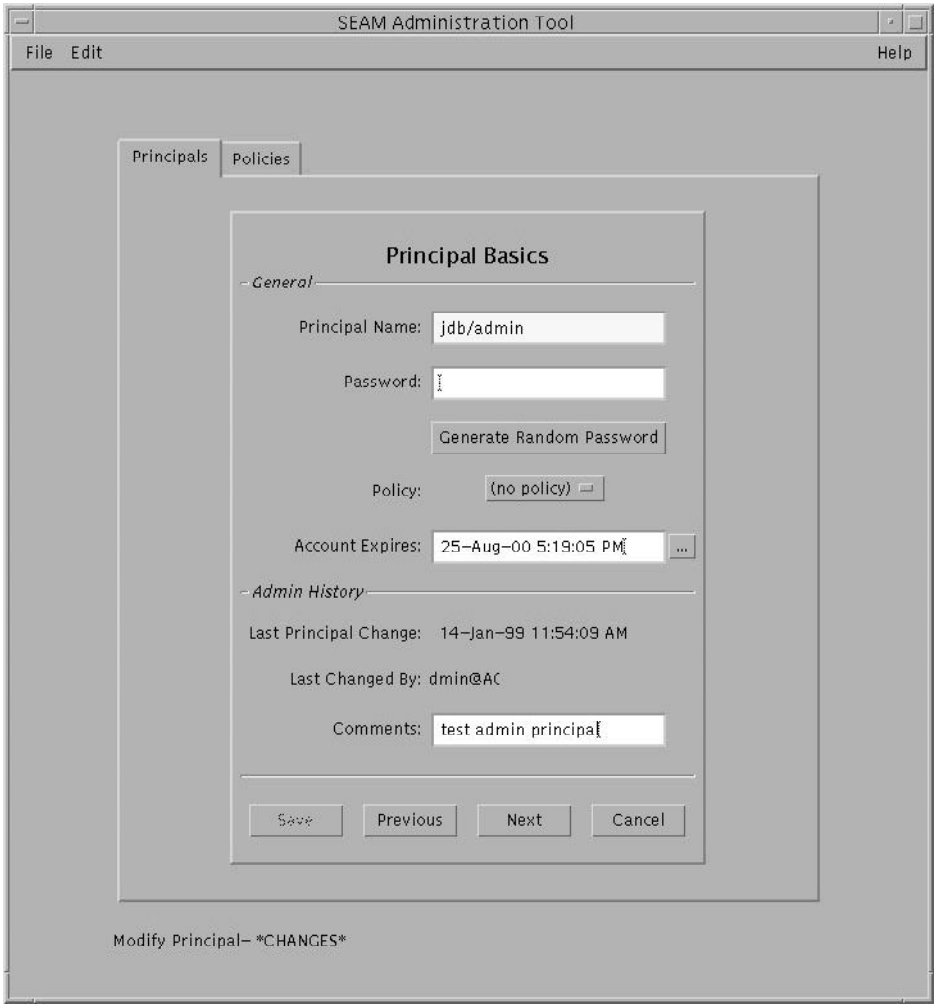
▼ 主体の属性を表示する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Principals**」タブをクリックします。
3. 表示する主体を一覧から選択して、「**Modify**」をクリックします。
   「Principal Basic」パネルが表示され、主体の属性の一部が示されます。
4. 「**Next**」をクリックして、主体のすべての属性を表示します。
   属性情報は、3 つのウィンドウに表示されます。「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。主体のすべての属性の説明については、305 ページの「SEAM ツールパネルの説明」 を参照してください。
5. 表示を終了する場合は、「**Cancel**」をクリックします。

例 — 主体の属性の表示

次の例は、jdb/admin 主体を表示したときの最初のウィンドウです。



例 — 主体の属性の表示 (コマンド行)

次の例では、kadmin の get_principal コマンドを使用して、jdb/admin 主体の属性を表示します。

```
kadmin: getprinc jdb/admin
Principal: jdb/admin@EXAMPLE.COM
Expiration date: Fri Aug 25 17:19:05 PDT 2000
Last password change: [never]
Password expiration date: Wed Apr 14 11:53:10 PDT 1999
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
```

```
Last modified: Thu Jan 14 11:54:09 PST 1999 (admin/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 1
Key: vno 1, DES cbc mode with CRC-32, no salt
Attributes: REQUIRES_HW_AUTH
Policy: [none]
kadmin: quit
```

▼ 新しい主体を作成する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。

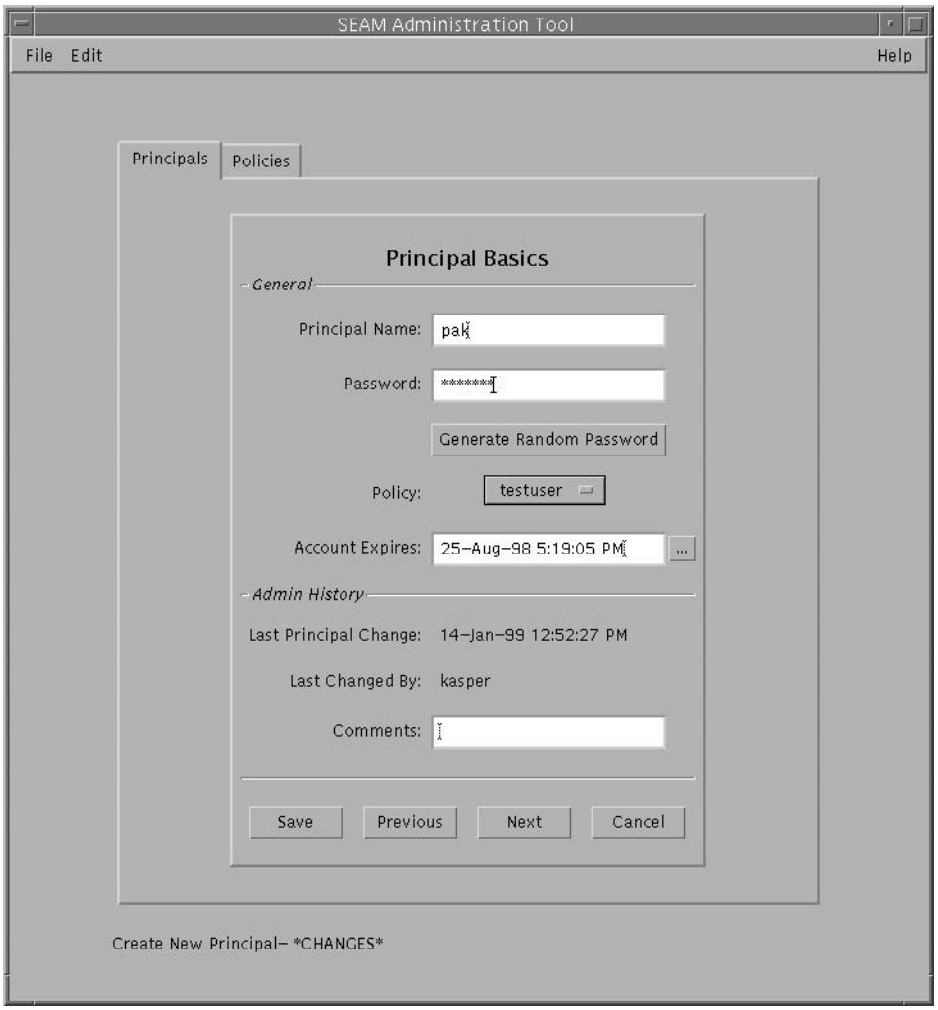
   注 – 新しい主体を作成するときに、新しいポリシーが必要な場合は、新しいポリシーを作成してから新しい主体を作成する必要があります。 301 ページの「新しいポリシーを作成する方法」を参照してください。

   ---

2. 「**Principals**」タブをクリックします。
3. 「**New**」をクリックします。
   「Principal Basics」パネルが表示され、主体の属性の一部が示されます。
4. 主体名とパスワードを指定します。
   主体名とパスワードは必須です。
5. 主体の属性に値を指定します。「**Next**」をクリックして、属性の値を必要に応じて指定します。
   属性情報は、3 つのウィンドウに表示されます。「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。主体のすべての属性の説明については、305 ページの「SEAM ツールパネルの説明」 を参照してください。
6. 主体を保存する場合は、「**Save**」をクリックします。または、最後のパネルで「**Done**」をクリックします。
7. 必要に応じて、新しい主体の **Kerberos** 管理権限を **`/etc/krb5/kadm5.acl`** ファイルに設定します。
   詳細は、295 ページの「Kerberos 管理権限を変更する方法」を参照してください。

例 — 新しい主体の作成

次の例は、pak という新しい主体を作成するときの「Principal Basics」パネルです。ポリシーには、testuser が設定されています。



例 — 新しい主体の作成 (コマンド行)

次の例では、kadmin の add_principal コマンドを使用して、 pak という新しい主体を作成します。この主体のポリシーには、testuser が設定されています。

```
kadmin: add_principal -policy testuser pak
Enter password for principal "pak@EXAMPLE.COM": <パスワードを入力する>
```

```
Re-enter password for principal "pak@EXAMPLE.COM": <パスワードを再入力する>
Principal "pak@EXAMPLE.COM" created.
kadmin: quit
```

▼ 主体を複製する方法

この手順では、既存の主体の一部またはすべてを使用して、新しい主体を作成する方法について説明します。この手順に対応するコマンド行はありません。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Principals**」タブをクリックします。
3. 複製する主体を一覧から選択して、「**Duplicate**」をクリックします。
   「Principal Basics」パネルが表示されます。選択した主体のすべての属性が複製されます。ただし、「Principal Name」と「Password」フィールドは複製されず、空で表示されます。
4. 主体名とパスワードを指定します。
   主体名とパスワードは必須です。選択した主体をそのまま複製するときは、「Save」をクリックして、手順 7 に進みます。
5. 主体の属性に別の値を指定します。「**Next**」をクリックして、属性の値を必要に応じて指定します。
   属性情報は、3 つのウィンドウに表示されます。「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。主体のすべての属性の説明については、305 ページの「SEAM ツールパネルの説明」を参照してください。
6. 主体を保存する場合は、「**Save**」をクリックします。または、最後のパネルで「**Done**」をクリックします。
7. 必要に応じて、主体の **Kerberos** 管理権限を **`/etc/krb5/kadm5.acl`** ファイルに設定します。
   詳細は、295 ページの「Kerberos 管理権限を変更する方法」を参照してください。

▼ 主体を変更する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Principals**」タブをクリックします。

3. 変更する主体を一覧から選択して、「**Modify**」をクリックします。
   「Principal Basic」パネルが表示され、主体の属性の一部が示されます。

4. 主体の属性を変更します。「**Next**」をクリックして、必要に応じて属性を変更します。
   属性情報は、3 つのウィンドウに表示されます。「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。主体のすべての属性の説明については、305 ページの「SEAM ツールパネルの説明」を参照してください。

注 – 主体名は変更できません。主体名を変更するときは、主体を複製し、新しい名前を指定して保存してから、古い主体を削除する必要があります。

---

5. 主体を保存する場合は、「**Save**」をクリックします。または、最後のパネルで「**Done**」をクリックします。

6. **`/etc/krb5/kadm5.acl`** ファイルで、主体の **Kerberos** 管理権限を変更します。
   詳細は、295 ページの「Kerberos 管理権限を変更する方法」を参照してください。

例 — 主体のパスワードの変更 (コマンド行)

次の例では、`kadmin` の `change_password` コマンドを使用して、`jdb` 主体のパスワードを変更します。`change_password` コマンドでは、主体のパスワード履歴に存在するパスワードには変更できません。

```
kadmin: change_password jdb
Enter password for principal "jdb": <新しいパスワードを入力する>
Re-enter password for principal "jdb": <パスワードを再度入力する>
Password for "jdb@EXAMPLE.COM" changed.
kadmin: quit
```

主体のその他の属性を変更するには、`kadmin` の `modify_principal` コマンドを使用する必要があります。

▼ 主体を削除する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。

2. 「**Principals**」タブをクリックします。

3. 削除する主体を一覧から選択して、「**Delete**」をクリックします。

削除を確定すると、主体が削除されます。

4. **Kerberos** アクセス制御リスト **(ACL)** ファイル **/etc/krb5/kadm5.acl** から主体を削除します。
   詳細は、295 ページの「Kerberos 管理権限を変更する方法」を参照してください。

例 — 主体を削除する (コマンド行)

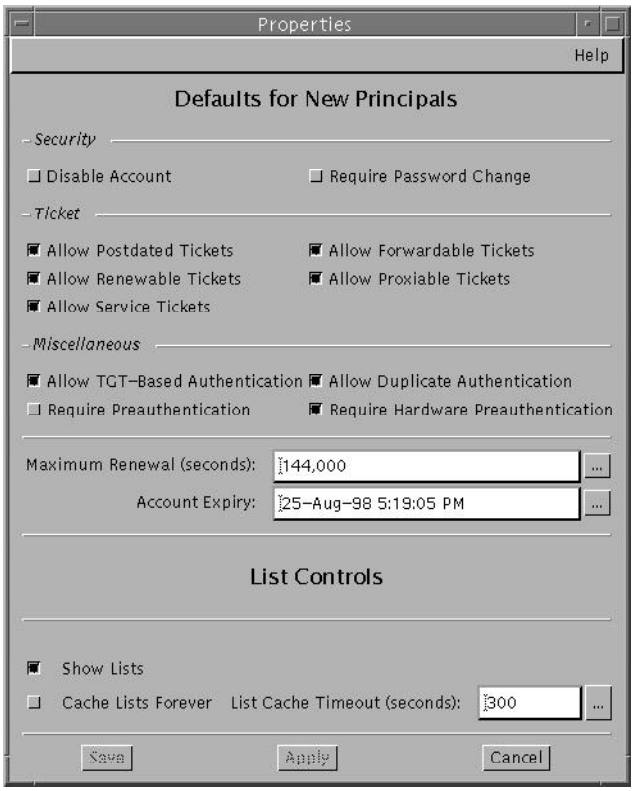
次の例では、kadminの delete_principal コマンドを使用して、jdb 主体を削除します。

```
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
```

▼ 新しい主体を作成するときのデフォルトを設定する方法

この手順に対応するコマンド行はありません。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Edit**」メニューから「**Properties**」を選択します。
   「Properties」ウィンドウが表示されます。



3. 新しい主体を作成するときのデフォルトを選択します。
   「Help」メニューから「Context-Sensitive Help」を選択すると、各ウィンドウの属性に関する情報が表示されます。
4. 「**Save**」をクリックします。

▼ Kerberos 管理権限を変更する方法

使用する環境には、多くのユーザー主体が登録されていると思われます。しかし、Kerberos データベースの管理者は通常、少数のユーザーだけに割り当てます。Kerberos データベースを管理する権限は、Kerberos アクセス制御リスト (ACL) ファイル (`kadm5.acl`) によって判断されます。`kadm5.acl` ファイルを使用すると、主体ごとに権限を設定できます。主体名にワイルドカード (*) を使用すると、複数の主体に権限を指定できます。

1. マスター **KDC** 上でスーパーユーザーになります。
2. **`/etc/krb5/kadm5.acl`** ファイルを編集します。

kadm5.acl ファイルのエントリは、次の書式で記述してください。

principal *privileges* [*principal-target*]

principal
: 権限を与える主体を指定する。主体名の任意の場所にワイルドカード (*) を使用できる。複数の主体グループに同じ権限を与えるときに使用する。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* を使用する

  admin インスタンスは通常、個別の権限 (Kerberos データベースへの管理アクセス権など) を個別の Kerberos 主体に許可するときに使用する。たとえば、ユーザー jdb が、jdb/admin という管理目的の主体を持つとする。この場合、ユーザー jdb は、この権限を実際に使用するときにだけ、jdb/admin チケットを取得する

privileges
: 主体が実行できる操作または実行できない操作を指定する。このフィールドは、次に示す 1 つまたは複数の文字列 (またはその大文字) の組み合わせから構成される。大文字の指定、または指定なしは許可されない。小文字の指定は許可される

  a — 主体またはポリシーの追加を許可する/しない

  d — 主体またはポリシーの削除を許可する/しない

  m — 主体またはポリシーの変更を許可する/しない

  c — 主体のパスワードの変更を許可する/しない

  i — Kerberos データベースの照会を許可する/しない

  l — Kerberos データベース内の主体またはポリシーの一覧表示を許可する/しない

  x または * — すべての権限 (admcil) を許可する

principal-target
: このフィールドに主体を指定すると、*principal* の操作が *principal_target* の場合にだけ、*privileges* が *principal* に適用される。主体名の任意の場所にワイルドカード (*) を使用できる。主体をグループ化するときに使用する

例 — Kerberos 管理権限の変更

kadm5.acl ファイル内の次のエントリは、EXAMPLE.COM レルム内で admin インスタンスを持つ すべての主体に対して、Kerberos データベース上のすべての権限を与えます。

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

kadm5.acl ファイル内の次のエントリは、jdb@EXAMPLE.COM 主体に対して、root インスタンスを持つすべての主体に関する追加、一覧表示、および照会の権限を与えます。

```
jdb@EXAMPLE.COM ali */root@EXAMPLE.COM
```

ポリシーの管理

この節では、SEAM ツールを使用してポリシーを管理する手順について説明します。また、対応するコマンド行がある場合は、その例も示します。

ポリシーの管理 (作業マップ)

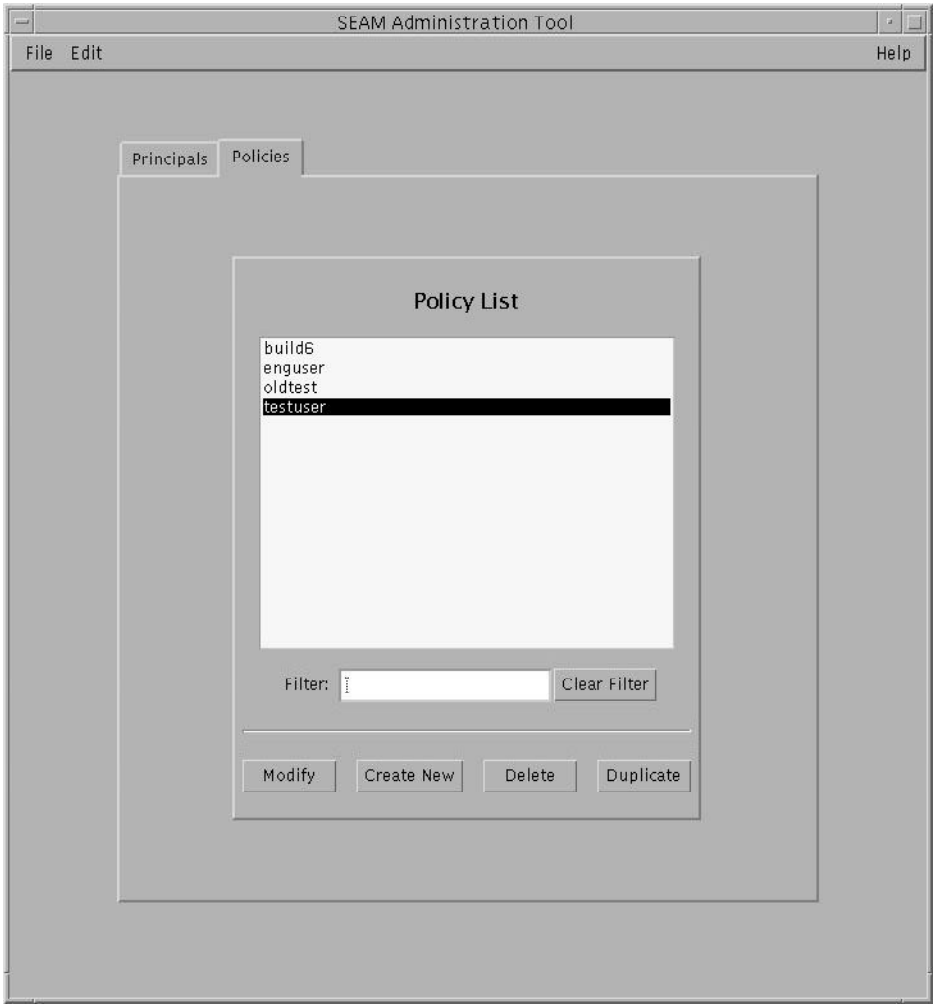
作業	説明	参照先
ポリシーの一覧の表示	「Policies」タブをクリックして、ポリシーの一覧を表示する	297 ページの「ポリシーの一覧を表示する方法」
ポリシーの属性の表示	「Policy List」からポリシーを選択し、「Modify」ボタンをクリックして、ポリシーの属性を表示する	299 ページの「ポリシーの属性を表示する方法」
新しいポリシーの作成	「Policy List」パネルの「Create New」ボタンをクリックして、新しいポリシーを作成する	301 ページの「新しいポリシーを作成する方法」
ポリシーの複製	複製するポリシーを「Policy List」ポリシーから選択し、「Duplicate」ボタンをクリックして、ポリシーを複製する	303 ページの「ポリシーを複製する方法」
ポリシーの変更	変更するポリシーを「Policy List」ポリシーから選択し、「Modify」ボタンをクリックして、ポリシーを変更する ポリシー名は変更できない。ポリシー名を変更するときは、ポリシーを複製し、新しい名前を指定して保存してから、古いポリシーを削除する必要がある	303 ページの「ポリシーを変更する方法」
ポリシーの削除	削除するポリシーを「Policy List」ポリシーから選択し、「Delete」ボタンをクリックして、ポリシーを削除する	304 ページの「ポリシーを削除する方法」

▼ ポリシーの一覧を表示する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Policies**」タブをクリックします。

ポリシーの一覧が表示されます。



3. 特定のポリシーを表示するか、ポリシーの部分リストを表示します。

   「Filter」フィールドにフィルタ文字列を入力して、`Return` キーを押します。フィルタが正常終了すると、フィルタに一致するポリシーの一覧が表示されます。

   フィルタ文字列は、1 文字以上の文字列である必要があります。フィルタメカニズムでは大文字と小文字が区別されるため、大文字と小文字を正しく指定する必要があります。たとえば、フィルタ文字列に `ge` と入力すると、ポリシー名に文字列 `ge` を含むポリシー (`george`、`edge` など) だけが表示されます。

   すべてのポリシーを表示するには、「Clear Filter」をクリックします。

例 — ポリシーの一覧の表示 (コマンド行)

次の例では、kadmin の list_policies コマンドを使用して、*user* と一致するすべてのポリシーを表示します。list_policies コマンドでは、ワイルドカードを使用できます。

```
kadmin: list_policies *user*
testuser
enguser
kadmin: quit
```

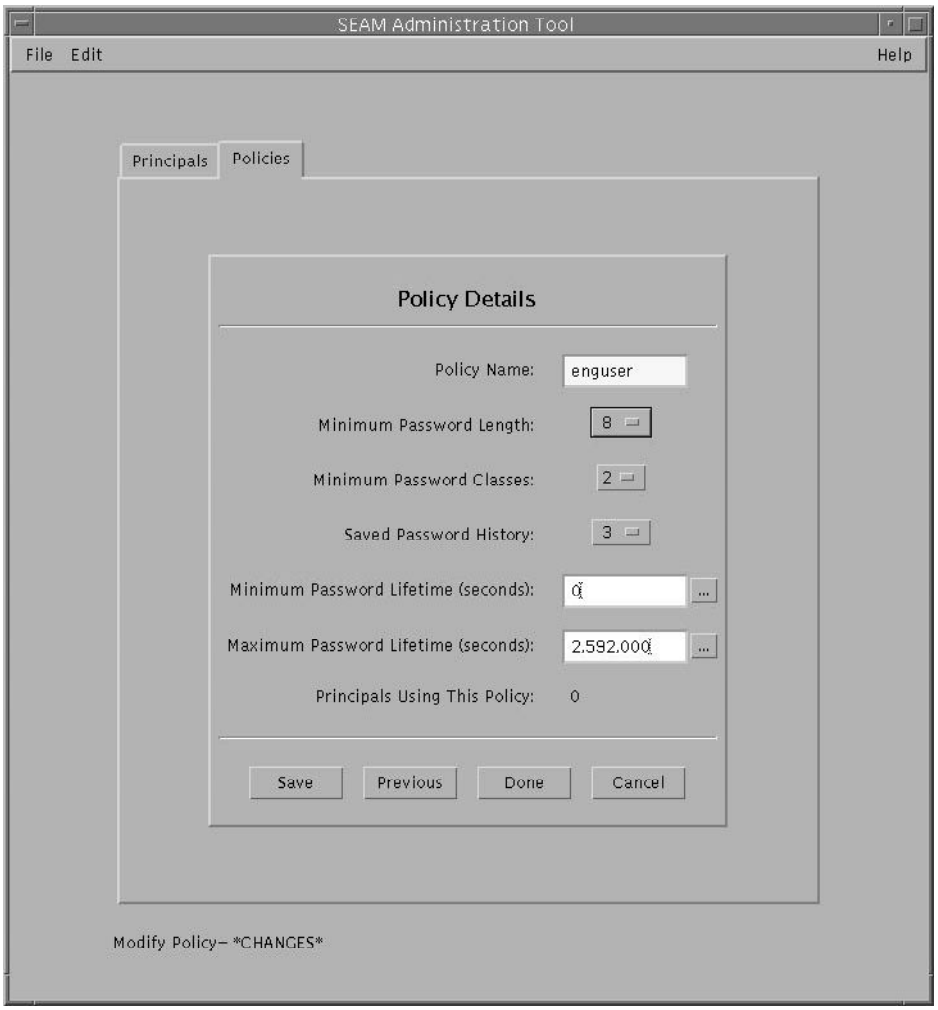
▼ ポリシーの属性を表示する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Policies**」タブをクリックします。
3. 表示するポリシーを一覧から選択して、「**Modify**」をクリックします。
   「Policy Details」パネルが表示されます。
4. 表示を終了する場合は、「**Cancel**」をクリックします。

例 — ポリシーの属性の表示

次の例は、test ポリシーを表示したときの「Policy Details」パネルです。



例 — ポリシーの属性の表示 (コマンド行)

次の例では、kadmin の get_policy コマンドを使用して、enguser ポリシーの属性を表示します。

```
kadmin: get_policy enguser
Policy: enguser
Maximum password life: 2592000
Minimum password life: 0
Minimum password length: 8
Minimum number of password character classes: 2
Number of old keys kept: 3
```

```
Reference count: 0
kadmin: quit
```

参照数は、このポリシーを使用する主体の数です。

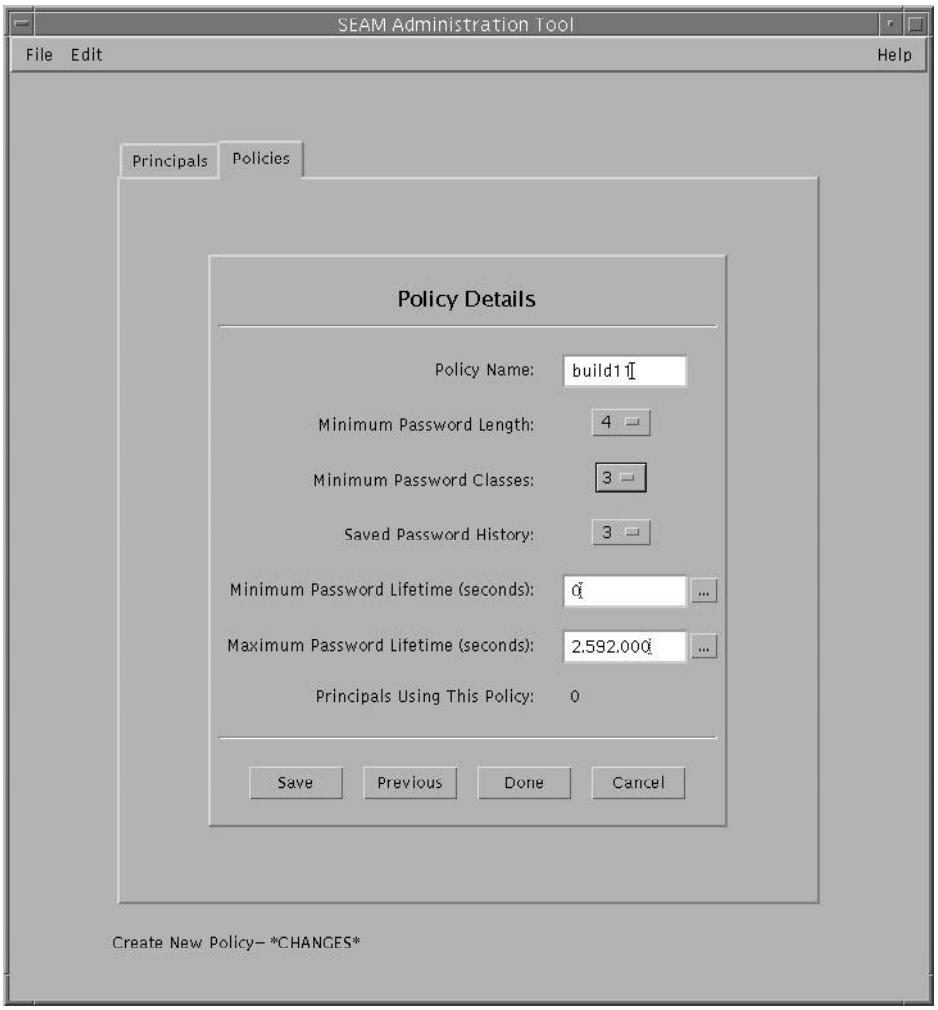
▼ 新しいポリシーを作成する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Policies**」タブをクリックします。
3. 「**New**」をクリックします。
   「Policy Details」パネルが表示されます。
4. 「**Policy Name**」フィールドにポリシー名を指定します。
   ポリシー名は必須です。
5. ポリシーの属性の値を指定します。
   「Help」メニューから「Context-Sensitive Help」を選択すると、このウィンドウの属性に関する情報が表示されます。ポリシーのすべての属性の説明については、表 17–5 を参照してください。
6. 「**Save**」をクリックしてポリシーを保存するか、「**Done**」をクリックします。

例 — 新しいポリシーの作成

次の例では、`build11` という新しいポリシーを作成します。「Minimum Password Classes」は、3 に設定されています。



例 — 新しいポリシーの作成 (コマンド行)

次の例では、kadminの add_policy コマンドを使用して、build11 ポリシーを作成します。このポリシーのパスワードには、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
```

▼ ポリシーを複製する方法

この手順では、既存のポリシーの一部またはすべてを使用して、新しいポリシーを作成する方法について説明します。この手順に対応するコマンド行はありません。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Policies**」タブをクリックします。
3. 複製するポリシーを一覧から選択して、「**Duplicate**」をクリックします。
   「Policy Details」パネルが表示されます。選択したフィールドのすべての属性が複製されます。ただし、「Policy Name」フィールドは空で表示されます。
4. 複製するポリシー名を「**Policy Name**」フィールドに指定します。
   ポリシー名は必須です。選択したポリシーをそのまま複製するには、「Save」をクリックして 手順 6 に進みます
5. ポリシーの属性に別の値を指定します。
   「Help」メニューから「Context-Sensitive Help」を選択すると、このウィンドウの属性に関する情報が表示されます。ポリシーのすべての属性の説明については、表 17–5 を参照してください。
6. 「**Save**」をクリックしてポリシーを保存するか、「**Done**」をクリックします。

▼ ポリシーを変更する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Policies**」タブをクリックします。
3. 変更するポリシーを一覧から選択して「**Modify**」をクリックします。
   「Policy Details」パネルが表示されます。
4. ポリシーの属性を変更します。
   「Help」メニューから「Context-Sensitive Help」を選択すると、このウィンドウの属性に関する情報が表示されます。ポリシーのすべての属性の説明については、表 17–5 を参照してください。

注 – ポリシー名は変更できません。ポリシー名を変更するときは、ポリシーを複製し、新しい名前を指定して保存してから、古いポリシーを削除する必要があります。

---

5. 「**Save**」をクリックしてポリシーを保存するか、「**Done**」をクリックします。

例 — ポリシーの変更 (コマンド行)

次の例では、kadmin の modify_policy コマンドを使用して、build11 ポリシーの最小パスワード長を 5 文字に変更します。

```
$ kadmin
kadmin: modify_policy -minlength 5 build11
kadmin: quit
```

▼ ポリシーを削除する方法

対応するコマンド行の例は、この手順のあとに示します。

1. 必要に応じて、**SEAM** ツールを起動します。
   詳細は、283 ページの「SEAM ツールを起動する方法」を参照してください。
2. 「**Policies**」タブをクリックします。

注 – ポリシーを削除する前に、現在使用しているすべての主体からそのポリシーを取り消す必要があります。ポリシーを取り消すには、その主体の「Policy」属性を変更する必要があります。任意の主体が使用しているポリシーは、削除できません。

---

3. 削除するポリシーを一覧から選択して、「**Delete**」をクリックします。
   削除を確定すると、ポリシーが削除されます。

例 — ポリシーの削除 (コマンド行)

次の例では、kadmin の delete_policy コマンドを使用して、build11 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
```

ポリシーを削除する前に、現在使用しているすべての主体からそのポリシーを取り消す必要があります。ポリシーを取り消すには、関係する主体に対して `kadmin` の `modify_principal -policy` コマンドを使用する必要があります。そのポリシーが主体に使用されている場合は、`delete_policy` コマンドは失敗します。

SEAM ツール参照

この節では、SEAM ツールの参照情報について説明します。

SEAM ツールパネルの説明

この節では、SEAM ツールで指定または表示できる主体とポリシーの属性について説明します。属性は、表示されるパネルごとに分類されています。

表 17–2 「Principal Basics」パネルの属性

属性	説明
Principal Name	主体名 (完全指定形式の主体名の *primary*/*instance* 部分)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 主体を変更しようとしても、主体名は編集できない
Password	主体のパスワード。「Generate Random Password」ボタンを使用して、主体のランダムパスワードを作成できる
Policy	主体に使用できるポリシーのメニュー
Account Expires	主体のアカウントが期限切れになる日時。アカウントが期限切れになると、主体はチケット認可チケット (TGT) を取得できず、ログインできなくなる
Last Principal Change	主体の情報が最後に変更された日付 (読み取り専用)
Last Changed By	この主体のアカウントを最後に変更した主体名 (読み取り専用)
Comments	主体に関連するコメント (「一時アカウント」など)

表 17–3 「Principal Details」パネルの属性

属性	説明
Last Success	主体が最後に正常にログインした日時 (読み取り専用)
Last Failure	主体が最後にログインに失敗した日時 (読み取り専用)

表 **17–3** 「Principal Details」パネルの属性　(続き)

属性	説明
Failure Count	主体のログインが失敗した回数 (読み取り専用)
Last Password Change	主体のパスワードが最後に変更された日時 (読み取り専用)
Password Expires	主体の現在のパスワードが期限切れになる日時
Key Version	主体の鍵のバージョン番号。この属性は通常、パスワードが危険にさらされた場合にだけ変更される
Maximum Lifetime (seconds)	チケットを主体が使用できる最大期間 (更新しない場合)
Maximum Renewal (seconds)	既存のチケットを主体が更新できる最大期間

表 **17–4** 「Principal Flags」パネルの属性

属性 (ラジオボタン)	説明
Disable Account	チェックすると、その主体はログインできなくなる。この属性は、主体のアカウントを一時的に凍結するときに使用する
Require Password Change	チェックすると、主体の現在のパスワードが期限切れとなり、ユーザーは `kpasswd` コマンドを使用して新しいパスワードを作成しなければならない。この属性は、セキュリティ違反が発生し、古いパスワードを置換する必要があるときに使用する
Allow Postdated Tickets	チェックすると、主体は遅延チケットを取得できる たとえば、`cron` ジョブを数時間後に実行する場合は、遅延チケットを使用する必要がある。ただし、チケットの有効期限が短い場合は、事前にチケットを取得できない
Allow Forwardable Tickets	チェックすると、主体は転送可能チケットを取得できる 転送可能チケットは、リモートホストに転送されて、シングルサインオンセッションを実現する。たとえば、転送可能チケットを使用して、ユーザー自身の `ftp` 認証または `rsh` 認証が完了すると、NFS サービスなどのほかのサービスを利用するときに、新たにパスワードを要求されない
Allow Renewable Tickets	チェックすると、主体が更新可能チケットを取得できる 主体は、チケットが更新可能な場合、有効期限日時を自動的に延長することができる。つまり、最初のチケットの期限が切れても、新しいチケットを取得する必要がない。現在の NFS サービスは、チケットを新しくするチケットサービスである
Allow Proxiable Tickets	チェックすると、主体は代理可能チケットを取得できる 代理可能チケットを使用すると、クライアントの代わりにサービスがクライアントの操作を実行できる。サービスは、代理可能チケットを使用すると、クライアントの ID を使用して別のサービスのチケットを取得できる。ただし、チケット認可チケットを取得することはできない

表 17–4 「Principal Flags」パネルの属性　(続き)

属性 (ラジオボタン)	説明
Allow Service Tickets	チェックすると、サービスチケットを特定の主体に発行できる サービスチケットの発行は、`kadmin/`*hostname* および `changepw/`*hostname* 主体に許可してはならない。これらの主体は、KDC データベース以外は更新してはならない
Allow TGT-Based Authentication	チェックすると、このサービス主体は別の主体にサービスを提供できる。つまり、KDC は、サービス主体にサービスチケットを発行できる この属性は、サービス主体にだけ使用できる。チェックを解除すると、サービスチケットをサービス主体に対して発行できない
Allow Duplicate Authentication	チェックすると、このユーザー主体はほかのユーザー主体のサービスチケットを取得できる この属性は、ユーザー主体にだけ使用できる。チェックを解除すると、ユーザー主体はサービス主体のサービスチケットを取得できるが、ほかのユーザー主体のサービスチケットは取得できない
Required Preauthentication	チェックすると、KDC が要求されたチケット認可チケット (TGT) を主体に送信する前に、その主体が TGT を要求している主体であることを KDC のソフトウェアが認証する。この事前認証は通常、DES カードなどの特別のパスワードを使用して行われる チェックを解除すると、KDC は要求された TGT を主体に送信する前に、主体の事前認証を必要としない
Required Hardware Authentication	チェックすると、KDC が要求されたチケット認可チケット (TGT) を主体に送信する前に、その主体が TGT を要求している主体であることを KDC のハードウェアが認証する。ハードウェア事前認証は、たとえば Java リングのリーダー上で行われる チェックを解除すると、KDC は要求された TGT を主体に送信する前に、主体の事前認証を必要としない

表 17–5 「Policy Basics」区画の属性

属性	説明
Policy Name	ポリシー名。ポリシーとは、主体のパスワードとチケットを管理する一連のルールのことである ポリシーを変更しようとしても、ポリシー名は編集できない
Minimum Password Length	主体の最小パスワード長

表 **17–5** 「Policy Basics」区画の属性　(続き)

属性	説明
Minimum Password Classes	主体のパスワードに必要な異なる文字タイプの数 たとえば、最小クラス値が 2 の場合は、パスワードに 2 種類以上の文字タイプを使用する必要がある。たとえば、英字と数字を使用して "hi2mom" と入力する必要がある。値が 3 の場合は、パスワードに 3 種類以上の文字タイプを使用する必要がある。たとえば、英字、数字、および句読点を使用して "hi2mom!" と入力する必要がある 値が 1 の場合は、パスワード文字タイプの数に制限が設定されない
Saved Password History	主体に使用された過去のパスワードの数と、過去のパスワードの一覧。これらのパスワードは再使用できない
Minimum Password Lifetime (seconds)	パスワードの最小使用期間。この期間が経過しないとパスワードを変更できない
Maximum Password Lifetime (seconds)	パスワードの最大使用期間。この期間が経過したらパスワードを変更する必要がある
Principals Using This Policy	このポリシーが現在適用されている主体の数 (読み取り専用)

Kerberos 管理権限を制限して SEAM ツールを使用する

`admin` 主体が Kerberos データベースの管理権限をすべて持っている場合は、SEAM 管理ツールの機能をすべて使用できます。ただし、Kerberos 管理権限は、制限することもできます。たとえば、主体の一覧だけを表示できるようにしたり、主体のパスワードを変更できるようにしたりできます。Kerberos 管理権限を制限した場合でも、SEAM ツールを使用できます。ただし、許可された Kerberos 管理権限によって、SEAM ツールの使い方が異なります。表 17–6 は、Kerberos 管理権限に基づいた SEAM ツールの変更の一覧です。

一覧表示権限がない場合、SEAM ツールの使い方が最も大きく変わります。この場合、操作する主体とポリシーの一覧が「List」パネルに表示されません。代わりに、「List」パネルの「Name」フィールドを使用して、操作する主体またはポリシーを指定する必要があります。

SEAM ツールにログインしても、必要な権限がない場合は、次のメッセージが表示されて 「SEAM Administration Login」ウィンドウに戻ります。

```
Insufficient privileges to use gkadmin: ADMCIL. Please try using another principal.
```

主体が Kerberos データベースを管理する権限を変更する方法については、295 ページの「Kerberos 管理権限を変更する方法」 を参照してください。

表 17–6 Kerberos 管理権限を制限して SEAM を使用する

許可しない権限	SEAM ツールの変更
a (追加)	「Principal List」および「Policy List」パネルの「Create New」と「Duplicate」ボタンを使用できない。追加権限がない場合は、新しい主体またはポリシーを作成または複製できない
d (削除)	「Principal List」および「Policy List」パネルの「Delete」ボタンを使用できない。削除権限がない場合は、主体またはポリシーを削除できない
m (変更)	「Principal List」および「Policy List」パネルの「Modify」ボタンを使用できない。変更権限がない場合は、主体またはポリシーを変更できない また、「Modify」ボタンを使用できない場合、パスワードの変更権限を持っていても、主体のパスワードを変更できない
c (パスワードの変更)	「Principal Basics」パネルの「Password」フィールドが読み取り専用になり、変更できない。パスワードの変更権限がない場合、主体のパスワードを変更できない パスワードの変更権限を持っている場合でも、主体のパスワードを変更するときは、さらに変更権限が必要になる
i (データベースの照会)	「Principal List」および「Policy List」パネルの「Modify」と「Duplicate」ボタンを使用できない。照会権限がない場合は、主体またはポリシーを変更または複製できない また、「Modify」ボタンを使用できない場合、パスワードの変更権限を持っていても、主体のパスワードを変更できない
l (一覧)	「List」パネルで主体とポリシーの一覧を表示できない。一覧権限がない場合は、「List」パネルの「Name」フィールドを使用して、操作する主体またはポリシーを指定する必要がある

キータブファイルの管理

サービスを提供するすべてのホストには、「キータブ」 (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 サーバーを使用する場合、このサーバーには nfs サービス主体を含むキータブが必要です。

キータブファイルにサービス鍵を追加するには、kadmin の ktadd コマンドを使用して、ホストのキータブファイルに該当するサービス主体を追加します。サービス主体をキータブファイルに追加するときは、Kerberos データベースにあらかじめ主体を登録し、kadmin が主体の存在を確認できるようにする必要があります。マスター KDC では、キータブファイルのデフォルトの位置は /etc/krb5/kadm5.keytab です。Kerberos サービスを提供するアプリケーションサーバーでは、キータブファイルのデフォルトの位置は /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。ユーザーの場合は、自分のパスワードを保護することが重要ですが、アプリケーションサーバーの場合は、キータブファイルを保護することが重要です。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。

root 主体をホストのキータブファイルに追加することがあります。SEAM クライアント上のユーザーが Kerberos NFS ファイルシステムを自動的にマウントして Kerberos 認証を使用する場合は、クライアントの root 主体をクライアントのキータブファイルに追加する必要があります。追加しなかった場合、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 コマンドをスーパーユーザーとして使用して、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。詳細は、251 ページの「NFS ファイルシステムをマウントするように root 認証を設定する」を参照してください。

注 – マスター KDC を設定するときは、kadmind および changepw 主体を kadm5.keytab ファイルに追加する必要があります。この手順によって、KDC は管理者の Kerberos チケットを復号化して、管理者にデータベースへのアクセス権を与えるかどうかを決定することができます。

キータブファイルを管理するときに、ktutil コマンドも使用できます。ktutil は対話型のコマンドで、Kerberos 管理権限がなくても、ローカルホストのキータブファイルを管理できます。これは、kadmin は Kerberos データベースと対話しますが、ktutil は対話しないためです。つまり、主体をキータブファイルに追加したあとに ktutil を使用すると、キータブファイル内のキー一覧を表示したり、サービスの認証を一時的に無効にしたりできます。

キータブファイルの管理 (作業マップ)

作業	説明	参照先
サービス主体をキータブファイルに追加する	kadmin の ktadd コマンドを使用して、サービス主体をキータブファイルに追加する	311 ページの「サービス主体をキータブファイルに追加する方法」
キータブファイルからサービス主体を削除する	kadmin の ktremove コマンドを使用して、キータブファイルからサービスを削除する	312 ページの「キータブファイルからサービス主体を削除する方法」
キータブファイル内のキー一覧 (主体) を表示する	ktutil コマンドを使用して、キータブファイル内のキー一覧を表示する	313 ページの「キータブファイル内のキー一覧 (主体) を表示する方法」
ホスト上でのサービスの認証を一時的に無効にする	この手順を行うと、kadmin 権限がなくても、ホスト上でのサービスの認証を一時的に無効にできる ktutil を使用してサーバーのキータブファイルからサービス主体を削除する前に、元のキータブファイルを一時的な位置にコピーする必要がある。サービスを再度有効にする場合は、元のキータブファイルを適切な場所に戻す必要がある	314 ページの「ホスト上のサービスの認証を一時的に無効にする方法」

▼ サービス主体をキータブファイルに追加する方法

1. 主体がすでに **Kerberos** データベースに登録されていることを確認します。
   詳細は、286 ページの「主体の一覧を表示する方法」を参照してください。
2. キータブファイルに主体を追加するホスト上でスーパーユーザーになります。
3. **kadmin** コマンドを起動します。

   ```
   # /usr/sbin/kadmin
   ```

4. **ktadd** コマンドを使用して、キータブファイルに主体を追加します。

   ```
   kadmin: ktadd [-k keytab] [-q] [principal | -glob principal-exp]
   ```

-k *keytab*	キータブファイルを指定する。デフォルトでは、/etc/krb5/krb5.keytab が使用される
-q	冗長な情報を表示しない

principal	キータブファイルに追加する主体を指定する。 host、root、nfs、および ftp のサービス主体を追加できる
-glob *principal-exp*	主体表現を指定する。主体表現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る。主体表現の規則は、kadmin の list_principals コマンドと同じである

5. **kadmin** コマンドを終了します。

```
kadmin: quit
```

例 — サービス主体をキータブファイルに追加する

次の例では、kadmin/admin および kadmin/changepw 主体をマスター KDC のキータブファイルに追加します。この例のキータブファイルは、kdc.conf ファイルで指定されている必要があります。

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/admin kadmin/changepw
Entry for principal kadmin/admin@EXAMPLE.COM with kvno 3, encryption type DES-CBC-CRC
  added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw@EXAMPLE.COM with kvno 3, encryption type DES-CBC-CRC
  added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

次の例では、denver の host 主体を denver のキータブファイルに追加し、KDC が denver のネットワークサービスを認証できるようにします。

```
denver # /usr/sbin/kadmin
kadmin: ktadd host/denver@example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver@example.com@EXAMPLE.COM with kvno 2,
  encryption type DES-CBC-CRC added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ キータブファイルからサービス主体を削除する方法

1. キータブファイルから削除するサービス主体が登録されているホスト上でスーパーユーザーになります。
2. **kadmin** コマンドを起動します。

```
# /usr/sbin/kadmin
```

3. **(**省略可能**)** キータブファイル内の現在の主体 **(**鍵**)** の一覧を表示するには、**ktutil** コマンドを使用します。

詳細は、313 ページの「キータブファイル内のキー一覧 (主体) を表示する方法」を参照してください。

4. **ktremove** コマンドを使用して、キータブファイルから主体を削除します。

   kadmin: **ktremove** [**-k** *keytab*] [**-q**] *principal* [*kvno* | **all** | **old**]

-k *keytab*	キータブファイルを指定する。デフォルトでは、/etc/krb5/krb5.keytab が使用される
-q	冗長な情報を表示しない
principal	キータブファイルから削除する主体を指定する
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する
all	指定された主体のすべてのエントリを削除する
old	指定した主体のすべてのエントリを削除する。ただし、鍵のバージョン番号が最上位の主体は削除しない

5. **kadmin** コマンドを終了します。

   kadmin: **quit**

例 — キータブファイルからサービス主体を削除する

次の例では、denver の host 主体を denver のキータブファイルから削除します。

```
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
```

▼ キータブファイル内のキー一覧 (主体) を表示する方法

1. キータブファイルが存在するホスト上でスーパーユーザーになります。

   注 – 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には root 所有権が必要です。

2. **ktutil** コマンドを起動します。

   # **/usr/bin/ktutil**

3. **read_kt** コマンドを使用して、キータブファイルをキー一覧バッファーに読み込みます。

```
ktutil: read_kt keytab
```

4. **list** コマンドを使用して、キー一覧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。

5. **ktutil** コマンドを終了します。

```
ktutil: quit
```

例 — キータブファイル内のキー一覧 (主体) を表示する

次の例では、denver ホストの /etc/krb5/krb5.keytab ファイル内のキー一覧を表示します。

```
denver # /usr/bin/ktutil
    ktutil: read_kt /etc/krb5/krb5.keytab
    ktutil: list
slot KVNO Principal
---- ---- ---------------------------------------
   1    5 host/denver@EXAMPLE.COM
    ktutil: quit
```

▼ ホスト上のサービスの認証を一時的に無効にする方法

ネットワークアプリケーションサーバー上の rlogin や ftp など、サービスの認証メカニズムを一時的に無効にしなければならない場合があります。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。ktutil コマンドを使用してサーバーのキータブファイルからサービス主体を削除することにより、サービスの認証を一時的に無効にすることができます。このとき、kadmin 権限は必要ありません。認証を再度有効にするには、保存した元のキータブファイルを元の位置にコピーするだけです。

注 – デフォルトでは、ほとんどのサービスが認証を要求するように設定されています。そのように設定されていないときは、サービスの認証を無効にした場合でもサービスは動作します。

---

1. キータブファイルが存在するホスト上でスーパーユーザーになります。

注 – 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には root 所有権が必要です。

2. 現在のキータブファイルを一時ファイルに保存します。
3. **ktutil** コマンドを起動します。

   ```
   # /usr/bin/ktutil
   ```

4. **read_kt** コマンドを使用して、キータブファイルをキー一覧バッファーに読み込みます。

   ```
   ktutil: read_kt keytab
   ```

5. **list** コマンドを使用して、キー一覧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。
6. ホストのサービスを一時的に無効にするには、**delete_entry** コマンドを使用して、キー一覧バッファーから目的のサービス主体を削除します。

   ```
   ktutil: delete_entry slot-number
   ```

   この例では、*slot-number* に、削除するサービス主体のスロット番号を指定します。スロット番号は、list コマンドで表示できます。
7. **write_kt** コマンドを使用して、キー一覧バッファーをキータブファイルに書き込みます。

   ```
   ktutil: write_kt keytab
   ```

8. **ktutil** コマンドを終了します。

   ```
   ktutil: quit
   ```

9. サービスを再度有効にする場合は、一時的な (元の) キータブファイルを元の場所にコピーします。

例 — ホスト上のサービスを一時的に無効にする

次の例では、denver ホスト上の host サービスを一時的に無効にします。denver 上のホストサービスを再度有効にするには、krb5.keytab.temp ファイルを /etc/krb5/krb5.keytab ファイルにコピーします。

```
denver # cp /etc/krb5/krb5.keytab /etc/krb5/krb5.keytab.temp
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/krb5.keytab
    ktutil: quit
```

第 18 章

SEAM の使用 (手順)

この章は、SEAM がインストールされているシステムのユーザーを対象としています。この章では、チケットの取得、表示、および破棄について説明します。 また、Kerberos パスワードの選択または変更についても説明します。

この章の内容は次のとおりです。

SEAM の概要については、第 13 章を参照してください。

チケットの管理

この節では、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、214 ページの「SEAM の動作」を参照してください。

チケットを意識する必要があるか

SEAM 1.0 または SEAM 1.0.1 がインストールされている場合、Kerberos は `login` コマンドに組み込まれており、チケットの取得はログイン時に自動的に行われます。

また、ほとんどの Kerberos コマンドは、終了時にチケットを自動的に破棄します。ただし、念のために、コマンドが終了したときに Kerberos チケットを明示的に破棄したい場合は、`kdestroy` を使用します。`kdestroy` の詳細は、320 ページの「チケットを破棄する方法」を参照してください。

チケットの有効期限については、332 ページの「チケットの有効期限」を参照してください。

チケットを作成する方法

通常は、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別な作業をする必要はありません。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。

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 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 コマンドの詳細な構文については、kinit(1) のマニュアルページを参照してください。

例 — チケットを作成する

この例では、ユーザー jennifer が自分のシステムにチケットを作成します。

```
% kinit
Password for jennifer@ENG.EXAMPLE.COM:      <パスワードを入力する>
```

次の例では、ユーザー david 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 david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:      <パスワードを入力する>
```

次の例では、ユーザー david は、 -f を使用して転送可能チケットを作成します。この転送可能チケットを使用すると、ユーザーは、たとえば、別のシステムにログインできます。

```
% kinit -f david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:         <パスワードを入力する>
```

転送チケットをどのように使用するかについては、330 ページの「チケットの種類」を参照してください。

チケットを表示する方法

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。チケットの属性には、「転送可能 (Forwardable)」、「遅延 (Postdated)」などがあります。また、1 つのチケットに「転送可能」と「遅延」の両方が指定されていることもあります。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。

F	転送可能 (Forwardable)
f	転送済み (Forwarded)
P	プロキシ可能 (Proxiable)
p	プロキシ (Proxy)
D	遅延可能 (Postdatable)
d	遅延 (Postdated)
R	更新可能 (Renewable)
I	初期 (Initial)
i	無効 (Invalid)

チケットに指定できる属性については、330 ページの「チケットの種類」を参照してください。

例 — チケットを表示する

次の例は、ユーザー jennifer の初期チケットが転送可能 (F) と遅延 (d) のプロパティを持っていて、まだ検証されていないこと (i) を示します。

```
% /usr/bin/klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: jenniferm@ENG.EXAMPLE.COM

Valid starting                 Expires                  Service principal
09 Mar 99 15:09:51  09 Mar 99 21:09:51  nfs/EXAMPLE.SUN.COM@EXAMPLE.SUN.COM
        renew until 10 Mar 99 15:12:51, Flags: Fdi
```

次の例は、ユーザー david が別のホストから自分のホストに転送済み (f) チケットを 2 つ持っていることを示します。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: david@EXAMPLE.SUN.COM

Valid starting                 Expires                  Service principal
07 Mar 99 06:09:51  09 Mar 99 23:33:51  host/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 99 17:09:51, Flags: fF

Valid starting                 Expires                  Service principal
08 Mar 99 08:09:51  09 Mar 99 12:54:51  nfs/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 99 15:22:51, Flags: fF
```

チケットを破棄する方法

チケットは通常、チケットを作成したコマンドが終了すると自動的に破棄されます。ただし、念のために、コマンドが終了したときに Kerberos チケットを明示的に破棄したい場合があります。チケットは盗まれることもあります。盗まれたチケットが復号化されると、期限切れになるまで使用される可能性があります。

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` 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。

システムを離れるときに侵入者が権限を使用する危険がある場合は、`kdestroy` を使用してチケットを破棄するか、スクリーンセーバーを使って画面をロックする必要があります。

注 – チケットを確実に破棄する 1 つの方法は、ホームディレクトリの `.logout` ファイルに `kdestroy` コマンドを追加することです。

通常はデフォルトで PAM モジュールが構成されているため、チケットはログアウト時に自動的に破棄されます。よって、`.login` ファイルに `kdestroy` への呼び出しを追加する必要はありません。ただし、PAM モジュールが構成されていない場合や、構成されているかどうかわからない場合は、システムを終了するときにチケットを確実に破棄するために、`kdestroy` を `.login` ファイルに追加します。

パスワード管理

SEAM をインストールすると、2 つのパスワードを持つことになります。通常の Solaris パスワードと Kerberos パスワードです。これらのパスワードは同じでも、異なっていてもかまいません。

通常、Kerberos 以外のコマンド (`login` など) は、PAM を使用して Kerberos と UNIX の両方で認証するように設定できます。2 つのパスワードが異なっている場合は、ログインで適切な認証を得るために両方のパスワードを入力する必要があります。2 つのパスワードが同じ場合は、UNIX 用に入力した最初のパスワードが Kerberos で使用されます。

ただし、UNIX と Kerberos に同じパスワードを使用すると、セキュリティを損うおそれがあります。つまり、他人が Kerberos パスワードを入手した場合、UNIX パスワードも安全ではありません。しかし、UNIX と Kerberos に同じパスワードを使用したとしても、Kerberos 環境ではパスワードがネットワークを超えて送信されることはないため、Kerberos 認証のないサイトに比べて安全です。通常、どの方法を選ぶかは、サイトごとの方針に従います。

Kerberos では、Kerberos パスワードだけを使用して、ユーザーの識別を検証します。Kerberos では、パスワードの所有者以外のユーザーに Kerberos パスワードを知られた場合、 セキュリティが保証されなくなります。そのユーザーが所有者になることができるためです。そのユーザーは、パスワードの所有者として電子メールを送信したり、所有者のファイルの読み込み、編集、または削除を行ったり、所有者として別のホストにログインしたりできます。この場合、正しいユーザーを識別することは不可能です。したがって、適切なパスワードを選択し、その秘密を保持することは極めて重要です。 パスワードは、システム管理者を含め誰にも教えてはいけません。また、パスワードは頻繁に変更してください。他人に知られた可能性のある場合は特に変更が必要です。

パスワード選択のヒント

パスワードには、キーボードから入力できるほとんどの文字を使用できます。ただし、`Ctrl` キーと `Return` キーは使用できません。良いパスワードとは、覚えやすく、しかも他人が簡単に推定できないパスワードです。悪いパスワードの例を次に示します。

- 辞書に出てくる言葉
- よく見られるありふれた名前
- 有名な人やキャラクタの名前
- ユーザーの氏名またはユーザー名 (たとえば、名前を逆に綴る、2 度繰り返す)
- 配偶者の名前、子供の名前、ペットの名前
- 自分の誕生日や親戚の誕生日
- 社会保険番号、運転免許書番号、パスポート番号、またはこれに類した身分証明書番号
- このマニュアルやほかのマニュアルに出てくるサンプルパスワード

良いパスワードとは 8 文字以上の長さで、大文字、小文字、数字、句読記号などが混在しているものです。次に例を示します。

- 「I2LMHinSF」などの短縮形。(「I too left my heart in San Francisco」と覚える)
- 「WumpaBun」、「WangDangdoodle!」など、発音しやすい意味のない語句
- 「6o′cluck」、「RrriotGrrrlsRrrule!」など、わざとスペルを間違えた語句

注意 – これらの例は使用しないでください。マニュアルの例に使用されているパスワードは侵入者が最初に試みるパスワードです。

パスワードの変更方法

Kerberos パスワードは次の 2 つの方法で変更できます。

- 通常の UNIX の `passwd` コマンド。SEAM がインストールされていると、Solaris の `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。

  `kpasswd` の代わりに `passwd` を使用する利点は、UNIX と Kerbeors 両方のパスワードを同時に設定できることです。ただし、一般的には `passwd` コマンドで両方のパスワードを同時に変更する必要はありません。UNIX パスワードだけを変更して Kerberos パスワードは変更しなかったり、その逆であっても構いません。

注 – `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` パスワードコマンド。`kpasswd` コマンドは、`passwd` コマンドと似ています。ただし、`kpasswd` コマンドでは、Kerberos パスワード以外は変更できません。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。

  もう 1 つの違いは、`kpasswd` コマンドでは、有効な UNIX ユーザーではない Kerberos 主体のパスワードを変更できる点です。たとえば、`david/admin` は Kerberos 主体ですが、実際の UNIX ユーザーではありません。したがって、この場合は、`passwd` コマンドの代わりに `kpasswd` コマンドを使用する必要があります。

パスワードを変更しても、変更がシステム全体に伝播されるまでには、ある程度の時間が必要です (特に大規模なネットワークでは)。システムの設定方法によりますが、この時間は数分から 1 時間以上になることがあります。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 V5 では、システム管理者が有効なパスワードの基準をユーザーごとに設定できます。この基準は、ユーザーごとの「ポリシー」に定義できます。デフォルトのポリシーを使用することもできます。ポリシーの詳細は、297 ページの「ポリシーの管理」を参照してください。

たとえば、ユーザー `jennifer` の `jenpol` ポリシーでは、パスワードは 8 文字以上で、2 種類以上の文字が混在すると定義されているとします。その場合、パスワードとして「`sloth`」を入力すると、`kpasswd` によって拒否されます。

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:        <jennifer が既存のパスワードを入力する>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
```

```
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:      <jennifer が「sloth」と入力する>
New password (again):      <jennifer が再び「sloth」と入力する>
kpasswd: New password is too short.
Please choose a password which is at least 4 characters long.
```

次に、jennifer はパスワードとして「slothrop49」を入力します。「slothrop49」は長さが 8 文字以上で、2 種類の文字 (数字と小文字) が混在しているため基準に合っています。

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:      <jennifer が既存のパスワードを入力する>
kpasswd: jennifer@ENG.EXAMPLE.COM's password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:      <jennifer が「slothrop49」と入力する>
New password (again):      <jennifer が「slothrop49」と再度入力する>
Kerberos password changed.
```

例 — パスワードを変更する

次の例では、ユーザー david が passwd を使用して、UNIX および Kerberos のパスワードを変更します。

```
% passwd
    passwd:  Changing password for david
    Enter login (NIS+) password:          <現在の UNIX パスワードを入力する>

    New password:                         <新しい UNIX パスワードを入力する>

    Re-enter password:                    <新しい UNIX パスワードを確認する>

    Old KRB5 password:                    <現在の Kerberos パスワードを入力する>

    New KRB5 password:                    <新しい Kerberos パスワードを入力する>

    Re-enter new KRB5 password:           <新しい Kerberos パスワードを確認する>
```

この例では、passwd により、UNIX パスワードと Kerberos パスワードが要求されます。ただし、PAM モジュールで *try_first_pass* が設定されていると、Kerberos パスワードは自動的に UNIX パスワードと同じ内容に設定されます。これはデフォルトの設定です。 この場合、ユーザー david は kpasswd を使用して、次の例で示すように Kerberos パスワードを変更する必要があります。

次の例では、ユーザー david が kpasswd を使用して、Kerberos パスワードだけを変更します。

```
% kpasswd
kpasswd: Changing password for david@ENG.EXAMPLE.COM.
Old password:           <現在の Kerberos パスワードを入力する>

New password:           <新しい Kerberos パスワードを入力する>

New password (again):   <新しい Kerberos パスワードを確認する>

Kerberos password changed.
```

次の例では、ユーザー david が、Kerberos 主体 david/admin (有効な UNIX ユーザーではない) のパスワードを変更します。kpasswd を使用する必要があります。

```
% kpasswd david/admin
kpasswd:  Changing password for david/admin.
Old password:                    <現在の Kerberos パスワードを入力する>

New password:                   <新しい Kerberos パスワードを入力する>

New password (again):       <新しい Kerberos パスワードを再度入力する>

Kerberos password changed.
```

第 19 章

SEAM (参照)

この章では、SEAM 製品に組み込まれている多数のファイル、コマンド、およびデーモンについて一覧表示します。また、Kerberos 認証システムの機能についても詳細に説明します。

この章の内容は次のとおりです。

SEAM ファイル

表 19–1 SEAM ファイル

ファイル名	説明
~/.gkadmin	SEAM 管理ツールで新しい主体を作成するときのデフォルト値
~/.k5login	Kerberos アカウントに対してアクセス権を許可する主体のリスト
/etc/init.d/kdc	krb5kdc を起動または停止する init スクリプト

表 **19–1** SEAM ファイル　*(続き)*

ファイル名	説明
`/etc/init.d/kdc.master`	`kadmind` を起動または停止する `init` スクリプト
`/etc/krb5/kadm5.acl`	Kerberos アクセス制御リストファイル。KDC 管理者の主体名とその Kerberos 管理特権を含む
`/etc/krb5/kadm5.keytab`	マスター KDC 上の `kadmin` サービスのキータブファイル
`/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用キータブファイル
`/etc/krb5/warn.conf`	Kerberos 警告構成ファイル
`/etc/pam.conf`	PAM 構成ファイル
`/tmp/krb5cc_`*uid*	デフォルト資格キャッシュ (*uid* はユーザーの 10 進 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 stash ファイル。KDC マスター鍵の暗号化されたコピーを含む
`/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.db`	Kerberos 主体データベース
`/var/krb5/principal.kadm5`	Kerberos 管理データベース。ポリシー情報を含む
`/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 主体データベースの初期化ファイル。Kerberos データベースの初期化が正常終了すると作成される
`/var/krb5/slave_datatrans`	KDC のバックアップファイルで、`kprop_script` スクリプトが伝播時に使用する

PAM 構成ファイル

デフォルトの PAM 構成ファイルには、認証サービス、アカウント管理、セッション管理、およびパスワード管理の各モジュールに対するエントリが含まれています。

認証モジュールについては、SEAM 1.0 または 1.0.1 がインストールされている場合、rlogin、 login、dtlogin の新しいエントリが作成されます。これらのエントリの例を、以下に示します。これらのサービスはすべて PAM ライブラリ /usr/lib/security/pam_krb5.so.1 を使用して Kerberos 認証を行います。

これらのエントリでは try_first_pass オプションが使用されています。この場合、ユーザーの最初のパスワードを使用して認証が行われます。最初のパスワードを使用すると、複数のメカニズムが記述されていても、ユーザーは別のパスワードを入力する必要がありません。

```
# cat /etc/pam.conf
 .
 .
rlogin auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
login auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
dtlogin auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
other auth optional /usr/lib/security/pam_krb5.so.1 try_first_pass
```

アカウント管理モジュールの場合、Kerberos ライブラリを使用する dtlogin の新しいエントリが次のように作成されます。other エントリはデフォルトの規則を提供するために 1 つ含まれています。現時点では、other エントリによって何の動作も行われません。

```
dtlogin account optional /usr/lib/security/pam_krb5.so.1
other account optional /usr/lib/security/pam_krb5.so.1
```

次に /etc/pam.conf ファイルの最後の 2 つのエントリを示します。セッション管理の other エントリではユーザー資格を破棄します。パスワード管理の新しい other エントリでは Kerberos ライブラリを選択します。

```
other session optional /usr/lib/security/pam_krb5.so.1
other password optional /usr/lib/security/pam_krb5.so.1 try_first_pass
```

SEAM コマンド

この節では、SEAM 製品に含まれているコマンドの一部を示します。

表 **19–2** SEAM コマンド

コマンド	説明
/usr/lib/krb5/kprop	Kerberos データベース伝播プログラム
/usr/sbin/gkadmin	Kerberos データベース管理 GUI プログラム。主体とポリシーの管理に使用される
/usr/sbin/kadmin	リモート Kerberos データベース管理プログラム (Kerberos 認証で実行される)。主体、ポリシー、およびキータブファイルの管理に使用される
/usr/sbin/kadmin.local	ローカル Kerberos データベース管理プログラム (Kerberos 認証がなくても動作するが、マスター KDC 上で実行する必要がある)。 主体、ポリシー、およびキータブファイルの管理に使用される
/usr/sbin/kdb5_util	Kerberos データベースと stash ファイルを作成する

SEAM デーモン

次の表は、SEAM 製品で使用されるデーモンの一覧です。

表 **19–3** SEAM デーモン

デーモン	説明
/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

SEAM の用語

この節では、用語とその定義について説明します。これらの用語は、SEAM のマニュアル全体で使用されます。SEAM 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、この節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 (Key Distribution Center、KDC)」は、資格の発行を担当する SEAM の構成要素です。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。

KDC のマスター鍵を暗号化したコピーは「stash ファイル」に含まれます。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` と `krb5kdc` コマンドが起動されます。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。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。

認証固有の用語

認証処理を理解するには、この節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」は、ユーザーのワークステーションで動作するソフトウェアです。クライアントで動作する SEAM ソフトウェアは、処理中に多数の要求を作成します。このため、SEAM 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」はよく同じ意味で使われます。明確に定義すると、「サーバー」は、SEAM ソフトウェアが動作する物理システムです。「サービス」は、サーバー上でサポートされる特定の機能 (`nfs` など) です。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。このマニュアルでは、サーバーという用語は、物理システムを指し、サービスという用語は、ソフトウェアを指します。

SEAM 製品には 3 種類の鍵があります。1 つは「非公開鍵」です。この鍵は各ユーザー (主体) に与えられ、主体のユーザーと KDC だけに知られています。ユーザー主体に対しては、鍵はユーザーのパスワードに基づいています。サーバーとサービスに対する鍵は「サービス鍵」と呼ばれます。サービス鍵は非公開鍵と同じ目的で使われますが、これはサーバーとサービスによって使用されます。3 つ目の鍵は「セッション鍵」です。セッション鍵は、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のセキュリティを保護するために生成されます。

「チケット」は、ユーザーの識別情報をサーバーやサービスに安全に渡すために使用される情報パケットです。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、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。

「資格」は、対応するセッション鍵とチケットを含む情報パケットです。一般に資格は、資格を暗号化するソフトウェアにより、非公開鍵かサービス鍵を使用して暗号化されます。

「オーセンティケータ （authenticator)」はさらに別の種類の情報です。これをチケットとともに使用すると、ユーザー主体の認証に使用できます。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。

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があります。これらのプロパティは、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はあとから変更できます。たとえば、チケットのプロパティを「転送可能」から「転送済み」に変更できます。チケットのプロパティを表示するには、`klist` コマンドを使用します (318 ページの「チケットを表示する方法」を参照)。

チケットは、次の 1 つまたは複数のプロパティで表されます。

転送可能 / 転送済み
: 転送可能チケットはホストからホストに転送されます。これによって、クライアントは再び認証を受ける必要がありません。たとえば、ユーザー `david` がユーザー `jennifer` のマシンで転送可能チケットを取得した場合、`david` は自分のマシンにログインするときに新しいチケットを取得する必要はありません (再び認証を受けることもありません)。転送可能チケットの例については、318 ページの「例 — チケットを作成する」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を使わ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、チケット認可チケットを使用せずに、クライアントが最近認証されたことを証明します。チケット認可チケットの場合は、認証されてから時間が経っている可能性があります。

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。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。

遅延可能 / 遅延
: 遅延チケットは、作成されても指定された時期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ます。開始時期がきてクライアントが KDC から検証を受けるまでは、無効チケットのままです。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。

プロキシ可能 / プロキシ

場合によっては、サービスがそれ自身のために操作できることが主体にとって必要な場合があります。たとえば、主体がサービスを要求して、サーバーとクライアント以外のホストで印刷ジョブを実行するとします。サービスにはクライアントの ID を使用する権限が必要になりますが、その権限はこの操作だけに制限する必要があります。その場合、サービスは、クライアントの「プロキシ」として動作するといいます。チケットを作成するときには、プロキシの主体名を指定する必要があります。

「プロキシ可能」チケットは「転送可能」チケットに似ていますが、プロキシ可能チケットが 1 つのサービスに対してだけ有効であることに対し、転送可能チケットはサービスに対し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

チケットに非常に長い有効期限を与えるとセキュリティを損なうおそれがあるため、チケットを「更新可能」にすることができます。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です。クライアントがチケットの使用を継続したいときは、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 時間を超えて使用したい場合は、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。

チケットを表示してその属性を見る方法については、318 ページの「チケットを表示する方法」を参照してください。

チケットの有効期限

主体がチケットを取得すると、チケット認可チケットであっても、チケットの有効期限は次の中で最も小さい値に設定されます。

- `kinit` を使用してチケットを取得する場合、`kinit` の `-l` オプションに指定した有効期限値
- `kdc.conf` ファイルで指定される最長有効期限値 (`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*
- 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

図 19–1 は、TGT の有効期限の決定方法と 4 つの有効期限値の指定元を示しています。この図は TGT の有効期限がどのようにして決まるかを示しますが、基本的には、どの主体がチケットを取得する場合でも同じです。違いは、`kinit` で有効期限値を指定しないことと、`krbtgt/`*realm* 主体の代わりに、チケットを提供するサービス主体が最長有効期限値を提供することです。

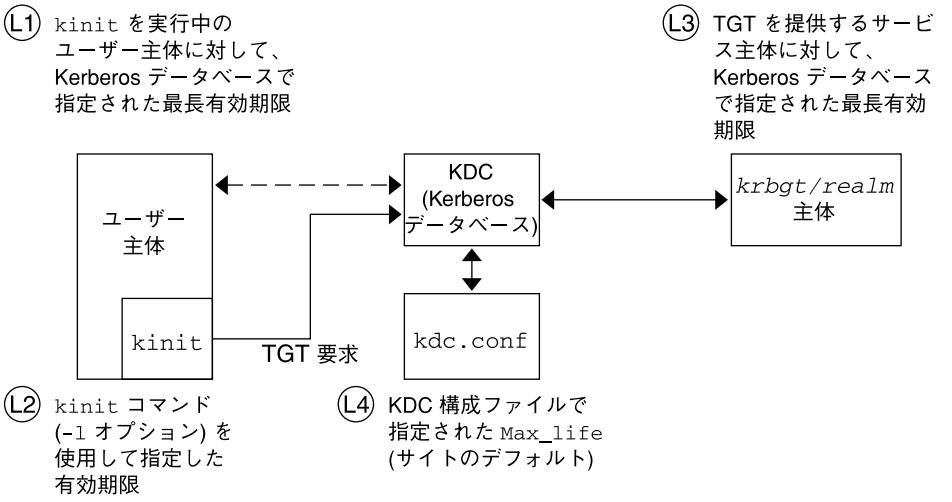


図 **19–1** TGT の有効期限の決定方法

「更新可能」チケットの有効期限も次の 4 つの最小値で決まります。ただし、この場合は更新可能有効期限が使用されます。

- `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` ファイルに指定された更新可能最長有効期限値 (`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*
- 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

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。 次の表に主体名の例を示します。

表 **19–4** 主体名の例

主体名	説明
`root/boston.example.com@EXAMPLE.COM`	NFS クライアントの `root` アカウントに関連付けられた主体。`root` 主体と呼ばれ、認証された NFS マウントを正常終了する場合に必要
`host/boston.example.com@EXAMPLE.COM`	ネットワークアプリケーションサーバー (`ftpd` や `telnetd` など) によって使用される主体。この主体は、`pam_krb5` 認証モジュールでも使用される。この主体は `host` またはサービス主体と呼ばれる
username`@EXAMPLE.COM`	ユーザー用の主体
username`/admin@EXAMPLE.COM`	KDC データベースを管理するために使用できる `admin` 主体
`nfs/boston.example.com@EXAMPLE.COM`	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る
`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る
`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る。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る
`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
`changepw/kdc1.example.com@EXAMPLE.COM`	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
`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する

認証システムの動作方法

アプリケーションを使用してリモートシステムにログインするには、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指定する必要があります。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` というサーバー上の `ftp` 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 `ftp` サービスにアクセスするには、別の資格が必要です。

資格の作成や格納は透過的に行われます。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。

SEAM によるサービスへのアクセス

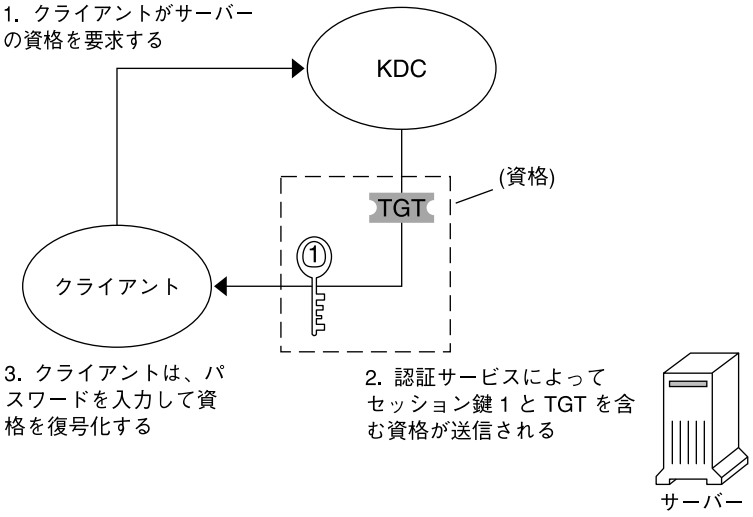
特定のサーバー上の特定のサービスにアクセスする場合、ユーザーは 2 つの資格を取得する必要があります。最初は TGT として知られるチケット認可サービスに対する資格です。チケット認可サービスは、この資格の暗号を解除すると、ユーザーからアクセスを要求されているサーバーの資格をさらに作成します。ユーザーは、この 2 つめの資格を使用してサーバー上のサービスへのアクセスを要求します。サーバーがこの資格の暗号を解除すると、ユーザーはアクセスを許可されます。次の節では、このプロセスを詳細に説明します。

チケット許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。要求にはセキュリティにかかわる情報が含まれていないため、暗号を使う必要はありません。
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。一致する主体が見つかると、認証サービスはその主体の非公開鍵を取得します。次に認証サービスは、クライアントとチケット許可サービスが使用するセッション鍵 (セッション鍵 1) とチケット許可サービス用のチケット (チケット 1) を生成します。このチケットは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、ユーザー主体の非公開鍵を使用して、この情報からセッション鍵 1 とチケット 1 の暗号を解除します。非公開鍵を知っているのはユーザーと KDC データベースだけである必要があるため、パケットの情報は安全に保たれなければなりません。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 KDC データベースから取り出されたパスワードが、入力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。これでクライアントは、チケット許可サービスに対して使用する資格を取得します。次にクライアントはサーバーに対する資格を要求します。



TGT = チケット認可チケット
KDC = 鍵配布センター

図 **19–2** チケット許可サービスに対する資格の取得

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。334 ページの「チケット許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、チケット許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、チケット許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. チケット許可サービスはチケット許可サービスのサービス鍵を知っているため、チケット 1 の暗号を解除できます。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、チケット許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許可サービスによって認証されます。

3. 認証が正常に終了すると、チケット許可サービスは、ユーザー主体とサーバーに対するセッション鍵 (セッション鍵 2) とサーバーに対するチケット (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許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。

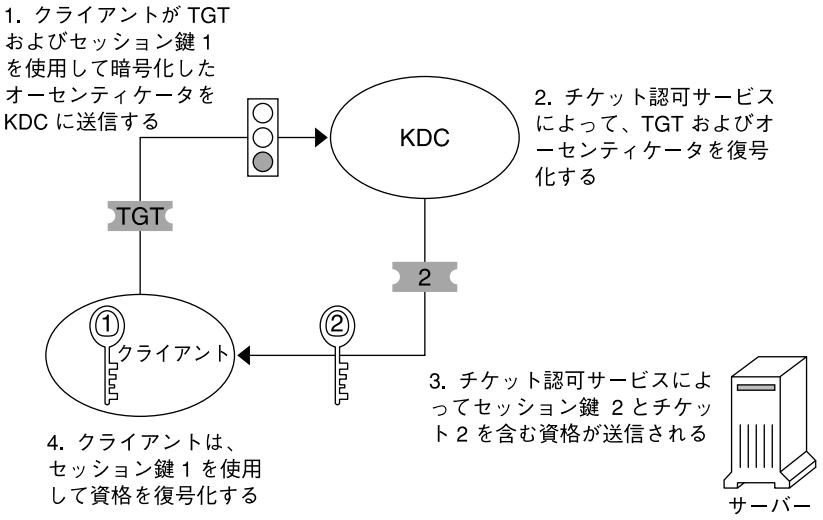


図 **19–3** サーバーに対する資格の取得

特定のサービスへのアクセス権の取得

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許可サービスに対する資格を取得し、チケット許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。334 ページの「チケット許可サービスに対する資格の取得」および 335 ページの「サーバーに対する資格の取得」を参照してください。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許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。

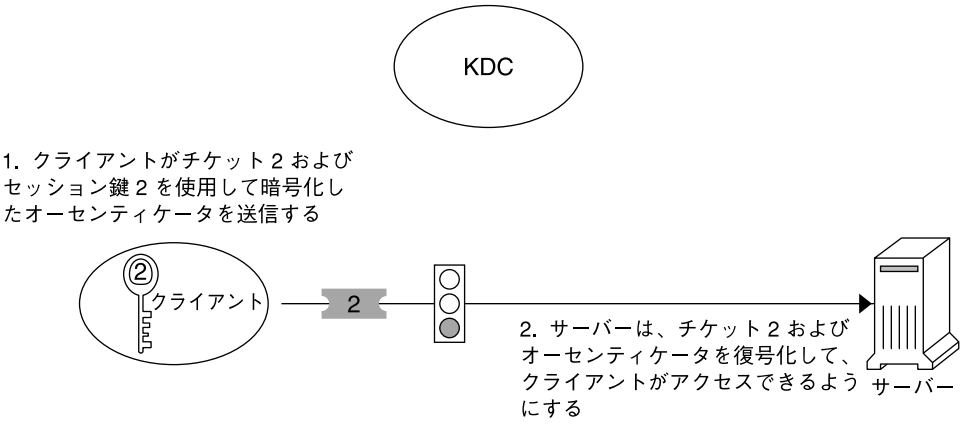


図 **19–4** 特定のサービスへのアクセス権の取得

gsscred テーブルの使用

gsscred テーブルは、NFS サーバーが SEAM ユーザーを識別するときに使用します。NFS サービスは、UNIX ID 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 テーブルは、パスワードファイルから得られる UNIX ID と主体名を割り当てるテーブルです。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。

クライアントの要求が到着すると、NFS サービスは主体名を UNIX ID に割り当てようとします。この割り当てに失敗した場合、gsscred テーブルが参照されます。kerberos_v5 メカニズムでは、root/hostname 主体は自動的に UID 0 に割り当てられるため、gsscred テーブルは参照されません。したがって、gsscred テーブルを使用して root を特別な UID に割り当てることはできません。

パート IV　監査とデバイス管理

第 20 章	基本セキュリティモジュール (Basic Security Module、BSM) の概要について説明します。このモジュールは、監査とデバイス管理によるセキュリティを提供します。
第 21 章	ユーザーのサイトで監査の使用を実装する際に役立つ情報を提供します。
第 22 章	監査を構成および管理する手順について説明します。
第 23 章	監査に関連付けられたファイルに関する情報を提供します。監査トークンの構造について説明します。デバイス管理のコンポーネントについて説明します。

第 20 章

BSM (概要)

基本セキュリティモジュール (Basic Security Module、BSM) には、2 つのセキュリティ機能があります。1 つ目の監査メカニズムには、監査データの分析を支援するツールが含まれています。2 つ目のデバイス割り当てメカニズムでは、リムーバブルデバイスまたは割り当て可能デバイスに必要なオブジェクト再利用特性を提供します。

この章では、BSM の概念について説明します。この章の内容は次のとおりです。

監査とは

監査とは、マシンリソースの使用状況に関するデータを収集することです。監査データは、セキュリティに関連するシステムイベントの記録を提供します。このデータは、ホストで発生する動作の責任の割り当てに使用できます。監査を正常に機能させるには、識別と認証という 2 つのセキュリティ機能が重要です。 ログイン時にユーザーがユーザー名とパスワードを与えると、一意の監査 ID がそのユーザーのプロセスに関連付けられます。監査 ID は、ログインセッション中に起動されるすべてのプロセスに継承されます。ユーザーが ID を変更しても、実行するすべての動作は同じ監査 ID によって追跡されます。`su(1M)` のマニュアルページを参照してください。

監査サブシステムを使うと、次の処理が可能になります。

- ホスト上で発生するセキュリティに関係するイベントの監視
- ネットワーク全体の監査トレールにイベントを記録
- 誤った使用または権限のない動作の検出

- アクセスパターンの確認と、ユーザーおよびオブジェクトのアクセス履歴の調査
- 保護メカニズムを迂回しようとする操作の検出
- スーパーユーザー権限を使用しているユーザーの検出

システムの構成時にどの動作を監視するかを選択します。各ユーザーに行う監査の程度は、細かく調整することもできます。

監査データを収集したあと、監査ファイル縮小ツールと変換ツールによって監査トレールの注目すべき部分を調査できます。たとえば、個々のユーザーまたは特定グループの監査レコードを確認できます。特定の日に発生した特定の種類のイベントに対するすべてのレコードを調査できます。または、特定の時間帯に生成されたレコードを選択することもできます。

監査の機能

監査とは、指定したイベントが発生したときに監査レコードを生成することです。通常、次のイベントが発生すると監査レコードが生成されます。

- システムの起動とシャットダウン
- ログインとログアウト
- プロセスまたはスレッドの作成と破棄
- オブジェクトを開く、閉じる、削除する、または名前を変更する
- スーパーユーザー権限または役割権限の使用
- 識別と認証動作
- プロセスまたはユーザーによる任意アクセス制御 (DAC)
- インストール固有の管理動作

次の 3 つが監査レコードの生成元になります。

- アプリケーション
- 非同期イベントの結果
- プロセスのシステムコールの結果

関連するイベント情報が取得されると、その情報は監査レコードの書式に変換されます。監査レコードは、監査キューと呼ばれるカーネルバッファーに格納されます。カーネルの監査キューに一時的に保管された監査レコードは、監査ファイルに書き込まれます。監査ファイルの場所は、`audit_control` ファイルのエントリによって決定されます。監査ファイルの配置先として、同一マシン上の複数のパーティション、異なる複数のマシン上のパーティション、またはネットワークで接続されている複数のマシン上のパーティションを選択できます。接続された監査ファイルの集合は、監査トレールと呼ばれます。

監査レコードは、発生順に監査ファイルに蓄積されます。各監査レコードには、イベントを識別する情報、イベントの発生元、イベントの時刻、およびその他の関連情報が格納されます。

監査とセキュリティとの関連

コンピュータシステム、特にネットワーク上のシステムのセキュリティを確保するのは簡単ではありません。セキュリティの確保には、システムまたはユーザーのプロセスが開始する前に動作を制御するメカニズムが必要となります。セキュリティの確保には、動作の経過を監視するツールが必要となります。また、セキュリティの確保には、動作終了後に動作内容を報告することも必要です。初期設定の Solaris 監査の場合、ユーザーのログイン前かマシンのプロセスが開始する前に、パラメータが設定されている必要があります。また、ほとんどの監査には、現在のイベントを監視し、指定されたパラメータを満たすイベントを報告する機能が含まれます。Solaris 監査がこれらのイベントを監視および報告する方法については、第 21 章および第 22 章を参照してください。

監査では、ハッカーによる不正な侵入を防止することはできません。ただし、監査サブシステムでは、たとえば、特定のユーザーが特定の日時に特定の動作を行なったことが報告されます。監査報告では、入力パスとユーザー名によってこのユーザーを特定できます。これらの情報は、すぐに端末に表示したり、ファイルに出力してあとで分析したりできます。このように、監査サブシステムの提供するデータから、次のことが判断できます。

- どのようにシステムセキュリティが低下したか
- 必要なセキュリティレベルを実現するために対応が必要なセキュリティホールはどれか

BSM の用語

BSM サービスでは、次の用語が使用されています。定義によっては、より詳細な説明への参照先も示します。

表 20–1 BSM の用語

用語	定義
監査クラス	監査イベントのグループ。監査クラスを使用して、イベントのグループを管理できる。詳細は、346 ページの「監査クラス」を参照
監査ディレクトリ	監査ファイルのリポジトリ。監査ディレクトリの種類については、347 ページの「監査ディレクトリ」を参照
監査イベント	監査対象のセキュリティ関連のシステム動作。監査イベントの種類については、344 ページの「監査イベント」を参照

表 **20–1** BSM の用語　*(続き)*

用語	定義
監査フラグ	クラスの短縮名。監査フラグは、監査を行うイベントクラスとタイミングを決定するために使用する。監査フラグの詳細は、346 ページの「監査フラグ」を参照
監査ポリシー	特定の構成について有効または無効を指定できる監査オプションの集合。監査オプションには、特定の種類の監査データを記録するかどうかを指定する。また、監査トレールがいっぱいになった場合に監査を中断するかどうかも指定する
監査レコード	監査データ。監査データはバイナリ形式で格納される。1 つの監査レコードにつき 1 つの監査イベントが記述される。各監査レコードは、一連の監査トークンから構成される。監査レコードの詳細は、346 ページの「監査レコードと監査トークン」を参照
監査トークン	一連の監査データ。監査トークンはバイナリ形式で格納される。各監査トークンには、監査イベントの 1 つの属性 (プロセス、パス、その他のオブジェクトなど) が記述される。監査トークンの詳細は、398 ページの「監査トークンの形式」を参照
監査トレール	1 つまたは複数の監査ファイルの集合。複数の監査ファイルパーティションに別々に存在することもある
デバイス割り当て	デバイスの使用を制限し、デバイスが使い終わったあとに残っているデータを消去するメカニズム。デバイス割り当ての詳細は、348 ページの「デバイス割り当て」を参照
公開オブジェクト	公開オブジェクトとは、`/etc` や `/usr/bin` などのシステムディレクトリ内に格納されているファイル。システムディレクトリは、すべてのユーザーによって読み取り可能である。システムディレクトリ内のファイルは、内容の参照、または実行のために頻繁に読み取られる。Solaris 9 8/03 リリースでは、デフォルトで、公開オブジェクトの読み取り専用イベントは監査されない。たとえば、`fr` 監査フラグがオンになっている場合でも、公開オブジェクトの読み取り動作は監査されない。`auditconfig -setpolicy` コマンドの `public` ポリシーフラグ引数によって、ポリシーの設定が決まる

監査イベント

セキュリティに関連するシステム動作について監査を行うことができます。これらの監査可能な動作を、「監査イベント」と呼びます。監査イベントは、`/etc/security/audit_event` ファイルに指定します。監査可能な各イベントは、このファイル内で、シンボル名、イベント番号、一連の選択済みクラス、および簡単な説明によって定義されています。`audit_event(4)` のマニュアルページを参照してください。

監査イベントには、いくつかのカテゴリがあります。まず、カーネルレベルのイベントとユーザーレベルのイベントに大きく分けられます。カーネルによって生成されたイベントは、カーネルレベルのイベントと呼ばれます。アプリケーションによって生成されたイベントは、ユーザーレベルのイベントと呼ばれます。次の表に示すように、カーネルレベルのイベントには、ユーザーレベルのイベントよりも小さい監査イベント番号が割り当てられています。

表 **20–2** 監査イベントのカテゴリ

番号の範囲	イベントの種類
1–2047	カーネルレベルの監査イベント
2048–65535	ユーザーレベルの監査イベント
2048–32767	SunOS ユーザーレベルのプログラム用に予約
32768–65535	サードパーティのアプリケーション用

カーネルレベルの監査イベント

カーネルによって生成されるイベントは、システムコールです。システムコールには、1 から 2047 までの監査イベント番号が割り当てられます。カーネルレベルのイベントでは、イベント名は `AUE_` で始まり、そのあとにイベントを表す大文字のニーモニックが続きます。たとえば、`creat()` システムコールのイベント番号は `4` で、イベント名は `AUE_CREAT` です。

ユーザーレベルの監査イベント

アプリケーションソフトウェアによって生成されるイベントは、カーネルの外側にあります。アプリケーションソフトウェアによってユーザーレベルのイベントが生成されます。ユーザーレベルのイベントは、2048 から 65535 までの番号です。イベント名は `AUE_` で始まり、そのあとにイベントを表す小文字のニーモニックが続きます。たとえば、`rlogin` コマンドのイベント番号は `6155` で、イベント名は `AUE_rlogin` です。表 20–2 にユーザーに関連するイベントの一般的なカテゴリを示します。

ユーザーに起因しない監査イベント

ほとんどのイベントは個々のユーザーの動作が原因で発生しますが、そうでないイベントも存在します。カーネル割り込みレベルでイベントが発生した場合や、ユーザーが識別および認証される前にイベントが発生した場合、それらのイベントは「ユーザーに起因しない」イベントです。ユーザーに起因しないイベントは、監査可能です。次の例は、`/etc/security/audit_event` ファイル内のユーザーに起因しないイベントを 2 つ示しています。

```
153:AUE_ENTERPROM:enter prom:na
6156:AUE_mountd_mount:mount:na
```

AUE_ENTERPROM はカーネルレベルの na イベントです。AUE_mountd_mount はユーザーレベルの na イベントです。

監査クラス

各監査イベントは、1 つまたは複数の「監査クラス」に属するものとして定義します。監査クラスは、多数の監査イベントが入った便利な入れ物です。あるクラスを監査対象として選択した場合、そのクラスに属するすべてのイベントを監査対象として選択したことになります。監査クラスは、/etc/security/audit_class ファイルに定義されます。各エントリには、クラスの名前、監査マスク、および短縮名が含まれます。

```
0x00000010:fc:file create
0x00000400:na:non-attribute
```

監査イベントのクラスへの割り当ては構成可能です。これらの構成の変更は audit_event ファイル内で行います。

監査可能な特定のイベントが監査トレール内に記録されるのは、その特定のイベントを含む監査クラスを監査対象として事前に選択した場合です。監査クラスは、32 クラスまで設定できます。クラスには、all および no という 2 つのグローバルクラスが含まれます。 監査クラスについては、表 23–1 を参照してください。audit_class(4) のマニュアルページも参照してください。

監査フラグ

「監査フラグ」は監査クラスの短縮名です。マシン全体で有効な監査関連のデフォルト値は、audit_control ファイル内の監査フラグによって指定します。audit_control ファイルについては、386 ページの「audit_controlファイル 」を参照してください。

マシン全体の監査デフォルトに対する例外を、個々のユーザーごとに定義できます。audit_user ファイル内で、ユーザーのエントリに監査フラグを設定します。監査フラグは、auditconfig コマンドの引数としても使用します。auditconfig(1M) と audit_user(4) のマニュアルページを参照してください。

監査レコードと監査トークン

各「監査レコード」には、監査された 1 つのイベントの発生が記述されます。レコードには、動作を行なったユーザー、影響を受けたファイル、試みられた動作、その動作が発生した位置と時刻などの情報が含まれます。次の行は、praudit コマンドによって処理された監査レコードの一例です。

```
header,81,2,login - local,,Mon May  6 16:55:48 PDT 2002, + 486 msec
subject,root,root,other,root,other,378,378,0 0 example_machine
text,successful login
```

```
return,success,0
```

監査レコードは監査ファイル内に収集されます。1 つのマシンやサイト内に存在する監査ファイルの集合は「監査トレール」と呼ばれます。監査ファイルの処理方法については、audit.log(4) のマニュアルページを参照してください。監査レコードをユーザーが読める書式に変換するには、praudit コマンドを使用します。praudit の出力例については、383 ページの「praudit コマンド」を参照してください。praudit(1M) のマニュアルページも参照してください。

監査イベントごとに保存される情報の種類は、一連の「監査トークン」によって定義されます。 イベントの監査レコードが生成されるたびに、そのイベントに対して定義されたトークンの一部またはすべてが、そのレコードに書き込まれます。どのトークンが記録されるかは、イベントの性質によって決まります。監査レコードの説明を生成するには、bsmrecord コマンドを使用します。次の出力は、creat() システムコール使用時に生成される監査レコードの構造を示しています。header で始まる各行が監査トークンです。

```
creat
  system call creat                 see creat(2)
  event ID    4                     AUE_CREAT
  class       fc                    (0x00000010)
      header
      path
      [attribute]
      subject
      return
```

詳細は、370 ページの「監査レコードの書式の表示方法」を参照してください。各監査トークンの構造については、398 ページの「監査トークンの形式」を参照してください。audit.log(4) のマニュアルページにも、監査トークンの情報が記載されています。

監査ディレクトリ

監査ディレクトリには、監査ファイルの集合を保管します。通常のインストールでは、多くの監査ディレクトリが使用されます。監査ディレクトリには、次の 3 つのタイプがあります。

- 「一次監査ディレクトリ」 – 通常の条件下で、システム監査ファイルが配置されるディレクトリ
- 「二次監査ディレクトリ」 – 一次ディレクトリがいっぱいか使用できない場合に、システム監査ファイルが配置されるディレクトリ
- 「最後のディレクトリ」 – 一次監査ディレクトリと二次監査ディレクトリが使用できない場合に使用されるローカル監査ディレクトリ

デバイス割り当て

デバイス割り当てメカニズムを使用すれば、CD-ROM などのデバイスに対するアクセスを制限できます。デバイスクリーンスクリプトは、デバイス使用後にデバイスに残されたデータをすべて削除します。これらの動作により、デバイスのセキュリティが向上します。詳細は、375 ページの「デバイス割り当ての管理 (作業)」または 416 ページの「デバイス割り当て参照」を参照してください。

第 21 章

監査の計画

この章では、インストールした Solaris に対して監査を設定する方法について説明します。特に、監査サービスを有効にする前に、考慮する必要のある問題について説明します。この章の内容は次のとおりです。

監査の概要については、第 20 章を参照してください。サイトで監査を構成する手順については、第 22 章を参照してください。

監査トレールの処理

監査トレールで使用するファイル領域は、監査にとって大きな問題の 1 つです。各ホストには、監査ファイル用に構成されたいくつかの監査ディレクトリが必要です。ホスト上で監査を有効にする前に、最初に監査ディレクトリの構成方法を決定する必要があります。次の表は、監査トレール記憶領域を計画するときに、解決する必要のある問題の一覧です。

監査トレール記憶領域に関する問題	計画内容
1. サイトで必要な監査の程度を決定する	サイトのセキュリティ上の必要性を考慮して、監査トレールに使用できるディスク容量を決定する。 サイトのセキュリティを確保しながら領域要件を削減する方法と、監査記憶領域を設定する方法については、354 ページの「監査コストの制御」および 355 ページの「効率的な監査」を参照
2. 監査対象のシステムを決定する。監査ファイルを格納するシステムを決定する	ネットワーク上で監査するホストを決定する。監査するホストごとに、1 つ以上のローカル監査ディレクトリを作成する。次に、監査トレールの大部分を格納するホストを決定する
3. 監査ディレクトリの名前と位置を決定する	使用するすべての監査ディレクトリの一覧を作成する
4. ホストと監査ディレクトリを対応付ける	ホストと監査ディレクトリの割り当てを作成する。この手順を行なって、監査作業の負荷を分散する

監査担当者と監査対象の決定

監査する動作の種類を上手に選択し、有用な監査情報を収集することが望まれます。監査ファイルはすぐに大きくなり、空き領域がなくなる可能性があるため、監査対象を計画する必要があります。

監査対象に関する問題	計画内容
1. サイトに必要な監査クラスを決定する	監査クラスの追加やデフォルトクラスの変更は、監査サービスを開始する前に行うことを推奨する。 監査クラスについては、391 ページの「監査クラスと監査フラグ」を参照
2. イベントからクラスへの割り当てを決定する	多くの場合、デフォルトの割り当てをそのまま使用できる。ただし、新しいクラスを追加したり、クラス定義を変更したりした場合は、イベントを別のクラスに移動しなければならないこともある
3. すべてのマシン上のすべてのユーザーについて、監査するクラスを決定する	`audit_control` ファイルにあるシステム全体の監査フラグは、すべてのユーザーとプロセスに適用される。監査フラグは、監査クラスの監査が正常終了したとき、失敗したとき、またはその両方の場合に決定される。Solaris がインストールされたマシン上のすべての `audit_control` ファイルには、同じ監査フラグが設定されている必要がある

監査対象に関する問題	計画内容
4. インストール全体の監査設定のユーザー例外を決定する	一部のユーザーの監査をシステム全体の設定と異なる設定にするときは、各マシン上の audit_user ファイルでそのユーザーのエントリを変更する。 詳細は、360 ページの「ユーザーの監査特性の変更方法」を参照
5. 最小空きディスク容量 (minfree) を決定する。警告が送信されるまで、監査ファイルシステムのディスク容量を使用できる	監査ファイルシステム上のディスク容量が minfree の割合を下回ると、監査デーモンは次に利用できる監査ディレクトリに切り替える。デーモンは、ソフト制限値を超えたことを示す警告を送信する。 詳細は、360 ページの「例 — 警告に対するソフト制限値を変更する」を参照
6. サイトに必要な監査ポリシーを決定する	ポリシー変数は動的なカーネル変数であるため、その値はシステムの終了時には保存されない。このため、適切な起動スクリプトを使用して、適切なポリシーを設定する必要がある。 詳細は、368 ページの「監査ポリシーを有効または無効にする方法」を参照
7. audit_warn メールの別名の管理方法を決定する	auditd デーモンは、監査ディレクトリを切り替えるたびに audit_warn スクリプトを実行する。また、ディスク容量の不足などの問題が発生した場合にも、このスクリプトを実行する。デフォルトでは、audit_warn スクリプトは、audit_warn の別名にメールを送信し、コンソールにメッセージを送信する。ほかの別名にメッセージを送信するには、別名を変更するか、スクリプトを変更する必要がある。 詳細は、367 ページの「audit_warn 別名の構成方法」を参照
8. すべての監査ディレクトリがいっぱいになったときの動作を決定する	監査トレールのオーバーフローが発生しても、システムを引き続き動作させるには、cnt ポリシーを有効にする。 cnt ポリシーの例については、368 ページの「例 — cnt ポリシーを設定する」を参照 また、監査を無効にした状態で、ログインおよび作業可能なアカウントを作成することもできる。 監査アカウントの例については、361 ページの「例 — 監査管理ログインを作成する」を参照

使用する監査ポリシーの決定

監査ポリシーを使用して、ローカルホストの監査レコードの特性を決定します。監査ポリシーは、起動スクリプトによって設定されます。監査サービスを有効にする bsmconv スクリプトによって、/etc/security/audit_startup スクリプトが作成されます。audit_startup スクリプトは、auditconfig コマンドを実行することで監査ポリシーを設定します。詳細は、audit_startup(1M) のマニュアルページを参照してください。

監査ポリシーがデフォルトで無効になっているのは、記憶領域要件とシステム処理要求を最小限に抑えるためです。監査ポリシーを動的に有効または無効にするには、auditconfig コマンドを使用します。監査ポリシーを静的に有効または無効にするには、audit_startup スクリプトを使用します。次の表を参照して、1 つまたは複数の監査ポリシーを有効にしたときに発生する追加のオーバーヘッドを考慮しながら、サイトの要件を決定してください。

表 **21–1** 監査ポリシーの働き

ポリシー名	説明	ポリシーを変更する理由
arge	無効にすると、実行済みプログラムスクリプトの環境変数が exec 監査レコードから除外される 有効にすると、実行済みプログラムスクリプトの環境変数が exec 監査レコードに追加される。監査レコードには、より詳細な情報が記録される	無効にすると、収集される情報が大幅に少なくなる このオプションは、少数のユーザーを監査するときに有効にする。このオプションは、exec プログラムで使用される環境変数に問題があるときにも有用
argv	無効にすると、実行済みプログラムスクリプトの引数が exec 監査レコードから除外される 有効にすると、実行済みプログラムスクリプトの引数が exec 監査レコードに追加される。監査レコードには、より詳細な情報が記録される	無効にすると、収集される情報が大幅に少なくなる このオプションは、少数のユーザーを監査するときに有効にする。このオプションは、exec プログラムが正常に動作しないことがはっきりしているときにも有用
cnt	無効にすると、ユーザーまたはアプリケーションの実行がブロックされる。このブロックが発生するのは、空きディスク容量の不足により監査トレールに監査レコードが追加できない場合である 有効にすると、監査レコードが生成されないまま、イベントを完了できる。生成されなかった監査レコードのカウントは行われる	セキュリティを最優先する場合は、無効にする セキュリティよりシステムの可用性が重要な場合は、有効にする

表 21–1 監査ポリシーの働き　(続き)

ポリシー名	説明	ポリシーを変更する理由
group	無効にすると、グループの一覧が監査レコードに追加されない	サイトのセキュリティが重要な場合、通常は無効にする
	有効にすると、グループの一覧が特別なトークンとしてすべての監査レコードに追加される	どのグループが監査可能なイベントを生成しているかを監査する必要があるときは、有効にする
path	無効にすると、1 つのシステムコールで使用されたパスが、あっても 1 つだけ監査レコードに記録される	無効にすると、監査レコードにパスが、あっても 1 つだけ記録される
	有効にすると、監査イベントで使用されたすべてのパスが、すべての監査レコードに記録される	有効にすると、1 つのシステムコールで使用された各ファイル名またはパスが、監査レコードに path トークンとして記録される
public	Solaris 9 8/03 リリースで追加。無効にすると、ファイルの読み取りが事前に選択されている場合に、公開オブジェクトの読み取り専用イベントが監査トレールに追加されなくなる。読み取り専用イベントを含む監査フラグとしては、fr、fa、および cl がある	サイトのセキュリティが重要な場合、通常は無効にする このオプションを有効にするのはまれである
	有効にすると、適切な監査フラグが事前に選択されている場合、公開オブジェクトの読み取り専用監査イベントのすべてが記録される	
seq	無効にすると、すべての監査レコードに順序番号が追加されない	監査が問題なく動作しているときは、無効にしてもかまわない
	有効にすると、すべての監査レコードに順序番号が追加される。順序番号は seq トークンに格納される	監査ファイルが正しく書き込まれているかどうかを確認するときは、有効にする。ファイルが壊れた場合、不正なレコードをすばやく検出できる可能性が高くなる。順序番号が順不同であったり、一部の番号が抜けていたりする場合がある。監査レコードの情報の一部が抜け落ちている場合、そのファイルは壊れている

表 **21–1** 監査ポリシーの働き　*(続き)*

ポリシー名	説明	ポリシーを変更する理由
`trail`	無効にすると、`trailer` トークンが監査レコードに追加されない	無効にすると、作成される監査レコードが小さくなる
	有効にすると、`trailer` トークンがすべての監査レコードに追加される	有効にすると、各監査レコードの最後に `trailer` トークンが常に付加される。`trailer` トークンは、多くの場合、デバッグ時に順序トークンとともに使用される。ファイルが壊れた場合、`auditreduce` コマンドを使用すると、正しいレコードに対する再同期速度が向上する。監査レコードの情報の一部が抜け落ちている場合、そのファイルは壊れている

監査コストの制御

監査処理によってシステム資源が消費されるため、どの程度詳しく記録するかを制御する必要があります。監査の対象を決めるときには、監査に伴う次の 3 つのコストを考慮してください。

- 処理時間の増大に伴うコスト
- 監査データの分析に伴うコスト
- 監査データの格納に伴うコスト

監査データの処理時間の増大に伴うコスト

処理時間の増大に伴うコストは、監査に関連する 3 つのコストの中ではもっとも重要性の低い問題です。第 1 に、通常は、イメージ処理や複雑な計算処理などの計算集中型のタスクの実行中には、監査処理が発生しないからです。その他の理由として、単一ユーザーシステムのコストが通常は無視できるほど小さいことが挙げられます。

監査データの分析に伴うコスト

分析に伴うコストは、収集される監査データの量にほぼ比例します。分析コストには、監査レコードをマージして検討するための時間が含まれます。コストにはまた、監査レコードをアーカイブし、それを安全な場所に保管するための時間も含まれます。

生成されるレコードの数が少ないほど、監査トレールの分析にかかる時間も短くなります。以下の項、355 ページの「監査データの格納に伴うコスト」と 355 ページの「効率的な監査」では、効率的に監査を行う方法について説明します。収集するデータの量を削減しながら、サイトのセキュリティ目標を達成することは可能です。

監査データの格納に伴うコスト

格納に伴うコストは、監査コストのうちでもっとも重要です。監査データの量は次の要素によって左右されます。

- ユーザー数
- マシン数
- 使用量
- 必要なセキュリティの程度

これらの要因はサイトごとに異なるため、監査データの格納用に前もって確保しておくディスク容量を決定できるような計算式はありません。

`all` フラグを指定して完全な監査を行うと、ディスクがすぐにいっぱいになります。プログラムのコンパイルといった単純なタスクによってさえ、巨大な監査ファイルが生成される可能性があります。中程度のサイズのプログラムでも、1 分も経たないうちに数千件の監査レコードが生成される可能性があります。たとえば、5 ファイルで合計 5000 行のプログラムの監査トレールが、何 M バイトものディスク容量を占有する可能性があります。事前選択機能を使用して、生成されるレコードの量を減らしておいたほうが賢明です。たとえば、`fr` クラスを省略すると、監査ボリュームを 3 分の 2 以上も削減できます。また、監査ファイルを効率的に管理することも重要です。監査レコードが生成されたあとにファイル管理を行うことによって、必要なディスク容量を減らせます。

監査を構成する前に、監査フラグを理解しておく必要があります。それらのフラグが監査対象とするイベントの種類を理解しておく必要もあります。適切な基準に基づいて、サイトの監査に関する基本ポリシーを設定します。そのような基準には、サイトで必要なセキュリティの程度や、管理対象ユーザーの種類などが含まれます。

効率的な監査

この節で説明する方法により、組織のセキュリティ目標を達成する一方で、監査効率を高めることができます。

- ある一定の割合のユーザーのみを任意の時間にランダムに監査する
- 監査ファイルのディスク容量要件を削減するために、監査ファイルを結合、縮小、および圧縮する。ファイルを保管する手順、リムーバブルメディアにファイルを転送する手順、およびファイルをオフラインで格納する手順を決定する

- 監査データの異常な動作をリアルタイムで監視する。特定の動作で生成された監査トレールを監視する手順を設定する。異常なイベントが検出された場合に、それに応じて特定のユーザーまたは特定のマシンの監査レベルを自動的に上げるようなスクリプトを作成する。

  たとえば、(1) すべての監査ファイルサーバー上における監査ファイルの作成を監視し、(2) その監査ファイルを `tail` コマンドを使用して処理するスクリプトを作成する。`tail(1)` のマニュアルページを参照。`tail -0f` の出力を `praudit` コマンドにパイプする。このコマンドは、監査レコードが生成されると監査レコードストリームを生成する。このストリームを分析して、異常なメッセージの種類などを調べ、監査担当者に配布する。また、このスクリプトを使用して、自動応答をトリガーすることもできる。

  さらに、このスクリプトには、(3) 監査ディレクトリを常時監視して、新しい `not_terminated` 監査ファイルが発生していないかを調べるコードを含める必要もある。また、このスクリプトは、(4) 監査ファイルに書き込めなくなったときに、未処理の `tail` プロセスを終了させる必要もある

第 22 章

BSM サービスの管理 (手順)

この章では、監査を組み込んだ Solaris 環境を設定および管理する手順について説明します。また、監査トレールおよびデバイス割り当ての管理方法についても説明します。この章では、次の作業マップについて説明します。

監査の概要については、第 20 章を参照してください。計画時のヒントについては、第 21 章を参照してください。

BSM サービスの管理 (作業マップ)

次の作業マップは、BSM サービスの管理に必要な主な作業の一覧です。

作業	説明	参照先
監査プロセスの計画	監査を構成する前に構成上の問題について考慮して判断する	第 21 章
監査ファイルの構成	監査を必要とするイベント、クラス、およびユーザーを定義する	358 ページの「監査ファイルの構成 (作業マップ)」
監査プロセスの構成	監査プロセスを利用できるように各ホストを構成する	364 ページの「監査サービスの構成 (作業マップ)」

作業	説明	参照先
監査レコードの管理	監査データを組み合わせて分析する	370 ページの「監査レコードの管理 (作業マップ)」
デバイス割り当ての管理	デバイス割り当てメカニズムを通して、アクセスするデバイスを定義する	375 ページの「デバイス割り当ての管理 (作業)」

監査ファイルの構成 (作業マップ)

ネットワーク上で監査プロセスを有効にする前に、監査構成ファイルを編集します。このあとに説明する手順の多くは、サービスの再開またはローカルシステムのリブートが必要です。監査構成ファイルの編集は、できるだけ BSM サービスを開始する前に完了してください。

次の表は、この節で説明する操作の一覧です。

作業	説明	参照先
監査フラグの選択、`audit_control` 設定の変更	監査対象イベントを事前に選択する。`audit_control` ファイルに設定された値を変更する	358 ページの「監査フラグの選択方法」
ユーザーの監査特性の変更	システム全体の監査フラグ設定に対してユーザー固有の例外を設定する	360 ページの「ユーザーの監査特性の変更方法」
監査クラスの追加	新しい監査クラスを定義する	361 ページの「監査クラスの追加方法」
イベントからクラスへの割り当ての変更	特定のイベントが属するクラスを変更する	362 ページの「監査イベントの所属先クラスの変更方法」
監査イベントの追加	新しいユーザーレベルのイベントを`audit_event` ファイルに追加する	363 ページの「監査イベントの追加方法」

▼ 監査フラグの選択方法

監査フラグは、`/etc/security/audit_control` ファイルに定義されます。監査フラグを使用して、監査ログに記録する監査レコードのクラスを選択します。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **(**省略可能**)** **`audit_control`** ファイルのバックアップコピーを保存します。

```
# cp /etc/security/audit_control /etc/security/audit_control.save
```

3. **audit_control** ファイルに新しいエントリを追加します。
   各エントリの書式は次のとおりです。

   title:*string*

   title　行の種類を定義する。dir:、flags:、minfree:、または naflags: を選択できる

   string　この種類の行に関連付けるデータを指定する

4. 監査デーモンを実行して、新しい **audit_control** ファイルを読み込みます。
   監査デーモンの内部に、読み込んだ情報が格納されます。新しい情報を使用するには、システムをリブートするか、次のコマンドを入力します。

   ```
   # audit -s
   ```

例 — 監査トレールファイルの位置を変更する

dir: で始まる行には、監査トレールファイルの格納に使用する監査ファイルシステムを定義します。この例では、監査トレールファイルの位置を 2 つ追加定義しています。

```
# cat /etc/security/audit_control
dir:/etc/security/audit/host.1/files
dir:/etc/security/audit/host.2/files
dir:/var/audit
flags:
minfree:10
naflags:lo
```

例 — すべてのユーザーに適用される監査フラグを変更する

audit_control ファイルの flags 行には、監査するイベントのクラスを定義します。このフラグは、ホスト上のすべてのユーザーに適用されます。クラスは空白を入れずにコンマで区切ります。この例では、すべてのユーザーを対象に lo クラスのイベントが監査されます。

```
# cat /etc/security/audit_control
dir:/var/audit
flags:lo
minfree:10
naflags:lo
```

例 — 警告に対するソフト制限値を変更する

audit_control ファイルの minfree 行には、すべての監査ファイルシステムの最小空き領域レベルを定義します。この例では、利用できるファイルシステムの領域が 10 % だけになったときに警告が発行されるように、ソフト制限値を設定しています。

```
# cat /etc/security/audit_control
dir:/var/audit
flags:
minfree:10
naflags:lo
```

例 — ユーザーに起因しないイベントの監査を変更する

audit_control ファイルの naflags: 行には、ホスト上のすべてのユーザーを対象に監査する、ユーザーに起因しないイベントのクラスを定義します。クラスは空白を入れずにコンマで区切ります。この例では、na イベントクラスが追加されます。

```
# cat /etc/security/audit_control
dir:/var/audit
flags:
minfree:10
naflags:lo,na
```

▼ ユーザーの監査特性の変更方法

ユーザーごとの定義は、/etc/security/audit_user ファイルに格納されます。これらの定義は、audit_control ファイル内のフラグに対する例外です。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **(**省略可能**)** **audit_user** ファイルのバックアップコピーを保存します。

   ```
   # cp /etc/security/audit_user /etc/security/audit_user.save
   ```

3. **audit_user** ファイルに新しいエントリを追加します。

   各エントリの書式は次のとおりです。

   username:*always*:*never*

   username 監査するユーザー名を選択する

   always 常に監査する監査クラスの一覧を選択する

   never 監査しない監査クラスの一覧を選択する

   複数のフラグを指定するには、監査クラスをコンマで区切ります。監査ファイルの詳細は、391 ページの「監査クラスと監査フラグ」を参照してください。
4. 監査デーモンで新しいデータが使用できるようにします。

新しいデータを使用するには、システムをリブートします。該当のユーザーをいったんログアウトさせてからログインし直させることもできます。

例 — 1 人のユーザーの監査を変更する

この例のエントリでは、ユーザー sue がログインクラス (lo) の任意のプログラムにアクセスすると、監査レコードが生成されます。

```
# grep sue /etc/security/audit_user
sue:lo:
```

例 — 監査管理ログインを作成する

ログインを監査対象としている場合にすべての監査パーティションがいっぱいになると、ユーザーがホストにログインできなくなる可能性があります。この状況を回避するために、監査を行わない特別なアカウントを設定できます。特別なアカウントは、監査パーティションがいっぱいになった場合でもホストにログインできるため、このようなパーティションの問題を解決することができます。この例では、アカウント auditadm を監査しないように定義します。

```
# grep auditadm /etc/security/audit_user
auditadmin:no:yes
```

注 – 監査管理アカウントの使用を許されたユーザーについては、別の方法で監視する必要があります。

▼ 監査クラスの追加方法

監査クラスは、/etc/security/audit_class ファイルに定義されます。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. (省略可能) **audit_class** ファイルのバックアップコピーを保存します。

   ```
   # cp /etc/security/audit_class /etc/security/audit_class.save
   ```

3. **audit_class** ファイルに新しいエントリを追加します。
   各エントリの書式は次のとおりです。

   0x*number*:*name*:*description*

0x	*number* が 16 進であることを示す
number	一意の監査クラスマスクを定義する

name 監査クラスの 2 文字の名前を定義する

description 監査クラスの記述名を定義する

4. **BSM** サービスで新しいデータが使用できるようにします。
   新しいデータを使用するには、システムをリブートするか、次のコマンドを入力します。

   ```
   # auditconfig -conf
   ```

例 — 新しい監査クラスを設定する

この例では、次のようなエントリを `audit_class` ファイルに追加します。このエントリによって、`ta` という名前の新しい監査クラスが作成されます。

```
0x01000000:ta:test application
```

▼ 監査イベントの所属先クラスの変更方法

イベントからクラスへの割り当ては、`/etc/security/audit_event` ファイル内に定義されています。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **(**省略可能**)** **`audit_event`** ファイルのバックアップコピーを保存します。

   ```
   # cp /etc/security/audit_event /etc/security/audit_event.orig
   ```

3. 特定のイベントの *flag* を変更して、そのイベントが属するクラスを変更します。
   各エントリの書式は次のとおりです。

   number : *event* : *program* : *flag*

   number 監査イベント ID を定義する

   event 監査イベントの名前を定義する

   program 監査レコードの作成を開始するシステムコールまたはユーザーレベルのプログラム実行可能ファイルを定義する

   flag 監査クラスの 2 文字の名前を定義する

4. **BSM** サービスで新しいデータが使用できるようにします。
   新しいデータを使用するには、システムをリブートするか、次のコマンドを入力します。

   ```
   # auditconfig -conf
   # audit -s
   ```

例 — サイト固有の監査イベントの割り当てを作成する

この例では、新しいクラスを定義して、そのクラスにイベントを追加します。割り当てを使用するには、新しいクラスを audit_control ファイル内に記述してから、システムをリブートします。

1. audit_class ファイルで、監視したい監査イベントのみを収集するため、サイト固有のクラスを定義します。

   ```
   0x00000800:sc:site class
   ```

2. audit_event ファイルで、一連の監査イベントの所属先を新しいクラスに変更します。

   ```
   26:AUE_SETGROUPS:setgroups(2):sc
   27:AUE_SETPGRP:setpgrp(2):sc
   40:AUE_SETREUID:setreuid(2):sc
   41:AUE_SETREGID:setregid(2):sc
   214:AUE_SETEGID:setegid(2):sc
   215:AUE_SETEUID:seteuid(2):sc
   ```

3. audit_controlファイルで新しいフラグを使用します。次のエントリは、ログインを監査するとともに、sc クラスに属するイベントのすべての正常な起動を監査します。

   ```
   flags:lo,+sc
   ```

4. 新しい構成によってすべてのプロセスが確実に監査されるようにするには、システムをリブートします。または、次の一連のコマンドを使えば、そのマシンを使用する各ユーザーが正しく監査されるようになります。*auid* はユーザー ID です。

   ```
   # auditconfig -conf
   # audit -s
   # setumask auid lo,+sc
   ```

▼ 監査イベントの追加方法

監査イベントの定義は、/etc/security/audit_event ファイルに格納されます。

1. スーパーユーザーになるか、同等の役割を引き受けます。

2. (省略可能) **audit_event** ファイルのバックアップコピーを保存します。

   ```
   # cp /etc/security/audit_event /etc/security/audit_event.save
   ```

3. **audit_event** ファイルに新しいエントリを追加します。
   各エントリの書式は次のとおりです。

   number:*name*:*description*:*classes*

   number　一意の監査イベント番号を定義する。32767 以降の番号を指定する

   name　一意の監査イベント名を定義する

description 監査イベントの説明を記述する。監査イベントのマニュアルページ名が含まれることが多い

classes このイベントを含む監査クラスを選択する

4. 監査デーモンで新しいデータが使用できるようにします。

   新しいデータを使用するには、システムをリブートするか、次のコマンドを入力します。

   ```
   # auditconfig -conf
   ```

例 — 新しい監査イベントを追加する

この例のエントリは、ローカルアプリケーションの新しい監査イベントを定義しています。

```
# grep localapp /etc/security/audit_event
32768:AUE_localapp:localapp(1):ta
```

監査サービスの構成 (作業マップ)

この節では、監査サービスを構成して有効にするために必要な作業について説明します。次の作業マップは、監査サービスの構成に必要な作業の一覧です。

作業	説明	参照先
1. (省略可能) 監査構成ファイルの変更	監査を必要とするイベント、クラス、およびユーザーを選択する	358 ページの「監査ファイルの構成 (作業マップ)」
2. 監査パーティションの作成	監査ファイルのパーティションを作成する	365 ページの「監査パーティションの作成方法」
3. `audit_warn` 別名の作成	電子メール警告を送信するユーザーを定義する	367 ページの「`audit_warn` 別名の構成方法」
4. (省略可能) 監査ポリシーの変更	追加の監査レコードや監査条件を定義する	368 ページの「監査ポリシーを有効または無効にする方法」
5. 監査の有効化	監査を有効にする	369 ページの「監査を有効にする方法」
6. (省略可能) 監査の無効化	監査を無効にする	369 ページの「監査を無効にする方法」

作業	説明	参照先
7. (省略可能) デバイス割り当ての開始	より安全にアクセスできるリムーバブルメディアを選択する	375 ページの「デバイス割り当ての管理 (作業)」

▼ 監査パーティションの作成方法

次の手順では、監査ファイル用のパーティションの作成方法、および監査に対応するファイルシステムとディレクトリの作成方法について説明します。すでに空のパーティションがある場合、またはすでに空のファイルシステムをマウントしている場合は、必要に応じて手順を省略してください。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. 必要なディスク容量を決定します。

   ホストごとに 200M バイト以上を割り当てます。ただし、ディスク容量の要件は、実行する監査のボリュームによって異なります。つまり、ディスク容量の要件が割り当てる数値を超えることがあります。予備ディレクトリのパーティション領域も含めてください。
3. 必要に応じて、監査パーティションを作成します。

   この手順は、サーバーのインストール時に実行するのが最も簡単です。サーバーにマウントされていないディスク上にパーティションを作成することもできます。パーティション作成方法の詳細については、『*Solaris* のシステム管理 *(*基本編*)* 』の「UFS ファイルシステムの作成」を参照してください。

   ```
   # newfs /dev/rdsk/cwtxdysz
   ```

   /dev/rdsk/c*w*t*x*d*y*s*z* は、パーティションの raw デバイス名です。

   ローカルホストを監査する場合は、予備の監査ディレクトリも作成します。
4. 新しいパーティションのマウント先を作成します。

   ```
   # mkdir /var/audit/server-name.n
   ```

   server-name.n は、サーバー名と各パーティションの識別番号を結合したものです。この識別番号は省略できますが、多数の監査ディレクトリを作成する場合はこの番号を使用すると便利です。
5. 新しいパーティションを自動的にマウントするエントリを追加します。

   /etc/vfstab ファイルに次のような行を追加します。

   ```
   /dev/dsk/cwtxdysz /dev/rdsk/cwtxdysz /var/audit/server-name.n   ufs  2  yes
   ```
6. **(**省略可能**)** 各パーティションの最小空き容量のしきい値を削除します。

   デフォルトの構成を使用した場合、ディレクトリの 80% がいっぱいになった時点で警告が生成されます。このため、パーティション上に空き容量を予約する必要はありません。

   ```
   # tunefs -m 0 /var/audit/server-name.n
   ```

7. 新しい監査パーティションをマウントします。

   # **mount /var/audit/***server-name***.***n*

8. 新しいパーティションに監査ディレクトリを作成します。

   # **mkdir /var/audit/***server-name***.***n***/files**

9. マウント先と新しいディレクトリへのアクセス権を訂正します。

   # **chmod -R 750 /var/audit/***server-name***.***n***/files**

10. **(**省略可能**)** ファイルサーバー上で、ほかのホストからアクセスできるファイルシステムを定義します。

    通常は、監査レコードを格納するために、ディスクファームをインストールします。監査ディレクトリを複数のシステムで使用する場合は、そのディレクトリを NFS サービスを通して共有する必要があります。/etc/dfs/dfstab ファイルに対して、次のようなエントリをディレクトリごとに追加します。

    share -F nfs /var/audit/*server-name*.*n*/files

11. **(**省略可能**)** ファイルサーバー上で、**NFS** サービスを起動し直します。

    share コマンドまたは share コマンドセットをはじめて実行する場合、NFS デーモンが動作していないことがあります。次のコマンドでデーモンを終了し、再起動してください。NFS サービスの詳細については、『*Solaris* のシステム管理 (資源管理とネットワークサービス)』の「NFS サービスの設定」 を参照してください。

    # **/etc/init.d/nfs.server stop**
    # **/etc/init.d/nfs.server start**

例 — 予備の監査ディレクトリを作成する

監査サブシステムを実行するすべてのシステムには、利用できるファイルシステムがほかにない場合に使用するローカルファイルシステムが必要です。この例では、ファイルシステムが egret という名前のシステムに追加されます。このファイルシステムは、ローカルシステムだけで使用されるため、続いてファイルサーバーの手順は必要ありません。

```
# newfs /dev/rdsk/c0t2d0
# mkdir /var/audit/egret
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1  /dev/rdsk/c0t2d0s1  /var/audit/egret ufs  2  yes  -
# tunefs -m 0 /var/audit/egret
# mount /var/audit/egret
# mkdir /var/audit/egret/files
# chmod -R 750 /var/audit/egret/files
```

例 — 新しい監査パーティションを作成する

この例では、新しいファイルシステムが、2 つの新しいディスクに作成されます。この 2 つのディスクは、ネットワーク上のほかのシステムと共有します。

```
# newfs /dev/rdsk/c0t2d0
# newfs /dev/rdsk/c0t2d1
# mkdir /var/audit/egret.1
# mkdir /var/audit/egret.2
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1  /dev/rdsk/c0t2d0s1  /var/audit/egret.1 ufs  2  yes  -
/dev/dsk/c0t2d1s1  /dev/rdsk/c0t2d1s1  /var/audit/egret.2 ufs  2  yes  -
# tunefs -m 0 /var/audit/egret.1
# tunefs -m 0 /var/audit/egret.2
# mount /var/audit/egret.1
# mount /var/audit/egret.2
# mkdir /var/audit/egret.1/files
# mkdir /var/audit/egret.2/files
# chmod -R 750 /var/audit/egret.1/files /var/audit/egret.2/files
# grep egret /etc/dfs/dfstab
 share -F nfs /var/audit/egret.1/files
 share -F nfs /var/audit/egret.2/files
# /etc/init.d/nfs.server stop
# /etc/init.d/nfs.server start
```

▼ audit_warn 別名の構成方法

audit_warn スクリプトは、audit_warn という別名に対してメールを生成します。このメールを有効な電子メールアドレスに送信するには、次のいずれかのオプションを行います。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **audit_warn** メール別名を構成します。

「オプション 1」 – audit_warn スクリプトで、audit_warn 別名をほかのメールアカウントに置き換えます。

audit_warn を root アカウントに置き換えると、電子メールメッセージを送信する行は次のようになります。

```
/usr/ucb/mail -s "$SUBJECT" root
```

audit_warn スクリプト内の 10 行にこの変更を適用する必要があります。

「オプション 2」 – audit_warn の電子メールをほかのメールアカウントにリダイレクトします。

この場合、audit_warn 別名を適切なメール別名ファイルに追加します。別名の追加先として、ローカルの /etc/mail/aliases ファイル、名前空間の mail_aliases データベースのいずれかを選択します。root メールアカウントを audit_warn 別名のメンバーとして登録する場合、新しいエントリは次のようになります。

```
audit_warn: root
```

▼ 監査ポリシーを有効または無効にする方法

監査ポリシーを使用して、ローカルホストの監査レコードの特性を決定します。デフォルトでは、すべての監査ポリシーが無効になっています。使用する監査ポリシーは、有効にする必要があります。各ポリシーについては、394 ページの「監査ポリシー」を参照してください。

プログラムレベルで `auditon()` システムコールを行なって、現在の監査ポリシーを調査したり、有効または無効にしたりすることができます。また、`auditconfig` コマンドを実行して同じタスクを行うこともできます。また、監査ポリシーの変更内容を固定するために、`audit_startup` スクリプト内で `auditconfig` コマンドの監査ポリシーオプションを変更することも可能です。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **(省略可能)** 既存の監査ポリシーを確認します。
   監査ポリシーを変更するときは、現在使用されているポリシーをすべて確認してください。次のコマンドを実行すると、有効なポリシーがすべて表示されます。

   ```
   # auditconfig -lspolicy
   ```

3. 監査ポリシーを有効または無効にします。

   ```
   # auditconfig -setpolicy flagpolicyname
   ```

   flag *flag* に + を指定すると、ポリシーが有効になる。*flag* に - を指定すると、ポリシーが無効になる

   policyname 有効または無効にするポリシーを選択する

   このポリシーの設定は、次回ブートするまで、または `auditconfig -setpolicy` コマンドを使ってポリシーを変更するまで持続します。

例 — cnt ポリシーを設定する

`cnt` ポリシーを設定すると、監査パーティションがいっぱいになっても、プロセスはブロックされません。パーティションがいっぱいになるとレコードは破棄されますが、システムは機能し続けます。`cnt` ポリシーを有効にすると、破棄された監査レコードのカウント数が記録されます。セキュリティを重視する場合は、`cnt` ポリシーは設定しないでください。ファイルシステムがいっぱいになると、イベントが記録されないことがあるためです。

次のコマンドを実行すると、`cnt` ポリシーが有効になります。

```
# auditconfig -setpolicy +cnt
```

リブートしてもポリシーの設定を維持させるには、auditconfig -setpolicy +cnt コマンドを audit_startup ファイルに追加する必要があります。

▼ 監査を有効にする方法

この操作では、監査サービスが開始されます。監査サービスが構成されている場合は、ホストをリブートしたときにもサービスが開始します。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. システムをシングルユーザーモードにします。

   ```
   # /etc/telinit 1
   ```

   詳細は、telinit(1M) のマニュアルページを参照してください。
3. スクリプトを実行して、システムが監査を実行するように構成します。

   /etc/security ディレクトリに移動し、bsmconv スクリプトを実行します。このスクリプトは、リブート後に標準 Solaris マシンが監査を実行するよう設定します。bsmconv(1M) のマニュアルページを参照してください。

   ```
   # cd /etc/security
   # ./bsmconv
   ```
4. システムをマルチユーザーモードにします。

   ```
   # /etc/telinit 6
   ```

   システムがマルチユーザーモードに移行すると、起動ファイル /etc/security/audit_startup によって監査デーモンが自動的に動作します。

▼ 監査を無効にする方法

監査が不要になった時点で、bsmunconv コマンドを実行して、監査サブシステムを無効にすることができます。bsmconv(1M) のマニュアルページを参照してください。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. システムをシングルユーザーモードにします。

   ```
   # /etc/telinit 1
   ```

   詳細は、telinit(1M) のマニュアルページを参照してください。
3. スクリプトを実行して、監査を無効にします。

   /etc/security ディレクトリに移動し、bsmunconv スクリプトを実行します。

```
# cd /etc/security
# ./bsmunconv
```

4. システムをマルチユーザーモードにします。

```
# /etc/telinit 6
```

監査レコードの管理 (作業マップ)

監査トレールを管理することによって、ネットワーク上のユーザーの動作を監視することができます。監査プロセスを行うと、大量のデータが生成される可能性があります。次の手順では、さまざまな監査データを使用して作業を行う方法について説明します。

次の表は、この節で説明する操作の一覧です。

作業	説明	参照先
監査レコードの書式の表示	特定の監査イベントのトークンの順番を表示する	370 ページの「監査レコードの書式の表示方法」
監査レコードの表示	監査レコードをユーザーが読める書式で表示する	374 ページの「監査レコードの表示方法」
監査レコードのマージ	複数のマシンの監査ファイルを 1 つの監査トレールに結合する	372 ページの「監査レコードのマージ方法」
監査トレールのオーバーフローの防止	監査ファイルシステムが完全にいっぱいになるのを防止する	375 ページの「監査トレールのオーバーフローを防ぐ方法」

▼ 監査レコードの書式の表示方法

`bsmrecord` コマンドは、監査イベントの監査 ID、監査クラス、選択マスク、およびレコード書式を表示します。このコマンドは、`audit_class` ファイル内と `audit_event` ファイル内のレコードを使用します。

次のように `-a` オプションを指定してコマンドを実行すると、すべての監査イベントのレコード書式が表示されます。`-h` オプションを指定すると、HTML 形式で一覧が出力されます。出力されたファイルは、ブラウザを使って表示できます。

- **`bsmrecord`** コマンドを使ってすべての監査イベントのレコード書式を **HTML** ファイルに出力します。

```
% bsmrecord -a -h> audit.events.html
```

*html ファイルはブラウザを使って表示できます。ブラウザの検索ツールを使って特定のレコードを検索します。

詳細は、bsmrecord(1M) のマニュアルページを参照してください。

例 — プログラムの監査レコード書式を表示する

この例では、login プログラムによって生成されたすべての監査レコードの書式を表示します。

```
% bsmrecord -p login

terminal login
  program     /usr/sbin/login      see login(1)
  event ID    6152                 AUE_login
  class       lo                   (0x00001000)
      header
      subject
      text                         error message or "successful login"
      return

login: logout
  program     /usr/sbin/login      see login(1)
  event ID    6153                 AUE_logout
  class       lo                   (0x00001000)
      header
      subject
      text                         "logout" username
      return

rlogin
  program     /usr/sbin/login      see login(1) - rlogin
  event ID    6155                 AUE_rlogin
  class       lo                   (0x00001000)
      header
      subject
      text                         success/fail message
      return

telnet login
  program     /usr/sbin/login      see login(1) - telnet
  event ID    6154                 AUE_telnet
  class       lo                   (0x00001000)
      header
      subject
      text                         success/fail message
      return
```

例 — 監査クラスの監査レコード書式を表示する

この例では、fd クラスに属するすべての監査レコードの書式を表示します。

```
% bsmrecord -c fd

ftruncate
    Not used.

truncate
    Not used.

unlink
  system call unlink                see unlink(2)
  event ID    6                     AUE_UNLINK
  class       fd                    (0x00000020)
      header
      path
      [attribute]
      subject
      return
```

▼ 監査レコードのマージ方法

次のタスクでは、すべての監査ディレクトリのすべての監査ファイルをマージする方法について説明します。監査トレールの内容を分析する場合は、次の手順を行います。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. 一次監査ディレクトリに移動します。

   # **cd /etc/security/audit/***server-name***.1/files**

   マージされたファイルは、/etc/security/audit/*server-name*.1/files ディレクトリ内に格納されます。このディレクトリは保護されています。
3. 監査レコードをマージします。

   # **auditreduce> merged.log**

   server-name 上の audit_control ファイル内の dir: 行に指定されているすべてのディレクトリがマージされます。マージされたレコードは、カレントディレクトリの merged.log ファイル内に格納されます。

例 — 監査トレール全体を表示する

監査トレール全体を一度に表示するには、auditreduce コマンドの出力を praudit コマンドにパイプします。

```
# auditreduce | praudit
```

例 — 監査トレール全体を印刷する

出力を lp コマンドにパイプすると、その出力はプリンタに送られます。

```
# auditreduce | praudit | lp
```

例 — 監査ファイルの結合と削減

auditreduce の -O オプションを使用して、複数の監査ファイルを 1 つのファイルに結合し、そのファイルを指定した出力ファイルに保存します。auditreduce を使用すると、このような結合と削除を自動的に実行できます。auditreduce(1M) のマニュアルページの -C オプションと -D オプションを参照してください。ただし、ファイルを手動で選択したほうが効率的です。find コマンドを使用したあと、auditreduce を使用して指定した一連のファイルだけを結合します。

この方法で auditreduce コマンドを使用すると、入力ファイルのすべてのレコードが 1 つの出力ファイルにマージされます。マージが完了したら、入力ファイルは削除する必要があります。また、出力ファイルは、/etc/security/audit/*server-name*/files という名前のディレクトリに保存し、auditreduce が出力ファイルを検索できるようにする必要があります。

```
# auditreduce -O combined-filename
```

auditreduce コマンドを使用すると、出力ファイル内のレコード数を減らすこともできます。このコマンドは、入力ファイルの結合時に不要なレコードを除外できます。たとえば、auditreduce コマンドを使用して、1 か月以上経過した監査ファイルから、ログインレコードとログアウトレコード以外のレコードを削除することができます。監査トレール全体が必要になった場合は、バックアップテープから監査トレールを復元できます。

```
# auditreduce -O daily.summary -b 19990413 -c lo; compress *daily.summary
# mv *daily.summary /etc/security/summary.dir
```

例 — 選択した日付のユーザーの動作を表示する

次の例では、1999 年 4 月 13 日におけるユーザー tamiko のログイン時刻とログアウト時刻を、システム管理者が確認します。管理者は、lo イベントクラスを要求します。短い書式の日付は、*yymmdd* 形式で出力されます。長い書式については、auditreduce(1M) のマニュアルページを参照してください。

```
# auditreduce -d 990413 -u tamiko -c lo | praudit
```

例 — 選択レコードを 1 つのファイルにコピーする

この例では、特定の日付におけるログイン、ログアウトのメッセージが監査トレールから選択されます。これらのメッセージは対象ファイルにマージされます。対象ファイルは、通常の監査ルートディレクトリ以外のディレクトリに書き込まれます。

```
# auditreduce -c lo -d 990413 -O /usr/audit_summary/logins
```

-O オプションを使用すると、開始時刻と終了時刻を示す 14 文字のタイムスタンプと接尾辞 logins が付いた監査ファイルが作成されます。

```
/usr/audit_summary/19990413000000.19990413235959.logins
```

例 — not_terminated 監査ファイルを整理する

監査ファイルが開いている状態で監査デーモンが終了してしまうことがあります。また、サーバーがアクセス不能になって、強制的に別のサーバーに切り替わってしまうことがあります。このような場合、その監査ファイルは監査レコードとして使用されなくなりますが、監査ファイルの終了時刻として文字列 not_terminated が付いたままになります。このようなファイルが見つかった場合、ファイルが使用されていないことを手動で確認します。開いたままのファイルを整理するには、正しいオプションを使用してファイル名を指定します。

```
# audit -s
19990414121112.not_terminated.egret
# auditreduce -O egret 19990413120429.not_terminated.egret
```

audit コマンドは、現在の監査ファイル名を確認します。auditreduce コマンドは、正しいファイル名と正しいタイムスタンプを持つ新しい監査ファイルを作成します。正しいファイル名には、正しい接尾辞 (egret) が含まれます。auditreduce は、すべてのレコードをこのファイルにコピーします。

▼ 監査レコードの表示方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **/usr/audit_summary/logins** などの監査ファイルディレクトリに移動します。

   ```
   # cd /usr/audit_summary/logins
   ```

3. **praudit** コマンドを使ってファイルを読み取ります。

   ```
   # praudit 19990413000000.19990413235959.logins | more
   ```

例 — 監査レコードを XML 形式に変換する

この例では、監査レコードを XML 形式に変換します。XML 形式はブラウザで表示できます。この形式は、報告を作成する際にも使用できます。

```
# praudit -x 19990413000000.19990413235959.logins> 19990413.logins.xml
```

*xml ファイルはブラウザを使って表示できます。スクリプトを使えば、XML ファイルの内容を操作して目的の情報を抽出できます。

▼ 監査トレールのオーバーフローを防ぐ方法

セキュリティポリシーの関係ですべての監査データを保存する必要がある場合は、次の手順に従います。

1. 監査ファイルを定期的に保存するスケジュールを設定します。保存した監査ファイルを監査ファイルシステムから削除するスケジュールを設定します。
2. 監査ファイルを手動でテープにバックアップします。これらのファイルを保存ファイルシステムに移動することもできます。
3. 監査レコードの解釈に必要な、内容に対応する情報を、監査トレールとともに格納します。
4. オフラインで移動した監査ファイルをを示すレコードを保管します。
5. 保存したテープを適切な方法で保管します。
6. サマリーファイルを作成して、格納する監査データのボリュームを削減します。

   監査トレールからサマリーファイルを抽出するには、`auditreduce` コマンドのオプションを使用します。サマリーファイルには、指定された種類の監査イベントのレコードだけが含まれます。373 ページの「例 — 監査ファイルの結合と削減」と 373 ページの「例 — 選択レコードを 1 つのファイルにコピーする」の例を参照してください。

デバイス割り当ての管理 (作業)

デバイス割り当てを使用して、さまざまなリムーバブルメディアに関連するセキュリティリスクを減らすことができます。

割り当て可能デバイスの追加 (作業マップ)

次の表は、新しい割り当て可能デバイスの定義に必要な、主な手順の一覧です。

作業	説明	参照先
1. `device_allocate` ファイル内のエントリの作成または変更	デバイス割り当てメカニズムを使用して、制御するデバイスを定義する	376 ページの「割り当て可能デバイスの変更方法」

作業	説明	参照先
2. ロックファイルの作成	デバイス割り当てメカニズムを有効にして、特定のデバイスを操作する	376 ページの「割り当て可能デバイスのロックファイルの設定方法」
3. (省略可能) デバイスクリーンスクリプトの作成	物理デバイスからデータを一掃する	421 ページの「デバイスクリーンスクリプト」
4. デバイスの割り当て	デバイス割り当てメカニズムにデバイスを追加する	377 ページの「デバイスを割り当てる方法」
5. (省略可能) デバイス割り当ての解除	デバイスの使用を解除する	377 ページの「デバイスの割り当てを解除する方法」

▼ 割り当て可能デバイスのロックファイルの設定方法

ロックファイルとは、 `/etc/security/dev` ディレクトリに作成されるサイズ 0 のファイルです。割り当て可能デバイスごとに 1 つのファイルが作成されます。割り当て可能デバイスのロックファイルがない場合は、そのデバイスを割り当てることができず、誰もアクセスできません。

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. **dminfo** コマンドを使用して、 **device_maps** ファイルのエントリからそのデバイスのデバイス名を取得します。

   418 ページの「`device_maps` ファイル」 と、`dminfo(1M)` および `device_maps(4)` のマニュアルページを参照してください。たとえば、デバイスタイプ `st` のデバイス名は `st0` です。次の手順では、そのデバイス名をロックファイル名として使用します。
3. **touch** コマンドを使用して、そのデバイスの空のロックファイルを作成します。

   ファイル名としてデバイス名を使用します。*device-name* に代入してください。

   ```
   # cd /etc/security/dev
   # touch device-name
   # chmod 600 device-name
   # chown bin device-name
   # chgrp bin device-name
   ```

▼ 割り当て可能デバイスの変更方法

次の手順では、デバイス割り当てメカニズムに使用できるデバイスを定義します。

1. スーパーユーザーになるか、同等の役割を引き受けます。

2. **/etc/security/device_allocate** ファイルに一覧されているデバイスを確認します。
3. **device_allocate** ファイルには指定されていないデバイスのうち、割り当て可能にするデバイスを決定します。
4. **device_allocate** ファイルを編集して新しいデバイスを追加します。
   各エントリの書式は次のとおりです。

   device-name:*device-type*:::*program*

   device-name デバイス名を指定する

   device-type デバイスタイプを指定する

   program 実行するパージプログラムを指定する

▼ デバイスを割り当てる方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
2. デバイス名を指定して **allocate** コマンドを使用します。

```
sar1% allocate st0
```

allocate コマンドの - g オプションを使用して、デバイスタイプでデバイスを割り当てることもできます。

コマンドでデバイスを割り当てられない場合は、コンソールウィンドウにエラーメッセージが表示されます。割り当てのエラーメッセージについては、allocate (1) のマニュアルページを参照してください。

例 — プリンタを割り当てる

allocate コマンドを実行したユーザーだけがプリンタを使用できます。

```
sarl% allocate /dev/lp/chestnut
```

▼ デバイスの割り当てを解除する方法

割り当てを解除すると、ほかのユーザーもユーザーの使用後にそのデバイスを割り当てて使用できるようになります。

- **deallocate** コマンドに続けてデバイスファイル名を使用し、デバイスの割り当てを解除します。

```
sar1% deallocate st0
```

例 — プリンタの割り当てを解除する

chestnut という名前のプリンタの割り当てを解除するには、次のコマンドを入力します。

```
# deallocate /dev/lp/chestnut
```

例 — 強制的に割り当てを解除する

ユーザーに割り当てられたデバイスは、プロセスが終了するとき、またはそのユーザーがログアウトするときに、自動的にその割り当てが解除されません。次の書式の deallocate コマンドは、通常、ユーザーが特定のデバイスの割り当てを解除し忘れたときのために使用します。次のコマンドは、デバイス割り当てを解除して、ほかのユーザーがデバイス割り当てを行えるようにします。

```
# deallocate -F st0
```

例 — すべてのデバイスの割り当てを解除する

注意 – すべてのデバイスの割り当てを解除できるのは、システムを初期化しているときだけです。

```
# deallocate -I
```

第 23 章

BSM サービス (参照)

この章では、BSM サービスの重要なコンポーネントである、監査サブシステムとデバイス割り当てメカニズムについて説明します。

監査メカニズムを使用して、システムが通常と異なる方法で使用されていないかどうか検査すると、潜在的なセキュリティ違反を検出できます。また、通常と異なる動作を追跡して、特定のユーザーを突き止めることができるため、セキュリティ違反を抑えることができます。つまり、ユーザーは自分の動作が監査されそうだと考えると、違反行為を思いとどまる可能性が大きくなります。

この章の内容は次のとおりです。

監査の概要については、第 20 章を参照してください。計画時のヒントについては、第 21 章を参照してください。サイトで監査を構成する手順については、第 22 章を参照してください。

監査コマンド

この節では、監査サービスで使用されるコマンドについて説明します。

監査デーモン

次に、監査デーモン auditd の役割を示します。

- auditd は、 audit_control ファイル内で指定されたディレクトリ内の監査ログファイルを、指定された順序で開き、閉じます。
- auditd は、監査データをカーネルから読み取り、監査ログファイルに書き込みます。
- auditd は、監査ディレクトリ内のデータ量が audit_controlファイル内で指定された上限を超えると、 audit_warn スクリプトを実行します。デフォルトでは、このスクリプトは audit_warn メールの別名とコンソールに警告を送信します。
- デフォルトでは、監査ディレクトリがすべていっぱいになると、監査レコードを生成するプロセスは中断されます。また、auditd コマンドは、コンソールと audit_warn メールの別名にメッセージを送ります。この監査ポリシーは、auditconfig コマンドを使用して構成し直すことができます。この時点では、システム管理者だけが、監査メカニズムの修復を行えます。管理者は、ログインして監査ファイルをテープに書き込んだり、監査ファイルをシステムから削除したり、その他のクリーンアップを実行したりできます。

auditd デーモンは、マシンがマルチユーザーモードになると自動的に起動されますが、コマンド行から起動することもできます。監査デーモンが起動すると、デーモンは監査ログファイルに必要な空き容量を判断します。

監査デーモンは、audit_control ファイル内に指定されている監査ディレクトリに、監査ファイルを作成します。監査デーモンは、このディレクトリの一覧へのポインタを、最初のディレクトリに位置付けます。監査デーモンは、監査ファイルを作成する必要があるたびに、一覧内の最初の使用可能ディレクトリ内に監査ファイルを格納します。一覧は、監査デーモンの現在のポインタ位置から始まります。このポインタを一覧の最初のディレクトリに設定し直すには、audit -s コマンドを実行します。audit -n コマンドは、新しい監査ファイルに切り替えるように監査デーモンに指示します。新しいファイルは、現在のファイルと同じディレクトリ内に作成されます。

audit コマンド

audit コマンドは、監査デーモンの動作を制御します。audit コマンドは、次の操作を実行できます。

- 監査機能を使用可能および使用不可にする
- 監査デーモンを設定し直す
- ローカルマシンの監査事前選択マスクを調整する
- 監査レコードを別の監査ログファイルに書き込む

利用できるオプションについては、audit(1M) のマニュアルページを参照してください。

bsmrecord コマンド

bsmrecord コマンドは、/etc/security/audit_event ファイル内に定義されている監査イベントの書式を表示します。監査イベントの監査 ID、監査クラス、監査フラグ、およびレコードのトークンが順に出力されます。オプションを指定しなかった場合、 bsmrecord は端末での表示に適した形で出力します。-h オプションを指定した場合、ブラウザでの表示に適した形式で出力します。使用例については、370 ページの「監査レコードの書式の表示方法」を参照してください。詳細は、bsmrecord(1M) のマニュアルページを参照してください。

auditreduce コマンド

auditreduce コマンドを使用すると、1 つまたは複数の入力監査ファイル内の監査レコードをマージできます。このコマンドでは、監査レコードの事後選択を実行することもできます。auditreduce(1M) のマニュアルページを参照してください。監査トレール全体をマージするには、監査サーバー上でこのコマンドを実行します。監査サーバーとは、すべての監査ファイルシステムがマウントされているマシンのことです。

auditreduce コマンドを使用すると、複数のマシン上のすべての監査対象動作を、1 か所から追跡できます。このコマンドは、すべての監査ファイルを論理的に結合し、単一の監査トレールとして読み取ることができます。サイト内のすべてのマシンが同一の監査構成を持つようにするとともに、サーバーと監査ログファイル用のローカルディレクトリを作成しておく必要があります。auditreduce では、レコードの生成方法や格納場所は無視されます。オプションを指定しなかった場合、auditreduce コマンドは、監査ルートディレクトリのすべてのサブディレクトリ内のすべての監査ファイルの監査レコードをマージします。通常、/etc/security/audit が監査ルートディレクトリです。auditreduce コマンドは、マージ結果を標準出力に送ります。マージ結果は、時系列に並べて 1 つの出力ファイルに格納することもできます。このファイルの形式はバイナリデータです。

auditreduce コマンドを使用して、特定の種類のレコードを選択し、解析に利用することもできます。auditreduce コマンドのマージ機能と選択機能は、論理的にほかに依存しません。auditreduce は、入力ファイルのレコードを読み取ると、マージしてディスクに書き込む前に、データを抽出します。

praudit コマンドは、auditreduce コマンドのバイナリ出力を、ユーザーが読める書式に変換します。

auditreduce コマンドにオプションを指定すると、次の操作も実行できます。

- 特定の監査フラグによって生成された監査レコードを要求する
- 特定のユーザーによって作成された監査レコードを要求する
- 特定の日付に作成された監査レコードを要求する

auditreduce に引数を指定しなかった場合は、デフォルトの監査ルートディレクトリ /etc/security/audit 内のサブディレクトリが検査されます。このコマンドは、*start-time.end-time.hostname* ファイルが配置されている files ディレクトリを検査します。auditreduce コマンドは、監査データが異なるディレクトリに格納されている場合に非常に有用です。図 23–1 は、監査データがホスト別のディレクトリ内に格納されている場合を示しています。図 23–2 は、監査データが監査サーバー別のディレクトリ内に格納されている場合を示しています。

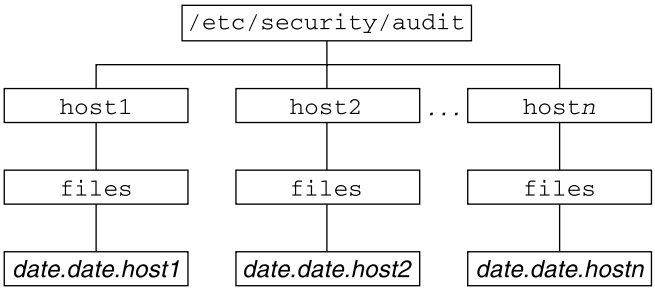


図 **23–1** ホストごとに格納された監査トレール

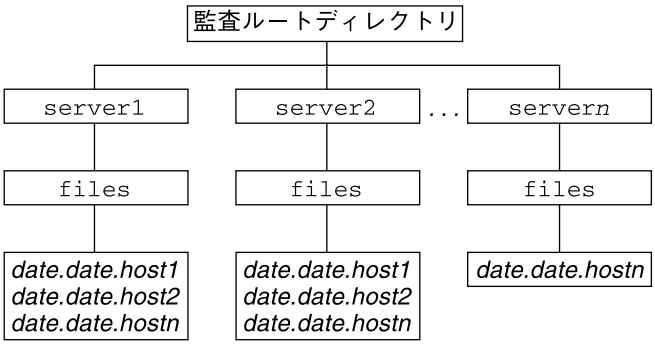


図 **23–2** サーバーごとに格納された監査トレール

/etc/security/audit のパーティションが小さい場合、デフォルトのディレクトリに監査データを格納しない方法もあります。-R オプションを使用して、auditreduce コマンドを別のディレクトリに渡すことができます。

```
# auditreduce -R /var/audit-alt
```

-S オプションを使用して、特定のサブディレクトリを指定することもできます。

```
# auditreduce -S /var/audit-alt/host1
```

特定の監査ログファイルだけを処理するには、auditreduce にそのファイルをコマンド引数として直接指定できます。

```
# auditreduce /var/audit/egret/files/2001*.2001*egret
```

その他のオプションと例については、auditreduce(1M) のマニュアルページを参照してください。

praudit コマンド

praudit コマンドは、標準入力からバイナリ形式の監査レコードを読み込み、そのレコードを表示可能な書式で表示します。auditreduce コマンドまたは 1 つの監査ファイルからの出力は、praudit コマンドの入力にパイプできます。catコマンドを使用すると、複数のファイルを連結して入力にパイプすることができます。tail コマンドを使用すると、現在の監査ファイルを入力にパイプできます。

praudit コマンドでは、次の 5 つの出力形式を生成できます。

- デフォルト – デフォルトでは、1 行に 1 つの監査トークンが表示されます。デフォルトでは、監査イベントは ioctl(2) などのその内容が表示され、テキストで表示できる値はすべてテキスト形式で表示されます。たとえば、ユーザーは、ユーザー ID ではなく、ユーザー名で表示されます。
- -l オプション – このオプションでは、1 行に 1 つの監査レコードが表示されます。-d オプションを指定すると、トークンフィールドおよびトークン間で使用される区切り文字を変更できます。デフォルトの区切り文字は、コンマです。
- -r オプション – このオプションでは、数値で表現できる値はすべて数値として表示されます。たとえば、ユーザーはユーザー ID で、インターネットアドレスは 16 進形式で、モードは 8 進形式で表示されます。監査イベントは、イベント番号 (158 など) で表示されます。
- -s オプション – このオプションでは、監査イベントがテーブル名 (AUE_IOCTL など) で表示されます。その他のトークンは、デフォルトと同じ形式で表示されます。
- -x オプション – このオプションでは、監査レコードが XML 形式で表示されます。このオプションは、出力結果をブラウザや XML を操作するスクリプトに入力する場合に便利です。

  XML は、監査サブシステムが提供する DTD によって記述されます。また、Solaris ソフトウェアにはスタイルシートも付属しています。DTD とスタイルシートは /usr/share/lib/xml ディレクトリ内に格納されています。

praudit のデフォルトの出力形式では、各レコードは監査トークンのシーケンスとして容易に識別できます。各トークンは 1 行ごとに出力されます。各監査レコードは header トークンで始まります。awk コマンドなどを使用すると、出力をさらに処理できます。

次の出力は、 header トークンを praudit コマンドのデフォルトで出力したものです。

```
header,240,1,ioctl(2),es,Tue Sept  7 16:11:44 1999, + 270 msec
```

次の出力は、同じ header トークンを praudit -r コマンドで出力したものです。

```
20,240,1,158,0003,699754304, + 270 msec
```

例 — praudit 出力をスクリプトで処理する

praudit コマンドの出力は、必要に応じてテキストとして操作できます。たとえば、auditreduce コマンドでは選択できないレコードを選択したいことがあります。単純なシェルスクリプトを使用すると、praudit の出力を処理できます。次の単純なスクリプトの例は、1 つの監査レコードを 1 行にまとめ、ユーザーが指定した文字列を検索し、最後に監査ファイルを元の形式に戻します。具体的には、スクリプトは次の処理を実行します。

1. header トークンに、Control-A という接頭辞を目印として付けます。
2. 1 つのレコード内のすべての監査トークンを 1 行に結合します。ただし、改行の情報は Control-A として保持されます。
3. grep コマンドを実行します。
4. 元の改行を復元します。

```
#!/bin/sh
praudit | sed -e '1,2d' -e '$s/^file.*$//' -e 's/^header/^aheader/' \\
| tr '\\012\\001' '\\002\\012' \\
| grep "$1" \\
| tr '\\002' '\\012'
```

スクリプトの ^a は、^ と a という 2 つの文字ではなく、Control-A です。この接頭辞によって、ヘッダートークンが、テキストとして表示される header 文字列と区別されます。

auditconfig コマンド

auditconfig コマンドは、監査構成パラメータを取得して設定するためのコマンド行インタフェースを提供します。auditconfig コマンドは、次の操作を実行できます。

- 監査ポリシーの表示、チェック、構成
- 監査状態 (オン/オフ) の確認
- 監査状態 (オン/オフ) の切り替え
- 監査ディレクトリと監査ファイルの管理
- 監査キューの管理
- 事前選択マスクの取得/設定
- 監査イベントの監査クラスへの割り当ての取得/設定

- セッション ID や監査 ID などの構成情報の取得/設定
- プロセス、シェル、セッションの監査特性の構成
- 監査統計情報のリセット

利用できるオプションについては、auditconfig(1M) のマニュアルページを参照してください。

監査サービスファイル

監査プロセスでは、次のファイルが使用されます。

/etc/system ファイル

/etc/system ファイルには、カーネルが初期設定で読み込み、システム動作をカスタマイズするためのコマンドが格納されます。bsmconv および bsmunconv シェルスクリプトは、監査機能を起動および終了するときに使用され、/etc/system ファイルを変更します。bsmconv シェルスクリプトは、 /etc/system ファイルに次の行を追加します。

```
set c2audit:audit_load=1
```

set c2audit:audit_load=1 コマンドは、システムのブート時に監査モジュール (カーネルモジュールの一種) をロードします。bsmunconv シェルスクリプトは、システムのリブート時に監査を無効にします。このコマンドは、/etc/system ファイルから c2audit の行を削除します。

audit_class ファイル

/etc/security/audit_class ファイルには、既存の監査クラスの定義が含まれます。監査クラスは、監査イベントのグループです。各監査クラスには、クラスの短縮名として監査フラグが関連付けられます。audit_control ファイル内のこの短縮名を使用して、監査するイベントのクラスを事前選択します。監査フラグでは、接頭辞を使ってきめ細かな選択が行えます。詳細については、393 ページの「監査フラグの構文」を参照してください。

root ユーザーまたはそれと同等の役割を持つ管理者は、監査クラスの定義を変更できます。管理者は、audit_class ファイルをテキストエディタで編集することによって、新しい監査クラスを定義したり、既存クラスの名前を変更したり、既存クラスにその他のさまざまな変更を施したりすることができます。詳細は、audit_class(4)のマニュアルページを参照してください。監査フラグについては、392 ページの「監査フラグの定義」を参照してください。

audit_controlファイル

各マシン上の /etc/security/audit_control ファイルは、監査デーモンによって読み込まれます。詳細については、audit_control(4) のマニュアルページを参照してください。audit_control ファイルは /etc/security ディレクトリにあります。各マシンには、独自の audit_control ファイルがローカルに格納されています。このファイルを使用すると、個々のマシン上で、さまざまな場所にある監査ファイルシステムをさまざまな順序でマウントすることが可能となります。たとえば、マシン A の一次監査ファイルシステムは、マシン B の二次監査ファイルシステムになっている場合があります。

audit_control ファイルには、次の 4 種類の情報を指定します。各行の情報は、キーワードで始まります。

- 「監査フラグ」行は flags: で始まります。 この行に指定する監査フラグでは、マシン上のすべてのユーザーを対象に監査するイベントのクラスを、事前選択します。ここで指定する監査フラグは、「マシン全体の監査フラグ」または「マシン全体の監査事前選択マスク」と呼びます。 監査フラグは、空白を入れずにコンマで区切ります。
- 「帰属不可能フラグ」行は naflags: で始まります。この行に指定する監査フラグでは、特定のユーザーに起因しない動作が発生したときに監査するイベントクラスを、事前選択します。監査フラグは空白を入れずにコンマで区切ります。
- 「監査しきい値」行は minfree: で始まります。この行では、すべての監査ファイルシステムに確保する最小空き領域のレベルを定義します。minfree の割合は、0 以上で指定します。デフォルトは 20% です。監査ファイルシステムの使用率が 80% に達すると、次に利用可能な監査ディレクトリに監査データが格納されるようになります。audit_warn(1M) のマニュアルページを参照してください。
- 「ディレクトリ定義」行は dir: で始まります。 各行には、監査ログファイルを格納するためにマシンが使用する、監査ファイルシステムとディレクトリを定義します。1 行または複数行のディレクトリを定義できます。dir: 行では、順番が重要になります。auditd デーモンは、ここで指定した順番でディレクトリに監査ファイルを作成します。1 番目のディレクトリがそのマシンの 1 次監査ディレクトリになり、2 番目のディレクトリが 2 次監査ディレクトリになります。1 番目のディレクトリがいっぱいになると、監査デーモンは 2 番目以降のディレクトリに監査トレールファイルを作成します。 audit(1M) のマニュアルページを参照してください。

audit_control ファイルは、各マシン上での構成処理中に作成されます。

audit_control ファイルを変更したときは、audit -s コマンドを実行すると、監査デーモンによってファイルが再度読み取られます。

注 – audit -s コマンドでは、既存のプロセスについて指定された事前選択マスクは変更されません。既存のプロセスについては、auditconfig、setaudit、auditon のいずれかを使用してください。詳細は、getaudit(2) および auditconfig(1M) のマニュアルページを参照してください。

---

audit_control ファイルの例

次の例は、マシン dopey で使用する audit_control ファイルです。dopey では、監査サーバー blinken 上で 2 つの監査ファイルシステムを使用し、2 つ目の監査サーバー winken からマウントされる 3 つ目の監査ファイルシステムを使用します。3 つ目のファイルシステムは、blinken 上の監査ファイルシステムがいっぱいであるか使用できないときにだけ使用されます。minfree の値として 20% を指定しているため、ファイルシステムの使用率が 80% に達した時点で警告スクリプトが実行されます。以下のフラグでは、監査対象としてログイン操作と管理操作が指定されています。これらの操作について、その成功と失敗が監査されます。ファイルシステムオブジェクト作成の失敗を除くすべての失敗が、監査対象となります。また、ユーザーに起因しないイベントも監査されています。

```
flags:lo,am,-all,^-fc
naflags:lo,nt
minfree:20
dir:/etc/security/audit/blinken/files
dir:/etc/security/audit/blinken.1/files
#
# blinken がいっぱいになったときに使用する監査ファイルシステム
#
dir:/etc/security/audit/winken
```

audit_data ファイル

auditd デーモンは、各マシン上で起動されると、ファイル /etc/security/audit_data を作成します。このファイルの書式は、コロンで区切られた 2 つのフィールドを含む 1 行のエントリから構成されています。audit_data(4) のマニュアルページを参照してください。最初のフィールドには、監査デーモンのプロセス ID を指定します。2 番目のフィールドには、監査デーモンが監査レコードを現在書き込んでいる監査ファイルのパス名を指定します。次に例を示します。

```
# cat /etc/security/audit_data
116:/etc/security/audit/blinken.1/files/19990320100002.not_terminated.dopey
```

audit_event ファイル

/etc/security/audit_event ファイルには、イベントからクラスへの割り当てのデフォルト値が格納されます。このファイルを編集して、クラスの割り当てを変更できます。ただし、変更した場合は、システムをリブートするか auditconfig -conf を実行して、変更した割り当てをカーネルに読み込む必要があります。
audit_event(4) のマニュアルページを参照してください。

audit_startup スクリプト

/etc/security/audit_startup スクリプトは、システムがマルチユーザーモードに移行すると監査デーモンを自動的に起動します。このスクリプトは、監査デーモンの実行直前に起動シーケンスの一部として起動されます。詳細は、audit_startup(1M) のマニュアルページを参照してください。

デフォルトの audit_startup スクリプトは、イベントからクラスへの割り当てを自動的に構成し、監査ポリシーを設定します。このスクリプトは、BSM パッケージのインストール時に作成されます。

audit_user ファイル

ユーザーごとに異なる方法で監査するには、/etc/security/audit_user ファイルを編集して、ユーザーごとに監査フラグを追加します。これらの監査フラグが指定されている場合は、audit_control ファイルのシステム全体で有効なフラグと組み合わせ、そのユーザーに対して監査するイベントクラスが決定されます。
audit_user ファイル内のユーザーエントリに追加するフラグは、audit_control ファイルにあるデフォルトを次の 2 つの方法で変更します。

- そのユーザーについて常に監査するイベントクラスを指定する
- そのユーザーについて監査しないイベントクラスを指定する

audit_user ファイルの各ユーザーエントリには、次の 3 つのフィールドがあります。

- *username* フィールド
- *always-audit* フィールド
- *never-audit* フィールド

これらの監査フィールドは、この順番で処理されます。*always-audit* フィールドは、指定されたクラスの監査を有効にします。*never-audit* フィールドは、指定されたクラスの監査を無効にします。

注 – *never-audit* フィールド内で `all` 監査フラグを設定したままにする誤りがよくあるので注意してください。`all` 監査フラグを指定したままにすると、そのユーザーの監査がすべてオフに設定され、*always-audit* フィールドに設定されているフラグが無効になります。このフラグは、`audit_control` ファイルで設定されたマシン全体の監査フラグよりも優先されます。

ユーザーの *never-audit* フラグは、システムのデフォルト値よりも優先されます。システムのデフォルトを有効にしたい場合も考えられます。たとえば、ファイルシステムオブジェクトの正常な読み込みを除いて、ユーザー `tamiko` のすべてのイベントを監査するとします。この方法は、ユーザーのほとんどすべての動作を監査しますが、監査データは、すべてのデータ読み取りを監査した場合の約 4 分の 3 しか生成されません。このとき、システムデフォルトを `tamiko` に適用したいとします。次に 2 つの `audit_user` エントリ例を示します。

正しいエントリ

```
tamiko:all,^+fr:
```

間違ったエントリ

```
tamiko:all:+fr
```

1 つ目の例は、「ファイル読み取り動作を除くすべての動作を監査する」ことを表しています。2 つ目の例は「常にすべての動作を監査するが、正常なファイルの読み取り動作はまったく監査しない」ことを表しています。2 つ目の例は正しいエントリではありません。*never-audit* フィールドがシステムのデフォルト値を無効にするからです。1 つ目の例では期待どおりの結果になります。*always-audit* フラグには、`all` フラグへの例外が含まれています。*never-audit* フィールドにはフラグが指定されていないため、`audit_control` ファイル内のシステムデフォルト値は無効になりません。

注 – 正常終了したイベントと失敗したイベントは別々に取り扱われます。プロセスが生成する監査レコードの数は、イベントが正常終了した場合よりも失敗した場合のほうが多くなる可能性があります。

audit_warn スクリプト

監査デーモンは、監査レコードの書き込み中に異常な状態が発生すると、`/etc/security/audit_warn` スクリプトを起動します。詳細については、`audit_warn(1M)` のマニュアルページを参照してください。このスクリプトをサイトに合わせてカスタマイズすることで、手動による対処が必要な状態を警告するようにしたり、そのような状態を自動的に処理するための方法を指定したりできます。エラーが発生すると、`audit_warn` はコンソールにメッセージを書き込みます。`audit_warn` はさらに、`audit_warn` メール別名にもメッセージを送信します。監査を有効にしたときは、この別名を設定する必要があります。

監査デーモンは、次の状態を検出すると、 audit_warn スクリプトを起動します。

- 監査ディレクトリが minfree の許容値を超えていっぱいになった場合。minfree 値はソフト制限値で、監査ファイルシステム上で使用できる領域の割合です。

  audit_warn スクリプトは、文字列 soft と、使用可能領域が下限値を下回ったディレクトリ名を使用して起動されます。監査デーモンは、次の適切なディレクトリに自動的に切り替えます。デーモンは、新しいディレクトリが minfree 制限値に達するまで、このディレクトリに監査ファイルを書き込みます。その後、監査デーモンは、audit_control ファイルに指定された順序で残りの各ディレクトリに順次アクセスします。デーモンは、各ディレクトリが minfree 制限値に達するまで監査レコードを書き込みます。

- すべての監査ディレクトリが minfree のしきい値に達した場合。

  文字列 allsoft を使用して audit_warn スクリプトが起動されます。コンソールにメッセージが書き込まれます。さらに、audit_warn の別名にメールが送信されます。

  audit_control ファイルに指定されたすべての監査ディレクトリが minfree しきい値に達すると、監査デーモンは最初のディレクトリに戻ります。デーモンは、そのディレクトリが完全にいっぱいになるまで監査レコードを書き込みます。

- 監査ディレクトリが完全にいっぱいになり、残りの容量がなくなった場合。

  文字列 hard とディレクトリ名を使用して、audit_warn スクリプトが起動されます。コンソールにメッセージが出力されます。さらに、audit_warn の別名にメールが送信されます。

  監査デーモンは、使用可能領域が残っている次の適切なディレクトリに自動的に切り替えます。その後、監査デーモンは、audit_control ファイルに指定された順序で残りの各ディレクトリに順次アクセスします。デーモンは、各ディレクトリがいっぱいになるまで完全レコードを書き込みます。

- すべての監査ディレクトリが完全にいっぱいになった場合。引数として文字列 allhard を使用して、audit_warn スクリプトが起動されます。

  デフォルトの構成では、コンソールにメッセージが書き込まれます。さらに、audit_warn の別名にメールが送信されます。監査レコードを生成するプロセスは中断されます。監査デーモンはループに入り、領域が使用可能になるのを待って監査レコードの処理を再開します。監査レコードが処理されるまで、監査対象の動作は待機します。監査レコードを生成しようとするプロセスは、すべて中断されます。このため、別の監査管理アカウントを設定し、監査機能を有効にしないで操作できるように設定する必要があります。このように設定すれば、操作を中断せずに継続することができます。

- 内部エラーが発生した場合。次のような内部エラーが考えられます。

  - ebusy – 別の監査デーモンプロセスがすでに動作している
  - tmpfile – 一時ファイルを使用できない
  - auditsvc – auditsvc() システムコールが失敗した
  - postsigterm – 監査のシャットダウン中に信号を受信した

  audit_warn の別名にメールが送られます。

- audit_control ファイルの構文に問題が検出された場合。デフォルトでは、コンソールにメッセージが書き込まれます。さらに、audit_warn の別名にメールが送信されます。

監査管理プロファイル

Solaris オペレーティング環境には、監査サービスを構成するためのプロファイルや監査トレールを分析するためのプロファイルが用意されています。

- **Audit Control –** 特定の役割が Solaris 監査を構成できるようにするためのプロファイル。このプロファイルは、監査ファイルを構成したり監査コマンドを実行したりする権限を付与します。Audit Control プロファイルで割り当てられた役割が実行できるコマンドは次のとおりです。 audit、auditd、auditconfig、bsmconv、および bsmunconv
- **Audit Review –** 特定の役割が Solaris 監査レコードを分析できるようにするためのプロファイル。このプロファイルは、praudit コマンドと auditreduce コマンドを使って監査レコードを読み取る権限を付与します。このプロファイルで割り当てられた役割は、 auditstat コマンドを実行することもできます
- **System Administrator –** Audit Review プロファイルを含むプロファイル。System Administrator プロファイルで割り当てられた役割は、監査レコードを分析できます。

プロファイルを役割に割り当てるには、106 ページの「役割の作成」を参照してください。

監査クラスと監査フラグ

「監査フラグ」は監査対象となるイベントのクラスを示します。マシン全体で有効な監査デフォルト値は、audit_control ファイル内のフラグによって各マシン上のすべてのユーザーに対して指定されます。このファイルについては、386 ページの「audit_controlファイル 」を参照してください。

監査フラグを audit_user ファイルにあるユーザーエントリに入れることにより、各ユーザーについて監査を行う対象を変更できます。監査フラグは、auditconfig コマンドの引数としても使用します。auditconfig(1M) のマニュアルページを参照してください。

監査フラグの定義

次の表に、事前に定義されている監査クラスを示します。この表には、監査フラグ、ロング名、および短い説明が記載されています。監査フラグは、監査クラスを表す短縮名です。監査するイベントのクラスを指定するときは、監査構成ファイルの監査フラグを使用します。監査フラグは、`auditconfig` などの監査コマンドの引数としても使用します。新しいクラスを定義するには、`audit_class` ファイルを変更します。既存クラスの名前の変更も可能です。詳細は、`audit_class(4)` のマニュアルページを参照してください。

表 23–1 事前に定義されている監査フラグ

短縮名	ロング名	短い説明
`all`	`all`	すべてのクラス (メタクラス)
`no`	`no_class`	イベントの事前選択を無効にする空の値
`na`	`non_attrib`	ユーザーが原因ではないイベント
`fr`	`file_read`	データを読み取る、読み取りのために開く
`fw`	`file_write`	データを書き込む、書き込みのために開く
`fa`	`file_attr_acc`	オブジェクト属性にアクセスする :`stat`、`pathconf`
`fm`	`file_attr_mod`	オブジェクト属性を変更する :`chown`、`flock`
`fc`	`file_creation`	オブジェクトの作成
`fd`	`file_deletion`	オブジェクトの削除
`cl`	`file_close`	`close` システムコール
`ap`	`application`	アプリケーションが定義するイベント
`ad`	`administrative`	管理上の操作 (旧 administrative メタクラス)
`am`	`administrative`	管理上の操作 (メタクラス)
`ss`	`system state`	システムの状態を変更
`as`	`system-wide administration`	システム全体の管理
`ua`	`user administration`	ユーザー管理
`aa`	`audit administration`	監査の使用
`ps`	`process start`	プロセスの起動、プロセスの停止
`pm`	`process modify`	プロセスの変更
`pc`	`process`	プロセス (メタクラス)

表 23–1 事前に定義されている監査フラグ　(続き)

短縮名	ロング名	短い説明
ex	exec	プログラムの実行
io	ioctl	ioctl システムコール
ip	ipc	System V の IPC 操作
lo	login_logout	ログインとログアウトのイベント
nt	network	ネットワークイベント: bind、connect、accept
ot	other	その他

監査フラグの構文

監査フラグの接頭辞によって、イベントクラスが成功した場合に監査するのか、失敗した場合に監査するのか、が決まります。接頭辞を指定しなかったクラスは、成功した場合も失敗した場合も監査されます。次の表に、監査フラグの書式といくつかの例を示します。

表 23–2 接頭辞 (プラス記号、マイナス記号) の付いた監査フラグ

prefixflag	意味
lo	成功したすべてのログインとログアウト、および失敗したすべてのログインを監査する。ログアウトが失敗することはない
+lo	成功したすべてのログインとログアウトを監査する
-all	失敗したすべてのイベントを監査する
+all	成功したすべてのイベントを監査する

注意 – all フラグを指定すると、大量のデータが生成され、監査ファイルシステムがすぐにいっぱいになる可能性があります。all フラグは、特別な理由ですべての活動を監査する場合にだけ使用してください。

監査フラグを変更する接頭辞

キャレット接頭辞 ^ を使えば、選択されている監査フラグをさらに変更できます。次の表は、キャレット接頭辞を使って選択済みの監査フラグを変更する方法を示したものです。

表 **23–3** 指定済みの監査フラグを変更するキャレット接頭辞

^*prefixflag*	意味
`-all,^-fc`	失敗したすべてのイベントを監査する。ただし、失敗したファイルシステムオブジェクト作成は監査しない
`am,^+aa`	成功または失敗したすべての管理イベントを監査する。ただし、成功した監査管理は監査しない
`am,^ua`	成功または失敗したすべての管理イベントを監査する。ただし、ユーザー管理イベントは監査しない

監査フラグの接頭辞は、次のファイルとコマンドで使用できます。

- `audit_control` ファイルの `flags` 行で使用する
- `audit_user` ファイルのユーザーエントリの `flags` フィールドで使用する
- `auditconfig` コマンドの引数で使用する

`audit_control` ファイル内での接頭辞の使用例については、386 ページの「`audit_control`ファイル 」を参照してください。

監査ポリシー

監査ポリシーには、監査トレールにトークンまたは情報を追加するかどうかを指定します。監査ポリシーについては、352 ページの「使用する監査ポリシーの決定」 を参照してください。

プロセスの監査特性

最初のログイン時に次の監査特性が設定されます。

- 「プロセス事前選択マスク」 – `audit_control` ファイル内の監査フラグと `audit_user` ファイル内の監査フラグを結合したもの。ユーザーがログインすると、`login` コマンドは、これらのフラグを結合し、そのユーザーのプロセスに対する「プロセス事前選択マスク」を確立します。 プロセス事前選択マスクは、各監査イベントクラス内のイベントで監査レコードを生成するかどうかを指定します。

  プロセス事前選択マスクを取得するアルゴリズムは、次の式で表されます。

user's process preselection mask = (flags: line + always-audit flags) - never-audit flags

audit_control ファイル内の flags: 行にある監査フラグを、audit_userファイル内のユーザーエントリの *always-audit* フィールドにあるフラグに追加します。次に、ユーザーの *never-audit* フィールドにあるフラグを、全体のフラグから減算します。

- 「監査 ID」 – ユーザーがログインすると、プロセスは監査 ID を取得します。監査 ID は、ユーザーの初期プロセスが起動するすべての子プロセスに継承されます。監査 ID はアカウントの追跡を強行するときにも役立ちます。ユーザーがスーパーユーザーになったあとも、監査 ID はそのまま変わらずに残ります。各監査レコード内に保存された監査 ID を使用すると、常に動作を追跡してログインした元のユーザーまでたどることができます。
- 「監査セッション ID」 – 監査セッション ID はログイン時に割り当てられます。このセッション ID はすべての子孫プロセスに継承されます。
- 「端末 ID (ポート ID、マシン ID)」 – 端末 ID は、ホスト名とインターネットアドレスで構成され、そのあとにユーザーがログインした物理デバイスを識別する一意の番号が続きます。通常、ログインはコンソールから行われ、そのコンソールデバイスに対応する番号は 0 になります。

監査トレール

「監査トレール」は監査デーモンによって作成されます。監査デーモンは、マシンが起動されるとその各マシン上で起動されます。 auditd デーモンは、ブート時に起動されると、監査トレールデータを収集し、監査レコードを「監査ファイル」に書き込む処理を受け持ちます。このファイルを「監査ログファイル」とも呼びます。ファイルの書式については、audit.log(4) のマニュアルページを参照してください。auditd(1M) のマニュアルページも参照してください。

監査ディレクトリは、監査専用でないほかのファイルシステム内に物理的に配置することもできますが、予備のディレクトリを除き、この配置は行わないでください。予備のディレクトリとは、他の適切なディレクトリが使用できないときに限り、監査ファイルが書き込まれるディレクトリです。

監査ディレクトリを監査専用でないファイルシステムに配置してもかまわない場合が、もう 1 つあります。つまり、ソフトウェア開発環境を使用していて、監査がオプションである場合は、そうしてもかまいません。監査トレールを保存するよりも、ディスク容量を有効に使用するほうが重視されるからです。しかし、セキュリティが重視される環境では、監査ディレクトリをほかのファイルシステム内に入れることは許されません。

また、次の要因も考慮する必要があります。

- 各ホストには、少なくとも 1 つのローカル監査ディレクトリを用意する必要があります。このローカルディレクトリは、ホストが監査サーバーと通信できなかった場合の予備ディレクトリとして使用できます。

- 監査ディレクトリは、読み取りオプションと書き込みオプション (rw) を使用してマウントしてください。監査ディレクトリをリモートマウントするときは、intr および noac オプションも使用してください。
- 監査ファイルシステムを、格納先の監査サーバー上で一覧してください。エクスポートリストには、監査サーバーを使用するように構成されたすべてのマシンが含まれます。

監査ファイルの命名規則

各監査ファイルは、それ自体で意味がわかるレコードの集合です。ファイル名には、レコードが生成された時間の範囲と、それを生成したマシン名が含まれます。

監査ファイルの命名

完全な監査ファイルには、次の書式の名前が付いています。

start-time.finish-time.machine

- *start-time* – 監査ファイル内の最初の監査レコードが生成された時刻
- *finish-time* – 最後のレコードがファイルに書き込まれた時刻
- *machine* – ファイルを生成したマシン名

監査ファイル名の例については、397 ページの「閉じられた監査ファイル名の例」を参照してください。

監査ログファイルが動作中である場合は、次の書式の名前が付いています。

start-time`.not_terminated.`*machine*

監査ファイル名の使用方法

auditreduce コマンドは、ファイル名に含まれるタイムスタンプを手掛かりにして、特定期間内のレコードを検索します。1 か月あるいはそれ以上蓄積された監査ファイルがオンライン上に存在する可能性もあるため、これらのタイムスタンプは重要な意味を持ちます。24 時間以内に生成されたレコードをすべてのファイルから検索するとなると、莫大な時間がかかることがあります。

タイムスタンプの書式と説明

start-time と *end-time* は 1 秒単位のタイムスタンプです。これらのタイムスタンプは、グリニッジ標準時 (GMT) で指定されます。タイムスタンプの書式は、次のように年が 4 桁で、2 桁ずつの月、日、時、分、秒があとに続きます。

YYYYMMDDHHMMSS

タイムスタンプには GMT が使用されるため、夏時間によるずれがあっても正しい順序でソートされることが保証されます。また、日時を把握しやすいように現在の時間帯に変換する必要があります。監査ファイルを `auditreduce` コマンドではなく標準ファイルコマンドで操作するときには、この点に注意してください。

動作中のファイル名の例

動作中のファイル名の書式は次のとおりです。

YYYYMMDDHHMMSS`.not_terminated.`*machine*

次に例を示します。

```
19990327225243.not_terminated.dopey
```

監査ログファイルの名前には、開始日が使用されます。上記の例の監査ファイルは、GMT の1990 年 3 月 27 日午後 10:52:43 に作成されています。ファイル名のうち `not_terminated` は、このファイルがまだ動作中であるか、または `auditd` デーモンが予期しない割り込みを行なったことを意味します。末尾の名前 `dopey` は、監査データが収集されているマシンのホスト名です。

閉じられた監査ファイル名の例

閉じられた監査ログファイル名の書式は次のとおりです。

YYYYMMDDHHMMSS.YYYYMMDDHHMMSS.hostname

次に例を示します。

```
19990320005243.19900327225351.dopey
```

この例の監査ログファイルは、GMT の 1999 年 3 月 20 日の午前 12:52:43 に作成されています。このファイルは、GMT の 3 月 27 日午後 10:53:51 に閉じられました。末尾の名前 `dopey` は、監査データが収集されたマシンのホスト名です。

`auditd` が予期しない割り込みを行うと、その時点で開いている監査ファイル名には `not_terminated` が付きます。たとえば、マシンがリモートマウントされた監査ファイルに書き込んでいるときに、ファイルサーバーにアクセスできなくなることがあります。マウントされた監査ファイルにアクセスできなくなると、`not_terminated` 指定がファイル名に付いたままになります。サービスが復旧すると、監査デーモンは、古い監査ファイルの名前はそのままにして、新しい監査ファイルを開きます。

監査レコードの構造

監査レコードは、一連の監査トークンです。監査トークンには、ユーザー ID、時刻、日付などのイベント情報が入っています。監査レコードは、header トークンで始まり、オプションの trailer トークンで終わります。ほかの監査トークンには、監査可能なイベントに関連する情報が入っています。次の図は、標準的な監査レコードを示しています。

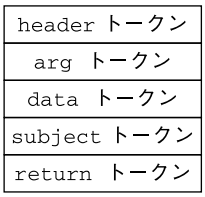


図 **23–3** 標準的な監査レコードの構造

監査トークンの形式

各トークンにはトークンの種類識別子とそのあとにトークン固有のデータが続いています。各トークンの種類には固有の形式があります。次の表は、各トークンの名前と説明の一覧です。

表 **23–4** 基本セキュリティモジュール (BSM) の監査トークン

トークン名	説明	参照先
acl	アクセス制御リスト情報	399 ページの「acl トークン」
arbitrary	書式情報と型情報が付いたデータ	400 ページの「arbitrary トークン」
arg	システムコールの引数値	401 ページの「arg トークン」
attr	ファイル vnode トークン	401 ページの「attr トークン」
exec_args	exec システムコールの引数	402 ページの「exec_args トークン」
exec_env	exec システムコールの環境変数	403 ページの「exec_env トークン」
exit	プログラム終了情報	403 ページの「exit トークン」
file	監査ファイル情報	404 ページの「file トークン」

表 **23–4** 基本セキュリティモジュール (BSM) の監査トークン　(続き)

トークン名	説明	参照先
groups	プロセスグループ情報	404 ページの「group トークン (現在は使用しない)」
header	監査レコードの始まりを示す	405 ページの「header トークン」
in_addr	インターネットアドレス	406 ページの「in_addr トークン」
ip	IP ヘッダー情報	406 ページの「ip トークン (現在は使用しない)」
ipc	System V IPC 情報	407 ページの「ipc トークン」
ipc_perm	System V IPC オブジェクトトークン	408 ページの「ipc_perm トークン」
iport	インターネットポートアドレス	408 ページの「iport トークン」
newgroups	プロセスグループ情報	409 ページの「newgroups トークン」
opaque	構造化されていないデータ (形式が未指定)	409 ページの「opaque トークン」
path	パス情報	410 ページの「path トークン」
process	プロセスのトークン情報	410 ページの「process トークン」
return	システムコールの状態	412 ページの「return トークン」
seq	シーケンス番号トークン	412 ページの「seq トークン」
socket	ソケットの種類とアドレス	413 ページの「socket トークン」
subject	サブジェクトのトークン情報 (process トークンと同じ書式)	413 ページの「subject トークン」
text	ASCII 文字列	415 ページの「text トークン」
trailer	監査レコードの終わりを示す	415 ページの「trailer トークン」

監査レコードには、必ず header トークンが入っています。header トークンは、監査トレール内で監査レコードの始まりを示します。ユーザーの動作に起因しないイベントからの監査レコードを除き、どの監査レコードにも subject トークンが入っています。ユーザーに起因するイベントの場合、この 2 つのトークンはイベントを引き起こしたプロセスの値を参照します。非同期イベントの場合、process トークンはシステムを参照します。

acl トークン

acl トークンには、アクセス制御リストに関する情報を記録します。次の 4 つの固定フィールドがあります。

- acl トークンであることを特定するトークン ID
- ACL タイプを指定するフィールド

- ACL ID フィールド
- ACL に関連付けるアクセス権を一覧するフィールド

praudit コマンドでは、acl トークンは次のように表示されます。

```
acl,tpanero,staff,0755
```

次の図に acl トークンの形式を示します。

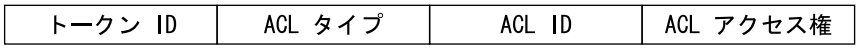


図 **23–4** acl トークンの形式

arbitrary トークン

arbitrary トークンは、監査トレール用にデータをカプセル化します。4 つの固定長フィールドと 1 つのデータ配列からなっています。固定長フィールドは次のとおりです。

- arbitrary トークンであることを特定するトークン ID
- 推奨される形式フィールド (16 進など)
- カプセル化するデータのサイズを指定する (短い形式など) サイズフィールド
- 後続の項目数を指定するカウントフィールド

トークンの残りの部分は、指定された形式の 1 つまたは複数の項目からなっています。praudit コマンドでは、arbitrary トークンは次のように表示されます。

```
arbitrary,decimal,int,1
42
```

次の図に arbitrary トークンの形式を示します。

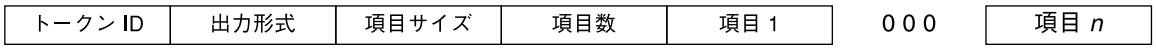


図 **23–5** arbitrary トークンの形式

次の表は、出力形式フィールドに指定できる値を示します。

表 **23–5** arbitrary トークンの出力形式フィールドの値

値	動作
AUP_BINARY	日付が 2 進形式で出力される
AUP_OCTAL	日付が 8 進形式で出力される

表 23–5 arbitrary トークンの出力形式フィールドの値　(続き)

値	動作
AUP_DECIMAL	日付が 10 進形式で出力される
AUP_HEX	日付が 16 進形式で出力される
AUP_STRING	日付が文字列で出力される

次の表は、項目サイズフィールドに指定できる値を示します。

表 23–6 arbitrary トークンの項目サイズフィールドの値

値	動作
AUR_BYTE	データはバイト単位 (1 バイト)
AUR_SHORT	データは短い形式の単位 (2 バイト)
AUR_LONG	データは長い形式の単位 (4 バイト)

arg トークン

arg トークンには、システムコールの引数情報 (システムコールの引数番号、引数の値、およびオプションの説明) が含まれています。このトークンを使用すると、監査レコード内で 32 ビット整数のシステムコール引数を指定できます。次の 5 つのフィールドがあります。

- argトークンであることを特定するトークン ID
- トークンが参照するシステムコールの引数の ID
- 引数の値
- テキスト文字列の長さ
- テキスト文字列

praudit コマンドでは、arg トークンは次のように表示されます。

```
argument,1,0x00000000,addr
```

arg トークンの形式を示します。

トークン ID	引数番号	引数の値	テキスト長	テキスト

図 23–6 arg トークンの形式

attr トークン

attr トークンには、ファイル v ノードからの情報が含まれています。次のトークンには 7 つのフィールドがあります。

- attr トークンであることを特定するトークン ID
- ファイルのアクセスモードと種類
- 所有者のユーザー ID
- 所有者のグループ ID
- ファイルシステム ID
- i ノード ID
- ファイルが示すデバイス ID

ファイルシステム ID とデバイス ID の詳細は、statvfs(2) のマニュアルページを参照してください。

attr トークンには通常、path トークンが付いています。attr トークンはパスの検索中に生成されます。パス検索エラーが発生すると、必要なファイル情報を取得するための v ノードが利用できません。このため、attr トークンは監査レコードの一部として組み込まれません。praudit コマンドでは、attr トークンは次のように表示されます。

```
attribute,100555,root,staff,1805,13871,-4288
```

attr トークンの形式を示します。

図 **23–7** attr トークンの形式

exec_args トークン

exec_args トークンは、 exec() システムコールへの引数を記録します。次の 2 つの固定フィールドがあります。

- exec_args トークンであることを特定するトークン ID
- exec() システムコールに渡す引数の数を表すカウント

このトークンの残りの部分は、0 個以上の NULL で終わる文字列からなっています。praudit コマンドでは、 exec_args トークンは次のように表示されます。

```
vi,/etc/security/audit_user
```

exec_args トークンの形式を示します。

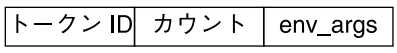


図 **23–8** exec_args トークンの形式

注 – exec_args トークンは、監査ポリシー argv が有効なときにだけ出力されます。

exec_env トークン

exec_envトークンは、exec() システムコールの現在の環境変数を記録します。次の 2 つの固定フィールドがあります。

- exec_env トークンであることを特定するトークン ID
- exec() システムコールに渡す引数の数を表すカウント

このトークンの残りの部分は、0 個以上の NULL で終わる文字列からなっています。praudit コマンドでは、exec_env トークンは次のように表示されます。

```
exec_env,25,
GROUP=staff,HOME=/export/home/matrix,HOST=mestrix,HOSTTYPE=sun4u,HZ=100,
LC_COLLATE=en_US.ISO8859-1,LC_CTYPE=en_US.ISO8859-1,LC_MESSAGES=C,
LC_MONETARY=en_US.ISO8859-1,LC_NUMERIC=en_US.ISO8859-1,
LC_TIME=en_US.ISO8859-1,LOGNAME=matrix,MACHTYPE=sparc,
MAIL=/var/mail/matrix,OSTYPE=solaris,PATH=/usr/sbin:/usr/bin,PS1=#,
PWD=/var/audit,REMOTEHOST=192.168.13.5,SHELL=/usr/bin/csh,SHLVL=1,
TERM=dtterm,TZ=US/Pacific,USER=matrix,VENDOR=sun
```

次の図に exec_env トークンの形式を示します。

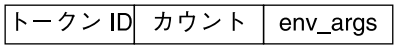


図 **23–9** exec_env トークンの形式

注 – exec_env トークンは、監査ポリシー arge が有効なときにだけ出力されます。

exit トークン

exit トークンは、プログラムの終了状態を記録します。次のフィールドがあります。

- exit トークンであることを特定するトークン ID
- exit() システムコールに渡されるプログラムの終了状態
- 終了状態を記述するか、システムエラー番号を示す戻り値

praudit コマンドでは、exit トークンは次のように表示されます。

```
exit,Error 0,0
```

次の図に exit トークンの形式を示します。

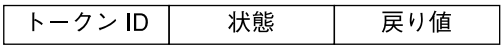


図 **23–10** exit トークンの形式

file トークン

file トークンは、監査デーモンによって生成される特殊なトークンです。このトークンは、古い監査ファイルが終了した時点で、新しい監査ファイルの開始と古い監査ファイルの終了をマークします。監査デーモンは、このトークンを含む特殊な監査レコードを構築して、連続する監査ファイルを 1 つの監査トレールに「リンク」します。次の 4 つのフィールドがあります。

- file トークンであることを特定するトークン ID
- ファイルが作成または閉じた日時を識別する時刻と日付のスタンプ
- NULL 終了文字を含むファイル名のバイト数
- NULL で終了するファイル名を保持するフィールド

praudit コマンドでは、file トークンは次のように表示されます。

```
file,Tue Sep  1 13:32:42 1992, + 79249 msec,
    /var/audit/localhost/files/19990901202558.19990901203241.quisp
```

次の図に file トークンの形式を示します。

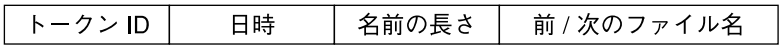


図 **23–11** file トークンの形式

group トークン (現在は使用しない)

このトークンは、newgroups トークンに置き換えられています。newgroups トークンは同じ種類の情報を少ない領域で提供します。ここでは完全を期すために group トークンについて説明しますが、アプリケーション設計者は newgroups トークンを使用してください。praudit の出力では、どちらのトークン ID にも group というラベルが付いているため、praudit がこれら 2 つのトークンを区別しない点に注意してください。

group トークンは、プロセスの資格からグループエントリを記録します。group トークンには次の 2 つの固定長フィールドがあります。

- group トークンであることを特定するトークン ID

- サイズが NGROUPS_MAX (16) のグループエントリの配列

このトークンの残りの部分は 0 個以上のグループエントリからなっています。praudit コマンドでは、group トークンは次のように表示されます。

```
group,staff,admin,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
```

次の図に group トークンの形式を示します。

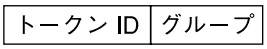


図 **23–12** group トークンの形式

注 – group トークンは、監査ポリシー group が有効なときにだけ出力されます。

---

header トークン

header トークンは、監査レコードの始まりをマークし、trailer トークンとの組み合わせでレコード内のほかのすべてのトークンを囲む特殊なトークンです。次の 6 つのフィールドがあります。

- header トークンであることを特定するトークン ID
- この監査レコード全体の長さのバイト数。ヘッダーとトレーラを含む
- この監査レコード構造体のバージョンを特定するバージョン番号
- このレコードが表す監査イベントの種類を特定する監査イベント ID
- この監査イベントの特殊な特性を特定する ID 修飾子
- レコードの作成日時

64 ビットシステムでは、header トークンは、32 ビットタイムスタンプではなく 64 ビットタイムスタンプで表示されます。

praudit では、ioctl() システムコールの header トークンは次のように表示されます。

```
header,240,1,ioctl(2),es,Tue Sept  1 16:11:44 2001, + 270000 msec
```

次の図に header トークンの形式を示します。

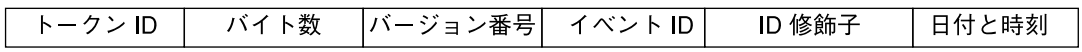


図 **23–13** header トークンの形式

ID 修飾子フィールドでは、次のフラグが定義されています。

```
0x4000          PAD_NOTATTR                     nonattributable event
0x8000          PAD_FAILURE                     fail audit event
```

in_addr トークン

in_addr トークンには、インターネットプロトコルアドレスが含まれます。Solaris 8 リリースから、IPv4 形式、IPv6 形式のいずれかでインターネットアドレスを表示できるようになりました。IPv4 アドレスは 4 バイトを使用します。IPv6 アドレスは、16 バイトを使って種類を記述し、さらに 16 バイトを使用してアドレスを記述します。次の 2 つのフィールドがあります。

- in_addr トークンであることを特定するトークン ID
- インターネットアドレス

praudit コマンドでは、in_addr トークンは次のように表示されます。

```
ip address,129.150.113.7
```

次の図に in_addr トークンの形式を示します。

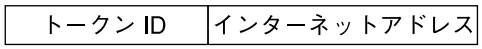


図 **23–14** in_addr トークンの形式

ip トークン (現在は使用しない)

ip トークンには、インターネットプロトコルのヘッダーのコピーが含まれます。次の 2 つのフィールドがあります。

- ip トークンであることを特定するトークン ID
- IP ヘッダーのコピー (全部で 20 バイト)

praudit コマンドでは、ip トークンは次のように表示されます。

```
ip address,0.0.0.0
```

IP ヘッダー構造は、/usr/include/netinet/ip.h ファイル内で定義されています。次の図に ip トークンの形式を示します。

図 **23–15** ip トークンの形式

ipc トークン

ipc トークンには、特定のIPC オブジェクトを識別するために呼び出し元に使用される System V IPC メッセージ、セマフォ、または共有メモリーハンドルが含まれています。次の 3 つのフィールドがあります。

- IPC トークンであることを特定するトークン ID
- IPC オブジェクトの形式を指定する形式フィールド
- IPC オブジェクトを識別するハンドル

praudit コマンドでは、ipc トークンは次のように表示されます。

```
IPC,msg,3
```

注 – IPC オブジェクト識別子は、コンテキストに依存しない Solaris 監査トークンに準拠していません。IPC オブジェクトを一意に識別するグローバルな「名前」はありません。代わりに、IPC オブジェクトはハンドルで識別されます。これらのハンドルは、IPC オブジェクトの動作中にのみ有効です。しかし IPC オブジェクトの識別は問題となりません。System V の IPC メカニズムはあまり使用されず、すべてのメカニズムが同じ監査クラスを共有するからです。

次の表は、IPC オブジェクトの形式フィールドに指定できる値の一覧です。値は /usr/include/bsm/audit.h ファイル内で定義されます。

表 **23–7** IPC オブジェクトの形式フィールドの値

名前	値	説明
AU_IPC_MSG	1	IPC メッセージオブジェクト
AU_IPC_SEM	2	IPC セマフォオブジェクト
AU_IPC_SHM	3	IPC 共有メモリーオブジェクト

次の図に ipc トークンの形式を示します。

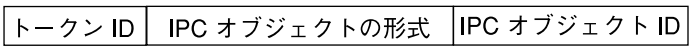


図 **23–16** ipc トークンの形式

ipc_perm トークン

ipc_perm トークンには、System V の IPC アクセス情報が含まれています。このトークンは、IPC 共有メモリーイベント、IPC セマフォイベント、および IPC メッセージイベントによって生成される監査レコードに追加されます。次の 8 つのフィールドがあります。

- ipc_perm トークンであることを特定するトークン ID
- IPC 所有者のユーザー ID
- IPC 所有者のグループ ID
- IPC 作成者のユーザー ID
- IPC 作成者のグループ ID
- IPC のアクセスモード
- IPC のシーケンス番号
- IPC 鍵の値

praudit コマンドでは、ipc_perm トークンは次のように表示されます。

```
IPC perm,root,wheel,root,wheel,0,0,0x00000000
```

値は、IPC オブジェクトに関連付けられた ipc_perm 構造から取り出されます。次の図に ipc_perm トークンの形式を示します。

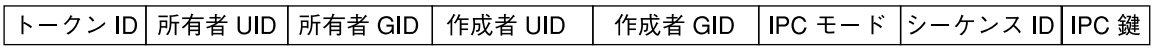


図 **23–17** ipc_perm トークンの形式

iport トークン

iport トークンには、TCP (または UDP) ポートアドレスが含まれています。次の 2 つのフィールドがあります。

- iport トークンであることを特定するトークン ID
- TCP または UDP ポートのアドレス

praudit コマンドでは、iport トークンは次のように表示されます。

```
ip port,0xf6d6
```

次の図に iport トークンの形式を示します。

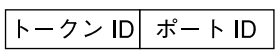


図 **23–18** iport トークンの形式

newgroups トークン

group トークンは、このトークンによって置き換えられます。praudit の出力では、どちらのトークン ID にも group というラベルが付いているため、praudit はこれら 2 つのトークンを区別しない点に注意してください。

newgroups トークンは、プロセスの資格からグループエントリを記録します。次の 2 つの固定長フィールドがあります。

- newgroups トークンであることを特定するトークン ID
- この監査レコードに含まれるグループ数を表すカウント

このトークンの残りの部分は 0 個以上のグループエントリからなっています。praudit コマンドでは、newgroups トークンは次のように表示されます。

```
group, staff, admin
```

次の図に newgroups トークンの形式を示します。

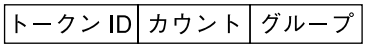


図 **23–19** newgroups トークンの形式

注 **–** newgroups トークンは、監査ポリシー group が有効なときにだけ出出力されます。

---

opaque トークン

opaque トークンには、フォーマットされていないデータが一連のバイトとして含まれています。次の 3 つのフィールドがあります。

- opaque トークンであることを特定するトークン ID
- このデータのバイト数
- バイトデータ配列

praudit コマンドでは、opaque トークンは次のように表示されます。

```
opaque,12,0x4f5041515545204441544100
```

次の図に opaque トークンの形式を示します。

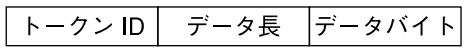


図 **23–20** opaque トークンの形式

path トークン

path トークンには、オブジェクトのアクセスパス情報が含まれています。次のフィールドがあります。

- path トークンであることを特定するトークン ID
- パス長のバイト数
- システムの実ルートを基点としたオブジェクトへの絶対パス

praudit コマンドでは、path トークンは次のように表示されます。パス長フィールドは、表示されません。

```
path,/etc/security/audit_user
```

次の図に path トークンの形式を示します。

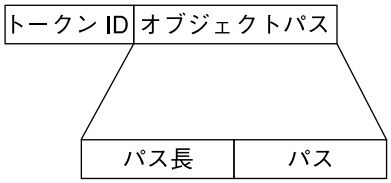


図 **23–21** path トークンの形式

process トークン

process トークンには、信号の受信者など、プロセスに関連するユーザーの情報が含まれています。次の 9 つのフィールドがあります。

- process トークンであることを特定するトークン ID
- 不変監査 ID
- 実効ユーザー ID
- 実効グループ ID
- 実ユーザー ID
- 実グループ ID
- プロセス ID
- 監査セッション ID
- デバイス ID とマシン ID で構成される端末 ID

監査 ID、ユーザー ID、グループ ID、プロセス ID、セッション ID は、短い形式ではなく長い形式です。

注 – セッション ID、実ユーザー ID、または実グループ ID の processトークンのフィールドを使用できないことがあります。その場合、値は -1 に設定されます。

端末 ID を含むトークンには、いくつかの種類があります。praudit コマンドは、これらの違いを吸収します。このため、端末 ID を含むすべてのトークンで、端末 ID は同じ方法で処理されます。端末 ID は、IP アドレスとポート番号の組み合わせか、デバイス ID です。モデムに接続されたシリアルポートなどのデバイス ID は、0 である可能性があります。端末 ID には、次の書式があります。

デバイス番号の場合、端末 ID は次のようになります。

- 「32 ビットアプリケーション」 – 4 バイトのデバイス番号、4 バイトは未使用
- 「64 ビットアプリケーション」 – 8 バイトのデバイス番号、4 バイトは未使用

Solaris 8 よりも前のリリースのポート番号の場合、端末 ID は次のようになります。

- 「32 ビットアプリケーション」 – 4 バイトのポート番号、4 バイトの IP アドレス
- 「64 ビットアプリケーション」 – 8 バイトのポート番号、4 バイトの IP アドレス

Solaris 8 リリースまたは Solaris 9 リリースのポート番号の場合、端末 ID は次のようになります。

- 「32 ビット IPv4」 – 4 バイトのポート番号、4 バイトの IP タイプ、4 バイトの IP アドレス
- 「32 ビット IPv6」 – 4 バイトのポート番号、4 バイトの IP タイプ、16 バイトの IP アドレス
- 「64 ビット IPv4」 – 8 バイトのポート番号、4 バイトの IP タイプ、4 バイトの IP アドレス
- 「64 ビット IPv6」 – 8 バイトのポート番号、4 バイトの IP タイプ、16 バイトの IP アドレス

praudit コマンドでは、process トークンは次のように表示されます。

```
process,root,root,wheel,root,wheel,0,0,0,0.0.0.0
```

次の図に process トークンの形式を示します。

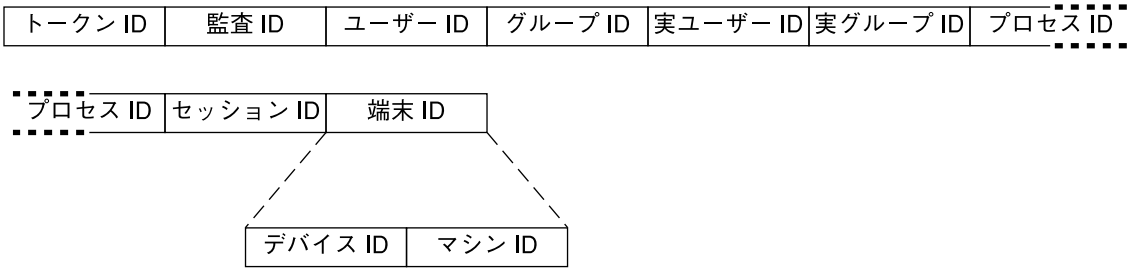


図 **23–22** process トークンの形式

return トークン

return トークンには、システムコールの戻り状態 (u_error) とプロセスの戻り値 (u_rval1) が含まれています。次の 3 つのフィールドがあります。

- return トークンであることを特定するトークン ID
- システムコールのエラー状態
- システムコールの戻り値

return トークンは、必ずシステムコールに関してカーネルによって生成される監査レコードの一部として返されます。このトークンは、アプリケーションを監査中の終了状態と他の戻り値を示します。

praudit コマンドにより return トークンは次のように表示されます。

```
return,success,0
```

次の図に return トークンの形式を示します。

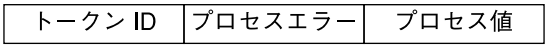


図 **23–23** return トークンの形式

seq トークン

seq トークン (シーケンストークン) は、シーケンス番号が含まれるオプションのトークンです。このトークンは、デバッグに使用されます。seq ポリシーが有効な場合は、このトークンが各監査レコードに追加されます。次の 2 つのフィールドがあります。

- seq トークンであることを特定するトークン ID
- シーケンス番号が含まれる 32 ビットの符号なし長形式フィールド

シーケンス番号は、監査レコードが生成されて監査トレールに組み込まれるたびに 1 ずつ増分します。praudit コマンドでは、seq トークンは次のように表示されます。

```
sequence,1292
```

次の図に seq トークンの形式を示します。

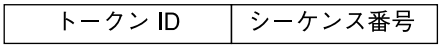


図 **23–24** seq トークンの形式

注 – seq トークンは、監査ポリシー seq が有効なときだけ出力されます。

socket トークン

socket トークンには、インターネットソケットを記述する情報が含まれています。次の 6 つのフィールドがあります。

- socket トークンであることを特定するトークン ID
- 参照するソケットの型 (TCP、UDP、UNIX) を示すソケット形式フィールド
- ローカルポートアドレス
- ローカルのインターネットアドレス
- リモートポートアドレス
- リモートのインターネットアドレス

praudit コマンドでは、socket トークンは次のように表示されます。

```
socket,0x0000,0x0000,0.0.0.0,0x0000,0.0.0.0
```

Solaris 8 リリースから、IPv4 形式、IPv6 形式のいずれかでインターネットアドレスを表示できるようになりました。IPv4 アドレスは 4 バイトを使用します。IPv6 アドレスは、16 バイトを使って種類を記述し、さらに 16 バイトを使ってアドレスを記述します。次の図に socket トークンの形式を示します。

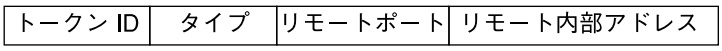


図 **23–25** socket トークンの形式

subject トークン

subject トークンには、操作を実行するユーザーまたは実行する予定のユーザーを記述します。形式は process トークンと同じです。次の 9 つのフィールドがあります。

- subject トークンであることを特定するトークン ID
- 不変監査 ID
- 実効ユーザー ID
- 実効グループ ID
- 実ユーザー ID
- 実グループ ID
- プロセス ID
- 監査セッション ID
- デバイス ID とマシン ID で構成される端末 ID

監査 ID、ユーザー ID、グループ ID、プロセス ID、セッション ID は、短い形式ではなく長い形式です。

注 – セッション ID、実ユーザー ID、または実グループ ID の `subject` トークンのフィールドを使用できないことがあります。その場合、値は -1 に設定されます。

端末 ID を含むトークンには、いくつかの種類があります。`praudit` コマンドは、これらの違いを吸収します。このため、端末 ID を含むすべてのトークンで、端末 ID は同じ方法で処理されます。端末 ID は、IP アドレスとポート番号の組み合わせか、デバイス ID です。モデムに接続されたシリアルポートなどのデバイス ID は、0 である可能性があります。端末 ID には、次の書式があります。

デバイス番号の場合、端末 ID は次のようになります。

- 「32 ビットアプリケーション」 – 4 バイトのデバイス番号、4 バイトは未使用
- 「64 ビットアプリケーション」 – 8 バイトのデバイス番号、4 バイトは未使用

Solaris 8 よりも前のリリースのポート番号の場合、端末 ID は次のようになります。

- 「32 ビットアプリケーション」 – 4 バイトのポート番号、4 バイトの IP アドレス
- 「64 ビットアプリケーション」 – 8 バイトのポート番号、4 バイトの IP アドレス

Solaris 8 リリースまたは Solaris 9 リリースのポート番号の場合、端末 ID は次のようになります。

- 「32 ビット IPv4」 – 4 バイトのポート番号、4 バイトの IP タイプ、4 バイトの IP アドレス
- 「32 ビット IPv6」 – 4 バイトのポート番号、4 バイトの IP タイプ、16 バイトの IP アドレス
- 「64 ビット IPv4」 – 8 バイトのポート番号、4 バイトの IP タイプ、4 バイトの IP アドレス
- 「64 ビット IPv6」 – 8 バイトのポート番号、4 バイトの IP タイプ、16 バイトの IP アドレス

`subject`トークンは、必ずシステムコールに関してカーネルによって生成される監査レコードの一部として返されます。`praudit` コマンドでは、`subject` トークンは次のように表示されます。

```
subject,cjc,cjc,staff,cjc,staff,424,223,0 0 quisp
```

次の図に `subject` トークンの形式を示します。

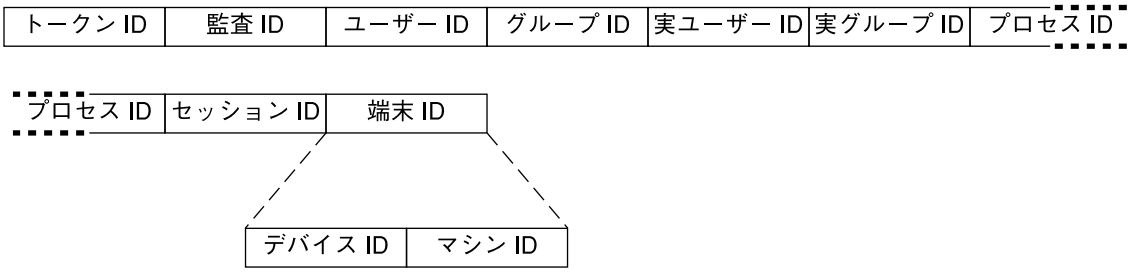


図 **23–26** `subject` トークンの形式

`text` トークン

`text` トークンにはテキスト文字列が含まれています。次の 3 つのフィールドがあります。

- `text` トークンであることを特定するトークン ID
- テキスト文字列の長さ
- テキスト文字列

`praudit` コマンドでは、`text` トークンは次のように表示されます。

```
text,aw_test_token
```

次の図に `text` トークンの形式を示します。

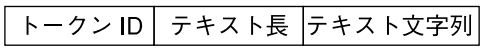


図 **23–27** `text` トークンの形式

`trailer` トークン

`header` と `trailer` の 2 つのトークンは、監査レコードの終端を区別し、他のすべてのトークンを囲む特殊なトークンです。`header` トークンは監査レコードを開始します。 `trailer` トークンは監査レコードを終了します。 `trailer` トークンは省略可能です。`trailer` トークンは、`trail` 監査ポリシーが設定されているときにだけ、各レコードの最後のトークンとして追加されます。

trailer トークンを含む監査レコードが生成された場合、`auditreduce` コマンドは、trailer がレコードの header を正しくポイントしているかどうかを確認します。また、`trailer` トークンを使用すると監査トレールを逆方向に検索できます。

3 つのフィールドがあります。

- trailer トークンであることを特定するトークン ID
- レコードの終了を示すパッド番号
- header トークンと trailerトークンを含む監査レコードの合計文字数

praudit コマンドでは、trailer トークンは次のように表示されます。

```
trailer,136
```

次の図に trailer トークンの形式を示します。

トークン ID	パッド番号	バイト数

図 **23–28** trailer トークンの形式

デバイス割り当て参照

デバイス割り当てを行うことによって、承認されていないユーザーがリムーバブルメディアを使用できないようにします。ユーザーへのデバイス割り当てを要求することができます。デバイスを使用する権限を拒否することができます。そのような割り当てによって、データの損失、コンピュータウィルスなどのセキュリティ違反からサイトを保護できます。次の節では、デバイス割り当てについて説明します。

デバイス割り当てメカニズムの構成要素

デバイス割り当てメカニズムは、次の要素で構成されます。

- allocate、deallocate、dminfo、list_devices コマンド。詳細については、417 ページの「デバイス割り当てコマンドの使用方法」を参照してください。
- /etc/security/device_allocate ファイル。詳細については、device_allocate(4) のマニュアルページを参照してください。
- /etc/security/device_maps ファイル。詳細については、 device_maps(4) のマニュアルページを参照してください。
- ロックファイル。割り当て可能デバイスごとに /etc/security/dev ディレクトリに配置する
- 各割り当て可能デバイスに関連付けられた「デバイス特殊ファイル」の変更後の属性
- 各割り当て可能デバイスのデバイスクリーンスクリプト

device_allocate ファイル、device_maps ファイル、およびロックファイルは、ローカル構成ファイルです。これらのファイルは、ネームサービスデータベースとして管理されません。テープドライブ、フロッピーディスクドライブ、およびプリンタはすべて、特定のマシンに接続されるためです。

デバイス割り当てコマンドの使用方法

この節では、管理者向けのコマンド allocate、deallocate、および list_devices のオプションのいくつかについて説明します。これらのオプションにアクセスできるのは、root またはそれと同等の役割のみです。各コマンドについての詳細は、それぞれのマニュアルページを参照してください。

表 **23–8** デバイス割り当てコマンドの管理オプション

コマンドとオプション	説明
allocate -F *device_special_filename*	指定するデバイスを再度割り当てる。通常は、このオプションに -U オプションを付けて、指定するデバイスを指定するユーザーに再度割り当てる。-U オプションを指定しない場合、デバイスは root に割り当てられる
allocate -U *username*	デバイスを、現在のユーザーではなく指定するユーザーに割り当てる。このオプションを使用すると、ユーザーの識別情報がなくても、デバイスを別のユーザーに割り当てることができる。
deallocate -F *device_special_filename*	デバイス割り当てを強制的に解除する。ユーザーに割り当てられたデバイスは、プロセスが終了するとき、またはそのユーザーがログアウトするときに、自動的に割り当てが解除される。ユーザーがテープドライブの割り当てを解除し忘れたときには、-F オプションを使って割り当てを強制的に解除できる
deallocate -I	割り当て可能なすべてのデバイス割り当てを強制的に解除する。このオプションは、システムの初期化時にのみ使用する
list_devices	device_maps ファイルにリストされたデバイスに関連付けられている、デバイス特殊ファイルを一覧する
list_devices -U *username*	指定したユーザー名に関連付けられたユーザー ID に割り当て可能なデバイスまたは割り当て済みのデバイスを一覧する。このオプションを使用すると、別のユーザーに割り当て可能なデバイスまたは割り当て済みのデバイスを確認できる

割り当てエラー状態

割り当て可能デバイスは、デバイス特殊ファイルのモードが 0100 でかつユーザー bin とグループ bin に所有されている場合に、「割り当てエラー状態」になります。割り当てエラー状態になっているデバイスを割り当てたい場合は、そのデバイスの割り当ての強制解除を試みます。-F オプションを指定して deallocate コマンドを実行すると、割り当てが強制的に解除されます。または、allocate -U を使用して、そのデバイスをユーザーに割り当てます。いったんデバイスが割り当てられると、発生したエラーメッセージを調査できます。デバイスの問題が解決したら、強制オプション -F を使用して、デバイスの割り当てエラー状態をクリアする必要があります。

device_maps ファイル

/etc/security/device_maps ファイルを調べると、各割り当て可能デバイスに関連付けられたデバイス名、デバイスの種類、デバイス特殊ファイルを判断できます。device_maps(4) のマニュアルページを参照してください。デバイスマップは、デバイス割り当てを設定したときに作成されます。device_maps の初期ファイルは、BSM を有効にしたときに、bsmconv によって作成されます。この初期 device_maps ファイルは、あくまでも開始点として使用する必要があります。device_maps は、使用する環境に合わせて拡張およびカスタマイズできます。

device_maps ファイルでは、デバイスごとにデバイス特殊ファイルの割り当てが定義されます。多くの場合、この割り当ては単純ではありません。このファイルによって、各種のプログラムはどのデバイス特殊ファイルがどのデバイスに割り当てられているかを検出できます。たとえば、dminfo コマンドを使用すると、デバイス名、デバイスの種類およびデバイス特殊ファイルを取得して、割り当て可能なデバイスを設定するときに指定できます。dminfo コマンドは、device_maps ファイルを使用してデバイス割り当て情報を報告します。

各デバイスは、次の形式の 1 行のエントリで表されます。

device-name:*device-type*:*device-list*

エントリを次の行に続けるには、行末にバックスラッシュ (\) を付けます。コメントも挿入できます。# を付けると、それに続くすべてのテキストは、行末にバックスラッシュ (\) のない次の改行までコメントになります。どのフィールドでも先行ブランクと後続ブランクを使用できます。

表 23–9 device_maps エントリ内のフィールドの説明

フィールド	説明
device-name	st0、fd0、または audio などのデバイス名を指定する。ここで指定するデバイス名は、/etc/security/dev ディレクトリ内で使用されるロックファイル名と対応している必要がある

表 **23–9** device_maps エントリ内のフィールドの説明　*(続き)*

フィールド	説明
device-type	汎用デバイスタイプを指定する。汎用名は、st、fd、audio などのデバイスクラス名である。*device-type* では、関連するデバイスが論理的にグループ化される
device-list	物理デバイスに関連付けられたデバイス特殊ファイルの一覧。*device-list* には、特定のデバイスにアクセスできるすべての特殊ファイルが含まれている必要がある。リストが不完全な場合は、悪意を持ったユーザーでも個人情報を入手または変更できる。*device-list* フィールドには、/devices 内または/dev 内のシンボリックリンクに置かれた実デバイスファイルを入力する。/dev ディレクトリ内のシンボルリンクは、バイナリ互換性を持つ

次の例では、SCSI テープ st0 とフロッピーディスク fd0 の device_maps ファイルエントリを示します。

```
fd0:\
    fd:\
    /dev/fd0 /dev/fd0a /dev/fd0b /dev/rfd0 /dev/rfd0a /dev/rfd0b:\
                        .
                        .
                        .
st0:\
    st:\
    /dev/rst0 /dev/rst8 /dev/rst16 /dev/nrst0 /dev/nrst8 /dev/nrst16:\
```

device_allocate ファイル

device_allocate ファイルを変更して、デバイスを割り当て可能から割り当て不可に変更したり、新しいデバイスを追加したりします。device_allocate ファイルの例を次に示します。

```
st0;st;;;;/etc/security/lib/st_clean
fd0;fd;;;;/etc/security/lib/fd_clean
sr0;sr;;;;/etc/security/lib/sr_clean
audio;audio;;;*;/etc/security/lib/audio_clean
```

割り当て可能にするデバイスは、BSM を初期構成するときに定義します。上述の device_allocate ファイルの例のように、デフォルトのデバイスとそれらに定義されている特性をそのまま使用することもできます。システム稼働後の実行中にマシンにデバイスを追加するときには、新しいデバイスを割り当て可能にするかどうかを決定する必要があります。

BSM をインストールしたあとで、デバイスのエントリは device_allocate ファイルを変更できます。割り当てたいデバイスは、使用する前に各マシン上の device_allocate ファイル内で定義する必要があります。現在、カートリッジテープドライブ、フロッピーディスクドライブ、CD-ROM デバイス、およびオーディオチップが、割り当て可能と見なされます。これらのデバイスタイプには、デバイスクリーンスクリプトが用意されています。

注 – Xylogics™ テープドライブまたは Archive テープドライブでは、SCSI デバイス用に用意されている st_clean スクリプトが使用できます。モデム、端末、グラフィックスタブレットなどの割り当て可能デバイスについては、独自のデバイスクリーンスクリプトを作成する必要があります。このスクリプトは、対応するデバイスタイプのオブジェクト再使用の要件を満たしている必要があります。

device_allocate ファイル内のエントリは、デバイスが割り当て可能であると特に記述されていない限り、そのデバイスが割り当て可能であることを説明しません。上述の device_allocate ファイルの例では、オーディオデバイスエントリの第 5 フィールドにアスタリスク (*) が指定されています。第 5 フィールド内のアスタリスクは、そのデバイスが割り当て可能でないことをシステムに示します。つまり、システム管理者はユーザーにデバイスを使用する前に割り当てたり、あとで割り当てを解除するように要求したりする必要がありません。このフィールドに他の文字列が入っている場合は、デバイスが割り当て可能であることを示します。

各デバイスは、次の形式の 1 行のエントリで表します。

device-name;*device-type*;reserved;reserved;*alloc*;*device-clean*

たとえば、次の行はデバイス名 st0 のエントリを示しています。

```
st0;st;;;;;/etc/security/lib/st_clean
```

エントリを次の行に続けるには、行末にバックスラッシュ (\) を付けます。コメントも挿入できます。# を付けると、それに続くすべてのテキストは、行末にバックスラッシュ (\) のない次の改行までコメントになります。どのフィールドでも先行ブランクと後続ブランクを使用できます。

次の表では、device_allocate ファイル内の各フィールドについて詳しく説明します。

表 23–10 device_allocate エントリ内のフィールドの説明

フィールド	説明
device-name	st0、fd0、sr0 などのデバイス名を指定する。デバイスを割り当て可能にした場合、device_maps ファイルの *device-name* フィールドから *device-name* を取得する。dminfo コマンドも使用できる。この名前はデバイスの DAC ファイル名でもある点に注意
device-type	汎用デバイスタイプを指定する。汎用名は、st、fd、sr などのデバイスクラス名である。このフィールドによって、関連するデバイスがグループ化される。割り当て可能デバイスを作成するときには、device_maps ファイル内の *device-type* フィールドから *device-type* を取得するか、または dminfo コマンドを使用する
reserved	reserved で示される 2 つのフィールドは、将来の使用に予約されている

表 **23–10** device_allocate エントリ内のフィールドの説明　*(続き)*

フィールド	説明
alloc	デバイスが割り当て可能かどうかを指定する。このフィールドにアスタリスク (*) が入っている場合は、デバイスが割り当て不可能であることを示す。このフィールドに他の文字列が入っている場合や、空の場合は、デバイスが割り当て可能であることを示す
device-clean	割り当てプロセス中にクリーンアップやオブジェクト再使用防止などの特殊処理のために呼び出されるスクリプトのパス名を指定する。deallocate - F を使用してデバイスの割り当てを強制的に解除するときなど、デバイスに対して deallocate コマンドを実行すると、*device-clean* スクリプトが実行される

デバイスクリーンスクリプト

「デバイスクリーン」スクリプトは、使用可能なすべてのデータを再使用する前に物理デバイスからパージするというセキュリティ要件に対応するものです。デフォルトでは、カートリッジテープドライブ、フロッピーディスクドライブ、CD-ROM デバイス、オーディオデバイスには、必要なデバイスクリーンスクリプトが用意されています。この節では、デバイスクリーンスクリプトが実行する処理について説明します。

オブジェクトの再使用

デバイス割り当てによって、オブジェクト再使用の要件の一部が満たされます。デバイスクリーンスクリプトによって、あるユーザーがデバイスに残したデータは確実にクリアされます。データのクリアは、そのデバイスが別のユーザーによって割り当て可能になる前に実行されます。

テープ用のデバイスクリーンスクリプト

st_clean デバイスクリーンスクリプトでは、3 つのテープデバイスがサポートされます。サポートされるテープデバイスは次のとおりです。

- SCSI 1/4 インチテープ
- Archive 1/4 インチテープ
- オープンリール 1/2 インチテープ

st_cleanスクリプトでは、mt コマンドの rewoffl オプションを使用して、デバイスのクリーンアップを制御します。詳細は、mt(1) のマニュアルページを参照してください。このスクリプトは、システムブート中に実行されると、デバイスを照会します。スクリプトは、そのデバイスがオンラインになっているかどうかを調べます。デバイスがオンラインになっていた場合、スクリプトはさらに、そのデバイスにメディアが挿入されているかどうかを調べます。1/4 インチのテープデバイスにメディアが挿入されていた場合、このデバイスは割り当てエラー状態になります。このため、管理者はそのデバイスを手動でクリーンアップする必要があります。

通常のシステム操作中に、allocate または deallocate コマンドを対話型モードで実行すると、メディアを取り出すように求めるプロンプトが表示されます。スクリプトは、メディアがデバイスから取り出されるまで一時停止します。

フロッピーディスクと *CD-ROM* 用のデバイスクリーンスクリプト

次の表に、フロッピーディスクとCD-ROM 用のデバイスクリーンスクリプトを示します。

表 23–11 フロッピーディスクと CD-ROM 用のデバイスクリーンスクリプト

ディスクデバイスの種類	デバイスクリーンスクリプト
フロッピーディスク	fd_clean
CD-ROM	sr_clean

スクリプトは、eject コマンドを使用してドライブからメディアを取り出します。詳細については、eject(1) のマニュアルページを参照してください。eject コマンドが失敗すると、デバイスは割り当てエラー状態になります。

オーディオ用のデバイスクリーンスクリプト

オーディオデバイスは、audio-clean スクリプトを使用してクリーンアップします。スクリプトは、AUDIO_DRAIN ioctl システムコールを実行してデバイスをフラッシュしてから、AUDIO_SETINFO ioctl システムコールを実行してデバイス構成をデフォルトにリセットします。また、スクリプトは AUDIOGETREG ioctl システムコールを使用して、オーディオチップレジスタを検出します。レジスタのデフォルト設定にリセットする場合は、AUDIOSETREG ioctl システムコールを使用します。

新しいデバイスクリーンスクリプトの作成

システムに新しく割り当て可能デバイスを追加する場合は、独自のデバイスクリーンスクリプトを作成する必要があります。deallocate コマンドは、デバイスクリーンスクリプトにパラメータを渡します。次のように、パラメータはデバイス名を含む文字列です。詳細は、device_allocate(4)のマニュアルページを参照してください。

```
st_clean -[I|F|S] device-name
```

デバイスクリーンスクリプトは、正常終了した場合は 0 を、失敗した場合は 1 以上の値を返します。オプション -I、-F、-S を使用すると、スクリプトに実行モードを判断させることができます。次の表は、オプションの一覧です。

表 **23–12** デバイスクリーンスクリプトのオプション

オプション	説明
-I	-I オプションは、システムをブートするときにだけ指定する。すべての出力は、システムコンソールに送られる。失敗した場合や、メディアを強制的に取り出せない場合は、デバイスを割り当てエラー状態にする
-F	-F オプションは、強制クリーンアップを行うときに指定する。このオプションは対話型である。このオプションは、ユーザーがプロンプトに応答するものと見なす。このオプションが付いたスクリプトは、クリーンアップの一部に失敗した場合に、クリーンアップ全体を完了しようとする
-S	-S オプションは、標準クリーンアップを行うときに指定する。このオプションは対話型である。このオプションは、ユーザーがプロンプトに応答するものと見なす。

デバイス割り当てメカニズムの機能

この節では、デバイス割り当てメカニズムの機能について例を挙げて説明します。

allocate コマンドは、まず/etc/security/dev ディレクトリ内で、指定されたデバイスのデバイス名が付いたロックファイルがあるかどうかを確認します。ファイルが allocate によって所有されている場合は、ロックファイルの所有権が allocate コマンドを起動したユーザー名に変更されます。

次に、allocate コマンドは、device_allocate ファイル内にそのデバイスのエントリが存在するかどうかを確認します。このコマンドはさらに、デバイスが割り当て可能に設定されているかどうかを、そのエントリ内で確認します。

次の例の最初の一覧は、/etc/security/dev 内に、st0 デバイスを所有者 bin、グループ bin、モード 600 で使用するロックファイルがあることを示しています。2 つ目の一覧は、それに関連するデバイス特殊ファイルが正しく設定されていて、所有者は bin、グループは bin、モードは 000: であることを示します。

```
untouchable% ls -lg /etc/security/dev/st0
-rw------- 1 bin bin                    0 Dec 6 15:21 /etc/security/dev/st0
untouchable% ls -lg /devices/sbus@1,f8000000/esp@0,800000
c--------- 1 bin bin               18,  4 May 12 13:11 st@4,0:
c--------- 1 bin bin           18, 20 May 12 13:11 st@4,0:b
c--------- 1 bin bin           18, 28 May 12 13:11 st@4,0:bn
c--------- 1 bin bin           18, 12 May 12 13:11 st@4,0:c
                     .
                     .
                     .
c--------- 1 bin bin           18,  0 May 12 13:11 st@4,0:u
c--------- 1 bin bin           18, 16 May 12 13:11 st@4,0:ub
c--------- 1 bin bin           18, 24 May 12 13:11 st@4,0:ubn
c--------- 1 bin bin           18,  8 May 12 13:11 st@4,0:un
```

次の例では、ユーザー vanessa がデバイス st0 を割り当てます。

```
untouchable% whoami
vanessa
untouchable% allocate st0
```

ユーザー vanessa が allocate コマンドを実行してテープ st0 を割り当てると、allocate はまず /etc/security/dev/st0 ファイルが存在することを確認します。ロックファイルが存在しない場合や、allocate 以外のユーザーに所有されている場合は、ユーザー vanessa はデバイスを割り当てることができません。

allocate コマンドは、正しい所有権とアクセス権が設定されたロックファイルを見つけると、次に、device_allocate ファイル内にそのデバイスのエントリが存在するかどうかを確認します。このコマンドはまた、デバイスが割り当て可能として指定されているかどうかを、そのエントリ内で確認します。

この例では、st0 デバイスのデフォルトの device_allocate エントリでは、デバイスが割り当て可能として指定されています。allocate コマンドではこれらの条件がすべて満たされていることを認識し、デバイスが vanessa に割り当てられます。

allocate コマンドは、/dev ディレクトリ内でデバイスに関連付けられたデバイス特殊ファイルの所有権とアクセス権を変更します。st0 デバイスをユーザー vanessa に割り当てるために、それに関連付けられたデバイス特殊ファイルのモードを 600 に変更し、所有者を vanessa に変更します。

また、allocate コマンドは、/etc/security/dev ディレクトリ内でデバイスに関連付けられたロックファイルの所有権も変更します。st0 デバイスをユーザー vanessa に割り当てるために、/etc/security/dev/st0 の所有者を vanessa に変更します。

次の例では、ユーザー vanessa が デバイス名 st0 を指定して allocate コマンドを実行すると、/etc/security/dev/st0 の所有者が vanessa に変更され、デバイス特殊ファイルに関連付けられた所有者も vanessa に変更されます。最後に、ユーザー vanessa にファイルの読み取り権と書き込み権が与えられます。

```
untouchable% whoami
vanessa
untouchable% allocate st0
untouchable% ls -lg /etc/security/dev/st0
-rw------- 1 vanessa staff               0 Dec 6 15:21 /etc/security/dev/st0
untouchable% ls -la /devices/sbus@1,f8000000/esp@0,800000
.
.
.
crw------- 1 vanessa 18,  4 May 12 13:11 st@4,0:
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:b
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:bn
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:c
.
.
.
crw------- 1 vanessa 18,  4 May 12 13:11 st@4,0:u
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:ub
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:ubn
```

```
crw------- 1 vanessa 18, 12 May 12 13:11 st@4,0:un
```

付録 **A**

『Solaris のシステム管理 (セキュリティサービス)』の更新情報

以下の各節では、このマニュアルで説明している Solaris 9 オペレーティング環境の更新機能について述べます。

Solaris 9 12/02 の更新情報

- 新しいパスワード暗号化モジュールについては、30 ページの「ログイン制御の管理」を参照してください。

  暗号化パスワードのモジュールを指定する方法については、54 ページの「パスワード暗号化のデフォルトアルゴリズムを変更する」を参照してください。
- パスワードポリシーが Sun ONE Directory Server に適用されます。この変更については、31 ページの「パスワード情報の管理」を参照してください。
- PAM の新しいバインディング制御フラグについては、185 ページの「PAM 制御フラグ」を参照してください。

Solaris 9 8/03 の更新情報

- 公開ファイルの読み取り専用イベントが監査されなくなりました。`auditconfig` コマンドに `-public` オプションを指定して使うと、公開ファイルを監査対象とするかどうかを制御できます。公開ファイルを監査しないようにすれば、監査トレールの量を大幅に減らすことができます。このため、機密性の高いファイルの読み取りイベントが監視しやすくなります。343 ページの「BSM の用語」を参照してください。

- praudit コマンドで、新しい出力形式 (XML) が使用できるようになりました。XML 形式の出力は、ブラウザを使用して表示できるほか、報告用に XML スクリプトへの入力としても使えます。praudit コマンドの -x オプションについては、383 ページの「praudit コマンド」を参照してください。
- 一連のデフォルトの監査クラスが整理されました。監査メタクラスによって、監査クラスをより微細にグループ化することができます。392 ページの「監査フラグの定義」を参照してください。
- bsmconv コマンドを使用しても、Stop-A キーの使用が無効化されなくなりました。セキュリティを確保するために、Stop-A イベントが監査されるようになりました。

用語集

admin 主体
: *username*/admin という形式 (joe/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) を持つことができる。「主体名」と「ユーザー主体」も参照

FQDN
: 完全指定形式のドメイン名。denver.mtn.example.com など (単なる denver 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サービス (SEAM など) をサポートするネットワーク層。GSS-API は、セキュリティ認証、完全性、およびプライバシサービスを提供する。「認証」、「完全性」、「プライバシ」も参照

KDC
: 鍵配布センター (Key Distribution Center)。次の 3 つの Kerberos V5 要素で構成されるマシン

  - 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、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る

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

  SEAM は、Kerberos V5 に準拠した認証の実装である

  「SEAM」と「Kerberos」は技術的には異なるが、SEAM のマニュアルでは同じ意味で使用されることが多い。「Kerberos」と「Kerberos V5」も同様である

  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

kvno 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。最も大きい kvno が、最新の鍵を示す

NTP ネットワークタイムプロトコル (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する。NTP を使用して、Kerberos 環境のクロックスキューを管理できる。「クロックスキュー」も参照

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ない。PAM は、ログイン時に SEAM セッションを初期化できる

QOP 保護の質 (Quality of Protection)。パラメータの 1 つ。完全性サービスまたはプライバシサービスで使用する暗号化アルゴリズムを選択するときに使用される

RBAC 役割によるアクセス制御。すべての機能を持つスーパーユーザーの代替アカウント。組織は、RBAC を使用してスーパーユーザーの機能を分割し、それらを役割と呼ばれる特別なユーザーアカウントに割り当てる。ジョブの要件に基づいて、特定のユーザーに役割を割り当てることができる

SEAM Sun Enterprise Authentication Mechanism の略。ネットワークを通してユーザーを認証するシステムの 1 つ。マサチューセッツ工科大学 (MIT) で開発された Kerberos V5 技術に準拠する

「SEAM」と「Kerberos」は、SEAM のマニュアルでは同じ意味で使用されることが多い

Secure Shell セキュリティ保護されていないネットワークを通して、セキュリティ保護されたリモートログインおよびその他のセキュリティ保護されたネットワークサービスを使用するための特別なプロトコル

stash ファイル KDC のマスター鍵を暗号化したコピーが含まれる。このマスター鍵を使用して、サーバーをリブートすると、KDC が自動的に認証され、サーバーが `kadmind` および `krb5kdc` プロセスを起動する。stash ファイルにはマスター鍵が入っているため、stash ファイルやそのバックアップは安全な場所に保管する必要がある。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る

TGS チケット認可サービス (Ticket-Granting Service)。KDC の構成要素の 1 つ。チケットを発行する

TGT チケット認可チケット (Ticket-Granting Ticket)。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る

VPN 仮想プライベートネットワーク (Virtual Private Network)。暗号化とトンネルを使用して、セキュリティ保護された通信を提供するネットワーク。公開ネットワークを通してユーザーを接続する

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

インスタンス　主体名の 2 番目の部分。インスタンスは、主体の主ノードを修飾する。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る。host/boston.eng.example.com のように、ホストの完全指定ドメイン名を指定する。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る。ただし、joe と joe/admin は、一意の主体である。「主ノード」、「主体名」、「サービス主体」、「ユーザー主体」も参照

オーセンティケータ　オーセンティケータは、KDC がチケットを要求したときおよびサーバーがサービスを要求したときに、クライアントから渡される。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る。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る。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る。チケットと異なり、オーセンティケータは一度しか使用できない。通常は、サービスへのアクセスが要求されたときに使用される。オーセンティケータは、クライアントとサーバーのセッション鍵を使用して暗号化される

鍵　1. 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

2. 暗号化鍵。次の 3 つの種類がある

- 「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る。「非公開鍵」も参照
- 「サービス鍵」 – 非公開鍵と同じ機能を持つが、サーバーとサービスによって使用される。「サービス鍵」も参照
- 「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る。「セッション鍵」も参照

関係　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

完全性　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サービス。「認証」、「プライバシ」も参照

キータブファイル　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る

機密性　「プライバシ」を参照

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す。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る

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す

一般的には、サービスを使用する主体

クライアント主体

(RPCSEC_GSS API) RPCSEC_GSS で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る

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、要求が拒否される。クロックスキューは、krb5.conf ファイルに指定できる

権利プロファイル

権利またはプロファイルとも呼ばれる。RBAC で使用される優先指定の集合。役割またはユーザーに割り当てることができる。権利プロファイルは、認証、UID または GID を指定したコマンド (セキュリティ属性とも呼ばれる)、およびその他の権利プロファイルで構成される

公開鍵の暗号化

暗号化方式の 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する。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する。SEAM は非公開鍵システムである。「非公開鍵の暗号化」も参照

更新可能チケット

有効期限の長いチケットは、セキュリティを低下させることがあるため、「更新可能」チケットに指定することができる。更新可能チケットには 2 つの有効期限がある。1 つはチケットの現在のインスタンスの有効期限、もう 1 つは任意のチケットの最長有効期限。クライアントがチケットを継続して使用したい場合は、最初の有効期限が切れる前に、そのチケットを更新する。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 時間だけ有効だとする。そのチケットを持つクライアントがもう 1 時間チケットを持ちたい場合は、そのチケットを更新する必要がある。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は自動的に切れ、それ以上更新できない

サーバー

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、boston.eng.acme.com というマシンに rlogin する場合、そのマシンが rlogin サービスを提供するサーバーとなる。「サービス主体」も参照

サーバー主体

(RPCSEC_GSS API) 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る。「クライアント主体」も参照

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る。たとえば、`boston.eng.acme.com` というマシンに `rlogin` する場合、そのマシンが `rlogin` サービスを提供するサーバーになる

2. 認証以外の保護レベルを提供するセキュリティサービス (完全性またはプライバシ)。「完全性」と「プライバシ」も参照

サービス鍵

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る。「鍵」も参照

サービス主体

1 つまたは複数のサービスに Kerberos 認証を提供する主体。サービス主体では、一次名はサービス名 (`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る。「ホスト主体」と「ユーザー主体」も参照

資格

チケットと照合セッション鍵を含む情報パッケージ。主体の識別情報を認証するときに使用する。「チケット」と「セッション鍵」も参照

資格キャッシュ

KDC から受信した資格を含む記憶領域。通常はファイル

主体

1. ネットワーク通信に参加する、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る。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のことである。「主体名」、「サービス主体」、「ユーザー主体」も参照

2. (RPCSEC_GSS API) 「クライアント主体」と「サーバー主体」を参照

主体名

1. 主体の名前。書式は、*primary/instance@REALM* 。「インスタンス」、「主ノード」、「レルム」も参照

2. (RPCSEC_GSS API) 「クライアント主体」と「サーバー主体」を参照

主ノード

主体名の最初の部分。「インスタンス」、「主体名」、「レルム」も参照

承認

1. 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. 役割によるアクセス制御 (RBAC) で、役割またはユーザーに割り当てることができるアクセス権。権利プロファイルに埋め込まれていることもある。承認が与えられていない動作クラスは、セキュリティポリシーによって拒否される

初期チケット

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)。一部のサービス (パスワードを変更するアプリケーションなど)

は、*initial* と指定されたチケットを要求して、クライアントが非公開鍵を知っていることを検査する。初期チケットを使用した検査は、クライアントが最近認証されたことを証明するときに重要になる。チケット認可チケットの場合は、取得してから時間が経過していることがある

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る。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) を配置する。「KDC」と「マスター KDC」も参照

セキュリティサービス
: 「サービス」を参照

セキュリティフレーバ
: 「フレーバ」を参照

セキュリティメカニズム
: 「メカニズム」を参照

セッション鍵
: 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を保護するために生成される。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る。「鍵」も参照

遅延チケット
: 遅延チケットは、作成されても指定された時期まで有効にならない。遅延チケットは、夜遅く実行されるバッチジョブなどに使用する。チケットが盗まれても、バッチジョブが実行されるまで使用できないためである。遅延チケットは、無効チケットとして発行され、a) 開始時刻を過ぎて、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る。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ある。ただし、チケットが「更新可能」な場合、チケットの有効期限は通常、チケット認可チケットの全有効期限と同じに設定される。「無効チケット」と「更新可能チケット」も参照

チケット
: ユーザーの識別情報をサーバーまたはサービスに安全に渡すために使用される情報パケット。チケットは、単一のクライアントと特定のサーバー上の特定のサービスだけに有効である。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が含まれる。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る。チケットは、作成されてから有効期限まで再使用できる。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ない。「オーセンティケータ」、「資格」、「サービス」、「セッション鍵」も参照

チケットファイル
: 「資格キャッシュ」を参照

転送可能チケット
: 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る。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ない。たとえば、ユーザー `david` が `jennifer` のマシンで転送可能チケットを取得した場合、

	david は自分のマシンにログインするときに新しいチケットを取得する必要がない (再び認証を受ける必要もない)。「プロキシ可能チケット」も参照
特権付きアプリケーション	システム制御を優先指定し、特定の UID、GID、または承認を確認するアプリケーション
認証	取り込まれた主体の ID を確認するプロセス
ネームサービススコープ	特定の役割が操作を許可されている適用範囲。つまり、 特定のネームサービス (NIS、NIS+、LDAP など) を使用する個々のホストまたはすべてのホスト。スコープは、Solaris 管理コンソールツールボックスに適用される
ネットワーク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を提供するサーバー (ftp など)。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る
非公開鍵	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る。「鍵」も参照
非公開鍵の暗号化	非公開鍵の暗号化では、送信者と受信者は同じ暗号化鍵を使用する。「公開鍵の暗号化」も参照
秘密鍵	「非公開鍵」を参照
プロファイルシェル	RBAC で使用されるシェル。コマンド行から役割の権利プロファイルに割り当てた任意の特権付きアプリケーションを、役割 (またはユーザー) が実行できる。プロファイルシェルには、pfsh、 pfcsh、pfksh がある。これらはそれぞれ、Bourne シェル(sh)、C シェル (csh)、Korn シェル(ksh) に対応する
プライバシ	セキュリティサービスの 1 つ。送信データを送信前に暗号化する。プライバシには、データの完全性とユーザー認証も含まれる。「認証」、「完全性」、「サービス」も参照
フレーバ	従来は、「セキュリティフレーバ」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た。RPCSEC_GSS もセキュリティフレーバだが、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する
プロキシ可能チケット	サービスによって使用されるチケットの 1 つ。クライアントの代わりに、クライアントの操作を実行する。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う。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る。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るが、チケット認可チケットは取得できない。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ある。「転送可能チケット」も参照
ホスト	ネットワークを通じてアクセス可能なマシン

ホスト主体
: サービス主体のインスタンスの 1 つ (一次名は `host`)。さまざまなネットワークサービス (`ftp`、 `rcp`、`rlogin` など) を提供するために設定する。`host/boston.eng.example.com@ENG.EXAMPLE.COM` はホスト主体の例である「サーバー主体」も参照

ポリシー
: チケットの使用方法を管理する一連の規則。SEAM がインストールされるときまたは管理されるときに開始される。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) を制限できる

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` で構成される。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る。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る

無効チケット
: 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る。無効チケットを有効にするには、開始時期が過ぎたあと、TGS 要求で `VALIDATE` フラグを設定し、クライアントがこのチケットを KDC に提示する必要がある。「遅延チケット」も参照

メカニズム
: データの認証や機密性を実現するための暗号化技術を指定するソフトウェアパッケージ。たとえば、Kerberos V5、Diffie-Hellman 公開鍵など

役割
: 特権付きアプリケーションを実行するための特別な ID。割り当てられたユーザーだけが実行できる

ユーザー主体
: 特定のユーザーに属する主体。ユーザー主体の一次名には、ユーザー名を指定する。オプションのインスタンスには、対応する資格の使用方法を説明する名前を指定する (`joe`、`joe/admin` など)。「ユーザーインスタンス」とも呼ばれる。「サービス主体」も参照

レルム
: 1. 1 つの SEAM 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 `joe/admin@ENG.EXAMPLE.COM` の場合、レルムは `ENG.EXAMPLE.COM` 。「主体名」 も参照

索引

B

E

F

G

H

I

K

O

P

Q

R

S

T

U

V

W

X

Y

あ

さ

し

す

せ

そ

た

ち

つ

て

と

な

へ

ほ

ま

み

む

め

も

や

わ