Oracle® Solaris 11.4 でのユーザーとプロセスのセキュリティー保護

Part No: E75240-01
2018 年 8 月

ORACLE

Part No: E75240-01

ドキュメントのアクセシビリティについて

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

Oracle Supportへのアクセス

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

目次

表目次

例目次

このドキュメントの使用法

- **概要** – ユーザーへの追加権利の割り当て、役割の作成と使用、および システム上の特定のリソースおよびプログラムへの権利の割り当ての方法について説明します。ラベル付き情報を処理するユーザーおよび SMF サービスにラベルを割り当てる方法についても説明します。
- **対象読者** – セキュリティー管理者。
- **前提知識** – サイトのセキュリティー要件。

製品ドキュメントライブラリ

この製品および関連製品のドキュメントとリソースは http://www.oracle.com/pls/topic/lookup?ctx=E75431-01 で入手可能です。

フィードバック

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

♦ ♦ ♦ 第 1 章

1 権利を使用したユーザーとプロセスの制御について

には、ユーザー、役割、プロセス、および選択されたリソースに割り当てることができる権利が用意されています。これらの権利は、スーパーユーザーモデルの代替となるよりセキュアな管理手法です。

この章では、ユーザー権利とプロセス権利の管理を支援する要素について説明し、ユーザーの権利拡大、ユーザーの権利の制限、コマンドへの特権の追加、必要な特権のみへのアプリケーションの制限の方法について説明します。

の権利の新機能

このセクションでは、ユーザー権利における重要な新機能である権利に基づくアクセス制御 (RBAC) と、プロセス権利における新機能である特権について、既存のお客様を対象に説明します。

- は、データおよびユーザープロセスにラベルを付けます。この機能によって、サイトのセキュリティーのために特別な保護が必要なディレクトリや情報に対するデータ損失保護が提供されます。ラベル付けが常に有効になっていると、管理者がラベル付けの階層を構成し、特定のファイルやディレクトリにラベルを適用して、信頼できるユーザーがラベル付きプロセスを実行できるようにするまで、システムの動作は変更されません。

  詳細は、『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』 の 第 2 章,「データ損失保護のためのファイルのラベル付け」および第6章「データ損失保護のためのプロセスのラベル付け」を参照してください。
- パスワードの最小文字数は 6 文字でなく 8 文字になりました。詳細は、`passwd(1)` のマニュアルページを参照してください。

- サービス管理機能 (SMF) は、システム全体のセキュリティー属性のためのリポジトリで、以前は次のファイルで管理されていました。

  ```
  /etc/security/policy.conf
  /etc/default/login
  /etc/default/passwd
  /etc/default/su
  ```

  属性およびその値は、`svc:/system/account-policy:default` サービスがオンラインで、レガシーの `/etc` ファイルからのセキュリティー属性 (たとえば、`/etc/default/su` ファイルからのすべてのセキュリティー属性) が有効なとき、SMF サービスとしてロードされて管理されます。

  詳細は、`account-policy(8S)` のマニュアルページ、88 ページの「システム全体の特権、承認、および権利プロファイルの変更」、および166 ページの「ファイルおよび対応する SMF プロパティー内のセキュリティー属性」を参照してください。
- リモート RAD インタフェースから不変ゾーンを管理できます。詳細は、106 ページの「不変ゾーンの管理」を参照してください。
- `user_attr` データベースに追加のセキュリティー属性が格納されます。
  - `account-policy` SMF ステンシルの `login_policy/annotation=`*value* セキュリティー属性を有効にするか、ユーザーアカウントに値を設定することによって、管理者はユーザーがログインの目的を注釈として付けることを求める (`yes`) またはリクエストする (`optional`) ことができます。注釈はログインイベントの監査レコードに追加されます。`account-policy` サービスが有効でない場合、値は `policy.conf` ファイルにシステム全体で設定できます。

    詳細は、『Managing Auditing in Oracle Solaris 11.4』 の 「New Feature – Annotating Reason for Access in the Audit Record」と、Unresolved link to " pam_unix_cred7" および `account-policy(8S)` のマニュアルページを参照してください。
  - `account-policy` SMF ステンシルの `login_policy/auto_unlock_time=`*time* セキュリティー属性を有効にするか、ユーザーアカウントに値を設定することによって、管理者はロックされたアカウントが認証の成功によって自動的にロック解除されるまでの時間を指定できます。管理者は時間を、分数、時間数、日数、または週数として指定できます。`account-policy` サービスが有効でない場合、値は `policy.conf` ファイルにシステム全体で設定できます。

    この属性の時間を指定しない場合、管理者は 70 ページの「標準ユーザーに対してアカウントロックを設定する方法」に示すように、アカウントを明示的にロック解除する必要があります。

注記 - `login_policy/auto_unlock_time` 属性は、ロックされて提供され、パスワードのないシステムアカウントには適用されません。この属性は、たとえば管理者が `passwd -l` コマンドなどを使用して、パスワードを持つユーザーアカウントをロックした場合は適用されません。

詳細は、`account-policy(8S)` および `user_attr(5)` のマニュアルページを参照してください。158 ページの「`user_attr` データベース」も参照してください。

- はワンタイムパスワード (OTP) を処理するための `pam_otp_auth` PAM モジュールを提供します。OTP はログイン前のもう 1 つの認証ステップを提供します。モジュールをインストールするパッケージによって、`/etc/security/pam_policy` ディレクトリに 2 つの PAM スタックもインストールされます。詳細は、『Managing Authentication in Oracle Solaris 11.4』 の 「Task Map: Using OTP in Oracle Solaris」を参照してください。
- は、スマートカード認証を管理するための `pam_pkcs11` PAM モジュールを提供します。スマートカードにより、ユーザーは、1) ログインサーバーによって認識されているスマートカードを持っていて、2) 正しい PIN を入力できる場合にのみログインできます。詳細は、『Managing Authentication in Oracle Solaris 11.4』 の 第 3 章, 「Using Smart Cards for Multifactor Authentication in Oracle Solaris」を参照してください。

ユーザー権管理

ユーザー権管理は、通常は`root` 役割に限定されるタスクへのユーザーアクセスを制御するためのセキュリティー機能です。セキュリティー属性、つまり権利をプロセスとユーザーに適用することによって、サイトは、スーパーユーザー特権を複数の管理者間で分割できます。プロセス権管理は、「特権」を介して実装されます。ユーザー権利管理は、あとでユーザーまたは役割に割り当てられる権利を集めた権利プロファイルによって実装されます。キオスクやゲストユーザーなどのユーザー権利も制限できます。

- カーネルプロセスでの権利の説明については、28 ページの「プロセス権管理」を参照してください。
- 権利を管理する手順については、第3章「 での権利の割り当て」、第4章「アプリケーション、スクリプト、およびリソースへの権利の割り当て」、および第5章「権利使用の管理」を参照してください。
- トラブルシューティング情報については、第8章「 での権利のトラブルシューティング」を参照してください。
- 参照情報については、第7章「 の権利の一覧表示」および第9章「 権利リファレンス」を参照してください。

スーパーユーザーモデルの代替としてのユーザー権利およびプロセス権利

従来の UNIX システムでは、root ユーザー (スーパーユーザーとも呼ばれる) が全権を有します。多数の setuid プログラムと同様に、root として実行されるプログラムにも全権があります。root ユーザーは全ファイルの読み取り権とアクセス権、全プログラムの実行権を持ち、任意のプロセスに終了シグナルを送ることができます。実際、スーパーユーザーになるユーザーは、使用するサイトのファイアウォールの変更、監査トレールの変更、機密レコードの読み取り、ネットワーク全体の停止などを行えます。setuid root プログラムがハイジャック (強奪) されると、システム上で何が起きても不思議はありません。

ユーザー、リソース、およびプロセスへの権利の割り当てによって、オールオアナッシングのスーパーユーザーモデルのよりセキュアな代替機能が提供されます。権利により、セキュリティーポリシーをよりきめ細かいレベルで適用できます。権利では、最小特権というセキュリティー原則が使用されます。最小特権は、ジョブを行う上で必要な特権だけをユーザーに与えることを意味します。通常のユーザーには、アプリケーションの使用、実行中のジョブのステータスチェック、ファイルの印刷、新しいファイルの作成などを行うための十分な特権が与えられます。通常のユーザー権利以外の権利は、権利プロファイルにグループ化されます。スーパーユーザー権利の一部が必要なジョブを行うユーザーには、権利プロファイルを割り当てることができます。

プロファイルにグループ化された権利は、ユーザーに直接割り当てることができます。間接的に割り当てるには、役割と呼ばれる特殊アカウントを作成します。これにより、ユーザーは一部の管理特権を必要とするジョブを実行するときに役割を引き受けることができます。 には事前定義の権利プロファイルが多数用意されています。管理者は役割を作成し、プロファイルを割り当てます。

ARMOR パッケージには、一連の標準化された役割が含まれています。このパッケージを自動インストールし、役割をユーザーに割り当てることによって、ブート時に責務分離を提供するシステムを作成できます。詳細は、The Open Group Blog: UNIX® Special Edition (Dec 2015-Feb 2017) の ARMOR の記事を参照してください。このガイドの、46 ページの「選択した権利モデルへの準拠」、および 例2「ARMOR の役割の使用」を参照してください。

権利プロファイルは、広範な管理権利を提供できます。たとえば System Administrator 権利プロファイルにより、アカウントはプリンタ管理や cron ジョブ管理などのセキュリティーに関連しないタスクを実行できます。権限を限定して権利プロファイルを定義することもできます。たとえば、Cron Management 権利プロファイルは at ジョブと cron ジョブを管理します。役割を作成すると、その役割に広範な管理権利または限定された権利を割り当てることができます。

次の図に、 が役割を作成して権利を信頼できるユーザーに割り振る方法を示します。スーパーユーザーは、信頼できるユーザーに権利プロファイルを直接割り当てることで、権利を割り振ることもできます。

図 1 権利の割り振り

図に示す権利モデルでは、スーパーユーザーが 3 つの役割を作成します。役割は、権利プロファイルに基づいて作成されます。続いてスーパーユーザーは、役割のタスクに適したユーザーにその役割を割り当てます。ユーザーは、各自のユーザー名でログインします。ユーザーはログイン後に、管理コマンドとグラフィカルユーザーインタフェース (GUI) ツールを実行できる役割を引き受けます。

役割は柔軟に設定できるため、さまざまなセキュリティーポリシーに対応できます。には標準装備された役割がほとんどありませんが、役割は簡単に構成できます。例 2「ARMOR の役割の使用」に、ARMOR 標準に基づく役割の使用方法を示します。ARMOR の役割に追加して使用する役割、または ARMOR の役割の代わりに使用する役割として、 に用意されている権利プロファイルに基づくユーザー独自の役割を作成できます。

- **root –** `root` ユーザーと同等の強力な役割。ただし、ほかのすべての役割と同様に、`root` 役割はログインできません。通常のユーザーは、ログインしてから、割り当てられた `root` 役割を引き受ける必要があります。デフォルトでは、この役割は構成され初期ユーザーに割り当てられます。
- **System Administrator –** セキュリティーに関係のない管理作業を行う役割で、権利が限定されています。この役割ではファイルシステム、メール、ソフトウェアのインストールなどを管理できます。ただし、パスワードの設定は行えません。
- **Operator –** バックアップやプリンタ管理などが行える、補佐的な管理者向けの役割。

注記 - Media Backup 権利プロファイルは、ルートファイルシステム全体へのアクセスを提供します。したがって、Media Backup 権利プロファイルと Operator 権利プロファイルは補佐的な管理者向けに設計されていますが、この管理者が信頼できるユーザーであることを確認する必要があります。

---

1 つ以上のセキュリティー役割を構成することもできます。セキュリティーは、Information Security、User Security、および Zone Security の 3 つの権利プロファイルと、それらの補助プロファイルによって処理されます。ネットワークセキュリティーは、Information Security 権利プロファイル内の補助プロファイルです。

役割は実装する必要はありません。役割は、組織のセキュリティー要件に応じて設定する機能です。1 つの方法として、セキュリティー、ネットワーク、ファイアウォー

ル管理などの領域における専用の管理者のための役割を設定します。別の方法として、強力な管理者役割を 1 つと上級ユーザー役割を作成することもできます。この上級ユーザー役割は、自分のシステムの各部を修正することを認められたユーザーに割り当てます。また、権利プロファイルをユーザーに直接割り当てて、役割を作成しないでおくこともできます。

スーパーユーザーモデルと権利モデルは共存できます。次の表では、権利モデルで設定できる権利 (スーパーユーザーから制限された通常のユーザーまで) を順に挙げます。両モデルで監視できる管理アクションを示しています。プロセス権利、つまり特権の効果のサマリーについては、表2を参照してください。

表 1 スーパーユーザーモデルと特権モデルの対比

システムにおけるユーザー権限	スーパーユーザーモデル	権利モデル
すべてのスーパーユーザー特権を持つスーパーユーザーになることができる	可能	可能
すべてのユーザー権利を持つユーザーとしてログインできる	可能	可能
権利が限定されたスーパーユーザーになることができる	不可能	可能
ユーザーとしてログインし、散発的にスーパーユーザー特権を持つことができる	可能 (`setuid root` プログラムのみを使用)	可能 (`setuid root` プログラムと権利を使用)
すべてのスーパーユーザー特権ではなく、管理権利だけを持つユーザーとしてログインできる	不可能	可能 (権利プロファイル、役割、および直接割り当てられた特権と承認を使用)
通常のユーザーよりも少ない権利を持つユーザーとしてログインできる	不可能	可能 (権利を削除)
スーパーユーザーの処理を監視する	可能 (`su` コマンドを監査することによって)	可能 (`pfexec()` への呼び出しを監査することによって) また、`root` 役割を引き受けたユーザーの名前も監査トレールに含まれる

ユーザー権利およびプロセス権利の基本情報

特権のないまたは権利のないという用語は には適用されません。通常のユーザープロセスを含む のすべてのプロセスには、少なくとも何らかの特権またはユーザー権利 (承認など) が設定されています。 がすべての UNIX プロセスに付与する特権の基本セットについては、28 ページの「プロセス権管理」を参照してください。

では次の要素によってユーザー権利が適用されます。これらの権限は、許容セキュリティーポリシーまたは制限セキュリティーポリシーを適用するように構成できます。

- **承認** – 追加の権利を必要とするアクションのクラスをユーザーまたは役割が実行できるようにするアクセス権。たとえばデフォルトのセキュリティーポリシーでは、コンソールユーザーに対し `solaris.device.cdrw` 承認が付与されます。この承認によってユーザーは CD-ROM デバイスの読み取りと書き込みが行えます。承認のリストについては、`auths list` コマンドを使用してください。承認はカーネルではなく、ユーザーアプリケーションレベルで適用されます。25 ページの「ユーザー承認に関する詳細」を参照してください。
- **特権** – コマンド、ユーザー、役割、または特定のリソース (ポートや SMF メソッドなど) に付与できる権利です。特権はカーネルで実装されます。たとえば、`proc_exec` 特権によってプロセスは `execve()` を呼び出すことができます。通常のユーザーには基本特権が与えられます。自分の基本特権を確認するには、`ppriv -vl basic` コマンドを実行します。詳細は、28 ページの「プロセス権管理」を参照してください。
- **セキュリティー属性** – プロセスが操作を実行できるようにする属性 (権利の実装)。標準的な UNIX 環境では、セキュリティー属性によって、通常のユーザーには禁止されている操作をプロセスで実行できるようになります。たとえば、`setuid` プログラムと `setgid` プログラムはセキュリティー属性を持ちます。権利モデルでは、`setuid` および `setgid` プログラムに加えて、承認と特権がセキュリティー属性です。これらの属性、つまり権利は、ユーザーに割り当てることができます。たとえば、`solaris.device.allocate` 承認が与えられたユーザーは、デバイスを独占的に使用するためにそのデバイスの割り当てを行うことができます。特権をプロセスに割り当てることができます。たとえば、`file_flag_set` 特権を持つプロセスは、変更不可能な、リンク解除できない、または追加のみのファイル属性を設定できます。

  セキュリティー属性によって権利を制限することもできます。たとえば `access_times` および `access_tz` セキュリティー属性は、特定のセキュリティー関連操作が許可される日時と、オプションで時間帯を設定します。ユーザーは直接、またはこれらのキーワードを含む認証権利プロファイルを割り当てることによって制限できます。詳細は、`user_attr(5)` のマニュアルページを参照してください。
- **特権付きアプリケーション** – 権利を確認してシステム制御をオーバーライドできるアプリケーションまたはコマンド。詳細は、41 ページの「権利を確認するアプリケーション」および『Oracle Solaris 12 セキュリティーサービス開発ガイド』を参照してください。
- **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権利の集合です。権利プロファイルには、承認、直接割り当てられた特権、セキュリティー属性を持つコマンド、およびほかの権利プロファイルを含めることができます。別のプロファイル内に存在するプロファイルは、補助権利プロファイルと呼ばれます。権利プロファイルは、権利をグループ化する手段として便利です。ユーザーに直接割り当てるか、または役割と呼ばれる特殊アカウントに割り当てることができます。権利プロファイルのコマンドを使用できるのは、プロセスで権利が認識される

場合に限ります。また、パスワードの入力が必要な場合があります。あるいは、パスワード認証がデフォルトで提供される場合があります。25 ページの「権利プロファイルの詳細」を参照してください。

- **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識別情報です。この特殊な識別情報を取得できるのは、あらかじめ割り当てられたユーザーだけです。役割により実行されるシステムでは、初期構成後にはスーパーユーザーが不要となることがあります。26 ページの「役割の詳細」を参照してください。
- **修飾ユーザー属性** – LDAP 内のユーザーおよび役割アカウントに適用でき、したがって集中管理できるセキュリティー属性。たとえば、指定されたアクセス時間にユーザーを制限したり、指定されたシステム上の役割または権利プロファイルのみをユーザーに割り当てたりできます。27 ページの「修飾ユーザー属性について」を参照してください。

次の図に、ユーザー権利とプロセス権利がどのように連携するかを示します。

図 2 ユーザー権利とプロセス権利の連携

次の図は、Network Security 役割と Network Security 権利プロファイルを使用して、割り当てられた権利がどのように機能するかを示します。

図 3 ユーザー権利とプロセス権利の割り当ての例

Network Security 役割は、IPsec、wifi、およびネットワークリンクの管理に使用します。この役割は、ユーザー `jdoe` に割り当てられています。`jdoe` は、この役割に切り替えてから役割のパスワードを指定することによって、この役割になることができます。管理者は、この役割で役割のパスワードではなくユーザーパスワードでの認証を可能にするようにできます。

この図では、Network Security 権利プロファイルが Network Security 役割に割り当てられています。Network Security 権利プロファイルには、Network Wifi Security、Network Link Security、および Network IPsec Management という、順番に評価される補助プロファイルが含まれています。これらの補助プロファイルには、役割の主要なタスクを実行する権利が含まれています。

Network Security 権利プロファイルには、直接割り当てられた 3 つの承認と、セキュリティー属性を持つ 2 つのコマンドがありますが、直接割り当てられた特権はありません。補助権利プロファイルには、直接割り当てられた承認があり、それらの 2 つにはセキュリティー属性を持つコマンドがあります。

jdoe が Network Security 役割を引き受けると、シェルがプロファイルシェルに変化します。プロファイルシェルプロセスは権利の使用を評価できるため、jdoe はネットワークセキュリティーを管理できます。

ユーザー権利の詳細

このセクションでは、ユーザーレベルでの権利の実装と使用についてさらに詳しく説明します。

ユーザー承認に関する詳細

承認とは、役割、プログラム、ゾーン、またはユーザーに付与できる権利です。承認は、ユーザーアプリケーションレベルでポリシーを適用します。特権と同様に、承認の割り当てを誤ると、本来予定していたよりも多くの権利が付与される結果となる可能性があります。詳細は、38 ページの「特権エスカレーションとユーザー権利」を参照してください。

承認と特権の違いは、セキュリティーポリシーが適用されるレベルにあります。プロセスに適切な特権がないと、特権化された処理の実行がカーネルによって防止される可能性があります。適切な承認が与えられていないユーザーは、特権付きアプリケーションを使用できなかったり、特権付きアプリケーションに含まれるセキュリティーの厳しい処理を実行できなかったりする可能性があります。特権の詳しい説明については、28 ページの「プロセス権管理」を参照してください。

権利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。

承認についての詳細は、次のセクションを参照してください。

- 156 ページの「承認のリファレンス」
- 160 ページの「auth_attr データベース」
- 162 ページの「承認を必要とする特別なコマンド」

権利プロファイルの詳細

権利プロファイルは、管理権利が必要なタスク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の集合です。権利プロファイルには、承認、特権、セ

キュリティー属性が割り当てられたコマンド、およびほかの権利プロファイルを含めることができます。権利プロファイルにはまた、特権の初期継承可能セットを削減または拡張したり、制限セットを削減したりするためのエントリも含まれています。

認証権利プロファイルは、ユーザーに対しパスワードの入力、つまり再認証を求める権利プロファイルです。管理者は、どのプロファイルをユーザーの再認証なしで使用できるかを決定します。再認証が不要なプロファイルの例として、Basic Solaris User 権利プロファイルがあります。サイトのセキュリティー要件に基づき、セキュリティーの厳しいタスクのための権利プロファイルでは、再認証が必要になることがあります。

権利プロファイルの参照情報については、次のセクションを参照してください。

- 154 ページの「権利プロファイルのリファレンス」
- 160 ページの「`prof_attr` データベース」
- 160 ページの「`exec_attr` データベース」

役割の詳細

「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、グループ割り当て、パスワードなどを持ちます。権利プロファイルと承認により、役割に管理権利が提供されます。役割は、ほかの役割やその役割を引き受けるユーザーから権利を継承することはできません。役割によりスーパーユーザー特権が割り振られるため、セキュリティーが強化された管理を実施できます。

各役割は、複数のユーザーに割り当てることができます。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、役割を引き受けるにはコマンド行で `su` コマンドを実行し、役割名と役割のパスワードを入力します。管理者は、ユーザーが、そのユーザーのパスワードを指定することによって認証できるようにシステムを構成できます。例18「役割パスワードでのユーザー独自のパスワード使用の有効化」を参照してください。

役割は直接ログインすることはできません。ユーザーがまずログインし、続いて役割を引き受けます。役割を引き受けたあとで別の役割を引き受けるには、まず現在の役割を終了する必要があります。

また、権利プロファイルはユーザーの環境に権利を追加しますが、役割はユーザーに対し、その役割を引き受けることができるほかのユーザーと共有するクリーンな実行環境を提供します。ユーザーが役割に切り替わっても、ユーザーの承認または権利プロファイルはすべて役割には適用されません。

`passwd`、`shadow`、および `user_attr` データベースに、静的な役割情報が格納されます。ユーザーは役割のアクションを監査できるので、この監査を実行すべきです。

役割の設定についての詳細は、次のセクションを参照してください。

- 46 ページの「選択した権利モデルへの準拠」
- 49 ページの「ユーザーへの権利の割り当て」

では root は役割であるため、匿名の root ログインが回避されます。プロファイルシェルコマンド pfexec が監査されると、監査トレールにはログインユーザーの実 UID、ユーザーが引き受けている役割、および実行された特権付き操作が含まれています。特権付き操作についてシステムを監査するには、114 ページの「管理アクションの監査」を参照してください。

修飾ユーザー属性について

修飾ユーザー属性は、ユーザーや役割、およびホストやネットグループと呼ばれるホストのグループに割り当てることができる属性です。ネットグループにより、ラボネットワークなどの一連のシステムの管理が簡素化されます。これらの修飾子は LDAP アカウントにのみ適用され、files ネームサービスには適用されません。

修飾および非修飾ユーザー属性は独立して保持され、組み合わせることはできません。この独立性により、管理者は 1 人のユーザーまたは役割に修飾と非修飾の両方の拡張ポリシー属性を割り当てることができます。実行時に、システムはまず、実行が発生しているホスト名を判定し、次に適切なポリシー属性のセットを適用します。

usermod および rolemod コマンドは、セキュリティー属性が適用されるホストまたはネットグループを示す修飾子オプション -q を受け入れます。修飾または非修飾属性の各セットを管理するには、個別の usermod および rolemod コマンドが必要です。userdel および roledel コマンドは、ほかの修飾または非修飾属性に影響を与えることなく、完全な修飾属性のセットを削除できます。

適切なユーザー属性のセットを適用するためのポリシーは、name-service/switch サービスによって指定された検索順序に従い、name-service/cache サービスによってキャッシュされます。その順序は次のとおりです。

1. 指定されたユーザーまたは役割に一致するローカルエントリ
2. 指定されたユーザーまたは役割の修飾属性の 1 つ以上の LDAP エントリ
3. ホスト名が現在のホストに一致する LDAP エントリ
4. 現在のホストをメンバーとして持つ (LDAP 内の) ネットグループ
5. 指定された LDAP ユーザーまたは役割の非修飾エントリ
6. 割り当てられた Stop 権利プロファイルがない場合は、account-policy サービスで指定されているデフォルト属性。通常、policy.conf ファイル内の属性値はサービス値を反映します。

一致が見つかった場合は、以降のクエリーを最適化するためにその内容がキャッシュされます。例については、例33「LDAP ユーザーや役割がその権利を使用できる場所と時間の修飾」を参照してください。

プロセス権管理

でのプロセス権管理は、特権により実装されます。特権により、コマンド、ユーザー、役割、および特定のシステムリソースのレベルにプロセスを制限できます。特権は、システムに対するすべてのスーパーユーザー権限を 1 人のユーザーまたは 1 つのプロセスだけが持っている場合に伴うセキュリティーリスクを軽減します。プロセス権利とユーザー権利によって、従来のスーパーユーザーモデルの魅力的な代替モデルが提供されます。

従来、特権は権利を追加するために使用されました。ただし、`setuid root` プログラムの、特権を認識するプログラムへの変更など、特権を使用して権利を制限することもできます。また拡張特権ポリシーにより、管理者は指定の特権のみをファイルオブジェクト、ユーザー ID、またはポートで使用できるように許可できます。このきめ細かな特権割り当てでは、このようなリソースに対し基本特権以外のそのほかのすべての特権が拒否されます。

- 拡張特権ポリシーと制限特権については、37 ページの「拡張特権ポリシーを使用した特権使用の制限」を参照してください。
- ユーザー権利については、19 ページの「ユーザー権管理」を参照してください。
- 特権を管理する方法については、第3章「 での権利の割り当て」を参照してください。
- 特権に関する参照情報については、164 ページの「特権のリファレンス」を参照してください。

カーネルプロセスを保護する特権

特権とは、プロセスが操作を実行するために必要とする権利です。この権利はカーネルにおいて実効性があります。特権の基本セットの範囲内で動作するプログラムは、システムセキュリティーポリシーの範囲内で動作します。`setuid root` プログラムは、システムセキュリティーポリシーの範囲を超えて動作するプログラムの例です。特権を使用することで、プログラムは `setuid root` を呼び出さなくてすみます。

特権は、システム上で行える処理をエミュレートします。プログラムは、その実行に必要な最小限の特権で実行できます。たとえば、ファイルを操作するプログラムには、`file_dac_write` および `file_flag_set` 特権が必要になることがあります。プロセス上のこれらの特権により、`root` としてプログラムを実行する必要がなくなります。

これまでシステムは、特権モデルで導入された22 ページの「ユーザー権利およびプロセス権利の基本情報」、つまり権利モデルに準拠していませんでした。逆に、システムはスーパーユーザーモデルを使用していました。スーパーユーザーモデルで

は、プロセスは root またはユーザーとして実行されていました。ユーザープロセスは、ユーザーのディレクトリとファイルにだけ作用するように限定されました。root プロセスは、システム上の任意の場所にディレクトリとファイルを作成できました。ユーザーのディレクトリ以外の場所にディレクトリを作成する必要があるプロセスは、UID=0 を使用して (つまり root として) 実行されました。セキュリティーポリシーは、システムファイルを保護するのに、任意アクセス制御 (Discretionary Access Control、DAC) に依存していました。デバイスノードは、DAC によって保護されました。たとえば、グループ sys が所有しているデバイスをオープンできるのはこのグループのメンバーだけでした。

しかし、setuid プログラムやファイルアクセス権、管理アカウントなどは悪用される危険性があります。setuid プロセスに許可されているアクションは、このプロセスがその処理に必要な数を上回っています。setuid root プログラムが侵入者に攻撃された場合には、全権を有する root ユーザーとしてふるまわれてしまいます。同様に、root パスワードにアクセスできるユーザーは誰でもシステム全体に損害を与えかねません。

対照的に、特権付きポリシーを適用するシステムでは、ユーザー権利からroot 権利までの間が段階的です。あるユーザーに通常のユーザーの権利を超える動作を実行するための特権を付与したり、root の特権を root が現在所有している数より少ない数に制限したりできます。権利により、特権で実行されるコマンドを権利プロファイルとして分離し、これを 1 人のユーザーまたは 1 つの役割に割り当てることができます。表1は、権利モデルが提供するユーザー権利と root 特権との間の段階を示しています。

権利モデルでは、スーパーユーザーモデルより高いレベルのセキュリティーが実現されます。プロセスから削除された特権が悪用される可能性はありません。プロセス特権は、弱点を突かれてアクセス権が取得される可能性がある DAC 保護だけの場合と比較して、重要なファイルとデバイスの保護を強化できます。

特権を使用することで、必要な権利しか持たないようにプログラムとプロセスを制限できます。最小特権が実装されたシステムでは、プロセスを取得した侵入者がアクセスできるのはそのプロセスに割り当てられた特権だけです。システムのほかの部分を攻撃することはできません。

特権の説明

特権は、それぞれの領域に基づいて論理的にグループ化されます。

- **FILE 特権** – 文字列 file で始まる特権は、ファイルシステムオブジェクトに対して作用します。たとえば、file_dac_write 特権は、ファイルへの書き込みの際に任意アクセス制御をオーバーライドします。
- **IPC 特権** – 文字列 ipc で始まる特権は、IPC オブジェクトアクセス制御をオーバーライドします。たとえば、ipc_dac_read 特権を使用すると、DAC によって保護されているリモート共有メモリーを読み取るプロセスが可能となります。

- **NET 特権 –** 文字列 `net` で始まる特権は、特定のネットワーク機能へのアクセスを可能にします。たとえば、`net_rawaccess` 特権を使用すると、デバイスをネットワークに接続できます。
- **PROC 特権 –** 文字列 `proc` で始まる特権は、プロセスがそれ自体の限定されたプロパティーを変更できるようにします。`PROC` 特権の中には、ごくわずかな効果しかない特権もあります。たとえば、`proc_clock_highres` 特権は、プロセスが高分解能タイマーを使用できます。
- **SYS 特権 –** 文字列 `sys` で始まる特権は、各種のシステムプロパティーに対する無制限のアクセス権をプロセスに付与します。たとえば、`sys_linkdir` 特権を使用すると、プロセスはディレクトリに対するハードリンクの確立と解除が行えます。

その他の論理グループには、CONTRACT、CPC、DAX、DTRACE、GRAPHICS、VIRT、WIN などがあります。

特権の中にはシステムに対する影響が少ないものもあれば、大きな影響を与えるものもあります。次の `proc_taskid` 特権の定義は、この特権の影響が小さいことを示しています。

```
proc_taskid
        Allows a process to assign a new task ID to the calling process.
```

`net_rawaccess` 特権の定義は、その影響が広範囲に及ぶことを示しています。

```
net_rawaccess
        Allows a process to have direct access to the network layer.
```

`privileges(7)` のマニュアルページに、各特権の説明が提供されています。138 ページの「特権の一覧表示」も参照してください。

特権を使用したシステムにおける管理上の相違点

特権を持つシステムと特権を持たないシステムとでは、明白な違いがいくつかあります。次の表に相違点の一部を示します。

表 2 特権を持つシステムと特権を持たないシステムとの明白な違い

機能	特権なし	特権
デーモン	デーモンが `root` として実行されます。	デーモンが、ユーザー `daemon` として実行されます。 たとえば、デーモン `lockd` および `rpcbind` には限定された特権が割り当てられており、`daemon` として実行されます。
ログファイルの所有権	ログファイルは `root` によって所有されます。	ログファイルは、そのログファイルを作成する `daemon` によって所有されます。`root` ユーザーがこのファイルを所有することはありません。
エラーメッセージ	エラーメッセージでスーパーユーザーが言及されます。 たとえば、`chroot: not superuser`。	エラーメッセージで特権の使用が言及されます。 たとえば、`chroot` エラーと同等のエラーメッセージは `chroot: exec failed`。

機能	特権なし	特権
setuid プログラム	プログラムは、通常のユーザーが実行を許可されていないタスクを完了するために setuid root を使用します。	多くの setuid root プログラムは、必要な特権のみで実行されます。 たとえば、コマンド audit、ikeadm、ipadm、ipsecconf、ping、traceroute、および newtask は特権を使用します。
ファイルアクセス権	デバイスアクセス権は DAC によって制御されます。たとえば、グループ sys のメンバーは /dev/ip を開くことができます。	デバイスを開くことができるユーザーをファイルアクセス権 (DAC) が予測することはありません。デバイスは、DAC と デバイスポリシーによって保護されます。 たとえば、/dev/ip ファイルには 666 アクセス権がありますが、デバイスを開くことができるのは適切な特権を持つプロセスだけです。
監査イベント	su コマンドの使用の監査によって、多くの管理機能がカバーされます。	特権の使用の監査によって、ほとんどの管理機能がカバーされます。cusa 監査クラスには、管理機能をモニターする監査イベントが含まれています。
プロセス	プロセスは、プロセス所有者の権利によって保護されます。	プロセスは特権によって保護されます。プロセス特権とプロセスフラグは、/proc/<pid>/priv ディレクトリ内の新しいエントリとして確認できます。
デバッグ	コアダンプ内で特権の言及はありません。	コアダンプの ELF 注記セクションで、NT_PRPRIV および NT_PRPRIVINFO の注記にプロセス特権とフラグについての情報が示されます。 ppriv コマンドやその他のコマンドでは、適切にサイズ設定されたセットの正しい数が示されます。これらのコマンドでは、ビットセット内のビットが特権名に正しく対応付けられます。

権限の詳細

このセクションでは、特権の実装、使用、および割り当てについて詳しく説明します。

特権の実装方法

各プロセスには、プロセスが特定の特権を使用できるかどうかを判断する 4 つの特権セットがあります。カーネルは、特権「有効セット」を自動的に計算します。初期の特権「継承可能セット」は変更できます。特権を使用するように作成されているプログラムは、そのプログラムで使用する特権の「許可されたセット」を減らすことができます。特権「制限セット」は縮小できます。

- **有効特権セット (E) –** 現在有効である特権の集合です。プロセスは、許可されたセット内の特権を有効セットに追加できます。プロセスは、E から特権を削除することもできます。
- **許可された特権セット (P) –** 使用できる特権の集合です。プログラムは、継承または割り当てを通して特権を使用できます。実行プロファイルは、プログラムに特

権を割り当てる方法の 1 つです。setuid コマンドは、root が持つすべての特権をプログラムに割り当てます。許可されたセットから特権を削除することはできますが、追加することはできません。P から削除された特権は、E からも自動的に削除されます。

特権を認識するプログラムは、そのプログラムがまったく使用することのない特権をそのプログラムの許可されたセットから削除します。この方法では、不要な特権がそのプログラムや悪質なプロセスによって悪用されることが防止されます。特権を認識するプログラムの詳細は、『Oracle Solaris 12 セキュリティーサービス開発ガイド』 の 第 2 章,「特権付きアプリケーションの開発」を参照してください。

- **継承可能な特権セット (I) –** exec への呼び出しでプロセスが継承できる特権の集合です。exec への呼び出しのあと、継承された特権は許可されたセットと有効セット内に配置されるため、setuid プログラムという特殊なケースを除き、これらのセットが等しくなります。

  setuid プログラムの場合は、exec への呼び出しのあと、継承可能セットがまず制限セットによって制限されます。続いて、継承された特権のセット (I) から制限セット (L) が除かれたものが、そのプロセスの P と E に割り当てられます。
- **制限特権セット (L) –** プロセスとその子プロセスでどの特権が利用できるかを示す上限を定義する集合です。デフォルトでは、制限セットはすべての特権です。プロセスは制限セットを縮小することはできますが、制限セットを拡張することはできません。L は I の制限に使用されます。このため、L は exec の時点で P と E を制限します。

  特権が割り当てられたプログラムを含むプロファイルがユーザーに割り当てられている場合、通常そのユーザーはそのプログラムを実行できます。未変更のシステムでは、プログラムの割り当て済み特権はユーザーの制限セットの範囲内です。プログラムに割り当てられている特権は、ユーザーの許可されたセットの一部になります。特権を割り当てられたプログラムを実行するには、ユーザーはプロファイルシェルからそのプログラムを実行する必要があります。

カーネルは、基本特権セットを認識します。変更されていないシステムの場合、各ユーザーの初期の継承可能セットはログイン時の基本セットと同じです。基本セットを変更することはできませんが、ユーザーが基本セットからどの特権を継承するかは変更できます。

未変更のシステムでは、ログイン時のユーザーの特権セットは次のようになります。

```
E (Effective): basic
I (Inheritable): basic
P (Permitted): basic
L (Limit): all
```

ログイン時には各ユーザーの基本セットは、それぞれの継承可能セット、許可されたセット、および有効セットに含まれます。ユーザーの制限セットは、ゾーン (大域または非大域) のデフォルトの制限セットと同等です。

追加の特権をユーザー、正確にはユーザーのログインプロセスに直接割り当てるか、権利プロファイルを介して複数のユーザーに間接的に割り当てるか、またはユーザー

に対して特権付きコマンドを割り当てることで間接的に割り当てることができます。また、ユーザーの基本セットから特権を削除できます。手順と例については、第3章「での権利の割り当て」を参照してください。

特権の使用法

特権は に組み込まれています。このセクションでは、 がデバイス、リソース管理、およびレガシーアプリケーションで特権をどのように使用するかを説明します。

プロセスが特権を取得する方法

プロセスが特権を継承するか、またはプロセスに特権を割り当てることができます。プロセスは、その親から特権を継承します。ログイン時に、ユーザーの初期継承可能特権セットによって、そのユーザーのプロセスで使用できる特権が決まります。ユーザーの当初のログインの子プロセスはすべて、このセットを継承します。

また、プログラム、ユーザー、役割、および特定のリソースに特権を直接割りてることもできます。プログラムで特権が必要な場合は、権利プロファイル内でそのプログラムの実行可能ファイルに特権を割り当てます。そのプログラムの実行を許可されたユーザーまたは役割には、そのプログラムが入ったプロファイルを割り当てます。ログイン時、あるいはプロファイルシェルが開かれている場合、プログラムの実行可能ファイルがプロファイルシェルで入力されると、そのプログラムは特権を使用して実行されます。たとえば、Object Access Management プロファイルが含まれる役割は、`file_chown` 特権を使用して `chmod` コマンドを実行できるため、その役割が所有していないファイルの所有権を変更できます。

付加的な特権が直接割り当てられたプログラムを役割またはユーザーが実行する場合、割り当てられているその特権は役割またはユーザーの継承可能セットに追加されます。特権が割り当てられたプログラムの子プロセスは、親プロセスの特権を継承します。子プロセスが親プロセスよりも多くの特権を必要とする場合には、子プロセスに直接それらの特権を割り当てる必要があります。

特権を使用するように作成されているプログラムは、特権を認識すると呼ばれます。特権を認識するプログラムは、プログラム実行中の特権の使用を有効または無効にできます。本番環境で使用するためには、プログラムに対し、そのプログラムが有効または無効にする特権を割り当てる必要があります。特権を認識するプログラムを使用可能にする前に、そのプログラムに必要な特権だけを実行可能ファイルに割り当てます。続いて管理者はこのプログラムのテストを行い、タスクが正常に行われるか確認します。また、プログラムが特権を悪用しないかも確認します。

特権を認識するコードの例については、『Oracle Solaris 12 セキュリティーサービス開発ガイド』 の 第 2 章,「特権付きアプリケーションの開発」を参照してください。特

権を必要とするプログラムに特権を割り当てるには、例38「レガシーアプリケーションへのセキュリティー属性の割り当て」および例49「特権付きコマンドを含む権利プロファイルの作成」を参照してください。

特権とデバイス

スーパーユーザーモデルではファイルアクセス権によってのみ保護されるシステムインタフェースを、権利モデルでは特権によって保護します。特権を使用したシステムでは、インタフェースを保護するほどの強さはファイルアクセス権にありません。`proc_owner` などの特権は、ファイルアクセス権をオーバーライドした上でファイルシステムへのフルアクセス権を取得する可能性があります。

このため、 では、デバイスを開くにはデバイスディレクトリの所有権では不十分です。たとえば、グループ `sys` のメンバーには、`/dev/ip` デバイスを開くことが自動的には許可されなくなります。`/dev/ip` のファイルアクセス権は `0666` ですが、デバイスを開くには `net_rawaccess` 特権も必要です。

デバイスポリシーは特権によって制御されるため、デバイスを開くためのアクセス権を付与する際の柔軟性が高くなります。特権要件は、デバイスポリシーに合わせて構成することも、ドライバ本体に合わせて構成することもできます。デバイスドライバのインストール、追加、または更新時に、特権要件を構成できます。

詳細は、`add_drv`(8)、`devfsadm`(8)、Unresolved link to " getdevpolicy8"、および `update_drv`(8) のマニュアルページを参照してください。

特権およびリソース管理

では、`project.max-locked-memory` および `zone.max-locked-memory` リソース制御を使用して、`PRIV_PROC_LOCK_MEMORY` 特権が割り当てられているプロセスのメモリー消費を制限できます。プロセスはこの特権を使うことで、物理メモリー内のページをロックできます。

`PRIV_PROC_LOCK_MEMORY` 特権を権利プロファイルに割り当てると、この特権を持つプロセスに、すべてのメモリーをロックする権限を与えることになります。安全対策として、この特権ユーザーがすべてのメモリーをロックできないように、リソース制御を設定してください。特権付きプロセスが非大域ゾーン内で実行される場合には、`zone.max-locked-memory` リソース制御を設定します。特権付きプロセスがシステム上で実行される場合には、プロジェクトを作成し、`project.max-locked-memory` リソース制御を設定します。これらのリソース制御については、『Administering Resource Management in Oracle Solaris 11.4』 の 第 6 章, 「About Resource Controls」および『Oracle Solaris Zones Configuration Resources』 の 第 1 章, 「Non-Global Zone Configuration Command and Resources」を参照してください。

レガシーアプリケーションと特権の使用

レガシーアプリケーションに対応するために、特権の実装はスーパーユーザーモデルと権利モデルの両方で動作します。カーネルは、プログラムが特権で動作するように設計されていること示す PRIV_AWARE フラグを自動的に追跡します。特権を認識しない子プロセスについて検討してください。親プロセスから継承された特権はどれも、子の許可されたセットおよび有効なセットで使用可能です。子プロセスで UID が 0に設定されていると、その子プロセスが完全なスーパーユーザー権利を持たない場合があります。プロセスの有効なセットおよび許可されたセットは、子の制限セットの特権に限定されます。このように、特権を認識するプロセスの制限セットによって、特権を認識しない子プロセス root 特権が制限されます。

特権の使用のデバッグ

には、特権のエラーを修正するツールが用意されています。ppriv コマンドと truss コマンドを使用して、デバッグ結果を出力できます。この例は、ppriv(1) のマニュアルページを参照してください。例については、143 ページの「RBAC と特権のトラブルシューティング」を参照してください。また、dtrace コマンドを使用することもできます。詳細は、dtrace(8) のマニュアルページおよび『Oracle Solaris 12 DTrace (動的トレース) ガイド』を参照してください。

特権の割り当て

「特権」という用語は従来、権利の強化を意味します。 システムの各プロセスは何らかの権利を使用して実行されるため、特権を削除することでプロセスの権利を減らすことができます。このリリースでは、拡張特権ポリシーを使用することで、特定のリソースにデフォルトで付与されている特権を除くほとんどの特権を削除できます。

ユーザーおよびプロセスへの特権の割り当て

セキュリティー管理者として、特権の割り当てを担当します。既存の権利プロファイルでは、プロファイル内のコマンドに特権がすでに割り当てられています。権利プロファイルを役割またはユーザーに割り当てます。

特権はまた、ユーザー、役割、または権利プロファイルに直接割り当てることもできます。セッションで特権を適切に使用すると信頼できるユーザーには、特権を直接割り当てることができます。直接の割り当てが適するものとしては、影響の少ない特権 (proc_clock_highres など) が挙げられます。直接の割り当てに適しない候

補としては、file_dac_write などの、影響が広範囲に及ぶ特権があります。詳細は、43 ページの「権利の割り当てにおけるセキュリティーに関する考慮事項」を参照してください。

ユーザー、役割、またはプロセスに対する特権が拒否されることもあります。ユーザーまたは役割の初期継承可能セットまたは制限セットから特権を削除する場合は、注意が必要です。

ユーザーまたは役割の特権の拡張

ユーザーや役割には、継承可能な特権セットがあります。制限セットには初めにすべての特権が設定されるため、このセットは削減のみ可能です。ユーザー、役割、およびプロセスの初期継承可能セットは、その継承可能セットにない特権を割り当てることで拡張できます。

使用可能な特権を 3 つの方法で拡張できます。

- 初期継承可能セットには含まれていないが制限セットに含まれている特権は、ユーザーと役割に割り当てることができます。この割り当ては、権利プロファイルの特権付きコマンドを使用して間接的に行うか、または直接行うことができます。
- 継承可能セットに含まれていない特権は、スクリプトまたはアプリケーションへの特権の追加などのように、プロセスに明示的に割り当てることができます。
- 継承可能セットに含まれていないが制限セットに含まれている特権は、ネットワークポート、UID、またはファイルオブジェクトに明示的に割り当てることができます。このような特権の使用は拡張特権ポリシーと呼ばれ、使用可能な特権を制限する手段でもあります。詳細は、37 ページの「拡張特権ポリシーを使用した特権使用の制限」を参照してください。

特権を必要とする管理タスクにのみその特権を割り当てることは、ユーザーまたは役割の特権をもっとも的確に拡張する方法です。コマンドまたはスクリプトとその必要な特権が含まれている権利プロファイルを作成します。次に、ユーザーまたは役割にその権利プロファイルを割り当てます。このように割り当てることで、ユーザーまたは役割はその特権付きコマンドを実行できます。このようにしないと、ユーザーはその特権を使用できません。

ユーザーまたは役割の初期継承可能特権セットの拡張は、特権を割り当てる方法として適切とは言えません。継承可能セット内の特権はすべて、許可されたセットと有効セット内に存在します。シェル内でユーザーまたは役割が入力するコマンドはすべて、直接割り当てられた特権を使用できます。詳細は、43 ページの「権利の割り当てにおけるセキュリティーに関する考慮事項」を参照してください。

不必要に特権が使用可能である状況を減らすには、拡張特権をネットワークポート、UID、およびファイルオブジェクトに割り当てることができます。このように割り当てることで、拡張特権割り当てに含まれない特権が有効セットから削除されます。詳

細は、37 ページの「拡張特権ポリシーを使用した特権使用の制限」を参照してください。

ユーザーまたは役割の特権の制限

信頼できないユーザーの権利を制限するため、特権と権利プロファイルを信頼できないユーザーに適用することもできます。特権を削除することで、ユーザーと役割による特定のタスク実行を不可能にできます。特権は、初期継承可能セットから削除することも、制限セットから削除することもできます。デフォルトセットよりも小さい初期継承可能セットまたは制限セットを配布する場合は、あらかじめ特権の削除を慎重にテストすることが望まれます。たとえば、初期継承可能セットから特権を削除したためにユーザーがログインできなくなる可能性があります。制限セットから特権を削除すると、削除した特権を必要とする古い `setuid root` プログラムが失敗する可能性があります。特権の削除の例については、75 ページの「ユーザーからの特権の削除」を参照してください。

ユーザー ID、ポート、またはファイルオブジェクトに対して使用可能な特権を制限するには、37 ページの「拡張特権ポリシーを使用した特権使用の制限」を参照してください。

スクリプトへの特権の割り当て

スクリプトは、コマンドと同様に実行可能ファイルです。このため、コマンドに特権を追加する場合と同じ方法で、権利プロファイルでスクリプトに特権を追加できます。権利プロファイルが割り当てられたユーザーまたは役割がプロファイルシェルでスクリプトを実行すると、スクリプトは、それらの追加された特権で実行されます。スクリプトに特権が必要なコマンドが含まれている場合は、特権が追加されたコマンドも、割り当てられた権利プロファイルに含まれている必要があります。例については、94 ページの「アプリケーションおよびスクリプトへの権利の割り当て」を参照してください。

拡張特権ポリシーを使用した特権使用の制限

拡張特権ポリシーは、基本特権および明示的に付与した特権を除き、ポート、ユーザー ID、またはファイルオブジェクトへのアクセスを制限できます。少ない数の特権では、システムを攻撃する目的でリソースを容易に使用することはできません。実際に、ユーザーは所有するファイルとディレクトリを、悪意のあるプロセスによるアクセスから保護できます。拡張特権ポリシーの例については、93 ページの「アプリケーション、スクリプトおよびリソースの特定の権利への制限」を参照してください。

特権エスカレーションとユーザー権利

では、管理者がセキュリティーを構成するとき、高い柔軟性が提供されます。 このソフトウェアがインストールされていると、特権エスカレーションが防止されます。特権エスカレーションは、意図していたよりも多くの管理権利がユーザーまたはプロセスに与えられたときに発生します。この場合「特権」とはカーネル特権だけではなく、すべての権利を意味します。38 ページの「特権エスカレーションとカーネル特権」を参照してください。

ソフトウェアには、`root` 役割にのみ割り当てられる権利が含まれています。ほかのセキュリティー保護が存在する状態で、`root` 役割用に設計された属性を管理者がほかのアカウントに割り当てる可能性がありますが、このような割り当ては慎重に行う必要があります。

次に示す権利プロファイルと一連の承認により、root 以外のアカウントの特権がエスカレートされる可能性があります。

- **Media Restore 権利プロファイル** – このプロファイルはほかのどの権利プロファイルにも含まれていません。Media Restore はルートファイルシステム全体へのアクセスを提供するため、これを使用することで特権のエスカレーションが可能です。故意に改ざんされたファイルや交換したメディアを復元できます。デフォルトでは、`root` 役割にはこの権利プロファイルが含まれています。
- `solaris.*.assign` **承認** – これらの承認はどの権利プロファイルにも割り当てられていません。`solaris.*.assign` 承認を持つアカウントは、そのアカウント自体に割り当てられていない権利をほかのユーザーに割り当てることができます。たとえば、`solaris.profile.assign` 承認を持つ役割は、その役割自体に割り当てられていない権利プロファイルをほかのアカウントに割り当てることができます。デフォルトでは、`solaris.*.assign` 承認を持つのは `root` 役割だけです。

  `solaris.*.assign` 承認ではなく `solaris.*.delegate` 承認を割り当てます。`solaris.*.delegate` 承認を使用すると、委託者は、その委託者が所有する権利のみをほかのアカウントに割り当てることができます。たとえば、`solaris.profile.delegate` 承認が割り当てられた役割は、その役割自体に割り当てられている権利プロファイルをほかのユーザーや役割に割り当てることができます。

カーネル特権のエスカレーションの防止については、38 ページの「特権エスカレーションとカーネル特権」を参照してください。

特権エスカレーションとカーネル特権

カーネルにより特権エスカレーションが防止されます。プロセスが必要な特権以外の特権を取得することを防ぐために、無防備なシステム変更に特権の完全セットがあるかどうかが確認されます。たとえば、`root` (`UID=0`) が所有するファイルまたはプロセスは、特権の完全セットを備えたプロセスによってのみ変更できます。`root` アカウ

ントは、特権がなくても root が所有するファイルを変更することができます。しかし、root ユーザー以外は、root が所有するファイルを変更するにはすべての特権が必要です。

同様に、デバイスへのアクセスを提供する操作には、有効なセットのすべての特権が必要です。

特に file_chown_self および proc_owner は、特権エスカレーションが生じやすい特権です。

- file_chown_self は、プロセスがそのファイルを渡せるようにする特権です。proc_owner は、プロセス自身が所有しないプロセスを調査できるようにする特権です。

  file_chown_self 特権は、rstchown システム変数によって制限されます。rstchown 変数が 0 に設定されると、file_chown_self 特権は、システムの全ユーザーの初期継承可能セットから削除されます。rstchown システム変数の詳細は、chown(1) のマニュアルページを参照してください。

  file_chown_self 特権は、特定のコマンド、権利プロファイルに配置されているコマンド、および役割または信頼できるユーザーに割り当てられているプロファイルにもっとも安全に割り当てることができます。
- proc_owner 特権は、プロセス UID を 0 にするには十分ではありません。任意の UID のプロセスを UID=0 にするには、すべての特権が必要です。proc_owner 特権はシステム上のすべてのファイルに無制限の読み取りアクセス権を与えるので、この特権の特定コマンドへの割り当て、プロファイルに配置されているコマンドへの割り当て、および役割に割り当てられているプロファイルへの割り当てはもっとも安全に行います。

注意 - file_chown_self 特権または proc_owner 特権がユーザーの初期継承可能セットに含まれるように、ユーザーのアカウントを構成できます。ただし、このような強力な特権をユーザーや役割の継承可能セットに配置するには、セキュリティー上の相応の理由がなければなりません。

デバイスでの特権エスカレーションを防止する方法については、34 ページの「特権とデバイス」を参照してください。一般的な説明については、privileges(7) のマニュアルページを参照してください。

権利の検証

割り当てられた権利を評価するかどうかは、プロセスが実行されるシェル、ネームサービスのスコープ、および検索順序の影響を受けます。権利を評価できないプロセスは失敗します。権利割り当ての確認の補足情報については、143 ページの「RBAC と特権のトラブルシューティング」を参照してください。

プロファイルシェルと権利の検証

ユーザーと役割は、プロファイルシェルから特権付きアプリケーションを実行できます。プロファイルシェルは、権利を認識する特殊シェルです。管理者は、プロファイルシェルをログインシェルとしてユーザーに割り当てることができます。そうでない場合は、そのユーザーが役割を引き受けるために `pfexec` コマンドまたは `su` コマンドを実行したときに、プロファイルシェルが起動されます。 では、どのシェルにも、対応するプロファイルシェルがあります。プロファイルシェルの一覧については、`pfexec(1)` のマニュアルページを参照してください。

権利プロファイルが直接割り当てられており、ログインシェルがプロファイルシェルではないユーザーが、割り当てられている特権付きコマンドを実行するには、プロファイルシェルを開く必要があります。認証権利プロファイルが割り当てられているユーザーと役割は、コマンド実行前に認証 (パスワード入力) するように求められます。操作性とセキュリティーに関する考慮事項については、43 ページの「権利の割り当てにおける考慮事項」を参照してください。

ネームサービススコープと権利の検証

ネームサービススコープは、割り当てられている権利がいつ使用可能になるかに影響します。役割の適用範囲は、個々のホストに限定されることがあります。また、LDAP などのネームサービスからサービスを受けるすべてのホストが適用範囲に含まれることもあります。あるシステムのネームサービスの適用範囲は、ネームスイッチサービス `svc:/system/name-service/switch` で指定されます。検索は、最初に一致した時点で停止します。たとえば、権利プロファイルが 2 つのネームサービススコープに存在する場合、最初のネームサービススコープに含まれるエントリだけが使用されます。最初に一致したものが `files` の場合、役割の適用範囲はローカルホストに限定されます。ネームサービスについては、`nsswitch.conf(5)` のマニュアルページ、『Oracle Solaris 12 ディレクトリサービスとネームサービスでの作業: DNS と NIS』、および『Oracle Solaris 12 ディレクトリサービスとネームサービスでの作業: LDAP』を参照してください。

割り当てられた権利の検索順序

ユーザーまたは役割に対し、セキュリティー属性を直接割り当てるか、または権利プロファイルを介して割り当てることができます。検索の順序は、使用されるセキュリティー属性の値に影響を及ぼします。その属性の最初に見つかったインスタンスの値が使用されます。

注記 - 承認の順序は重要ではありません。承認は累積されます。

ユーザーがログインすると、次に示す検索順序で権利が割り当てられます。

- `useradd` および `usermod` コマンドを使ってユーザーに直接割り当てられる**権利**。可能な権利割り当てのリストについては、158 ページの「`user_attr` データベース」を参照してください。
- **useradd** および `usermod` コマンドを使ってユーザーに割り当てられる**権利プロファイル**。これらの割り当ては順番に検索されます。
  - 最初に、認証権利プロファイルが検索されます。

    この順序は、認証プロファイルリストの最初のプロファイル、その補助プロファイル、認証プロファイルリストの 2 番目のプロファイル、その補助プロファイル、のようになります。累積される `auths` 値を除き、最初のインスタンスの値がシステムで使用される値になります。権利プロファイルに割り当てることができる属性には、ユーザーに割り当てることができるすべての権利と、補助プロファイルが含まれます。リストについては、158 ページの「`user_attr` データベース」を参照してください。
  - 次に、再認証を必要としない権利プロファイルが同様の方法で検索されます。
- **Console User 権利プロファイル**の値。詳細は、154 ページの「権利プロファイルのリファレンス」を参照してください。
- **Stop 権利プロファイル**が割り当てられた場合、セキュリティー属性の評価は停止します。Stop プロファイルが割り当てられたあとは属性は一切割り当てられません。Stop プロファイルは、Console User 権利プロファイルのあと、ほかのセキュリティー属性 (`authorizations_granted` など) の前に評価されます。詳細は、154 ページの「権利プロファイルのリファレンス」を参照してください。
- `rbac/default_profiles astring Basic\ Solaris\ User`
- `rbac/default_authorizations`
- `rbac/default_auth_profiles`
- `rbac/default_profiles`
- `rbac/default_privileges`
- `rbac/default_limit_privileges`

権利を確認するアプリケーション

システム制御をオーバーライドするアプリケーションとコマンドは、特権付きアプリケーションとみなされます。アプリケーションは、`UID=0` のようなセキュリティー属性、特権、および承認によって特権化されます。

UID と GID を確認するアプリケーション

root (UID=0) またはその他の特殊な UID または GID を確認する特権付きアプリケーションは、長期にわたって UNIX 環境に存在しています。権利プロファイルのメカニズムによって、特定の ID を必要とするコマンドを分離できます。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、UID が割り当てられたコマンドを権利プロファイル内に配置できます。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、スーパーユーザー以外でもその UID としてプログラムを実行できます。

ID は実 ID または 実効 ID として指定できます。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、ファイルアクセス権ビットの setuid 機能に相当します。実行 ID は、監査のために UID の識別も行います。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、reboot コマンドには実効 UID ではなく、実 UID が必要です。

ヒント - あるコマンドを実行するために実効 ID では十分でない場合は、そのコマンドに実 ID を割り当てます。

特権を確認するアプリケーション

特権付きアプリケーションは、特権の使用を確認できます。権利プロファイルメカニズムを使用すると、セキュリティー属性を必要とする特定のコマンドの特権を指定できます。次に、セキュリティー属性が割り当てられたコマンドを権利プロファイル内に分離できます。この権利プロファイルを持つユーザーまたは役割は、そのコマンドに必要な特権だけを使用してコマンドを実行できます。

特権を確認するコマンドとして次のようなものがあります。

- Kerberos コマンド (kadmin、kprop、kdb5_util など)
- ネットワークコマンド (ipadm、routeadm、snoop など)
- ファイルコマンドとファイルシステムコマンド (chmod、chgrp、mount など)
- プロセスを制御するコマンド (kill、pcred、rcapadm など)

特権を持つコマンドを権利プロファイルに追加するには、115 ページの「権利プロファイルを作成する方法」および profiles(1) のマニュアルページを参照してください。特定のプロファイル内の特権を確認するコマンドを判断するには、第7章「 の権利の一覧表示」を参照してください。

承認を確認するアプリケーション

次を含む一部の コマンドは、承認を確認します。

- 監査管理用のコマンド (`auditconfig`、`auditreduce` など)
- プリンタ管理用のコマンド (`cupsenable`、`lpadmin` など)
- バッチジョブコマンド (`at`、`atq`、`batch`、`crontab` など)
- デバイス向けのコマンド (`allocate`、`deallocate`、`list_devices`、`cdrw` など)

スクリプトまたはプログラムでの承認の確認のガイダンスについては、例40「スクリプトまたはプログラム内の承認の確認」を参照してください。承認が必要なプログラムを作成するには、『Oracle Solaris 12 セキュリティーサービス開発ガイド』 の 「承認について」を参照してください。

権利の割り当てにおける考慮事項

セキュリティーと操作性の問題が、管理者による役割の割り当て方法に影響する可能性があります。

権利の割り当てにおけるセキュリティーに関する考慮事項

一般に、ユーザーまたは役割は権利プロファイルを介して管理権利を取得しますが、権利を直接割り当てることもできます。

- 役割とユーザーには、特権を直接割り当てることができます。

  特権の割り当てを直接行うことは安全とは言えません。特権が直接割り当てられたユーザーと役割は、カーネルがその特権を要求するときにはいつでもセキュリティーポリシーをオーバーライドできます。また、ユーザーまたは役割のプロセスに損害を与える悪質なプロセスが、カーネルでこの特権が必要な場合はいつでもこの特権を使用できます。

  安全なやり方は、特権をコマンドのセキュリティー属性として権利プロファイル内で割り当てる方法です。そうすると、その特権は、その権利プロファイルを持つユーザーが、そのコマンドでのみ使用できます。
- 役割とユーザーには、承認を直接割り当てることができます。

  承認はユーザーレベルで評価されるため、承認の直接割り当ては特権の直接割り当てよりリスクが小さいと言えます。しかし、承認が与えられることで、ユーザーは監査フラグの割り当てなどの高いセキュリティーが求められるタスクも実施でき

るようになります。セキュリティー強化のため、コマンド実行前にユーザーがパスワードを入力する必要がある認証権利プロファイル内で、承認を割り当てます。

権利の割り当てにおける操作性に関する考慮事項

権利を直接割り当てると、操作性に影響を及ぼす可能性があります。

- 直接割り当てられた承認、およびユーザーの権利プロファイル内のコマンドと承認を有効にするには、プロファイルシェルでこれらを解釈する必要があります。デフォルトでは、ユーザーにはプロファイルシェルが割り当てられません。したがってユーザーは忘れずにプロファイルシェルを開き、そのシェルでコマンドを実行する必要があります。
- 承認を個々に割り当てる方法には拡張性がありません。また、直接割り当てられた承認は、タスクを実行するには十分でない可能性があります。タスクに特権付きコマンドが必要な場合もあります。

  権利プロファイルは、承認と特権付きコマンドをまとめるように設計されています。また、ユーザーグループに合わせて適切に拡大縮小します。

♦ ♦ ♦ 第 2 章

2 管理権利構成の計画

この章では、システムの管理に従来の権利モデルを使用するか、または の権利モデルを活用するかを決定する上で役立つ情報を提供します。この章の内容は次のとおりです。

- 45 ページの「管理に使用する権利モデルの決定」
- 46 ページの「選択した権利モデルへの準拠」

権利の概要については、19 ページの「ユーザー権管理」を参照してください。参照情報については、第9章「 権利リファレンス」を参照してください。

管理に使用する権利モデルの決定

の権利には、権利プロファイル、承認、および特権が含まれます。 ではさまざまな方法でシステムの管理権利を構成できます。

次のリストでは、セキュリティーがもっとも高いモデルから、セキュリティーが低い従来のスーパーユーザーモデル の順に示します。

1. それぞれが限られた権利を持つ複数の信頼できるユーザーの間で管理タスクが分けられます。この方式は の権利モデルです。

   この方式を実施する方法については、46 ページの「選択した権利モデルへの準拠」を参照してください。

   この方式の利点については、第1章「権利を使用したユーザーとプロセスの制御について」を参照してください。
2. デフォルトの権利構成を使用します。この方式では権利モデルが使用されますが、モデルはサイトに合わせてカスタマイズされません。

   デフォルトでは、初期ユーザーはいくつかの管理権利を持ち、`root` 役割を引き受けることができます。`root` 役割は、オプションで別の信頼できるユーザーに `root` 役割を割り当てることができます。セキュリティーを強化するため、`root` 役割は管理コマンドの監査を有効にすることがあります。

   このモデルを使用する管理者にとって役立つタスクを次に示します。

   - 110 ページの「割り当てられている管理権利の使用」

- 49 ページの「ユーザーへの権利の割り当て」
- 114 ページの「管理アクションの監査」
- 61 ページの「役割のパスワードの変更」
- 第7章「 の権利の一覧表示」

3. sudo コマンドを使用します。

   sudo コマンドを使い慣れている管理者が sudo を構成して使用できます。オプションで、sudo ユーザーが一定期間にわたり再認証なしで管理コマンドを実行できるように /etc/sudoers ファイルを構成できます。

   sudo のユーザーにとって役立つタスクを次に示します。

   - 110 ページの「割り当てられている管理権利の使用」
   - 114 ページの「管理アクションの監査」
   - 例45「役割の使用を容易にするために認証をキャッシュする」

   sudo コマンドは、 RBAC コマンドと同等の Linux コマンドです。RBAC コマンドとは異なり、sudo は権利プロファイルを参照できません。これは、/etc/sudoers ファイル内で各プログラムに対して指定されている権利を現在のユーザーに付与できるように、すべての特権を持つ root として実行されます。詳細は、sudo(8) および sudoers(4) のマニュアルページを参照してください。

4. root 役割をユーザーに変更することによってスーパーユーザーモデルを使用します。

   従来の UNIX モデルを使用する管理者は、120 ページの「root 役割をユーザーに変更する方法」を完了する必要があります。オプションで root ユーザーは監査を構成できます。

選択した権利モデルへの準拠

ユーザーおよびプロセスの権利の管理は、システム配備の管理に不可欠となることがあります。計画を行うには、組織のセキュリティー要件に関する詳細な知識と、 での権利を理解しておく必要があります。このセクションでは、サイトでの権利使用の計画の一般的なプロセスを説明します。

1. 権利に関する基本概念を理解します。

   第1章「権利を使用したユーザーとプロセスの制御について」を参照してください。権利を使用したシステムの管理は、従来の UNIX 管理方法を使用する場合と大幅に異なります。

2. セキュリティーポリシーを検査します。

   組織のセキュリティーポリシーでは、システムに対する潜在的な脅威を詳細に記述し、各脅威のリスクを評価して、それらの脅威に対抗するための方策を提供し

ます。この戦略の一環として、権利を使用してセキュリティー関連タスクを切り離すことがあります。

たとえば、サイトでセキュリティー管理とセキュリティー以外の管理を切り分ける必要がある場合などです。責務分離を実装するには、例5「責務を分離するための役割の作成」を参照してください。『Oracle Solaris 11.4 Security and Hardening Guidelines』 の 付録 A, 「Site Security Policy and Enforcement,」も参照してください。

ユーザーおよび役割に対し、ログインを注釈として示すようにサイトで要求されることがあります。これらの注釈は監査トレールに表示されます。詳細は、『Managing Auditing in Oracle Solaris 11.4』 の 「New Feature – Annotating Reason for Access in the Audit Record」を参照してください。

セキュリティーポリシーで Authorization Rules Managed On RBAC (ARMOR) を使用する場合は、ARMOR パッケージをインストールして使用する必要があります。でのその使用については、例2「ARMOR の役割の使用」を参照してください。

3. デフォルトの権利プロファイルを確認します。

   デフォルトの権利プロファイルには、タスクの実行に必要な権利がまとめられています。使用可能な権利プロファイルを確認するには、135 ページの「権利プロファイルの一覧表示」を参照してください。

4. 役割を使用するか、または権利プロファイルをユーザーに直接割り当てるかを決定します。

   役割では、権利の管理が容易になります。役割名は、その役割が実行できるタスクを識別し、ユーザー権利から役割権利を切り離します。役割を使用する場合には 3 つのオプションがあります。

   - ARMOR パッケージをインストールできます (この場合 Authorization Roles Managed on RBAC (ARMOR) 標準により定義される 7 つの役割がインストールされます)。例2「ARMOR の役割の使用」を参照してください。
   - ユーザー独自の役割を定義したり、ARMOR 役割を使用したりできます。54 ページの「役割の作成」および例2「ARMOR の役割の使用」を参照してください。
   - ユーザー独自の役割を定義し、ARMOR 役割は使用しないでおくことができます。54 ページの「役割の作成」を参照してください。

   サイトで役割が不要な場合は、権利プロファイルをユーザーに直接割り当てることができます。ユーザーが各自の権利プロファイルから管理タスクを実行するときにパスワードを求めるには、認証権利プロファイルを使用します。例13「ユーザーに対し DHCP 管理の前にパスワード入力を求める」を参照してください。

5. 追加の権利プロファイルを作成する必要が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。セキュリティーに影響するアプリケーション、サービス拒否の問題を発生させ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、権利を使用することをお勧めします。たとえば Sun Ray システムのユーザーには、すべての基本特権が必要であるわけではありません。ユーザーを

制限する権利プロファイルの例については、例30「権利プロファイルからの基本特権の削除」を参照してください。

a. 新しいタスクに必要な権利を決定します。
b. 既存の権利プロファイルがこのタスクに対して適切であるかどうかを判断します。
c. コマンドがその必要な特権を使用して実行されるように権利プロファイルを順序付けます。

   順序付けについては、40 ページの「割り当てられた権利の検索順序」を参照してください。

6. どのユーザーにどの権利を割り当てるかを決定します。

   principle of least privilegeに従って、ユーザーの信頼レベルに適した役割にユーザーを割り当てます。実行する必要のないタスクをユーザーが実行できないようにすると、問題が発生する可能性が減少します。

注記 - システムのすべてのユーザーに適用される権利は、`account-policy:default` SMF サービス内で設定および指定できます。詳細は、`account-policy(8S)` のマンページおよび 49 ページの「ユーザーへの権利の割り当て」を参照してください。

計画が完成したら、権利プロファイルまたは役割を割り当てることができる信頼できるユーザーのログインを作成します。ユーザーの作成に関する詳細は、『Managing User Accounts and User Environments in Oracle Solaris 11.4』 の 「Setting Up and Managing User Accounts (Task Map)」を参照してください。

権利を割り当てるには、49 ページの「ユーザーへの権利の割り当て」の手順から開始します。以降のセクションでは、権利の拡張、権利の制限、リソースへの権利の割り当て、および権利割り当てのトラブルシューティングの例を示します。

♦ ♦ ♦ 第 3 章

3 での権利の割り当て

この章では、ユーザーと役割に権利を割り当てるためのタスクについて説明します。この章の内容は次のとおりです。

- 49 ページの「ユーザーへの権利の割り当て」
- 62 ページの「ユーザーの権利の拡張」
- 68 ページの「ユーザーの権利の制限」
- 82 ページの「SMF プロパティーとしてのシステム全体での権利の変更」

権利の概要については、19 ページの「ユーザー権管理」を参照してください。参照情報については、第9章「 権利リファレンス」を参照してください。

注記 - この章は、『Oracle Solaris 11.4 Security and Hardening Guidelines』 の 「Passwords and Password Policy」に記載されているように、ユーザーパスワードおよび役割パスワードが保護されていることを前提としています。

ユーザーへの権利の割り当て

の権利はすべてのプロセスに存在します。ユーザーと役割に権利を追加したり、権利を削除したりできます。権利には、ユーザープロセスの特権、ユーザーが実行するコマンドの特権または特別な ID、および特定アクションの実行のための承認が含まれます。権利割り当てに伴う管理作業の負担を軽減するため、 ではサービスと管理アクションに関する権利が権利プロファイルにまとめられています。個別の権利をユーザーと役割に割り当てる代わりに、権利プロファイルに権利を集めることができます。次に、その権利プロファイルをユーザーと役割に割り当てることができます。

役割により、ユーザーが実行できる管理タスクに `auditadm` などの名前が指定されます。管理アクションを実行するため、ユーザーはそのアクションの実行のために割り当てられている役割を引き受けます。役割はセキュリティーポリシーで必要とされることがあり、単純に便利です。役割を作成するか、または 7 つの役割とそのローカルホームディレクトリを作成する `armor` パッケージをインストールできます。役割の詳

細については、20 ページの「スーパーユーザーモデルの代替としてのユーザー権利およびプロセス権利」を参照してください。

権利をシステムに割り当て、そのシステムにログインしたすべてのユーザーにそれらの権利を付与できます。通常の場合、管理者はキオスクやほかのシステムの管理のみを目的としたシステムなどの特殊なシステムから権利を削除します。システムの権利を変更するための推奨される方法は、`account-policy` サービス管理機能 (SMF) サービスを有効にしてシステムのセキュリティー属性を SMF プロパティーとして変更することです。レガシー方式は、`/etc` ディレクトリ内の個々のファイルを編集することです。

SMF 内のセキュリティー属性を変更するには、ローカルファイルを編集する代わりに `setprop` コマンドを使用します。

```
example-11u4 $ pfbash svccfg -s account-policy:default  \
  setprop config/etc_security_policyconf/disabled = boolean: false
example-11u4 $ svccfg -s svc:/system/account-policy:default \
  setprop rbac/default_privileges astring: = "basic,!file_link_any"
```

前のコマンドはこのレガシー方式を置き換えます。

```
example-11u3 $ pfexec vim /etc/security/policy.conf
PRIV_DEFAULT=basic,!file_link_any
```

注記 - は `account-policy:default` サービスをデフォルトでは有効にしません。ただし、システム、ユーザー、および役割セキュリティーを管理するにはこれを有効にして SMF を使用する必要があります。セキュリティーポリシーファイルの編集は非推奨です。

詳細は、`account-policy(8S)` および 88 ページの「システム全体の特権、承認、および権利プロファイルの変更」を参照してください。

役割を割り当てることができるユーザー

まず、役割を割り当てるには `root` 役割である必要があります。

`root` 役割により、管理タスクが信頼できるユーザーとしてユーザーに配布されているか、または役割をユーザーに割り当てることで管理タスクが割配布されている場合、次に示す権利プロファイルが割り当てられると、ユーザーと役割の作成またはユーザーと役割への権利の割り当てが可能になります。

- ユーザーまたは役割を作成するには、User Management 権利プロファイルが割り当てられている管理者になる必要があります。
- ユーザーまたは役割にほとんどの権利を割り当てるには、User Security 権利プロファイルが割り当てられている管理者になる必要があります。

  監査フラグを割り当てることはできません。ユーザーまたは役割に監査フラグを割り当てることができるのは `root` 役割だけです。

役割のパスワードは変更できません。役割のパスワードを変更できるのは root 役割だけです。

管理権利が割り当てられている場合は、管理コマンドを実行する前に110 ページの「割り当てられている管理権利の使用」を参照してください。

管理者に割り当てる権利の判断

管理者は、特権付きコマンドを実行する権利を要求し、多くの場合、そのコマンドを実行する承認を要求します。権利プロファイルは簡便なバンドルで、特権付きコマンド、承認、場合によっては補助権利プロファイルも提供します。

どの権利プロファイルを割り当てるのが最適かを判断する方法は複数あります。権利プロファイルの名前はその機能を示しているので、プロファイル名を一覧表示して、機能領域を検索できます。コマンド名から開始して、どの権利プロファイルにそのコマンドが含まれているかを判断することもできます。

対象のコマンドを含む権利プロファイルの名前がわかっている場合は、その権利プロファイル内の権利を確認するときに、管理者にその特定のプロファイルを割り当てるかどうかを判断できます。個々の特権または承認を管理者に割り当てないでください。詳細は、43 ページの「権利の割り当てにおける考慮事項」を参照してください。

管理権利を割り当てたあとで、管理コマンドの実行前に110 ページの「割り当てられている管理権利の使用」を参照するように管理者に伝えてください。

▼ 割り当てる権利を判断する方法

権利プロファイルまたはコマンド名で開始することにより、割り当てる権利を検索できます。この手順は、権利プロファイルを使用して検索する方法を示します。例1「コマンドが必要とする権利の判断」ではコマンドを使用して検索する方法を示しています。

1. **使用可能な権利プロファイルを一覧表示します。**

```
$ profiles -a | more
...
Administrative Command History
Administrator Message Edit
Audit Configuration
...
```

2. **機能領域を検索します。**

次の例では、ゾーンの管理に関する権利プロファイルを検索します。

```
$ profiles -a | grep -i zone
```

```
Zone Security
Zone Configuration
Zone Management
Zone Migration
Zone Cold Migration
```

3. **割り当てる予定の権利をもっとも適切に表している権利プロファイルの内容を確認します。**

   引き続きゾーンの例で、ゾーンをセキュリティー保護する権利を割り当てます。

```
$ profiles -p "Zone Security" info
        name=Zone Security
        desc=Zones Virtual Application Environment Security
        auths=solaris.zone.*,solaris.auth.delegate
        cmd=/usr/sbin/txzonemgr
        cmd=/usr/sbin/zonecfg
        cmd=/usr/lib/rad/module/mod_zonemgr.so.1
```

   この出力は、割り当て対象者が、`solaris.zone` の文字列から始まるすべての承認と、`solaris.auth.delegate` 承認を持っていることを示しています。割り当て対象者は、`txzonemgr` および `zonecfg` コマンドを実行でき、RAD コマンド `mod_zonemgr.so.1` モジュールを使用できます。

   コマンドに割り当てる権利の詳細は、次のステップに進んでください。`solaris.zone` 承認の説明については、ステップ 5を参照してください。

4. **特権付きコマンドデータベースでコマンドを検索します。**

```
$ getent exec_attr | grep "^Zone Security"
Zone Security:solaris:cmd:RO::/usr/sbin/txzonemgr:uid=0
Zone Security:solaris:cmd:RO::/usr/sbin/zonecfg:uid=0
Zone Security:solaris:cmd:RO::/usr/lib/rad/module/mod_zonemgr.so.1:uid=0
```

   この出力は、割り当て対象者の UID ではなく、`0` の UID でコマンドが実行することを示しています。`RO` はこの権利プロファイルが読み取り専用であることを示します。

5. **(オプション) 選択した権利プロファイルにある承認の定義を確認します。**

```
$ getent auth_attr | grep solaris.zone
solaris.zone.:RO::Zone Management::
solaris.zone.clonefrom:RO::Clone another Zone::
solaris.zone.login:RO::Zone Login::
solaris.zone.manage:RO::Zone Deployment::
solaris.zone.config:RO::Modify the Persistent Zone Configuration::
solaris.zone.liveconfig:RO::Inspect and Modify the Live Zone Configuration::
solaris.zone.migrate:RO::Zone Migration::
solaris.zone.migrate.cold:RO::Zone Cold Migration::
$ getent auth_attr | grep solaris.auth.delegate
solaris.auth.delegate:RO::Assign owned authorizations::
```

例 1 コマンドが必要とする権利の判断

この例では、管理者は、`pfctl` コマンドをネットワーク管理者に割り当てようと考えているが、割り当て対象者にはパケットフィルタ (PF) ファイアウォールの処理に必要とするそのほかの権利がわかっていません。

1. 管理者は、特権付きコマンドデータベース exec_attr で pfctl コマンドを検索します。

```
$ getent exec_attr | grep pfctl
Network Firewall Management:solaris:cmd:RO::/usr/sbin/pfctl:privs=sys_ip_config
```

   この出力は、pfctl コマンドがネットワークファイアウォール管理権利プロファイルの一部であり、sys_ip_config 特権で実行することを示しています。

2. 管理者は、権利プロファイルの内容を確認します。

```
$ profiles -p "Network Firewall Management" info
        name=Network Firewall Management
        desc=Firewall Administration
        auths=solaris.smf.value.network.firewall,solaris.smf.manage.network.firewall
        cmd=/usr/sbin/pfconf
        cmd=/usr/sbin/pfctl
```

   この出力は、ネットワークファイアウォール管理プロファイルが、ファイアウォールの SMF プロパティーを変更することを割り当て対象者に承認し、pfconf コマンドも含んでいることを示しています。

3. 管理者は、特権付きコマンドデータベースで pfconf コマンドを検索します。

```
$ getent exec_attr | grep pfconf
Network Firewall Management:solaris:cmd:RO::/usr/sbin/pfconf:privs=sys_ip_config
```

4. 管理者は、選択したプロファイルにある承認の定義を確認します。

```
$ getent auth_attr | grep firewall
solaris.smf.manage.network.firewall:RO::Manage Network Firewall::
solaris.smf.value.network.firewall:RO::Change Network Firewall Configuration::
solaris.smf.manage.firewall:RO::Manage Firewall Service::
solaris.smf.value.firewall.config:RO::Change Service Firewall Config::
```

5. 割り当て対象者が必要とするすべての機能が権利プロファイルに含まれる場合、管理者は、その権利プロファイルをユーザーに割り当てたり、役割を作成してその役割をユーザーに割り当てたりします。例については、54 ページの「役割の作成」および 例12「DHCP を管理する信頼できるユーザーの作成」を参照してください。

6. 割り当て対象者がさらに多くのネットワーク機能を必要とする場合、管理者は確認を続けます。

   管理者は、すべてのネットワークプロファイルを一覧表示し、別のプロファイルを選択して、検索を繰り返します。

```
$ profiles -a | grep ^Network
Network Autoconf Admin
Network Autoconf User
Network ILB
Network Dot1x Management
Network LLDP
```

```
Network VRRP
Network DLMP
Network Management
Network Observability
Network TCP Key Management
Network Security
Network Wifi Management
Network Wifi Security
Network Link Security
Network IPsec Management
Network Firewall Management
```

管理者は、115 ページの「権利プロファイルと承認の作成」の指示に従って、カスタムネットワーキング権利プロファイルを作成することもできます。

ユーザーおよび役割への権利の割り当て

このセクションでは、役割とユーザーを作成および変更するコマンドについて説明します。権利プロファイルを作成または変更するには、115 ページの「権利プロファイルを作成する方法」および116 ページの「システム権利プロファイルをクローニングおよび変更する方法」を参照してください。

役割の詳細については、22 ページの「ユーザー権利およびプロセス権利の基本情報」を参照してください。

役割とユーザーの作成または変更における主なアクションを次に示します。

- 役割の作成
- 追加権利によって信頼されるユーザーの作成
- 役割の権利の変更
- ユーザーの権利の変更
- ユーザー独自のパスワードを使用した役割の引き受けの有効化
- 役割のパスワードの変更
- 役割の削除

役割の作成

役割を使用する場合にはさまざまなオプションがあります。ARMOR から事前定義の役割をインストールし、排他的に使用できます。役割を作成することも可能です。ARMOR の役割の使用と、作成した役割を組み合わせることもできます。

ARMOR 役割を使用するには、例2「ARMOR の役割の使用」を参照してください。

ユーザー独自の役割を作成するには、roleadd コマンドを使用します。このコマンドのすべての引数のリストについては、roleadd(8) のマニュアルページを参照してください。

注意 - roleauth=user キーワードと auth_profiles=*profiles* キーワードの両方を使用して役割を構成しないでください。auth_profiles キーワードは、roleauth パスワード (ユーザー) ではない現在のプロセス所有者のパスワード (役割) に対して認証されるため、認証が失敗します。

たとえば、次のコマンドは、ホームディレクトリと pfbash ログインシェルでローカルユーザー管理者の役割を作成し、その役割のパスワードを作成します。

```
# roleadd -c "User Administrator role, local" \
-m -K profiles="User Security,User Management" accountadm
80 blocks
# ls /export/home/accountadm
local.bash_profile     local.login     local.profile
# passwd accountadm
Password: xxxxxxxx
Confirm Password: xxxxxxxx
```

各表記の意味は次のとおりです。

-c *comment*	役割を記述します。
-m	ホームディレクトリを作成します。
-K profiles=	1 つまたは複数の権利プロファイルを役割に割り当てます。権利プロファイルのリストについては、135 ページの「権利プロファイルの一覧表示」を参照してください。
rolename	役割の名前です。許容される文字列の制限については、roleadd(8) のマニュアルページを参照してください。

注記 - 役割アカウントを複数のユーザーに割り当てることができます。そのため、管理者は役割パスワードを作成し、その役割パスワードを通常の通信手段以外の手段でユーザーに伝えます。役割のパスワードの代替機能については、60 ページの「役割パスワードでのユーザー独自のパスワード使用の有効化」、例18「役割パスワードでのユーザー独自のパスワード使用の有効化」、および例20「ユーザーが役割パスワードとして独自のパスワードを使用できるようにするための権利プロファイルの変更」を参照してください。

例 2 ARMOR の役割の使用

この例では、セキュリティー管理者が ARMOR 標準により定義されている役割をインストールします。管理者はまず、役割名が既存のどのアカウントとも競合していない

ことを確認してから、パッケージをインストールし、役割定義を表示して、信頼できるユーザーに役割を割り当てます。

管理者は最初に、次の UID と名前がネームサービスに存在していないことを確認します。

- 57 auditadm
- 55 fsadm
- 58 pkgadm
- 53 secadm
- 56 svcadm
- 59 sysop
- 54 useradm

管理者は UID と名前が使用されていないことを確認してから、パッケージをインストールします。

```
# pkg install system/security/armor
```

このパッケージは、7 つの役割とローカルホームディレクトリを /export/home ディレクトリに作成します。

各役割の権利を確認するため、管理者は各役割に割り当てられているプロファイルを一覧表示できます。

```
# profiles auditadm
# profiles fsadm
# profiles pkgadm
# profiles secadm
# profiles svcadm
# profiles sysop
# profiles useradm
```

ARMOR の役割に割り当てられた権利は変更できません。権利の別の構成を作成するには、116 ページの「システム権利プロファイルをクローニングおよび変更する方法」を参照してください。

最後に、管理者は信頼できるユーザーに役割を割り当てます。役割の認証にはユーザー独自のパスワードが使用されます。一部のユーザーには複数の役割が割り当てられます。時間的制約のあるタスクを持つ役割は、複数の信頼できるユーザーに割り当てられます。

```
# usermod -R=auditadm adoe
# usermod -R=fsadm,pkgadm bdoe
# usermod -R=secadm,useradm cdoe
# usermod -R=svcadm ddoe
# usermod -R=svcadm edoe
# usermod -R=sysop fdoe
# usermod -R=sysop gdoe
```

例 3 アプリケーション管理者の役割の作成

管理者は、Oracle データベース管理者 (DBA) の役割を作成することにより、その管理者のログイン名が確実に監査トレールに含まれるようにします。`oracle` アカウントは役割であるため、ユーザーは自分のアカウントにログインしたあと、`oracle` 役割にユーザーを切り替える必要があります。このサイトには 3 人の DBA が存在します。

```
# usermod -K type=role oracle
# usermod -R oracle adoe
# usermod -R oracle bdoe
# usermod -R oracle cdoe
```

例 4 LDAP リポジトリでの User Administrator 役割の作成

管理者は LDAP 内に User Administrator 役割を作成します。ユーザーは役割を引き受けるときにパスワードを入力するため、個々のコマンドにパスワードを入力する必要がなくなります。

```
# roleadd -c "User Administrator role, LDAP" -m -S ldap \
-K profiles="User Security,User Management" accountadm
```

例 5 責務を分離するための役割の作成

管理者は 2 つの役割を作成します。`usermgt` 役割は、ユーザーを作成したり、ユーザーにホームディレクトリを提供したり、その他のセキュリティー以外のタスクを実行したりできます。`usersec` 役割は、ユーザーの作成はできませんが、パスワードの割り当てとほかの権利の割り当て変更は実行できます。いずれの役割でも、ユーザーまたは役割の監査フラグの設定や、役割のパスワードの変更はできません。これらのアクションは `root` 役割が実行する必要があります。

```
# roleadd -c "User Management role, LDAP" -s /usr/bin/pfksh \
-m -S ldap -K profiles="User Management"  usermgt
# roleadd -c "User Security role, LDAP" -s /usr/bin/pfksh \
-m -S ldap -K profiles="User Security"  usersec
```

管理者は、例7「ユーザーへの役割の割り当て」の通常のユーザーをすべて作成するには 2 人のユーザーが必要であることを確認します。

例 6 暗号化サービス管理のための役割の作成と割り当て

この例では、LDAP ネットワーク上の管理者が暗号化フレームワークを管理するための役割を作成して、その役割を UID `1111` に割り当てます。

```
# roleadd -c "Cryptographic Services manager" \
-g 14 -m -u 104 -S ldap -K profiles="Crypto Management" cryptomgt
# passwd cryptomgt
New Password: xxxxxxxx
Confirm password: xxxxxxxx
```

```
# usermod -u 1111 -R +cryptomgt
```

UID が 1111 のユーザーは、ログイン後にその役割を引き受けて、割り当てられている権利を表示します。

```
$ su - cryptomgt
Password: xxxxxxxx
$ profiles -l
      Crypto Management
          /usr/bin/kmfcfg            euid=0
          /usr/sbin/cryptoadm        euid=0
          /usr/sfw/bin/CA.pl         euid=0
          /usr/sfw/bin/openssl       euid=0
```

暗号化フレームワークについては、『Oracle Solaris 12 での暗号化と証明書の管理』を参照してください。

信頼できるユーザーのログインの作成

ログインを作成するには useradd コマンドを使用します。useradd コマンドのすべての引数のリストについては、useradd(8) のマニュアルページを参照してください。このコマンドの権利関連の引数は roleadd コマンドと同じですが、さらに -R *rolename* オプションがあります。

役割をユーザーに割り当てる場合、ユーザーはその役割を引き受けたあとでその役割の権利を使用できます。たとえば次のコマンドは、58 ページの「信頼できるユーザーのログインの作成」で作成した accountadm 役割を引き受けることが可能な信頼できるユーザーを作成します。

```
# useradd -c "Trusted Assistant User Manager user"  -m -R accountadm \
        -s /usr/bin/pfbash jdoe
80 blocks
# ls /export/home/jdoe
local.bash_profile     local.login     local.profile
```

各表記の意味は次のとおりです。

-m　　ユーザーのホームディレクトリを作成します。

-R *rolename*　　既存の役割の名前を割り当てます。

-s *shell*　　*username* のログインシェルを決定します。このシェルは、pfbash などのプロファイルシェルにできます。信頼できるユーザーにプロファイルシェルを割り当てる理由については、44 ページの「権利の割り当てにおける操作性に関する考慮事項」を参照してください。プロファイルシェルの一覧については、pfexec(1) のマニュアルページを参照してください。

ほかの例については、『Managing User Accounts and User Environments in Oracle Solaris 11.4』 の 「Setting Up and Managing User Accounts (Task Map)」を参照してください。

ユーザーの権利の変更

ユーザーアカウントを変更するには、usermod コマンドを使用します。usermod コマンドのすべての引数のリストについては、usermod(8) のマニュアルページを参照してください。このコマンドの権利関連の引数は useradd コマンドと同じです。

権利プロファイルをユーザーに割り当てると、そのユーザーはプロファイルシェルを開いたあとで権利を使用できます。たとえば、権利プロファイルとプロファイルシェルをユーザーのログインシェルとして割り当てます。

```
# usermod -s /usr/bin/pfbash -K profiles="User Management" kdoe
```

これらの変更は、そのユーザーの次回のログイン時に有効になります。ユーザーが割り当てられている権利を使用する方法を習得する場合は、110 ページの「割り当てられている管理権利の使用」を参照するように指示してください。

例 7 ユーザーへの役割の割り当て

この例では、通常のユーザーを作成するために 2 人の信頼できるユーザーが必要であることを管理者が確認します。役割は例5「責務を分離するための役割の作成」で作成されました。

```
# usermod -R +accountadm jdoe
# usermod -R +usersec mdoe
```

役割の権利の変更

役割アカウントを変更するには、rolemod コマンドを使用します。rolemod コマンドのすべての引数のリストについては、rolemod(8) のマニュアルページを参照してください。このコマンドの権利関連の引数は roleadd コマンドと同じです。

key=*value* ペアの値と、-A、-P、および -R オプションは、マイナス (-) またはプラス (+) 記号で変更できます。- 記号は、現在割り当てられている値から値を引くことを示します。+ 記号は、現在割り当てられている値に値を加えることを示します。権利プロファイルの場合、値は現在のプロファイルリストの先頭に付加されます。権利プロファイルが先の順序であることの影響については、40 ページの「割り当てられた権利の検索順序」を参照してください。

例 8 役割の最初の権利プロファイルとしての権利プロファイルの追加

たとえば、accountadm 役割の先頭に権利プロファイルを付加します。

```
# rolemod -K profiles+="Device Management" accountadm
```

```
# profiles accountadm
accountadm:
Device Management
User Management
User Security
```

例 9 ローカル役割に割り当てられているプロファイルの置換

この例では、セキュリティー管理者が `prtmgt` 役割を変更し、Printer Management プロファイルの後に VSCAN Management 権利プロファイルを組み込みます。

```
# rolemod -c "Handles printers and virus scanning" \
-K profiles="Printer Management,VSCAN Management,All" prtmgt
```

例 10 役割への特権の直接割り当て

この例では、セキュリティー管理者は、システム時間に影響を及ぼすきわめて特殊な特権を `realtime` 役割に委ねます。特権をユーザーに割り当てるには、例16「ユーザーへの特権の直接割り当て」を参照してください。

```
# rolemod -K defaultpriv+='proc_clock_highres' realtime
```

`defaultpriv` キーワードの値は、常にその役割のプロセスに含まれる特権のリスト内にあります。

役割パスワードでのユーザー独自のパスワード使用の有効化

ユーザーが役割を引き受けるときに役割のパスワードではなくユーザー独自のパスワードを使用できるようにするには、その役割を変更します。

次のコマンドは、`accountadm` 役割が割り当てられているすべてのユーザーが、`accountadm` 役割を含め、割り当てられている任意の役割を引き受けるときに、ユーザー独自のパスワードを使用できるようにします。

```
# rolemod -K roleauth=user accountadm
```

注意 - `roleauth=user` キーワードと `auth_profiles=`*profiles* キーワードの両方を使用して役割を構成しないでください。`auth_profiles` キーワードは、`roleauth` パスワード (ユーザー) ではない現在のプロセス所有者のパスワード (役割) に対して認証されるため、認証が失敗します。

`auth_profiles` キーワードは役割のために設計されていませんが、役割でも使用できます。このキーワードは、別の役割アカウントを使用する必要のない、ユーザーの資格情報による再認証のために設計されています。

役割のパスワードの変更

役割は多数のユーザーに割り当てることができるため、役割が割り当てられているユーザーは、その役割のパスワードを変更できません。役割のパスワードを変更するには root 役割である必要があります。

```
# passwd accountadm
Enter accountadm's password: xxxxxxxx
New: xxxxxxxx
Confirm: xxxxxxxx
```

リポジトリを指定しない場合は、すべてのリポジトリでパスワードが変更されます。コマンドオプションの詳細については、passwd(1) のマニュアルページを参照してください。

例 11 特定リポジトリでの役割のパスワードの変更

次の例では、root 役割がローカル devadmin 役割のパスワードを変更します。

```
# passwd -r files  devadmin
New password: xxxxxxxx
Confirm password: xxxxxxxx
```

次の例では、root 役割が LDAP ネームサービス内で devadmin 役割のパスワードを変更します。

```
# passwd -r ldap devadmin
New password: xxxxxxxx
Confirm password: xxxxxxxx
```

役割の削除

役割を削除すると、その役割は即時に使用不可になります。

```
# roledel accountadm
```

役割の管理タスクを現在実行しているユーザーは、続行できなくなります。profiles コマンドは次の出力を表示します。

```
accountadm $ profiles
Unable to get user name
```

ユーザーの権利の拡張

このセクションのタスクと例では、ユーザーにデフォルトで付与される権利に権利を追加します。権利の詳細については、第1章「権利を使用したユーザーとプロセスの制御について」を参照してください。

- 信頼できるユーザーに役割を割り当てる –

  例2「ARMOR の役割の使用」
  例6「暗号化サービス管理のための役割の作成と割り当て」
  例7「ユーザーへの役割の割り当て」
- 信頼できるユーザーに権利プロファイルを割り当てる –

  例12「DHCP を管理する信頼できるユーザーの作成」
  例22「信頼できるユーザーによる拡張アカウントファイル読み取りの有効化」
  例38「レガシーアプリケーションへのセキュリティー属性の割り当て」
- 信頼できるユーザーに認証権利プロファイルを割り当てる –

  例13「ユーザーに対し DHCP 管理の前にパスワード入力を求める」
  例39「割り当てられた権利でのアプリケーションの実行」
- 信頼できるユーザーまたは役割に承認を割り当てる –

  例14「ユーザーへの承認の直接割り当て」
  例15「役割への承認の割り当て」
- ユーザーまたは役割に特権を直接割り当てる –

  例10「役割への特権の直接割り当て」
  例16「ユーザーへの特権の直接割り当て」

注意 - 直接割り当てられた特権と承認を不適切に使用すると、意図しないセキュリティー違反が発生することがあります。詳細は、43 ページの「権利の割り当てにおけるセキュリティーに関する考慮事項」を参照してください。

- 役割を引き受けるときにユーザーが自分のパスワードを使用できるようにする -

  例18「役割パスワードでのユーザー独自のパスワード使用の有効化」
  例20「ユーザーが役割パスワードとして独自のパスワードを使用できるようにするための権利プロファイルの変更」
- サードパーティーアプリケーションの管理者の権利プロファイルを作成する – 例19「サードパーティーアプリケーションの管理者の権利プロファイルの作成」
- 権利プロファイルを変更する – 75 ページの「ユーザーからの特権の削除」
- 権利プロファイルのコマンドにセキュリティー属性を追加する –

例32「選択したアプリケーションによる新規プロセス生成の防止」
例49「特権付きコマンドを含む権利プロファイルの作成」

- root 所有ファイルをユーザーが読み取れるようにする。

  例22「信頼できるユーザーによる拡張アカウントファイル読み取りの有効化」
  例23「root 以外のアカウントによる root 所有ファイルの読み取りの有効化」
- ユーザーまたは役割が root 所有ファイルを編集できるようにする – 例51「権利プロファイルでの選択した権利のクローニングおよび削除」
- 新しい承認が含まれている権利プロファイルを割り当てる – 例53「権利プロファイルへの承認の追加」

例 12 DHCP を管理する信頼できるユーザーの作成

セキュリティー管理者が、DHCP を管理できるユーザーを作成します。

```
# useradd -K profiles="DHCP Management" -s /usr/bin/pfbash -S ldap  jdoe
```

ユーザーには pfbash ログインシェルが割り当てられるため、DHCP Management 権利プロファイル内の権利は常に評価され、DHCP 管理コマンドが正常に完了します。

例 13 ユーザーに対し DHCP 管理の前にパスワード入力を求める

この例では、セキュリティー管理者が jdoe に対し、DHCP を管理する前にパスワードを入力するよう求めます。

```
# usermod -K auth_profiles="DHCP Management" profiles="Edit Administrative Files" jdoe
```

jdoe が DHCP コマンドを入力すると、パスワードプロンプトが表示されます。jdoe の認証後に DHCP コマンドが完了します。検索順序に従って、通常のプロファイルの前に認証権利プロファイルが処理されます。

```
jdoe$ dhcpconfig -R 120.30.33.7,120.30.42.132
Password: xxxxxxxx
     /** Command completes **/
```

例 14 ユーザーへの承認の直接割り当て

この例では、セキュリティー管理者が、画面の明るさを制御できるローカルユーザーを作成します。

```
# useradd -c "Screened KDoe, local" -s /usr/bin/pfbash \
-K auths=solaris.system.power.brightness  kdoe
```

この承認は、ユーザーの既存の承認の割り当てに追加されます。

例 15 役割への承認の割り当て

この例では、セキュリティー管理者が DNS サーバーサービスの構成情報を変更できる役割を作成します。

```
# roleadd -c "DNS administrator role" -m -K auths="solaris.smf.manage.bind" dnsadmin
```

例 16 ユーザーへの特権の直接割り当て

この例では、セキュリティー管理者が、システム時間に影響を及ぼすきわめて特殊な特権をユーザー kdoe に委ねます。特権を役割に割り当てるには、例10「役割への特権の直接割り当て」を参照してください。

```
# usermod -K defaultpriv='basic,proc_clock_highres' kdoe
```

既存の値が defaultpriv キーワードの値で置き換えられます。このため、ユーザーが basic 特権を保持するために、値 basic が指定されます。デフォルトの構成では、すべてのユーザーが基本特権を保持します。基本特権のリストについては、138 ページの「特権の一覧表示」を参照してください。

ユーザーは追加された特権とその定義を表示できます。

```
kdoe$ ppriv -v $$
1800:   pfksh
flags = <none>
        E: file_link_any,…,proc_clock_highres,sys_ib_info
        I: file_link_any,…,proc_clock_highres,sys_ib_info
        P: file_link_any,…,proc_clock_highres,sys_ib_info
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,win_upgrade_sl
$ ppriv -vl proc_clock_highres
        Allows a process to use high resolution timers.
```

例 17 役割の基本特権への追加

次の例では、役割 realtime に日時のプログラムを処理する特権が直接割り当てられます。例10「役割への特権の直接割り当て」で proc_clock_highres を realtime に割り当てました。

```
# rolemod -K defaultpriv='basic,sys_time' realtime

$ su - realtime
Password: xxxxxxxx
$ ppriv -v $$
1600:   pfksh
flags = <none>
        E: file_link_any,...,proc_clock_highres,sys_ib_info,sys_time
        I: file_link_any,...,proc_clock_highres,sys_ib_info,sys_time
        P: file_link_any,...,proc_clock_highres,sys_ib_info,sys_time
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

例 18 役割パスワードでのユーザー独自のパスワード使用の有効化

デフォルトでは、ユーザーが役割になるには、その役割のパスワードを入力する必要があります。ユーザーパスワードを要求することで、管理者は Linux 環境での役割の引き受けと同様に で役割を引き受けられるようにします。

```
# rolemod -K roleauth=user auditrev
```

割り当てられたユーザーは、この役割を引き受けるために、その役割用に特別に作成されたパスワードではなく、自分のパスワードを使用できるようになります。

ユーザーにほかの役割が割り当てられている場合は、ユーザーのパスワードによってそれらの役割への認証も行われます。

例 19 サードパーティーアプリケーションの管理者の権利プロファイルの作成

通常、サードパーティーアプリケーションの管理者に `root` 特権は与えられません。アプリケーションが 権利プロファイルに含まれていないコマンドを使用する場合、`root` 管理者は、アプリケーション管理者の権利プロファイルを作成するようにしてください。この例は、ユーザー管理コマンドが User Management および User Security 権利プロファイル内の コマンドに依存するアプリケーションの権利プロファイルを示しています。このアプリケーションはまた、特権を必要とする方法で `mv`、`rm`、および `cp` コマンドも使用します。

`cp`、`mv`、および `rm` コマンドにどの特権が必要であるかを判定するために、この管理者は、149 ページの「プログラムが必要とする特権を判断する方法」で説明されているようにアプリケーションをテストしました。

```
# profiles -p "Application User Management"
profiles:Application User Management> set desc="Application User Management Profile"
profiles:Application User Management> add profiles="User Management,User Security"
profiles:Application User Management> add cmd=/usr/bin/cp
profiles:Application User Management:cp> set privs=zone
profiles:Application User Management:cp> end
profiles:Application User Management> add cmd=/usr/bin/mv
profiles:Application User Management:mv> set privs=zone
profiles:Application User Management:mv> end
profiles:Application User Management> add cmd=/usr/bin/rm
profiles:Application User Management:rm> set privs=zone
profiles:Application User Management:rm> end
profiles:Application User Management> add cmd=/opt/Appl/bin/addusr
profiles:Application User Management:addusr> set privs=zone
profiles:Application User Management:addusr> end
profiles:Application User Management> add cmd=/opt/Appl/bin/addgrp
profiles:Application User Management:addgrp> set privs=zone
profiles:Application User Management:addgrp> end
profiles:Application User Management> commit
profiles:Application User Management> info
    name=Application User Management
       desc=Application User Management Profile
       profiles=User Management,User Security
```

```
       cmd=/usr/bin/cp
       ...
profiles:Application Administrator> exit
```

例 20 ユーザーが役割パスワードとして独自のパスワードを使用できるようにするための権利プロファイルの変更

```
# profiles -p "Local System Administrator"
profiles:Local System Administrator> set roleauth="user"
profiles:Local System Administrator> end
profiles:Local System Administrator> exit
```

Local System Administrator 権利プロファイルが割り当てられているユーザーがその役割を引き受ける場合、そのユーザーはパスワードの入力を求められます。次のシーケンスでは、役割名は `admin` です。

```
$ su - admin
Password: xxxxxxxx
$       /** You are now in a profile shell with administrative rights**/
```

例 21 LDAP リポジトリ内の役割の `roleauth` の値を変更する

この例では、`root` 役割によって、役割 `secadmin` を引き受けることができるすべてのユーザーが役割の引き受け時に自分のパスワードを使用できるようになります。この機能は、LDAP サーバーによって管理されるすべてのシステムでこれらのユーザーに付与されます。

```
# rolemod -S ldap -K roleauth=user secadmin
```

例 22 信頼できるユーザーによる拡張アカウントファイル読み取りの有効化

信頼できるユーザーまたはユーザーグループによる、`root` アカウントが所有するファイルの読み取りを可能にできます。この権利は、`root` 所有ファイルを含む管理アプリケーションを実行できるユーザーにとって役立ちます。この例では、1 つまたは複数の Perl スクリプトを Extended Accounting Net Management 権利プロファイルに追加します。

管理者は `root` 役割を引き受けたあとで、名前が `network` で始まるアカウンティングファイルの読み取り機能を追加する権利プロファイルを作成します。

次のプロファイルは、拡張特権ポリシーを使用して `file_dac_read` 特権をスクリプトに付与し、これによりこのスクリプトは `/var/adm/exacct/network*` ファイルのみにアクセスできるようになります。このプロファイルは既存の Extended Accounting Net Management 権利プロファイルを補助プロファイルとして追加します。

```
# profiles -p "Extended Accounting Perl Scripts"
profiles:Extended Accounting Perl Scripts >
```

```
set desc="Perl Scripts for Extended Accounting"
... Scripts> add profiles="Extended Accounting Net Management"
... Scripts> add cmd=/usr/local/bin/exacctdisp.pl
... Scripts:exacctdisp.pl> set privs={file_dac_read}:/var/adm/exacct/network*
... Scripts:exacctdisp.pl> end
... Scripts> commit
... Scripts> exit
```

サンプルスクリプトについては、『Administering Resource Management in Oracle Solaris 11.4』の「Using the Perl Interface to libexacct」を参照してください。

管理者は、権利プロファイルエントリで、スペルミス、省略、繰り返しなどのエラーがないかどうかを確認してから、Extended Accounting Perl Scripts 権利プロファイルを役割またはユーザーに割り当てます。

```
# profiles -p "Extended Accounting Perl Scripts" info
Found profile in files repository.
name=Extended Accounting Perl Scripts
desc=Perl Scripts for Extended Accounting
profiles=Extended Accounting Net Management
cmd=/usr/local/bin/exacctdisp.pl
privs={file_dac_read}:/var/adm/exacct/network*

# rolemod -K profiles+="Extended Accounting Perl Scripts" rolename
# usermod -s /usr/bin/pfbash -K profiles+="Extended Accounting Perl Scripts" username
```

例 23 root 以外のアカウントによる root 所有ファイルの読み取りの有効化

この例では、管理者が拡張特権ポリシーを使用する権利プロファイルを作成し、認証されるユーザーと役割が、root が所有する /var/adm/sulog ファイルを読み取ることができるようにします。管理者は、ユーザーがファイルの読み取りに使用できるコマンドを追加します。head コマンドなど、リストにないコマンドを使用不可にすることはできません。

```
# profiles -p "Read sulog File"
profiles:Read sulog File
set desc="Read sulog File"
... File> add profiles="Read Log Files"
... File> add cmd=/usr/bin/cat
... File:cat> set privs={file_dac_read}:/var/adm/sulog
... File:cat> end
... File> add cmd=/usr/bin/less
... File:less> set privs={file_dac_read}:/var/adm/sulog
... File:less> end
... File> add cmd=/usr/bin/more
... File:more> set privs={file_dac_read}:/var/adm/sulog
... File:more> end
... File> add cmd=/usr/bin/page
... File:page> set privs={file_dac_read}:/var/adm/sulog
... File:page> end
... File> add cmd=/usr/bin/tail
... File:tail> set privs={file_dac_read}:/var/adm/sulog
... File:tail> end
... File> add cmd=/usr/bin/view
... File:head> set privs={file_dac_read}:/var/adm/sulog
... File:head> end
... File> commit
```

```
... File> exit
```

`view` コマンドは、ユーザーによるファイル読み取りを可能にしますが、編集はできません。

ユーザーの権利の制限

このセクションの手順および例では、ログイン試行を制限したり、通常ユーザーの権利を制限したり、いくつかの管理上の権利を管理者から削除したりできます。ユーザー、役割、および権利プロファイルを変更する方法を示します。権利の詳細については、第1章「権利を使用したユーザーとプロセスの制御について」を参照してください。

- ワンタイムパスワード (OTP) の入力をユーザーに求める – 『Managing Authentication in Oracle Solaris 11.4』 の 「Task Map: Using OTP in Oracle Solaris」
- ユーザーに対してより強力なデフォルトファイル権限を提供する
  - レガシー – 69 ページの「標準ユーザーに対してより制限された `umask` 値を設定する方法」
  - SMF – 84 ページの「すべてのログインに対してより制限された `umask` 値を設定する方法」
- 連続して失敗するログイン試行を制限する
  - レガシー – 70 ページの「標準ユーザーに対してアカウントロックを設定する方法」
  - SMF – 86 ページの「すべてのログインに対してアカウントロックを設定する方法」
- ユーザーから制限特権を削除する – 例24「ユーザーの制限セットからの特権の削除」
- ユーザー独自のシェルプロセスから基本特権を削除する – 例25「ユーザー自身からの基本特権の削除」
- ユーザープロセスがサブプロセスを生成できないようにする – 例26「ゲストによるエディタサブプロセス生成の防止」
- ゲスト用の制限付きエディタを作成する – 例26「ゲストによるエディタサブプロセス生成の防止」
- 制限付きエディタをパブリックシステムに割り当てる –
  - レガシー – 例27「全ユーザーへの Editor Restrictions 権利プロファイルの割り当て」
  - SMF – 例36「すべてのログインへの Editor Restrictions 権利プロファイルの割り当て」
- 権利プロファイルを使用して権利を削除する – 例30「権利プロファイルからの基本特権の削除」

例29「リモートユーザー権利プロファイルの作成」
例51「権利プロファイルでの選択した権利のクローニングおよび削除」

- 管理者を明示的に割り当てられている権利に制限する – 例31「明示的に割り当てられた権利への管理者の制限」
- アプリケーションがサブプロセスを作成できないようにする – 例32「選択したアプリケーションによる新規プロセス生成の防止」
- システムのすべてのユーザーから権利を削除します。
  - レガシー – 例27「全ユーザーへの Editor Restrictions 権利プロファイルの割り当て」、例28「システムユーザーが使用できる権利を制限するように `policy.conf` ファイルを変更する」
  - SMF – 例37「Console User のみのログインの有効化」、69 ページの「一般的なユーザー制限の設定」
- 制限された用途でシステムを作成する -
  - レガシー – 例28「システムユーザーが使用できる権利を制限するように `policy.conf` ファイルを変更する」
  - SMF – 例37「Console User のみのログインの有効化」
- ユーザー、役割、システム、またはシステムのセットによって LDAP の属性を修飾する – 例33「LDAP ユーザーや役割がその権利を使用できる場所と時間の修飾」および `user_attr(5)` のマニュアルページを参照してください。
- システムへのユーザーアクセスを時間または場所で制限する – 例33「LDAP ユーザーや役割がその権利を使用できる場所と時間の修飾」
- ユーザーから承認を削除する – 例52「割り当て済みの承認のテストおよび削除」
- ユーザーからの役割割り当てを削除する – 例55「システム保守での `root` 役割の使用の防止」

一般的なユーザー制限の設定

このセクションでは、すべてのユーザーのデフォルトの `umask` 値を変更し、ログインの失敗を制限することによって悪質なログイン試行を防ぎ、コンソールユーザーがシステムをシャットダウンする機能を削除します。システムごと、ユーザーごと、または権利プロファイルを通じたログイン試行の失敗を制限できます。パスワードの制約については、『Oracle Solaris 11.4 Security and Hardening Guidelines』の「Passwords and Password Policy」を参照してください。

▼ 標準ユーザーに対してより制限された umask 値を設定する方法

`umask` ユーティリティーは、ユーザーが作成したファイルのファイルアクセス権ビットを設定します。デフォルトの `umask` 値 `022` では十分に制限されない場合は、この手順を使用して、より制限されたマスクを設定します。

始める前に スケルトンファイルを編集する権限がある管理者になる必要があります。root 役割にはこれらの権限が割り当てられます。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **でユーザーシェルのデフォルトとして提供されているサンプルファイルを表示します。**

```
# ls -1a /etc/skel
.bashrc
.profile
local.cshrc
local.login
local.profile
```

2. **ユーザーに割り当てる /etc/skel ファイルに umask 値を設定します。**

   次の値のいずれかを選択します。

   - umask 026 – 適度なファイル保護を提供します。

     (751) – グループには r、他のユーザーには x
   - umask 027 – 厳密なファイル保護を提供します

     (750) – グループには r、他のユーザーにはアクセス権なし
   - umask 077 – 完全なファイル保護を提供します。

     (700) – グループや他のユーザーのアクセスを禁止します。

参照 詳細については、次を参照してください。

- 『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』の 「umask のデフォルト値」
- 選択したマニュアルページには、useradd(8) および umask(1) が含まれています。

▼ 標準ユーザーに対してアカウントロックを設定する方法

この手順を使用して、特定の数のログイン試行に失敗したあとに通常ユーザーアカウントをロックします。ユーザーごと、システムごと、または権利プロファイルを通じた選択済みユーザーに対して、アカウントロックを設定できます。

注記 - 役割は共有されるアカウントです。ロックされた 1 人のユーザーが役割をロック解除できるため、役割を引き受けることができるユーザーにはアカウントロックを設定しないでください。

始める前に 管理アクティビティーで使用されるシステムでは、この保護をシステム全体に設定しないでください。むしろ、管理システムに異常な使用状況がないかどうかをモニターし、管理者が常に管理システムを使用できるようにしてください。

root 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **LOCK_AFTER_RETRIES セキュリティー属性を YES に設定します。**

   属性値のスコープを選択します。

   - **システム全体に設定します。**
     - **SMF のセキュリティー属性を変更します。**

       手順については、85 ページの「ログインポリシーの変更」を参照してください。

     - **非推奨。システムセキュリティーファイルを編集します。**

       ```
       # pfedit /etc/security/policy.conf
       ...
       #LOCK_AFTER_RETRIES=NO
       LOCK_AFTER_RETRIES=YES
       ...
       ```

   - **ユーザーごとに設定します。**

     この保護は、このコマンドの実行対象のユーザーに対してのみ適用されます。ユーザー数が多い場合、これはスケーラブルな解決方法ではありません。

     ```
     # usermod -K lock_after_retries=yes username
     ```

   - **権利プロファイルを作成して割り当てます。**

     この保護は、この権利プロファイルを割り当てるすべてのユーザーまたはシステムに適用されます。

     a. **権利プロファイルを作成します。**

        ```
        # profiles -p shared-profile -S ldap
        shared-profile: set lock_after_retries=yes
        ...
        ```

        権利プロファイルの作成の詳細は、115 ページの「権利プロファイルと承認の作成」を参照してください。

     b. **権利プロファイルをユーザーまたはシステム全体に割り当てます。**

        権利プロファイルを共有するユーザーの数が多い場合は、権利プロファイルにこの値を設定することがスケーラブルな解決方法になります。

        ```
        # usermod -P shared-profile username
        ```

        また、policy.conf ファイルでシステムごとにプロファイルを割り当てることもできます。

```
# pfedit /etc/security/policy.conf
...
#PROFS_GRANTED=Basic Solaris User
PROFS_GRANTED=shared-profile,Basic Solaris User
```

2. **RETRIES セキュリティー属性を 3 に設定します。**
   属性値のスコープを選択します。

   - **システム全体に設定します。**
     - **SMF のセキュリティー属性を変更します。**
       手順については、85 ページの「ログインポリシーの変更」を参照してください。
     - **非推奨。システムセキュリティーファイルを編集します。**

       ```
       # pfedit /etc/default/login
       ...
       #RETRIES=5
       RETRIES=3
       ...
       ```

   - **ユーザーごとに設定します。**

     ```
     # usermod -K lock_after_retries=3 username
     ```

   - **権利プロファイルを作成して割り当てます。**
     115 ページの「権利プロファイルと承認の作成」を確認し、`lock_after_retries=3` を含む権利プロファイルを作成します。

3. **(オプション) ユーザーがロック済みアカウントに再認証できるまでの時間を指定します。**

   ```
   # usermod -K unlock_after=185m username
   ```

   このユーザーは、アカウントがロックされてから 3 時間 5 分後に、管理者が操作しなくてもログインできます。

4. **ロックされたユーザーをロック解除するには、`passwd` コマンドを使用します。**

   ```
   # passwd -u username
   ```

   ロックされたユーザーは、管理者の操作なしではログインできません。`files` と `ldap` の両方のネームサービスでユーザーアカウントをロック解除できます。

参照

- ユーザーおよび役割のセキュリティー属性については、第9章「 権利リファレンス」を参照してください。
- 選択したマニュアルページには、`passwd(1)`、`policy.conf(5)`、`profiles(1)`、`user_attr(5)`、および `usermod(8)` が含まれています。

▼ ユーザーから電源管理機能を削除する方法

この手順を使用して、システムのコンソール上のユーザーがシステムを保存停止したり、電源を切ったりすることを回避します。コンソールユーザーがシステムのハードウェアを取り外すことができる場合、このソフトウェア解決方法は有効ではありません。

始める前に root 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **コンソールユーザー権利プロファイルの内容をレビューします。**

```
$ profiles -p "Console User" info
 name=Console User
 desc=Manage System as the Console User
 auths=solaris.system.shutdown,solaris.device.cdrw,
             solaris.smf.manage.vbiosd,solaris.smf.value.vbiosd
 profiles=Suspend To RAM,Suspend To Disk,Brightness,CPU Power Management,
                Network Autoconf User
```

2. **ユーザーが保持する権限がコンソールユーザープロファイルに含まれる権利プロファイルを作成します。**

   手順については、115 ページの「権利プロファイルを作成する方法」を参照してください。

   サンプルの置換プロファイルを次に示します。

```
$ profiles -p "Site Console User" info
 name=Site Console User
 desc=Read devices and control brightness on the console
 auths=solaris.device.cdrw,
 profiles=Brightness,CPU Power Management,Network Autoconf User
```

3. **/etc/security/policy.conf ファイルでコンソールユーザー権利プロファイルをコメントアウトします。**

```
#CONSOLE_USER=Console User
```

4. **ステップ 2で作成した権利プロファイルを割り当てます。**

   - この権利プロファイルをユーザーに割り当てます。

     # **usermod -P** *shared-profile username*

   - また、SMF または policy.conf ファイルのいずれかで、プロファイルをシステムごとに割り当てることもできます。
     - account-policy サービスが有効な場合、例35「すべてのログインへの権利プロファイルの追加」に示すようにプロファイルを SMF で割り当てます。
     - 非推奨。account-policy サービスが有効でない場合、policy.conf ファイルを編集します。

```
# pfedit /etc/security/policy.conf...
#PROFS_GRANTED=Basic Solaris User
PROFS_GRANTED=shared-profile,Basic Solaris User
```

参照 詳細は、account-policy(8S)、policy.conf(5)、および usermod(8) のマニュアルページを参照してください。

リモートログイン制限の設定

このセクションでは、特定のルート以外のユーザーについてのリモートログインを防ぐためのいくつかの方法について説明します。root によるリモートログインを防ぐには、/etc/default/login ファイル内の CONSOLE 変数値の説明および『Oracle Solaris 11.4 でのシステムおよび接続されたデバイスのセキュリティー保護』 の 「ログインとパスワードのセキュリティー」を参照してください。

では、特定のルート以外のユーザーのリモートログインを防ぐためのいくつかの方法があります。

- ユーザーが ssh などの PAM サービスにアクセスできる曜日と時間を指定できます。たとえば、access_times セキュリティー属性を使用して、jdoe が を使用できないようにします。

  ```
  # usermod -K access_times={ssh}:Al0000-0000 jdoe
  ```

  この値は、すべての曜日 (Al) について、利用可能なアクセス時間がないことを示しています。詳細は、user_attr(5) のマニュアルページを参照してください。

  RBAC セキュリティー属性を使用することの 1 つの利点として、ユーザーおよび役割の構成コマンドに -S ldap オプションを使用して、LDAP 内の制限を維持できるということがあります。別の利点としては、アクセスが許可されるサービス名、曜日、時間を指定できることがあります。
- の DenyUsers プロパティーに特定のユーザーを追加できます。詳細は、ssh_config(5) のマニュアルページを参照してください。
- pam_deny モジュールを含むように PAM スタックをカスタマイズし、ユーザーおよび役割構成コマンドに pam_policy セキュリティー属性を使用することによって、カスタマイズされた PAM スタックを特定のユーザーに割り当てることができます。詳細は、『Managing Authentication in Oracle Solaris 11.4』 の 「Configuring PAM」、および Unresolved link to " pam_deny7" と Unresolved link to " pam_user_policy7" のマニュアルページを参照してください。
- root を含むほかのすべてのユーザーが、そのディレクトリに格納されているデータにアクセスできないようにする、特定ユーザー向けのラベル付きディレクトリおよび認可上限を作成できます。ラベルによって、ユーザーはプロジェクト用のサンドボックスを作成できます。詳細は、第6章「データ損失保護のためのプロセスのラベル付け」および sandboxing(7) のマニュアルページを参照してください。

ユーザーからの特権の削除

特定の状況では、標準ユーザーまたはゲストユーザーから特権を削除できます。たとえば、リモートユーザーに対してそのユーザーが所有していないプロセスのステータスを確認できないようにしたり、個々のユーザーが自分のプロセスを強調表示するために同様の処理を行なったり、ゲストが多くのリソースを使いすぎないようにしたりする場合があります。

▼ ユーザーから不要な基本特権を削除する方法

特定の状況では、標準ユーザーまたはゲストユーザーの基本セットから一部の基本特権を削除できます。たとえば、リモートユーザーは自分が所有していないプロセスのステータスを確認できない場合があります。

始める前に `root` 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **基本特権セットの完全な定義を一覧表示します。**

   次の 3 つの基本特権は、削除の対象になる可能性があります。

```
$ ppriv -lv basic
file_link_any
 Allows a process to create hardlinks to files owned by a uid
 different from the process' effective uid.
...
proc_info
 Allows a process to examine the status of processes other
 than those it can send signals to.  Processes which cannot
 be examined cannot be seen in /proc and appear not to exist.
proc_session
 Allows a process to send signals or trace processes outside its
 session.
...
```

2. **特権削除のスコープを選択します。**

   - **システム全体に設定します。**

     システムを使用しようとするユーザーは、これらの特権を拒否されます。この特権削除の方法は、だれでも使用可能なコンピュータに適している可能性があります。

```
# pfedit /etc/security/policy.conf
...
#PRIV_DEFAULT=basic
PRIV_DEFAULT=basic,!file_link_any,!proc_info,!proc_session
```

   - **個別のユーザーから特権を削除します。**

- **ユーザーが所有していないファイルへのリンクを作成できないようにします。**

  ```
  # usermod -K 'defaultpriv=basic,!file_link_any' user
  ```

- **ユーザーが所有していないプロセスを調査できないようにします。**

  ```
  # usermod -K 'defaultpriv=basic,!proc_info' user
  ```

- **ユーザーの現在のセッションから ssh セッションを開始するなど、ユーザーが 2 つ目のセッションを開始できないようにします。**

  ```
  # usermod -K 'defaultpriv=basic,!proc_session' user
  ```

- **ユーザーの基本セットから 3 つの特権をすべて削除します。**

  ```
  # usermod -K 'defaultpriv=basic,!file_link_any,!proc_info,!proc_session' user
  ```

- **権利プロファイルを作成して割り当てます。**

  この保護は、この権利プロファイルが割り当てられたユーザーまたはシステムに適用されます。

  a. **権利プロファイルを作成します。**

  ```
  # profiles -p shared-profile -S ldap
  shared-profile: set defaultpriv=basic,!file_link_any,!proc_info,!proc_session
  ...
  ```

  権利プロファイルの作成の詳細は、115 ページの「権利プロファイルと承認の作成」を参照してください。

  b. **権利プロファイルをユーザーまたはシステム全体に割り当てます。**

  リモートユーザーなど、権利プロファイルを共有するユーザーの数が多い場合は、権利プロファイルにこの値を設定することがスケーラブルな解決方法になります。

  ```
  # usermod -P shared-profile username
  ```

  また、`policy.conf` ファイルでシステムごとにプロファイルを割り当てることもできます。

  ```
  # pfedit /etc/security/policy.conf
  ...
  #PROFS_GRANTED=Basic Solaris User
  PROFS_GRANTED=shared-profile,Basic Solaris User
  ```

例 24　ユーザーの制限セットからの特権の削除

次の例では、jdoe の最初のログインから開始されるすべてのセッションで sys_linkdir 特権を使用できないようにします。su コマンドの実行後でも、ユーザーはディレクトリへのハードリンクの作成やディレクトリのリンク解除を実行できません。

```
# usermod -K 'limitpriv=all,!sys_linkdir' jdoe
# userattr limitpriv jdoe
all,!sys_linkdir
```

例 25　ユーザー自身からの基本特権の削除

次の例では、通常のユーザーが .bash_profile を変更して proc_info 基本特権を削除します。ps や prstat などのプログラムの出力にはユーザー自身のプロセスだけが含まれており、有用な情報を示していることがあります。

```
##  .bash_profile
## Remove proc_info privilege from my shell
##
ppriv -s EI-proc_info $$
```

ppriv 行は、現在のシェルプロセス ($$) でユーザーの有効特権セットと継承可能特権セット (EI-) から proc_info 特権を削除します。

次の prstat 出力では、プロセスの合計が 74 から 3 に減少しています。

```
## With all basic privileges
Total: 74 processes, 527 lwps, load averages: 0.01, 0.00, 0.00

## With proc_info removed from the effective and inheritable set
Total: 3 processes, 3 lwps, load averages: 0.00, 0.00, 0.00
```

例 26　ゲストによるエディタサブプロセス生成の防止

この例では、管理者がエディタコマンドから proc_exec 基本特権を削除して、ユーザーが 1 つまたは複数のエディタからサブシェルを作成することを防止します。

1. 管理者は、vim エディタの制限特権セットから proc_exec を削除する権利プロファイルを作成します。

```
# profiles -p -S ldap "Editor Restrictions"
profiles:Editor Restrictions> set desc="Site Editor Restrictions"
... Restrictions> add cmd=/usr/bin/vim
... Restrictions:vim> set limitprivs=all,!proc_exec
... Restrictions:vim> end
... Restrictions> commit
... Restrictions> exit
```

2. 管理者がほかの一般的なエディタを権利プロファイルに追加します。

```
# profiles -p "Editor Restrictions"
profiles:Editor Restrictions> add cmd=/usr/bin/gedit
... Restrictions:gedit> set limitprivs=all,!proc_exec
... Restrictions:gedit> end
... Restrictions> add cmd=/usr/bin/vim
... Restrictions:vim> set limitprivs=all,!proc_exec
... Restrictions:vim> end
... Restrictions> add cmd=/usr/xpg4/ed
... Restrictions:ed> set limitprivs=all,!proc_exec
... Restrictions:ed> end
... Restrictions> add cmd=/usr/xpg4/ex
... Restrictions:ex> set limitprivs=all,!proc_exec
... Restrictions:ex> end
... Restrictions> add cmd=/usr/xpg4/vi
... Restrictions:vi> set limitprivs=all,!proc_exec
... Restrictions:vi> end
... Restrictions> commit
... Restrictions> exit
```

3. 管理者は、権利プロファイルのエントリに誤字、脱字、繰り返しなどのエラーがないかどうかを確認します。

```
# profiles -p "Editor Restrictions" info
Found profile in files repository.
name=Editor Restrictions
desc=Site Editor Restrictions
cmd=/usr/bin/vim
limitprivs=all,!proc_exec
...
```

4. 管理者は Editor Restrictions 権利プロファイルを `guest` ユーザーに割り当てます。

```
# usermod -K profiles+="Editor Restrictions" guest
```

管理者は `profiles+` を使用して、この権利プロファイルをアカウントの現在の権利プロファイルに追加します。

5. エディタ特権が制限されていることを確認するため、管理者はエディタを開き、別のウィンドウでエディタプロセスの特権を調べます。

```
# ppriv -S $(pgrep vim)
2805:   vim .bash_profile
flags = PRIV_PFEXEC       User is running a profile shell
        E: basic,!proc_info       proc_info is removed from basic set
        I: basic,!proc_info
        P: basic,!proc_info
        L: all,!proc_exec       proc_exec is removed from limit set
```

例 27 全ユーザーへの Editor Restrictions 権利プロファイルの割り当て

この例では、管理者が Editor Restrictions 権利プロファイルを `policy.conf` ファイルに追加します。管理者は、ゲストがログインできるパブリックシステムにこのファイルが配布されていることを確認します。

```
# cd /etc/security; cp policy.conf policy.conf.orig
# pfedit /etc/security/policy.conf
...
AUTHS_GRANTED=
AUTH_PROFS_GRANTED=
#PROFS_GRANTED=Basic Solaris User
PROFS_GRANTED=Editor Restrictions,Basic Solaris User
```

User Security 管理者は、すべてのユーザーにプロファイルシェルを割り当てました。理由と手順については、49 ページの「ユーザーへの権利の割り当て」を参照してください。

参照 詳細は、`privileges(7)` のマニュアルページを参照してください。

権利プロファイルの使用による多くのユーザーからの権利の削除

権利プロファイルによって多くのユーザーの権利を制限できます。権利プロファイルを変更すると、プロファイルを割り当てられているユーザーは次回ログイン時に権利が変更されるため、保守が簡単です。

システム全体での権利の削除

例 28 システムユーザーが使用できる権利を制限するように `policy.conf` ファイルを変更する

この例では、管理者がネットワークの管理にのみ役立つシステムを作成します。管理者は、`/etc/security/policy.conf` ファイルから Basic Solaris User 権利プロファイルとすべての承認を削除します。Console User 権利プロファイルは削除されません。結果となる `policy.conf` ファイルで影響を受けた行は次のとおりです。

```
...
##AUTHS_GRANTED=
##AUTH_PROFS_GRANTED=
##PROFS_GRANTED=Basic Solaris User
CONSOLE_USER=Console User
...
```

承認、コマンド、または権利プロファイルが明示的に割り当てられているユーザーのみがこのシステムを使用できます。ログイン後、その承認ユーザーは管理責務を果

たすことができます。承認されたユーザーがシステムコンソールの前に座っている場合、そのユーザーには Console User の権利があります。

例 29 リモートユーザー権利プロファイルの作成

この例では、管理者は、リモートログインするユーザーのために LDAP リポジトリ内で権利プロファイルを作成します。

```
# profiles -p -S LDAP "Remote Users"
profiles:Remote Users> set desc="For all logins from remote systems"
... Remote Users> set defaultpriv="basic,!proc_info"
... Remote Users> set limitpriv="basic,!proc_info"
... Remote Users> end
... Remote Users> exit
```

管理者は、内容を確認します。

```
# profiles -p "Remote Users" info
Found profile in LDAP repository.
        name=Remote Users
        desc=For all logins from remote systems
        defaultpriv=basic,!proc_info,
        limitpriv=basic,!proc_info
```

例 30 権利プロファイルからの基本特権の削除

次の例では、セキュリティー管理者は十分なテストを行なったあと、例29「リモートユーザー権利プロファイルの作成」で作成されたリモートユーザー権利プロファイルから別の基本特権を削除します。このプロファイルが割り当てられているユーザーは、現在のセッションの外部ではどのプロセスも検査できず、さらに別のセッションを追加することもできません。

```
# profiles -p "Remote Users"
profiles:Remote Users> set defaultpriv="basic,!proc_info,!proc_session"
profiles:Remote Users> end
profiles:Remote Users> exit
```

例 31 明示的に割り当てられた権利への管理者の制限

2 つの方法で、ユーザーまたは役割を限られた数の管理操作に制限できます。

- ユーザーのプロファイルリストの最終プロファイルとして、Stop 権利プロファイルを割り当てます。

  Stop 権利プロファイルは、制限付きシェルを作成するもっとも簡単な方法です。`policy.conf` ファイル内の承認と権利プロファイルは、ユーザーまたは役割には割り当てられません。
- システム上の `policy.conf` ファイルを変更して、役割またはユーザーがそのシステムを管理タスクに使用するよう要求します。例28「システムユーザーが使用で

きる権利を制限するように policy.conf ファイルを変更する」を参照してください。

次のコマンドは、auditrev 役割を監査レビューの実行のみに制限します。

```
# rolemod -K profiles="Audit Review,Stop" auditrev
```

auditrev 役割には Console User 権利プロファイルがないため、監査担当者はシステムをシャットダウンできません。この役割には solaris.device.cdrw 承認がないため、監査担当者は CD-ROM ドライブに対して読み取りまたは書き込みを行うことができません。この役割には Basic Solaris User 権利プロファイルがないため、そのプロファイルのコマンドをこの役割で実行できません。All 権利プロファイルが割り当てられていないため、ls コマンドは実行されません。この役割は、ファイルブラウザを使用してレビューする監査ファイルを選択します。

詳細は、40 ページの「割り当てられた権利の検索順序」および154 ページの「権利プロファイルのリファレンス」を参照してください。

例 32 選択したアプリケーションによる新規プロセス生成の防止

この例では、適切な動作のためにサブプロセスが不要なアプリケーションの権利プロファイルを管理者が作成します。便宜上、管理者はこれらの実行可能ファイルを保管するディレクトリを作成します。サブプロセスを必要としない新しいアプリケーションが追加されたら、実行可能ファイルをこのディレクトリに追加できます。あるいは、実行可能ファイルが特定のディレクトリに存在している必要がある場合、管理者は /opt/local/noex/*app-executable* からそれにリンクできます。

```
# profiles -p "Prevent App Subprocess"
profiles:Prevent App Subprocess> set desc="Keep apps from execing processes"
profiles:Prevent App Subprocess> add cmd=/opt/local/noex/mkmod
... Subprocess:mkmod> set limitprivs=all,!proc_exec
... Subprocess:mkmod> end
... Subprocess> add cmd=/opt/local/noex/gomap
... Subprocess:gomap> set limitprivs=all,!proc_exec
... Subprocess:gomap> end
... Subprocess> commit
... Subprocess> exit
```

特定のシステムで権利を引き受けるためのアカウントの修飾

この一連の例は、ユーザーや役割へのセキュリティー属性の割り当てを集中管理する方法を示しています。これらのコマンドは LDAP ネームサービスでのみ機能し、files ネームサービスでは機能しません。

例 33 LDAP ユーザーや役割がその権利を使用できる場所と時間の修飾

次の例では、ユーザー jdoe がシステム labsys1 および labsys2 を管理できるようにします。jdoe は LDAP アカウントです。

```
# usermod -q labsys1 -K profiles="System Administrator" jdoe
# usermod -q labsys2 -K profiles="System Administrator" jdoe
```

次の例では、system1 上の役割 admin への管理アクセスを平日の午前 5 時から午後 3 時までに制限します。admin は LDAP アカウントです。システムのローカルタイムゾーンが使用されます。

```
# rolemod -q system1 -k access_times="(*}:Wk0500-1500" \
  -K profiles="System Administrator" admin
```

例 34 ユーザーや役割が管理権利を持つシステムの修飾

この一連の例は、セキュリティー属性の割り当てをホスト名、またはネットグループと呼ばれるホストのグループで修飾する方法を示しています。netgroup(5) のマニュアルページを参照してください。

次の例では、ユーザー jdoe が lab1 ネットグループとして定義されている一連のシステムを管理できるようにします。jdoe および lab1 ネットグループは、LDAP ディレクトリで管理されます。

```
# usermod -q @lab1 -K profiles="System Administrator" jdoe
```

次の例では、ユーザー jdoe を平日の午前 5 時から午後 3 時までの lab1 ネットグループの管理に制限します。

```
# usermod -q @lab1 -k access_times="(*}:Wk0500-1500" -K profiles="System Administrator" jdoe
```

SMF プロパティーとしてのシステム全体での権利の変更

以降のセクションでは、サービス管理機能 (SMF) を使用して、システムにログインするすべてのユーザーに割り当てられたデフォルトの権利を管理する方法について説明します。SMF によって、システムごとまたは LDAP 内の権利を管理できます。 にインストールされているデフォルトの権利で通常は十分ですが、サイトのセキュリティー要件に合わせてデフォルト設定を変更できます。権利の詳細については、第1章「権利を使用したユーザーとプロセスの制御について」を参照してください。

新機能 – account-policy サービスの有効化

ではシステムのデフォルトの権利を SMF に設定できます。

レガシーシステムでは、/etc ディレクトリ内のファイルを編集します。SMF を使用する場合、account-policy のプロパティーが現在のポリシーを示しています。セキュリティーポリシーは 4 つのステンシルにグループ化されます。

```
config/etc_security_policyconf      config/etc_default_passwd
config/etc_default_login            config/etc_default_su
```

ステンシルを有効にすると、そのステンシル内にあるセキュリティー属性を変更できるようになります。

1. サービスを有効にして、サービスが有効であることを検証するには、次のコマンドを実行します。

```
$ pfexec svcadm enable account-policy
$ svcs account-policy
STATE          STIME    FMRI
online         0:10:00  svc:/system/account-policy:default
```

2. 次に、変更するセキュリティー属性を有効にして、サイトポリシーに合わせてその値を変更します。

```
$ pfbash svccfg -s account-policy  \
 setprop config/etc_security-stencil/disabled = boolean: false
 $ svccfg -s account-policy:default  \
 setprop security-stencil-group/property = [type:] value
$ svcadm refresh account-policy
```

注記 - account-policy: default SMF サービスがオンラインのとき、/etc/default/login などのレガシーファイルを変更しても、システムが強制的に適用する権利に影響しません。同じように、ファイルの内容が現在のポリシーに反映されない場合もあります。

システム全体で変更できる権利には、次のようなものがあります。

- 84 ページの「ログイン環境変数の変更」
- 85 ページの「ログインポリシーの変更」
- 87 ページの「パスワードポリシーの変更」
- 88 ページの「システム全体の特権、承認、および権利プロファイルの変更」
- 90 ページの「ロギングポリシーの変更」

システム全体で変更できるセキュリティー属性の一覧については、166 ページの「ファイルおよび対応する SMF プロパティー内のセキュリティー属性」を参照してください。

ログイン環境変数の変更

このセクションでは、すべてのユーザーのデフォルトの umask 値を変更し、ログインの失敗を制限することによって悪質なログイン試行を防ぎ、コンソールユーザーがシステムをシャットダウンする機能を削除します。システムごと、ユーザーごと、または権利プロファイルを通じたログイン試行の失敗を制限できます。パスワードの制約については、『Oracle Solaris 11.4 Security and Hardening Guidelines』の「Passwords and Password Policy」を参照してください。

このセクションは、82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。

account-policy サービスの config/etc_default_login ステンシルのプロパティーであるセキュリティー属性には次のものが含まれています。

```
$ svcprop -p login/environment account-policy:default
login/environment/path astring
login/environment/root_path astring
login/environment/set_shell boolean
login/environment/timezone astring
login/environment/ulimit integer
login/environment/umask integer
```

たとえば、84 ページの「すべてのログインに対してより制限された umask 値を設定する方法」を参照してください。account-policy(8S) のマニュアルページも参照してください。

▼ すべてのログインに対してより制限された umask 値を設定する方法

この手順では、すべてのユーザーに対するデフォルトの umask 値を変更します。umask ユーティリティーは、ユーザーが作成したファイルのファイルアクセス権ビットを設定します。デフォルトの umask 値 022 では十分に制限されない場合は、この手順を使用して、より制限されたマスクを設定します。

始める前に 82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。User Security 権利プロファイルが割り当てられている管理者になる必要があります。root 役割にはこのプロファイルが割り当てられます。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **次のようにして、サイトのセキュリティー要件を満たす値を判別します。**

   - umask 026 – 適度なファイル保護を提供します。
     (751) – グループには r、他のユーザーには x

- umask 027 – 厳密なファイル保護を提供します
  (750) – グループには r、他のユーザーにはアクセス権なし
- umask 077 – 完全なファイル保護を提供します。
  (700) – グループや他のユーザーのアクセスを禁止します。

2. **account-policy SMF ステンシルの umask プロパティー値を設定します。**

   a. **umask プロパティーの完全名を見つけます。**

   ```
   $ svcprop account-policy | grep umask
   login/environment/umask integer
   ```

   b. **新しい値を設定してサービスをリフレッシュします。**

   ```
   $ pfbash svccfg -s account-policy
   svc:/.../account-policy> setprop config/etc_default_login/disabled = boolean: false
   svc:/.../account-policy> setprop login/environment/umask = 026
   svc:/.../account-policy> exit
   $ svcadm refresh account-policy
   ```

参照 詳細については、次を参照してください。

- 『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』の 「umask のデフォルト値」
- 選択したマニュアルページには、useradd(8)、account-policy(8S)、および umask(1) が含まれています。

ログインポリシーの変更

このセクションは、82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。

account-policy サービスの config/etc_default_login ステンシルのプロパティーであるセキュリティー属性には次のものが含まれています。

```
$ svcprop -p login_policy account-policy:default
login_policy/annotation astring
login_policy/auto_unlock_time astring
login_policy/clearance astring
login_policy/disabletime count
login_policy/lock_after_retries astring
login_policy/pam_policy astring
login_policy/password_required boolean
login_policy/retries count
login_policy/root_login_device astring
login_policy/sleeptime count
login_policy/timeout count
```

詳細は、account-policy(8S) のマニュアルページを参照してください。

▼ すべてのログインに対してアカウントロックを設定する方法

この手順を使用して、特定の数のログイン試行に失敗したあとにユーザーのアカウントをロックすることによって悪質なログイン試行を防ぎます。

注意 - 管理アクティビティーで使用されるシステムでは、この保護をシステム全体に設定しないでください。むしろ、管理システムに異常な使用状況がないかどうかをモニターし、管理者が常に管理システムを使用できるようにしてください。

始める前に 82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。User Security 権利プロファイルが割り当てられている管理者になる必要があります。root 役割にはこのプロファイルが割り当てられます。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **account-policy ステンシルから retries SMF プロパティーの完全名を見つけます。**

```
$ svcprop account-policy | grep retries
login_policy/lock_after_retries astring
login_policy/retries count
```

2. **lock_after_retries プロパティーの値を yes に設定します。**

```
$ pfbash svccfg -s account-policy
svc:/.../account-policy> setprop config/etc_default_login/disabled = boolean: false
svc:/.../account-policy> setprop login_policy/lock_after_retries = yes
svc:/.../account-policy> exit
```

3. **retries カウントを 3 に設定してサービスをリフレッシュします。**

```
$ svccfg -s account-policy \
setprop login_policy/retries = 3
$ svcadm refresh account-policy
```

4. **(オプション) ロック済みアカウントでユーザーが再認証できるようになるまでの時間を指定します。**

   a. **account-policy ステンシルから unlock SMF プロパティーの完全名を見つけます。**

```
$ svcprop account-policy | grep unlock
login_policy/auto_unlock_time astring
```

   b. **auto_unlock_time プロパティー値を設定してサービスをリフレッシュします。**

   次のコマンドにより、ユーザーはアカウントがロックされてから 3 時間 5 分後に、管理者が操作しなくてもログインできます。

```
$ svccfg -s account-policy \
```

```
    setprop login_policy/auto_unlock_time = 185m
$ svcadm refresh account-policy
```

5. **ロックされたユーザーをロック解除するには、`passwd` コマンドを使用します。**

```
# passwd -u username
```

管理者として、`files` と `ldap` の両方のネームサービスでユーザーアカウントをロック解除できます。

参照

- ユーザーおよび役割のセキュリティー属性については、第9章「 権利リファレンス」を参照してください。
- 選択したマニュアルページには、`passwd(1)` および `account-policy(8S)` が含まれています。

パスワードポリシーの変更

このセクションは、82 ページの「新機能 – `account-policy` サービスの有効化」を完了したことが前提となります。

`account-policy` サービスの `config/etc_default_passwd` ステンシルのプロパティーであるセキュリティー属性には次のものが含まれています。

```
password/aging_defaults/max_days count
password/aging_defaults/min_days count
password/aging_defaults/warn_days count
password/complexity/max_repeats count
password/complexity/min_alpha count
password/complexity/min_diff count
password/complexity/min_digit count
password/complexity/min_lower count
password/complexity/min_nonalpha count
password/complexity/min_special count
password/complexity/min_upper count
password/complexity/namecheck boolean
password/complexity/passlength count
password/complexity/whitespace boolean
password/crypt/algorithms_allow astring 2a 5 6
password/crypt/algorithms_deprecate astring
password/crypt/default astring 5
password/dictionary/db_dir astring
password/dictionary/min_word_length count
password/dictionary/word_list astring
```

次の例では、管理者はデフォルトの 8 文字よりも長いパスワードの長さを必須として設定します。

```
$ pfbash svccfg -s account-policy
svc:/.../account-policy> setprop config/etc_default_passwd/disabled = boolean
svc:/.../account-policy> setprop password/complexity/passlength = 13
svc:/.../account-policy> exit
$ svcadm refresh account-policy
```

システム全体の特権、承認、および権利プロファイルの変更

このセクションは、82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。

次のコマンドは、RBAC ポリシー変数を SMF プロパティーとして表示します。

```
$ svcprop -p rbac account-policy
rbac/console_user_profiles astring Console\ User
rbac/default_auth_profiles astring
rbac/default_authorizations astring
rbac/default_limit_privileges astring
rbac/default_privileges astring
rbac/default_profiles astring Basic\ Solaris\ User
```

例 35 すべてのログインへの権利プロファイルの追加

この例では、管理者は Site Console User 権利プロファイルを追加し、システムのユーザーによる Console User 権利プロファイルへのアクセスを削除します。この例は、管理者が82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。

```
$ pfbash svccfg -s account-policy
svc:/.../account-policy> setprop config/etc_security_policyconf/disabled = boolean
svc:/.../account-policy> setprop rbac/console_user_profiles astring = ""
svc:/.../account-policy> setprop rbac/default_profiles astring = "Site Console User, Basic Solaris
 User"
svc:/.../account-policy> exit
$ svcadm refresh account-policy
```

73 ページの「ユーザーから電源管理機能を削除する方法」に、Site Console User 権利プロファイルの内容が示されています。

システムで使用可能な特権の変更

このセクションは、82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。

特定の状況で、システムから特権を削除できます。たとえば、リモートユーザーに対してそのユーザーが所有していないプロセスのステータスを確認できないようにしたり、パブリックシステムの特権を下げることでメリットを得られるようにしたりする場合があります。

次のコマンドは、ユーザーのセッションの外部にあるすべてのプロセスのファイルリンクおよび表示ができなくなるようにパブリックシステムを変更します。

```
$ pfbash svccfg -s account-policy
```

```
svc:/.../account-policy> setprop config/etc_security_policyconf/disabled = boolean
svc:/.../account-policy> setprop rbac/default_privileges = "basic,!file_link_any"
svc:/.../account-policy> exit
$ svcadm refresh account-policy
```

システムへの権利プロファイルの割り当て

このセクションは、82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。

権利プロファイルによって多くのユーザーの権利を指定できます。これらは容易に保守でき、システムに適用できます。

```
$ pfbash svccfg -s account-policy
svc:/.../account-policy> setprop config/etc_security_policyconf/disabled = boolean
svc:/.../account-policy> setprop rbac/default_profiles = "Example Rights Profile"
svc:/.../account-policy> exit
$ svcadm refresh account-policy
```

例 36 すべてのログインへの Editor Restrictions 権利プロファイルの割り当て

この例では、システムのエディタとなるすべてのユーザーに対し、編集前に認証を求める方法を示しています。

```
$ pfbash svccfg -s account-policy
svc:/.../account-policy> setprop config/etc_security_policyconf/disabled = boolean
svc:/.../account-policy> setprop rbac/default_profiles = "Editor Restrictions"
svc:/.../account-policy> exit
$ svcadm refresh account-policy
```

「Editor Restrictions」プロファイルは例26「ゲストによるエディタサブプロセス生成の防止」で作成されたものです。

例 37 Console User のみのログインの有効化

この例では、管理者がネットワークの管理にのみ役立つシステムを作成します。管理者は、システムから Basic Solaris User 権利プロファイルとすべての承認を削除します。Console User 権利プロファイルは削除されません。

```
$ pfbash svccfg -s account-policy
svc:/.../account-policy> setprop config/etc_security_policyconf/disabled = boolean
svc:/.../account-policy> setprop rbac/default_authorizations = ""
svc:/.../account-policy> setprop rbac/default_profiles = ""
svc:/.../account-policy> exit
$ svcadm refresh account-policy
```

承認、コマンド、または権利プロファイルが明示的に割り当てられているユーザーのみがこのシステムを使用できます。ログイン後、その承認ユーザーは管理責務を果たすことができます。承認されたユーザーがシステムコンソールの前に座っている場合、そのユーザーには Console User の権利があります。

ロギングポリシーの変更

このセクションは、82 ページの「新機能 – account-policy サービスの有効化」を完了したことが前提となります。

次のコマンドは、ロギングポリシー変数を SMF プロパティーとして表示します。プロパティーは 2 つのステンシルにあります。

```
login/log/syslog boolean
login/log/syslog_failed_attempts count
su/log/device astring
su/log/logfile astring
su/log/syslog boolean
```

次のコマンドは、syslog のロギングポリシーの設定を有効化します。

```
$ pfexec svccfg -s account-policy \
setprop config/etc_default_login/disabled boolean true
```

次のコマンドは、su のロギングポリシーの設定を有効化します。

```
$ pfexec svccfg -s account-policy \
setprop config/etc_default_su/disabled boolean true
```

ロギングプロパティーの値を変更する効果については、account-policy(8S) のマニュアルページを参照してください。

自動インストール中のアカウントポリシーの設定

account-policy の設定は、SMF プロファイルを通じた自動インストール (AI) の一部として行うのが最善です。

次の SMF プロファイルの例は、すべての account-policy 構成ファイルのステンシルを有効化し、選択されたプロパティーを設定します。この AI プロファイルは、初期 AI 中に顧客の IPS パッケージによって /etc/svc/profile/{*node,site,enterprise*} に提供されます。

```
<?xml version='1.0' encoding='US-ASCII'?>
<!DOCTYPE service_bundle SYSTEM
"/usr/share/lib/xml/dtd/service_bundle.dtd.1">

<service_bundle type="profile" name="site-account-policy">
  <service version="1" type="service" name="system/account-policy">
      <!-- Enable stenciling -->
      <instance name="default" enabled="true" />

      <property_group name="config" type="system">
        <property_group name="etc_default_login" type="configfile">
          <propval type="boolean" name="disabled" value="false"/>
        </property_group>
        <property_group name="etc_default_passwd" type="configfile">
```

```
          <propval type="boolean" name="disabled" value="false"/>
        </property_group>
        <property_group name="etc_default_su" type="configfile">
          <propval type="boolean" name="disabled" value="false"/>
        </property_group>
        <property_group name="etc_security_policyconf" type="configfile">
          <propval type="boolean" name="disabled" value="false"/>
        </property_group>
      </property_group>'

      <!-- Set account policy -->
      <property_group name="login_policy">
          <propval name="disabletime" type="count" value="0"/>
          <propval name="pam_policy" type="astring" value="ldap"/>
          <propval name="annotation" type="astring" value="yes"/>
      </property_group>

  </service>
</service_bundle>
```

♦ ♦ ♦ 第 4 章

4 アプリケーション、スクリプト、およびリソースへの権利の割り当て

この章では、特権、拡張特権ポリシー、およびその他の権利をユーザー、ポート、およびアプリケーションに適用するタスクについて説明します。

権利の概要については、19 ページの「ユーザー権管理」を参照してください。

アプリケーション、スクリプトおよびリソースの特定の権利への制限

このセクションのタスクと例では、実行可能ファイルとシステムリソースに特権を割り当てます。通常、信頼できるユーザーが実行可能ファイルを実行できるようにする目的で、特権を実行可能ファイルに割り当てます。94 ページの「アプリケーションおよびスクリプトへの権利の割り当て」では、特権割り当てによって、プロファイルシェル内で信頼できるユーザーがアプリケーションまたはスクリプトを実行できるようになります。97 ページの「拡張特権を使用したリソースのロックダウン」では、拡張特権によって、ユーザー ID、ポート、またはファイルオブジェクトが、デフォルトの有効セットよりも小さな特権セットに制限されます。未指定の特権は、ユーザーのプロセス、ポート、またはオブジェクトに対して拒否されます。このような割り当ては、最小特権ポリシーに近いものです。

アプリケーションおよびスクリプトへの権利の割り当て

アプリケーションとスクリプトは 1 つのコマンドまたは一連のコマンドを実行します。権利を割り当てるには、権利プロファイル内の各コマンドに対して、セット ID や特権などのセキュリティー属性を設定します。必要に応じて、アプリケーションは承認を確認できます。

注記 - スクリプト内のコマンドを成功させるためにそのコマンドに対して `setuid` ビットまたは `setgid` ビットを設定する必要がある場合は、権利プロファイル内でスクリプト実行可能ファイルおよびそのコマンドに対してセキュリティー属性を追加する必要があります。スクリプトがプロファイルシェルで実行されると、セキュリティー属性を使用してコマンドが実行されます。

- 権利を必要とするスクリプトを実行する – 94 ページの「特権付きのコマンドを含むシェルスクリプトの実行方法」
- 機密ファイルをバッチ編集する – 例41「ディレクトリ内のファイルのバッチ編集のスクリプト作成」
- `root` 以外のユーザーが特権を認識するアプリケーションを実行できるようにする – 例38「レガシーアプリケーションへのセキュリティー属性の割り当て」
- `root` 以外のユーザーが `root` 所有のアプリケーションを実行できるようにする – 例39「割り当てられた権利でのアプリケーションの実行」
- スクリプト内の承認を確認する – 例40「スクリプトまたはプログラム内の承認の確認」

▼ 特権付きのコマンドを含むシェルスクリプトの実行方法

特権シェルスクリプトを実行するには、そのスクリプトとスクリプト内のコマンドに特権を追加します。このため、該当する権利プロファイルには、特権が割り当てられているコマンドが含まれている必要があります。

始める前に `root` 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **1 行目に `/bin/pfsh` またはほかのプロファイルシェルが指定されているスクリプトを作成します。**

```
#!/bin/pfsh
# Copyright (c) 2015 by Oracle
```

2. **通常のユーザーとして、スクリプト内のコマンドに必要な特権を判断します。**

特権なしでスクリプトを実行すると、ppriv コマンドの debug オプションにより不足している特権が一覧表示されます。

```
$ ppriv -eD script-full-path
```

詳細は、149 ページの「プログラムが必要とする特権を判断する方法」を参照してください。

3. **スクリプトの権利プロファイルを作成または変更します。**

   シェルスクリプトとそのシェルスクリプトに含まれるコマンドを、それらの必要なセキュリティー属性とともに権利プロファイルに追加します。115 ページの「権利プロファイルを作成する方法」を参照してください。

4. **信頼できるユーザーまたは役割に権利プロファイルを割り当てます。**

   例については、49 ページの「ユーザーへの権利の割り当て」を参照してください。

5. **スクリプトを実行するには、次のいずれかを行います。**

   - **ユーザーとしてスクリプトが割り当てられている場合、プロファイルシェルを開いてスクリプトを実行します。**

     ```
     $ pfexec script-full-path
     ```

   - **役割としてスクリプトが割り当てられている場合、役割を引き受けてスクリプトを実行します。**

     ```
     $ su - rolename
     Password: xxxxxxxx
     # script-full-path
     ```

例 38 レガシーアプリケーションへのセキュリティー属性の割り当て

レガシーアプリケーションは特権を認識しないため、管理者は権利プロファイル内のアプリケーション実行可能ファイルにセキュリティー属性 euid=0 を割り当てます。次に、管理者はそれを信頼できるユーザーに割り当てます。

```
# profiles -p LegacyApp
profiles:LegacyApp> set desc="Legacy application"
profiles:LegacyApp> add cmd=/opt/legacy-app/bin/legacy-cmd
profiles:LegacyApp:legacy-cmd> set euid=0
profiles:LegacyApp:legacy-cmd> end
profiles:LegacyApp> exit
# profiles -p LegacyApp 'select cmd=/opt/legacy-app/bin/legacy-cmd;info;end'
 id=/opt/legacy-app/bin/legacy-cmd
 euid=0

# usermod -K profiles+="Legacy application" jdoe
```

例 39 割り当てられた権利でのアプリケーションの実行

この例では、管理者は 例49「特権付きコマンドを含む権利プロファイルの作成」の権利プロファイルを信頼できるユーザーに割り当てます。ユーザーはスクリプトの実行時にパスワードを入力する必要があります。

```
# usermod -K auth_profiles+="Site application" jdoe
```

例 40 スクリプトまたはプログラム内の承認の確認

承認を確認するには、`auths` コマンドに基づくテストを作成します。このコマンドの詳細については、`auths(1)` のマニュアルページを参照してください。

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
```

より完全なテストには、ワイルドカードの使用を確認するロジックが含まれています。たとえば、`solaris.system.date` 承認がユーザーにあるかどうかをテストするには、次の文字列を確認する必要があります。

- `solaris.system.date`
- `solaris.system.*`
- `solaris.*`

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。

例 41 ディレクトリ内のファイルのバッチ編集のスクリプト作成

この例は、2 つのスクリプトを使用して機密ファイルをバッチ編集する方法を示しています。最初のスクリプトはデフォルトエディタを消去し、2 番目のスクリプトを呼び出します。2 番目のスクリプトは編集スクリプトです。2 番目のスクリプトを変更することによって、ファイルへの変更のレコードを保持します。

1. 最初のスクリプトを作成します。

```
# pfedit batchpfedit.sh
    #!/usr/bin/bash
    OLDEDITOR=$EDITOR
    export EDITOR=~/bin/key-edit.sh
    pfedit /var/ITdemos/firstdemo-directory
    export EDITOR=$OLDEDITOR
```

各表記の意味は次のとおりです。

`OLDEDITOR=$EDITOR` 現在のエディタの値を保持するための OLDEDITOR 変数を定義します。

`~/bin/key-edit.sh` 編集を含むスクリプトです。

`pfedit /var/ITdemos/firstdemo-directory` 使用するスクリプトの一時的な pfedit ファイルを作成するコマンドです。

`export EDITOR=$OLDEDITOR` 元の $EDITOR 定義を置き換えます。

2. 編集スクリプトを作成します。

```
# pfedit batchpfedit.sh
    #!/usr/bin/bash
    OLDEDITOR=$EDITOR
    export EDITOR=~/bin/key-edit.sh
    pfedit /var/ITdemos/firstdemo-directory
    export EDITOR=$OLDEDITOR
```

2 番目のスクリプトには、編集する変更が含まれています。この例では、key-edit.sh スクリプトは単語 pey を単語 key に置き換えます。

```
#!/usr/bin/bash
    perl -pi.pfedit -e 's/pey/key/g' $1
    rm $1.pfedit
```

- $1 は、一時ファイルを検索します。
- perl -pi.*extension* は一時ファイルを作成し、元のファイルに書き戻します。
- rm コマンドは、この余分な一時ファイルを削除します。

3. 最初のスクリプトを呼び出します。

さらに多くの編集スクリプトを作成してから、batchpfedit.sh 内の key-edit.sh スクリプトを新しいスクリプト名に置き換えることができます。

拡張特権を使用したリソースのロックダウン

拡張特権ポリシーは、アプリケーションに対する攻撃が成功した場合のシステムへの攻撃者のアクセスを制限します。拡張ポリシールールは、特権割り当ての影響の範囲を、ルール内のリソースだけに制限します。拡張ポリシールールを記述するには、特権を中括弧で囲み、その後にコロンと関連付けられたリソースを記述します。詳細

は、36 ページの「ユーザーまたは役割の特権の拡張」を参照してください。構文の例については、ppriv(1) および privileges(7) のマニュアルページを参照してください。

管理者と通常のユーザーはいずれも、拡張特権を使用してリソースをロックダウンできます。管理者はユーザー、ポート、およびアプリケーションに対する拡張特権ルールを作成できます。通常のユーザーはコマンド行を使用するか、または ppriv -r コマンドを使用するスクリプトを作成して、アプリケーションによるユーザー指定ディレクトリ外部のファイルへの書き込みを防止できます。

- ポートから侵入する悪意のあるユーザーが使用できるアクセスを制限する – 98 ページの「ポートに拡張特権ポリシーを適用する方法」
- 非 root デーモンとしてデータベースを実行する – 99 ページの「MySQL サービスをロックダウンする方法」
- 非 root デーモンとして を実行する – 101 ページの「特定の特権を に割り当てる方法」
- が特権を使用して実行されていることを確認する – 102 ページの「 が使用している特権を判断する方法」
- Firefox がシステムのディレクトリに書き込まないようにする – 例42「保護されている環境でのブラウザの実行」
- アプリケーションをシステムの特定のディレクトリに制限する – 例43「アプリケーションプロセスからのシステム上のディレクトリの保護」

▼ ポートに拡張特権ポリシーを適用する方法

Network Time Protocol (NTP) のサービスは、udp トラフィックに対して特権ポート 123 を使用します。このサービスを実行するには特権が必要です。この例の手順では、ほかのポートを保護し、このポートに割り当てられている特権を取得した悪意のあるユーザーがアクセスできないように、サービスマニフェストを変更します。

始める前に root 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. ポートのデフォルトのサービスマニフェストエントリを参照します。

次の /lib/svc/manifest/network/ntp.xml start メソッドエントリでは、net_privaddr、proc_lock_memory、および sys_time 特権をほかのプロセスで使用できます。

```
privileges='basic,!file_link_any,!proc_info,!proc_session,net_privaddr,
proc_lock_memory,sys_time'
```

!file_link_any,!proc_info,!proc_session に指定されている削除された特権が原因で、サービスは、その他のプロセスに対するシグナル送信または監視と、ファイル名を変更する方法としてのハードリンクの作成を実行できません。つまり、このサー

ビスによって起動されたプロセスは NTP のポート 123 にしかバインドできず、ほかのどの特権ポートにもバインドできません。

ハッカーがこのサービスを悪用してほかのプロセスを開始できる可能性がある場合、そのプロセスも同様に制限されます。

2. **start メソッドと restart メソッドを変更して、net_privaddr 特権をこのポートのみに制限します。**

```
# svccfg -s ntp editprop -a
```

   a. **文字列 net_privaddr を検索します。**

   b. **net_privaddr が含まれているエントリのコメントを解除します。**

   c. **両方のエントリで net_privaddr を {net_privaddr}:123/udp に置き換えます。**
   拡張特権ポリシーにより、指定されている特権と未指定の基本特権を除くすべての特権がこのサービスから削除されます。このため、80 を超える悪用される可能性がある特権が、8 個未満に減少します。

3. **拡張特権ポリシーを使用するため、サービスを再起動します。**

```
# svcadm restart ntp
```

4. **サービスが拡張特権を使用していることを確認します。**

```
# svcprop -s ntp | grep privileges
start/privileges    astring  basic,!file_link_any,!proc_info,!proc_session,
                             {net_privaddr}:123/udp,proc_lock_memory,sys_time
restart/privileges  astring  basic,!file_link_any,!proc_info,!proc_session,
                             {net_privaddr}:123/udp,proc_lock_memory,sys_time
```

▼ MySQL サービスをロックダウンする方法

インストール時に、MySQL データベースは保護されていないポート上で root のフル特権で実行されるように構成されます。このタスクでは、権利プロファイルで MySQL サービスに拡張特権ポリシーを割り当てます。権利プロファイルがサービスの exec メソッドになったあとで、MySQL は保護されているポート上で、MySQL 以外のプロセスによるデータベースアクセスを制限した状態でユーザー mysql として実行されます。

始める前に 初期ユーザーはパッケージをインストールできます。残りのステップは root 役割が実行する必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **MySQL パッケージをインストールします。**

```
# pkg search basename:mysql
```

```
...
basename ... pkg:/database/mysql-57@version
# pfexec pkg install mysql-57
```

2. **MySQL サービスの状態と FMRI を表示します。**

```
# svcs mysql
STATE          STIME    FMRI
disabled       May_15   svc:/application/database/mysql:version_57
```

3. **サービスの実行メソッドを変更する権利プロファイルを作成します。**

   このサービスのサービスマニフェストは、実行メソッドがシェルスクリプトラッパー `/lib/svc/method/mysql_57` であることを指定します。

```
# svcprop -s mysql | grep manifest
... astring     /lib/svc/manifest/application/database/mysql_57.xml
# grep exec= /lib/svc/manifest/application/database/mysql_57.xml
                exec='/lib/svc/method/mysql_57 start'
                exec='/lib/svc/method/mysql_57 stop'
```

   プロファイル内のコマンドには `/lib/svc/method/mysql_57` ラッパーを使用します。

```
$ su -
Password: xxxxxxxx
# profiles -p "MySQL Service"
MySQL Service> set desc="Locking down the MySQL Service"
MySQL Service> add cmd=/lib/svc/method/mysql_57
MySQL Service:mysql_57> set privs=basic
MySQL Service:mysql_57> add privs={net_privaddr}:3306/tcp
MySQL Service:mysql_57>  add privs={file_write}:/var/mysql/5.7/data/*
MySQL Service:mysql_57>  add privs={file_write}:/tmp/mysql.sock
MySQL Service:mysql_57>  add privs={file_write}:/var/tmp/ib*
MySQL Service:mysql_57> end
MySQL Service>  set uid=mysql
MySQL Service>  set gid=mysql
MySQL Service>  exit
```

   `file_write` 特権は、デフォルトですべてのプロセスに付与されている基本特権です。書き込み可能パスを明示的に列挙することで、書き込みアクセスがそれらのパスだけに制限されます。この制約は、指定されている実行可能ファイルとその子プロセスに適用されます。

4. **MySQL のデフォルトポートを特権ポートにします。**

```
# ipadm set-prop -p extra_priv_ports+=3306 tcp
# ipadm show-prop -p extra_priv_ports tcp
PROTO PROPERTY            PERM CURRENT     PERSISTENT   DEFAULT    POSSIBLE
tcp   extra_priv_ports    rw   2049,4045,  3306         2049,4045  1-65535
                                 3306
```

   特権ポートにバインドするには `net_privaddr` 特権が必要です。MySQL の場合、デフォルトポート番号 3306 へのバインディングでは、通常はこの権限は不要です。

5. **権利プロファイルを MySQL サービスに割り当て、サービスに対してそれを使用するように指示します。**

```
# svccfg -s mysql:version_57
...version_57> setprop method_context/profile="MySQL Service"
```

```
...version_57> setprop method_context/use_profile=true
...version_57> refresh
...version_57> exit
```

6. **サービスを有効にします。**

   このサービスを一意に指定するには、FMRI の最後のコンポーネント `mysql:version_57` で十分です。

```
#  svcadm enable  mysql:version_57
```

7. **(オプション) MySQL Service 権利プロファイルに指定されている権利を使用してサービスが実行されていることを確認します。**

```
# ppriv $(pgrep mysql)
103697:   /usr/mysql/5.7/bin/mysqld --basedir=/usr/mysql/5.7
                                    --datadir=/var/mysql/5.7/data
flags =   PRIV_XPOLICY
      Extended policies:
                  {net_privaddr}:3306/tcp
                  {file_write}:/var/mysql/5.7/data/*
                  {file_write}:/tmp/mysql.sock
                  {file_write}:/var/tmp/ib*
        E: basic,!file_write
        I: basic,!file_write
        P: basic,!file_write
        L: all
103609:  /bin/sh /usr/mysql/5.7/bin/mysqld_safe --user=mysql
                               --datadir=/var/mysql/5.7/data
flags =  PRIV_XPOLICY
      Extended policies:
                  {net_privaddr}:3306/tcp
                  {file_write}:/var/mysql/5.7/data/*
                  {file_write}:/tmp/mysql.sock
                  {file_write}:/var/tmp/ib*
        E: basic,!file_write
        I: basic,!file_write
        P: basic,!file_write
        L: all
```

▼ 特定の特権を に割り当てる方法

この手順では、必要な特権のみを Web サーバーデーモンに割り当てることで、その Web サーバーデーモンをロックダウンします。Web サーバーはポート 80 だけにバインドでき、また `webservd` デーモンが所有するファイルだけに書き込むことができます。`root` として実行される `apache24` サービスプロセスはありません。

始める前に `root` 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **Web サーバー権利プロファイルを作成します。**

```
# profiles -p "Apache2"
profiles:Apache2> set desc="Apache HTTP Server Extended Privilege"
profiles:Apache2> add cmd=/lib/svc/method/http-apache24
```

```
profiles:Apache2:http-apache24> add privs={net_privaddr}:80/tcp
...http-apache24> add privs={zone}:/system/volatile/apache2
...http-apache24> add privs={zone}:/var/apache2/2.4/logs/*
...http-apache24> add privs={zone}:/var/user
...http-apache24> add privs={file_write}:/var/user/webserv*
...http-apache24> add privs={file_write}:/tmp/*
...http-apache24> add privs={file_write}:/system/volatile/apache*
...http-apache24> add privs={file_write}:/proc/*
...http-apache24> add privs=basic,proc_priocntl
...http-apache24> set uid=webservd
...http-apache24> set gid=webservd
...http-apache24> end
---Apache2> exit
```

2. **apache24 SMF 起動メソッドに権利プロファイルを追加します。**

```
# svccfg -s apache24
svc:/network/http:Apache2> listprop start/exec
start/exec  astring "/lib/svc/method/http-apache24 start"
...
svc:/network/http:Apache2> setprop start/profile="Apache2"
svc:/network/http:Apache2> setprop start/use_profile=true
svc:/network/http:Apache2> refresh
svc:/network/http:Apache2> exit
```

   apache24 サービスが有効化されると、Apache2 プロファイルが使用されます。

3. **apache24 サービスを有効にします。**

```
# svcadm enable apache24
```

4. **Web サーバーが機能していることを確認します。**

   ブラウザを開き、Firefox の URL フィールドに `localhost` と入力します。

次の手順 特権が正しく適用されていることを確認するため、102 ページの「 が使用している特権を判断する方法」に進みます。

▼ が使用している特権を判断する方法

このタスクでは、Apache2 権利プロファイルのデバッグバージョンを作成して、Web サーバーが使用している特権を判断します。

始める前に 101 ページの「特定の特権を に割り当てる方法」を完了しています。`apache24` サービスが無効になっています。`root` に含まれています。

1. **別のコマンドを呼び出すため、Apache2 プロファイルをクローニングします。**

   コマンドのデバッグは、SMF サービスのデバッグよりも単純です。`apachectl` コマンドは、Apache サービスを対話式に起動します。

```
# profiles -p "Apache2"
profiles:Apache2> set name="Apache-debug"
```

```
profiles:Apache-debug> sel <Tab><Tab>
profiles:Apache-debug:http-apache24> set id=/usr/apache2/2.4/bin/apachectl
profiles:Apache-debug:apachectl> end
profiles:Apache-debug> exit
```

詳細は、apachectl(8) のマニュアルページを参照してください。

2. **クローニングしたプロファイルを webservd アカウントに割り当てます。**

```
# usermod -K profiles+=Apache-debug webservd
```

3. **webservd 識別情報に切り替えます。**

```
# su - webservd
```

4. **(オプション) この識別情報を検証します。**

```
# id
uid=80(webservd) gid=80(webservd)
```

5. **プロファイルシェルで、Web サービスをデバッグモードで起動します。**

SMF を直接使用しないでください。Apache-debug 権利プロファイルのコマンドを使用します。

```
$ pfbash
# ppriv -De /usr/apache2/2.4/bin/apachectl start
```

6. **root 役割で、最初の http デーモンの特権を調べます。**

```
# ppriv $(pgrep httpd|head -1)
2999:   httpd
flags = PRIV_DEBUG|PRIV_XPOLICY|PRIV_EXEC
  5         Extended policies:
  6                 {net_privaddr}:80/tcp
  7                 {zone}:/system/volatile/apache2
  8                 {zone}:/var/apache2/2.4/logs/*
  9                 {zone}:/var/user
 10                 {file_write}:/var/user/webserv*
 11                 {file_write}:/tmp/*
 12                 {file_write}:/system/volatile/apache*
 13                 {file_write}:/proc/*
 14         E: basic,!file_write,!proc_info,proc_priocntl
 15         I: basic,!file_write,!proc_info,proc_priocntl
 16         P: basic,!file_write,!proc_info,proc_priocntl
 17         L: all
```

ユーザーによる自身が実行しているアプリケーションのロックダウン

ユーザーは、拡張特権ポリシーを使用してアプリケーションから基本特権を削除できます。このポリシーは、アプリケーションがアクセスすべきではないディレクトリへのアクセスを防ぎます。

注記 - 順序は重要です。ほとんどの`$HOME/.*` ディレクトリに対して限定的な特権を割り当てたあとで、`$HOME/Download*` などのディレクトリに対してより広範な特権を割り当てる必要があります。

例 42 保護されている環境でのブラウザの実行

この例では、保護されている環境でユーザーが Firefox ブラウザを実行できる方法を示します。この構成では、ユーザーの `Documents` ディレクトリが Firefox では非表示になります。

ユーザーは次のコマンドを使用して、`/usr/bin/firefox` コマンドから基本特権を削除します。`ppriv -r` コマンドの拡張特権引数は、ブラウザによる読み取りと書き込みを、ユーザーが指定したディレクトリに制限します。`-e` オプションとその引数により、拡張特権ポリシーを使用してブラウザが開かれます。

```
$ ppriv -r "\
{file_read}:/dev/*,\
{file_read}:/etc/*,\
{file_read}:/lib/*,\
{file_read}:/usr/*,\
{file_read}:/var/*,\
{file_read}:/proc,\
{file_read}:/proc/*,\
{file_read}:/system/volatile/*,\
{file_write}:$HOME,\
{file_read}:$HOME/.*,\
{file_read,file_write}:$HOME/.mozill*,\
{file_read,file_write}:$HOME/.gnome*,\
{file_read,file_write}:$HOME/Downloa*,\
{file_read,file_write}:/tmp,\
{file_read,file_write}:/tmp/*,\
{file_read,file_write}:/var/tmp,\
{file_read,file_write}:/var/tmp/*,\
{proc_exec}:/usr/*\
" -e /usr/bin/firefox file:///$HOME/Desktop
```

拡張ポリシーで `file_read` および `file_write` 特権が使用されている場合は、読み取りまたは書き込みが必要なすべてのファイルに対する明示的なアクセス権を付与する必要があります。このようなポリシーではワイルドカード文字 `*` を使用する必要があります。

自動マウントされたホームディレクトリを操作するため、ユーザーが次の例のように自動マウントパスの明示的なエントリを追加することがあります。

```
{file_read,file_write}:/export/home/$USER
```

サイトで `automount` 機能が使用されていない場合は、保護ディレクトリの最初のリストで十分です。

ユーザーはシェルスクリプトを作成して、このコマンド行保護ブラウザを自動化できます。その後ブラウザを起動するには、ユーザーは `/usr/bin/firefox` コマンドではなくスクリプトを呼び出します。

例 43 アプリケーションプロセスからのシステム上のディレクトリの保護

この例では、通常のユーザーがシェルスクリプトラッパーを使用してアプリケーションのサンドボックスを作成します。このスクリプトの前半では、アプリケーションが特定のディレクトリに制限されます。Firefox などの例外の処理は、このスクリプトの後半で行われます。スクリプトの後に、スクリプトの各部分に関するコメントを示します。

```
1 #!/bin/bash
2
3 # Using bash because ksh misinterprets extended policy syntax
4
5 PATH=/usr/bin:/usr/sbin:/usr/gnu/bin
6
7 DENY=file_read,file_write,proc_exec,proc_info
8
9 SANDBOX="\
10 {file_read}:/dev/*,\
11 {file_read}:/etc/*,\
12 {file_read}:/lib/*,\
13 {file_read,file_write}:/usr/*,\
14 {file_read}:/proc,\
15 {file_read,file_write}:/proc/*,\
16 {file_read}:/system/volatile/*,\
17 {file_read,file_write}:/tmp,\
18 {file_read,file_write}:/tmp/*,\
19 {file_read,file_write}:/var/*,\
20 {file_write}:$HOME,\
21 {file_read}:$HOME/.*,\
22 {file_read,file_write}:$PWD,\
23 {file_read,file_write}:$PWD/*,\
24 {proc_exec}:/usr/*\
25 "
26
27 # Default program is restricted bash shell
28
29 if [[ ! -n $1 ]]; then
30     program="/usr/bin/bash --login --noprofile
       --restricted"
31 else
32     program="$@"
33 fi
34
35
36 # Firefox needs more file and network access
37 if [[ "$program" =~ firefox ]]; then
38     SANDBOX+=",\
39 {file_read,file_write}:$HOME/.gnome*,\
40 {file_read,file_write}:$HOME/.mozill*,\
41 {file_read,file_write}:$HOME/.dbu*,\
42 {file_read,file_write}:$HOME/.puls*\
43 "
44
45 else
46     DENY+=",net_access"
47 fi
48
49 echo Starting $program in sandbox
50 ppriv -s I-$DENY -r $SANDBOX -De $program
```

ポリシーを調整して、特定のアプリケーションに対して許可するアクセス権を増やすかまたは減らすことができます。行 38 から 42 での調整では、Firefox に対し、ユーザーのホームディレクトリ内のセッション情報を保持するドットファイルへの書き込みアクセスが付与されています。また Firefox は、ネットワークアクセスを削除する行 46 の対象ではありません。ただし Firefox は、ユーザーのホームディレクトリ内の任意のファイルの読み取りが制限されており、現行ディレクトリにのみファイルを保存できます。

保護を強化するため、行 30 でデフォルトプログラムが制限付き Bash シェルになっています。制限付きシェルは、その現行ディレクトリを変更したり、ユーザーのドットファイルを実行したりすることはできません。したがって、このシェルから開始するすべてのコマンドは同様にサンドボックスにロックされます。

スクリプトの最後の行で、`ppriv` コマンドに 2 つの特権セット `$DENY` および `$SANDBOX` がシェル変数として渡されます。

最初のセットである `$DENY` は、プロセスに対し、ファイルの読み取りと書き込み、サブプロセスの実行、ほかのユーザーのプロセスの監視、および (条件付きでの) ネットワークへのアクセスを禁止します。これらの制限は非常に厳しいため、2 番目のセットである `$SANDBOX` で、読み取り、書き込み、および実行が可能なディレクトリを列挙することでこのポリシーが調整されています。

また行 50 ではデバッグオプション `-D` が指定されています。アクセス失敗は端末ウィンドウにリアルタイムで表示され、これには名前付きオブジェクトと、正常な実行に必要な該当する特権が含まれています。このデバッグ情報は、ユーザーがほかのアプリケーション向けにポリシーをカスタマイズする際に役立ちます。

不変ゾーンの管理

不変ゾーンはシステムのファイルの整合性を保証します。ゾーンのポリシーは、変更可能なファイルを指定します。システムのファイルを変更するには、管理者は Trusted Path Domain (TPD) に入ることが許可されている必要があります。 では、柔軟性と普遍性のトレードオフを提供する、不変ゾーン用のいくつかのポリシーが用意されています。Unresolved link to " mwac7" のマニュアルページでは、ゾーンを不変にするために適用できるポリシーが説明されており、Unresolved link to " tpd7" のマニュアルページでは Trusted Path Domain について説明されています。

『Creating and Using Oracle Solaris Zones』 の 第 12 章, 「Configuring and Administering Immutable Zones」では、不変ゾーンを構成および管理する方法が説明されています。

不変ゾーンを管理するときは 2 つのオプションがあります。

- 大域ゾーンの端末ウィンドウにアクセスできる場合、ゾーンを可変に変更し、管理したあとでゾーンを不変に戻します。

このオプションでは TPD は使用されません。一時的に可変にすることによって不変ゾーンを管理する方法については、『Creating and Using Oracle Solaris Zones』の「Administering Immutable Non-Global Zones」を参照してください。

- コンソールまたは RAD インタフェースにアクセスできる場合、ゾーンを不変のままにしてもよく、Trusted Path Domain (TPD) に認証してゾーンに入り、ゾーンを管理してから TPD を終了します。

  セキュリティーの高いこのオプションでは、ゾーン管理者が TPD に入る権利を持っている必要があります。RAD アクセスの場合、RAD プロセスが信頼できるパスで実行されていることが必要です。『Creating and Using Oracle Solaris Zones』の「Administering an Immutable Zone by Using the Trusted Path Domain」を参照してください。

これらの方法のためのステップは、次の各セクションで説明します。

- 可変ゾーン管理 – `zlogin -T` または `zoneadm` コマンドの使用については、『Creating and Using Oracle Solaris Zones』の「Administering an Immutable Zone by Making It Writable」を参照してください。
- 物理または仮想コンソールエントリ – 『Creating and Using Oracle Solaris Zones』の「How to Enable Administrative Access to an Immutable Zone From the Console」を参照してください。
- リモート RAD エントリ – 『Creating and Using Oracle Solaris Zones』の「How to Enable Remote Administrative Access to an Immutable Zone by Using RAD」を参照してください。

♦ ♦ ♦ 第 5 章

5 権利使用の管理

この章では、管理に権利モデルを使用するシステムを保守するタスクについて説明します。いくつかのタスクでは、新しい権利プロファイルと承認を作成することで、 が提供する権利を拡張します。

この章の内容は次のとおりです。

権利の詳細については、第1章「権利を使用したユーザーとプロセスの制御について」を参照してください。ユーザーと役割に割り当てられた権利の保守については、第3章「 での権利の割り当て」を参照してください。

権利使用の管理

このセクションのタスクと例では、割り当てられた権利を使用する方法と、デフォルトで提供される権利構成を変更する方法について説明します。

注記 - トラブルシューティングの支援については、143 ページの「RBAC と特権のトラブルシューティング」を参照してください。

- 割り当てられた権利を使用する – 110 ページの「割り当てられている管理権利の使用」
- 管理アクションを監査する – 例48「2 つの役割を使用した監査の構成」
- 権利プロファイルと承認を追加する – 115 ページの「権利プロファイルと承認の作成」
- root をユーザーとして構成する – 120 ページの「root 役割をユーザーに変更する方法」
- root を役割に戻す – 例54「root ユーザーを root 役割に変更する」

- root によるシステムの管理を妨げる – 例55「システム保守での root 役割の使用の防止」

割り当てられている管理権利の使用

root 役割では、初期ユーザーにすべての管理権限が与えられます。このユーザーは root として、役割、権利プロファイル、特定の特権などの管理権利と承認を信頼できるユーザーに割り当てることができます。このセクションでは、このようなユーザーが各自に割り当てられている権利を使用する方法を説明します。

注記 - には管理ファイル用の特殊なエディタがあります。管理ファイルを編集するときには pfedit コマンドを使用します。例44「システムファイルの編集」は、root 以外のユーザーが指定したシステムファイルを編集できるようにする方法を示しています。

管理タスクを実行するには、端末ウィンドウを開いて次のオプションのいずれかを選択します。

- sudo を使用している場合、sudo コマンドを入力します。

  sudo コマンドを使い慣れている管理者の場合は、sudoers ファイルで管理者に割り当てられている管理コマンドの名前を使用してこのコマンドを実行します。詳細は、sudo(8) および sudoers(5) のマニュアルページを参照してください。
- タスクにスーパーユーザー特権が必要な場合は、root になります。

```
$ su -
Password: xxxxxxxx
#
```

注記 - このコマンドは、root がユーザーまたは役割のいずれであっても機能します。ポンド記号 (#) のプロンプトは、ユーザーが現在 root であることを示します。

- タスクが役割に割り当てられている場合、そのタスクを実行できる役割を引き受けます。

  次の例では、監査構成の役割になります。この役割には、Audit Configuration 権利プロファイルが含まれています。管理者からこの役割のパスワードを受け取っています。

```
$ su - audadmin
Password: xxxxxxxx
#
```

ヒント - 役割のパスワードを受け取っていない場合は、管理者が、ユーザーのパスワードを求めるように役割を構成しています。役割を引き受けるには、ユーザーパスワードを入力します。このオプションの詳細については、例18「役割パスワードでのユーザー独自のパスワード使用の有効化」を参照してください。

このコマンドを入力したシェルは、プロファイルシェルになりました。このシェルでは、`auditconfig` コマンドを実行できます。プロファイルシェルの詳細については、40 ページの「プロファイルシェルと権利の検証」を参照してください。

ヒント - 役割の権利を表示するには、135 ページの「権利プロファイルの一覧表示」を参照してください。

- 自身のタスクがユーザーとして自身に直接割り当てられ、例76「プロファイルシェルを使用しているかどうかの判断」で説明しているようにプロファイルシェルを実行していない場合は、次のいずれかの方法でプロファイルシェルを作成してください。
  - `pfbash` コマンドを使用して、管理権利を評価するシェルを作成します。

    次の例では、Audit Configuration 権利プロファイルがユーザーに直接割り当てられています。次のコマンドセットにより、`pfbash` プロファイルシェルで監査事前選択値と監査ポリシーを表示できます。

    ```
    $ pfbash
    $ auditconfig -getflags
    active user default audit flags = ua,ap,lo(0x45000,0x45000)
    configured user default audit flags = ua,ap,lo(0x45000,0x45000)
    $ auditconfig -getpolicy
    configured audit policies = cnt
    active audit policies = cnt
    ```

  - 1 つの管理コマンドを実行するには、`pfexec` コマンドを使用します。

    次の例では、Audit Configuration 権利プロファイルが認証権利プロファイルとしてユーザーに直接割り当てられています。特権付きコマンドの名前を指定した `pfexec` コマンドを使用して、このプロファイルからその特権付きコマンドを実行できます。たとえば、ユーザーの事前選択監査フラグを表示できます。

    ```
    $ pfexec auditconfig -getflags
    Enter password:       Type your user password
    active user default audit flags = ua,ap,lo(0x45000,0x45000)
    configured user default audit flags = ua,ap,lo(0x45000,0x45000)
    ```

    一般に、権利に含まれている別の特権付きコマンドを実行するには、その特権付きコマンドを入力する前に、`pfexec` を再度入力する必要があります。詳

細は、pfexec(1) のマニュアルページを参照してください。パスワードキャッシュを使用して構成されている場合は、例45「役割の使用を容易にするために認証をキャッシュする」に示すように、構成可能な期間内ではパスワードを入力せずに後続のコマンドを実行できます。

例 44 システムファイルの編集

0 の UID を持つ root でない場合は、デフォルトでは、システムファイルを編集できません。ただし、solaris.admin.edit/*path-to-system-file* 承認が割り当てられている場合は、*system-file* を編集できます。たとえば solaris.admin.edit/etc/security/audit_warn 承認が割り当てられている場合は、pfedit コマンドを使用して audit_warn ファイルを編集できます。

```
# pfedit /etc/security/audit_warn
```

詳細は、pfedit(4) のマニュアルページを参照してください。このコマンドは、すべての管理者が使用できます。

例 45 役割の使用を容易にするために認証をキャッシュする

この例では、監査構成を管理するための役割を管理者が構成しますが、ユーザーの認証をキャッシュすることで使用が容易になります。最初に、管理者は役割を作成して割り当てます。

```
# roleadd -K roleauth=user -K profiles="Audit Configuration" audadmin
# usermod -R +audadmin jdoe
```

jdoe が、その役割に切り替えるときに -c オプションを使用した場合は、auditconfig の出力が表示される前にパスワードが必要になります。

```
$ su - audadmin -c auditconfig option
Password: xxxxxxxx
     auditconfig output
```

認証がキャッシュされない場合は、jdoe がコマンドを再度実行するときにパスワードプロンプトが表示されます。

管理者が、認証のキャッシュを有効にする su スタックを保持するファイルを、pam.d ディレクトリに作成します。認証をキャッシュするときには、初回はパスワードが必要ですが、その後は一定の時間が経過するまでは、パスワードは不要です。

```
$ pfedit /etc/pam.d/su
## Cache authentication for switched user
#
auth required           pam_unix_cred.so.1
auth sufficient         pam_tty_tickets.so.1
auth requisite          pam_authtok_get.so.1
auth required           pam_dhkeys.so.1
```

```
auth required           pam_unix_auth.so.1
```

このファイルを作成したあと、管理者は、各エントリにタイポ、脱字、または繰り返しがないかチェックします。

管理者は、前の su スタック全体を提供する必要があります。pam_tty_tickets.so.1 モジュールにはキャッシュが実装されています。PAM の詳細は、Unresolved link to " pam_tty_tickets7" および pam.conf(5) のマニュアルページおよび 『Managing Authentication in Oracle Solaris 11.4』 の 第 1 章, 「Using Pluggable Authentication Modules」 を参照してください。

管理者が su PAM ファイルを追加してシステムをリブートしたあと、audadmin 役割を含むすべての役割は、一連のコマンドを実行しているときに 1 回だけパスワードの入力を要求されます。

```
$ su - audadmin -c auditconfig option
Password: xxxxxxxx
     auditconfig output
$ su - audadmin -c auditconfig option
     auditconfig output
...
```

例 46 root 役割になる

次の例では、最初のユーザーが root 役割になり、その役割のシェルで特権を一覧表示します。

```
$ roles
root
$ su - root
Password: xxxxxxxx
#       Prompt changes to root prompt
# ppriv $$
1200:   pfksh
flags = <none>
        E: all
        I: basic
        P: all
        L: all
```

特権については、28 ページの「プロセス権管理」および ppriv(1) のマニュアルページを参照してください。

例 47 ARMOR 役割の引き受け

この例では、ユーザーは管理者が割り当てた ARMOR 役割を引き受けます。

端末ウィンドウで、ユーザーは割り当てられている役割を確認します。

```
$ roles
fsadm
```

```
sysop
```

その後ユーザーは fsadm 役割を引き受け、ユーザーパスワードを入力します。

```
$ su - fsadm
Password: xxxxxxxx
#
```

su - *rolename* コマンドは、端末のシェルをプロファイルシェルに変更します。これでユーザーは、この端末ウィンドウで fsadm 役割になりました。

この役割で実行できるコマンドを確認するため、ユーザーは 135 ページの「権利プロファイルの一覧表示」の手順に従います。

管理アクションの監査

しばしば、サイトのセキュリティーポリシーで管理アクションの監査が要求されることがあります。116:AUE_PFEXEC:execve(2) with pfexec enabled:ps,ex,ua,as 監査イベントはこのようなアクションを捕捉します。管理アクションを監査する際にもう 1 つのオプションとして、役割での使用に適したイベントグループを提供する cusa メタクラスがあります。詳細については、/etc/security/audit_class ファイルのコメントを参照してください。

例 48 2 つの役割を使用した監査の構成

この例では、2 人の管理者がサイトセキュリティー担当者の監査構成計画を実装します。この計画では、すべてのユーザーに pf クラスを使用し、個別の役割に cusa メタクラスを指定します。root 役割は、監査フラグを役割に割り当てます。1 番目の管理者が監査を構成し、2 番目の管理者が新しい構成を有効にします。

1 番目の管理者に、Audit Configuration 権利プロファイルが割り当てられます。この管理者は現在の監査構成を表示します。

```
# auditconfig -getflags
active user default audit flags = lo(0x1000,0x1000)
configured user default audit flags = lo(0x1000,0x1000)
```

pf クラスには lo クラスが含まれていないため、管理者はこのクラスをシステム構成に追加します。

```
# auditconfig -setflags lo,pf
```

新しい監査構成をカーネルに読み込むには、Audit Control 権利プロファイルが割り当てられているユーザーが、監査サービスをリフレッシュします。

```
# audit -s
```

権利プロファイルと承認の作成

提供される権利プロファイルに必要な権利の集合が含まれていない場合は、権利プロファイルを作成または変更できます。限られた権利を持つユーザー、新しいアプリケーション、あるいはそのほかの目的で権利プロファイルを作成できます。

が提供する権利プロファイルは読み取り専用です。提供される権利プロファイルの一連の権利では不十分な場合は、その権利プロファイルをクローニングできます。たとえば、提供される権利プロファイルに `solaris.admin.edit/`*path-to-system-file* 承認を追加できます。背景情報については、25 ページの「権利プロファイルの詳細」を参照してください。

提供される承認に、特権付きアプリケーションでコーディングされている承認が含まれていない場合は、承認を作成できます。既存の承認は変更できません。背景情報については、25 ページの「ユーザー承認に関する詳細」を参照してください。

権利プロファイルの特権を制限する例については、例29「リモートユーザー権利プロファイルの作成」および例30「権利プロファイルからの基本特権の削除」を参照してください。

▼ 権利プロファイルを作成する方法

始める前に `root` 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **権利プロファイルを作成します。**

   ```
   # profiles -p [-S repository] profile-name
   ```

   説明を入力するよう求められます。

2. **権利プロファイルに内容を追加します。**

   1 つの値を持つプロファイルプロパティーには `set` サブコマンドを使用します (`set desc` など)。複数の値を指定できるプロパティーには `add` サブコマンドを使用します (`add cmd` など)。

   次のコマンドは、『Managing Authentication in Oracle Solaris 11.4』 の 「How to Assign a Modified PAM Policy」 のカスタム PAM 権利プロファイルを作成します。読みやすくするため、名前が短縮されています。

   ```
   # profiles -p -S LDAP "Site PAM LDAP"
   profiles:Site PAM LDAP> set desc="Profile which sets pam_policy=ldap"
   ...LDAP> set pam_policy=ldap
   ...LDAP> commit
   ...LDAP> end
   ...LDAP> exit
   ```

例 49 特権付きコマンドを含む権利プロファイルの作成

この例では、セキュリティー管理者は、その管理者が作成した権利プロファイル内のアプリケーションに特権を追加します。このアプリケーションは特権を認識できます。

```
# profiles -p SiteApp
profiles:SiteApp> set desc="Site application"
profiles:SiteApp> add cmd="/opt/site-app/bin/site-cmd"
profiles:SiteApp:site-cmd> add privs="proc_fork,proc_taskid"
profiles:SiteApp:site-cmd> end
profiles:SiteApp> exit
```

確認するには、管理者は `site-cmd` を選択します。

```
# profiles -p SiteApp "select cmd=/opt/site-app/bin/site-cmd; info;end"
Found profile in files repository.
  id=/opt/site-app/bin/site-cmd
  privs=proc_fork,proc_taskid
```

次の手順 信頼できるユーザーまたは役割に権利プロファイルを割り当てます。例については、例12「DHCP を管理する信頼できるユーザーの作成」および 例22「信頼できるユーザーによる拡張アカウントファイル読み取りの有効化」を参照してください。

参照 権利割り当てのトラブルシューティングを行うには、144 ページの「権利割り当てをトラブルシューティングする方法」を参照してください。背景情報については、40 ページの「割り当てられた権利の検索順序」を参照してください。

▼ システム権利プロファイルをクローニングおよび変更する方法

始める前に `root` 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **既存のプロファイルから新しい権利プロファイルを作成します。**

   ```
   # profiles -p [-S repository] existing-profile-name
   ```

   - **既存の権利プロファイルに内容を追加するには、新しいプロファイルを作成します。**

     新しいプロファイルに既存の権利プロファイルを補助権利プロファイルとして追加してから、拡張を追加します。例50「Network IPsec Management 権利プロファイルのクローニングと拡張」を参照してください。

   - **既存の権利プロファイルから内容を削除するには、そのプロファイルをクローニングし、名前を変更して変更を行います。**

     例51「権利プロファイルでの選択した権利のクローニングおよび削除」を参照してください。

2. **補助権利プロファイル、承認、およびその他の権利を追加または削除して、新しい権利プロファイルを変更します。**

例 50 Network IPsec Management 権利プロファイルのクローニングと拡張

この例では、管理者がサイトの IPsec Management 権利プロファイルに `solaris.admin.edit` 承認を追加し、これにより `root` 役割が不要になります。この権利プロファイルは、信頼されており `/etc/hosts` ファイルを変更できるユーザーだけに割り当てられます。

1. 管理者は、Network IPsec Management 権利プロファイルを変更できないことを確認します。

```
# profiles -p "Network IPsec Management"
profiles:Network IPsec Management> add auths="solaris.admin.edit/etc/hosts"
Cannot add. Profile cannot be modified
```

2. 管理者は、Network IPsec Management プロファイルを含む権利プロファイルを作成します。

```
# profiles -p "Total IPsec Mgt"
... IPsec Mgt> set desc="Network IPsec Mgt plus /etc/hosts"
... IPsec Mgt> add profiles="Network IPsec Management"
... IPsec Mgt> add auths="solaris.admin.edit/etc/hosts"
... IPsec Mgt> end
... IPsec Mgt> exit
```

3. 管理者は、内容を確認します。

```
# profiles -p "Total IPsec Mgt" info
        name=Total IPsec Mgt
        desc=Network IPsec Mgt plus /etc/hosts
        auths=solaris.admin.edit/etc/hosts
        profiles=Network IPsec Management
```

例 51 権利プロファイルでの選択した権利のクローニングおよび削除

この例では、管理者は VSCAN サービスのプロパティーの管理を、このサービスを有効および無効にする機能から分離します。

最初に、管理者は、 が提供する権利プロファイルの内容を一覧表示します。

```
# profiles -p "VSCAN Management" info
        name=VSCAN Management
        desc=Manage the VSCAN service
        auths=solaris.smf.manage.vscan,solaris.smf.value.vscan,
              solaris.smf.modify.application
```

次に、管理者はサービスを有効化および無効化できる権利プロファイルを作成します。

```
# profiles -p "VSCAN Management"
profiles:VSCAN Management> set name="VSCAN Control"
profiles:VSCAN Control> set desc="Start and stop the VSCAN service"
... VSCAN Control> remove auths="solaris.smf.value.vscan"
... VSCAN Control> remove auths="solaris.smf.modify.application"
... VSCAN Control> end
... VSCAN Control> exit
```

次に、管理者は、サービスのプロパティーを変更できる権利プロファイルを作成します。

```
# profiles -p "VSCAN Management"
profiles:VSCAN Management> set name="VSCAN Properties"
profiles:VSCAN Properties> set desc="Modify VSCAN service properties"
... VSCAN Properties> remove auths="solaris.smf.manage.vscan"
... VSCAN Properties> end
... VSCAN Properties> exit
```

管理者は、新しい権利プロファイルの内容を確認します。

```
# profiles -p "VSCAN Control" info
        name=VSCAN Control
        desc=Start and stop the VSCAN service
        auths=solaris.smf.manage.vscan
# profiles -p "VSCAN Properties" info
        name=VSCAN Properties
        desc=Modify VSCAN service properties
        auths=solaris.smf.value.vscan,solaris.smf.modify.application
```

次の手順 信頼できるユーザーまたは役割に権利プロファイルを割り当てます。例については、例12「DHCP を管理する信頼できるユーザーの作成」および 例22「信頼できるユーザーによる拡張アカウントファイル読み取りの有効化」を参照してください。

参照 権利割り当てのトラブルシューティングを行うには、144 ページの「権利割り当てをトラブルシューティングする方法」を参照してください。背景情報については、40 ページの「割り当てられた権利の検索順序」を参照してください。

▼ 承認を作成する方法

始める前に 開発者は、ユーザーがインストールするアプリケーションで承認を定義および使用しています。手順については、『Oracle Solaris 12 セキュリティーサービス開発ガイド』の 「承認について」を参照してください。

- **`auths add` コマンドを使用して承認を作成します。**

  たとえば、次のコマンドは、`com.newco.siteapp.data.modify` 承認をローカルシステムに作成します。

  ```
  # auths add -t "SiteApp Data Modify Authorized" com.newco.siteapp.data.modify
  ```

  これで承認をテストし、権利プロファイルに追加し、役割またはユーザーにそのプロファイルを割り当てることができます。

例 52 割り当て済みの承認のテストおよび削除

この例では、管理者が例49「特権付きコマンドを含む権利プロファイルの作成」の SiteApp 権利プロファイルを使用して `com.newco.siteapp.data.modify` 承認をテストします。

```
# usermod -A com.newco.siteapp.data.modify -P SiteApp tester1
```

テストが正常に完了したら、管理者は承認を削除します。

```
# rolemod -A-=com.newco.siteapp.data.modify siteapptester
```

保守を容易にするため、管理者は例53「権利プロファイルへの承認の追加」の SiteApp 権利プロファイルにこの承認を追加します。

例 53 権利プロファイルへの承認の追加

承認が適切に機能することをテストしたら、セキュリティー管理者は `com.newco.siteapp.data.modify` 承認を既存の権利プロファイルに追加します。例49「特権付きコマンドを含む権利プロファイルの作成」に、管理者がプロファイルを作成した方法を示します。

```
# profiles -p "SiteApp"
profiles:SiteApp> add auths="com.newco.siteapp.data.modify"
profiles:SiteApp> end
profiles:SiteApp> exit
```

確認するために、管理者はプロファイルの内容を一覧表示します。

```
# profiles -p SiteApp
Found profile in files repository.
  id=/opt/site-app/bin/site-cmd
  auths=com.newco.siteapp.data.modify
```

次の手順 信頼できるユーザーまたは役割に権利プロファイルを割り当てます。例については、例12「DHCP を管理する信頼できるユーザーの作成」および例22「信頼できるユーザーによる拡張アカウントファイル読み取りの有効化」を参照してください。

参照 権利割り当てのトラブルシューティングを行うには、144 ページの「権利割り当てをトラブルシューティングする方法」を参照してください。背景情報については、40 ページの「割り当てられた権利の検索順序」を参照してください。

root がユーザーまたは役割のいずれであるかの変更

デフォルトでは、root は の役割です。ユーザーに変更し、再度役割に変更するか、またはこれを削除して使用しないようにするオプションがあります。

Oracle Enterprise Manager を使用している場合、または権利モデルではなく従来の管理のスーパーユーザーモデルに従っている場合は、root をユーザーに変更する必要があります。背景情報については、45 ページの「管理に使用する権利モデルの決定」を参照してください。

権利モデルに従っている場合は、ネットワークから取り外されたシステムを廃棄する場合に root をユーザーに変更することがあります。このシナリオでは、root としてシステムにログインするとクリーンアップが容易になります。

注記 - root 役割を使用してリモート管理を行う場合は、セキュアなリモートログインの手順について、『Managing Secure Shell Access in Oracle Solaris 11.4』 の 「How to Remotely Administer ZFS With Secure Shell」を参照してください。

一部のサイトでは、本番システムでは root が正当なアカウントではないことがあります。root の使用を削除するには、例55「システム保守での root 役割の使用の防止」を参照してください。

▼ root 役割をユーザーに変更する方法

この手順は、root がシステムに直接ログインできる必要があるシステムで行う必要があります。

始める前に root 役割になる必要があります。

1. **root 役割の割り当てをローカルユーザーから削除します。**

   たとえば、その役割の割り当てを 2 人のユーザーから削除します。

```
$ su -
Password: xxxxxxxx
# roles jdoe
root
# roles kdoe
root
# roles ldoe
secadmin
# usermod -R "" jdoe
# usermod -R "" kdoe
#
```

2. **root 役割をユーザーに変更します。**

```
# rolemod -K type=normal root
```

   現在 root 役割になっているユーザーはそのままです。root アクセスを持つほかのユーザーは、su を実行して root に変更することも、root ユーザーとしてシステムにログインすることもできます。

3. **変更内容を確認します。**

次のいずれかのコマンドを使用できます。

- **root の user_attr エントリを調べます。**

  ```
  # getent user_attr root
  root::::auths=solaris.*;profiles=All;audit_flags=lo\:no;lock_after_retries=no;
  min_label=admin_low;clearance=admin_high
  ```

  type キーワードが出力内に見つからないか、または normal に等しい場合、そのアカウントは役割ではありません。

- **userattr コマンドからの出力を表示します。**

  ```
  # userattr type root
  ```

  出力が空であるか、または normal が表示される場合、そのアカウントは役割ではありません。

例 54 root ユーザーを root 役割に変更する

この例では、root ユーザーが root ユーザーを役割に戻します。

最初に、root ユーザーは root アカウントを役割に変更し、その変更内容を確認します。

```
# usermod -K type=role root
# getent user_attr root
root::::type=role...
```

次に、root は root 役割をローカルユーザーに割り当てます。

```
# usermod -R root jdoe
```

例 55 システム保守での root 役割の使用の防止

この例では、サイトのセキュリティーポリシーで、root アカウントによってシステムが保守されないように要求します。管理者は、システムを保守する役割をすでに作成し、テストしています。これらの役割には、すべてのセキュリティープロファイルと System Administrator 権利プロファイルが含まれています。信頼できるユーザーには、バックアップを復元できる役割が割り当てられています。ユーザー、役割、または権利プロファイルの監査フラグを変更するか、または役割のパスワードを変更することができる役割はありません。

root アカウントがシステムの保守に使用されないようにするために、セキュリティー管理者は、root 役割の割り当てを削除します。root アカウントはシングルユーザーモードでシステムにログインできる必要があるため、そのアカウントのパスワードは保持されます。

```
# usermod -K roles= jdoe
# userattr roles jdoe
```

注意事項 デスクトップ環境では、root が役割の場合は root として直接ログインすることはできません。このシステム上で root が役割になっていることを示す診断メッセージが表示されます。

root 役割を引き受けることができるローカルアカウントがない場合は、次のステップを実行します。

- root としてシングルユーザーモードでシステムにログインし、ローカルユーザーアカウントとパスワードを作成します。
- root 役割を新しいアカウントに割り当てます。
- この新しいユーザーとしてログインし、root 役割を引き受けます。

♦ ♦ ♦ 第 6 章

6 データ損失保護のためのプロセスのラベル付け

この章では、構成されたラベルポリシーを使用して、 システムのラベル付きファイルおよびファイルシステムにアクセスする方法について説明します。ラベルポリシーを使用して、選択されたファイルおよびファイルシステムにラベルを付けることができます。これらのラベル付きファイルを表示または変更できるのは、それらを処理するための認可上限を持っているユーザーだけです。特権ユーザーおよび役割であっても、ラベル付きファイルの内容へのアクセスを妨げられる場合があります。ラベル付きファイルおよびファイルシステムについては、『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』 の 第 2 章,「データ損失保護のためのファイルのラベル付け」を参照してください。

この章で扱う内容は、次のとおりです。

のプロセスラベルおよび認可上限について

は、ファイルとプロセスにラベルを付けます。デフォルトポリシーは透過的で、システムはラベルが存在しない場合と同様に動作します。ラベルポリシーを作成する管理者は、情報の機密性を指定するためにラベルをファイルに割り当てることができます。代表的なラベルとして「Public」と「Confidential - Restricted」があります。ラベルエンコーディングファイルはシステムのラベルを定義します。ラベルポリシーの作成とラベルの定義については、『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』 の 「ラベル付けを構成するためのプロセス全体」を参照してください。

ラベルポリシーは、ユーザープロセスおよび SMF サービスの開始ラベルの定義を含んでいます。システムのラベルを定義するラベルエンコーディングファイルは、ユーザープロセスの初期ラベルも定義します。エンコーディングファイル内の `clearance` 値は、「Confidential - Internal」などの組織内のユーザーに適したユーザー指定のラベルです。`account-policy` サービスの `login_policy/clearance` 値は、ほとんど

のプロセスが実行するときに指定されるラベルです。ラベルポリシーを作成するとき、CLEARANCE 値を ADMIN_LOW に設定します。リブート後、ユーザープロセスはエンコーディングファイルの clearance 値で開始し、SMF サービスはADMIN_LOW で開始します。承認されたユーザーおよび機密性の高いプロセスをさらに高いラベルで開始するには、承認されたユーザーおよび選択された SMF サービスに高い認可上限を構成する必要があります。ユーザーおよびプロセスに認可上限を割り当てるときに考慮すべき問題については、『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』 の 「ラベルポリシーのカスタマイズ」を参照してください。

ラベルが構成されていないときのデフォルトの認可上限はもっとも高いラベル「ADMIN_HIGH」であるため、ラベルによってアクセスは制限されません。

エンコーディングファイル内の clearance の値は、clearance セキュリティー属性の明示的なキー値が設定されていないユーザーまたは役割に適用されます。root 役割と、 のインストール中に作成された初期アカウントには、明示的な認可上限「ADMIN_HIGH」が割り当てられています。

注意 - root アカウントの明示的な「ADMIN_HIGH」認可上限は決して変更しないでください。

ユーザープロセスは、そのユーザーのプライマリログインプロセスの認可上限を継承します。現在のプロセスの認可上限を表示するには、端末ウィンドウで「plabel」と入力します。ユーザーは、自分の認可上限から「ADMIN_LOW」までのすべてのラベルにアクセスできます。次の例は、初期ユーザーおよび root が plabel コマンドを実行したときの表示内容を示します。

```
$ plabel
ADMIN_HIGH
```

ラベル付きファイルへのアクセスについて

プロセスのラベルが認可上限と呼ばれるのは、これらのラベルが、プロセスが認可されるもっとも高いラベルを示すためです。認可上限はラベル範囲の上限を示します。ユーザーがラベル付きデータにアクセスできるのは、ユーザーのプロセスのラベルが、そのデータを含むファイルのラベルよりも優位な場合です。同じように、ほかのプロセスがデータにアクセスできるのは、プロセスのラベルがファイルのラベルよりも優位な場合です。「優位である」とは、プロセスのラベルがデータのラベルと少なくとも同等か、それよりも高い可能性があることを意味します。たとえば、「Confidential - Restricted」の認可上限を持つユーザーは、そのラベルおよび「Confidential - Internal」や「Public」などのそれより低いすべてのラベルのデータにアクセスできます。

アクセス試行中に、 はテキスト文字列を内部表現に変換したり、その逆方向に変換したりします。プロセスが、そのプロセスのラベルと同等か、またはそれより優位なラ

ベルを変換しようとした場合、変換は許可されません。この制限をオーバーライドするには、sys_trans_label 特権が必要です。

通常のユーザーは組織のデフォルトの認可上限を継承するため、機密データへのアクセスは認可されません。管理者は、このデータにアクセスすることが必要なサービスおよびユーザーのみにより高い認可上限を割り当てます。ユーザー認可上限は、ユーザーが最初にログインしたときに有効になります。役割の引き受けなどの 2 回目のログインでは、最初のログインから認可上限が保持されます。

高い認可上限を持つユーザーは、低い認可上限で新しいプロセスを開始する sandbox コマンドを使用して、低い認可上限で動作できます。サンドボックス内で実行中のプロセスは分離されているため、サンドボックス外のプロセスを監視できません。詳細情報および例については、129 ページの「プロジェクトの分離のためのサンドボックスの構成」、例43「アプリケーションプロセスからのシステム上のディレクトリの保護」、および sandbox(1) と sandboxing(7) のマニュアルページを参照してください。

ラベルによりユーザーおよびプロセスを構成する

『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』の手順および例には、必要な認可上限をユーザーに構成するステップが含まれています。

- 『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』の 「ファイルシステムにラベルを割り当てる方法」。
- 『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』の 「ラベル付きファイルシステムをゾーン内に分離する方法」。
- 『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』の 「ラベル付き監査トレールを作成する方法」。

このセクションでは、次の情報について説明します。

ラベル付きファイルへのアクセスを有効にする

管理者は、ラベル付きファイルにアクセスすることが必要なユーザーに対して適切な認可上限を割り当てることを担当します。ラベル付きファイルを表示または変更できるのは、ファイルのラベルと少なくとも同じ認可上限を持つユーザーだけです。すべてのユーザーは、ラベルエンコーディングファイル経由で認可上限を受け取ります。

機密性の高いファイルへのアクセス権をユーザーに付与するには、さらに高い認可上限を持つようにユーザーを直接許可するか、高い認可上限で実行するコマンドを含む権利プロファイルを承認済みユーザーに割り当てることができます。また、高い認可上限で実行する権利プロファイルを持つ役割をユーザーに割り当てることもできます。

▼ ユーザーに認可上限を割り当てる方法

この手順では、権利プロファイルを通じて、または役割の割り当てを通じて高い認可上限をユーザーに直接割り当てる方法を示します。

始める前に User Management 権利プロファイルが割り当てられているか、または `root` 役割である必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **システムで使用可能なラベルを一覧表示します。**

   ```
   $ labelcfg list
   label-list-from-highest-to-lowest-label
   ```

2. **ラベル付きファイルを処理する能力を特定のユーザーまたは役割に割り当てます。**

   ```
   # usermod -K clearance=label username
   ```

   ```
   # rolemod -K clearance=label rolename
   ```

   また、権利プロファイルを通じて間接的に、ユーザーに認可上限を割り当てることもできます。

3. **ラベル付きファイルを処理するための高い認可上限で実行するコマンドを含む権利プロファイルを作成します。**

   コマンドには、高い認可上限に加えて十分な特権が必要です。十分な特権には、コマンドが実行するために十分な認可上限を持つ UID または EUID、またはコマンドが必要とする特権が含まれる場合があります。

   次の例の「Labeled Audit Review」権利プロファイルは、『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』 の 「ラベル付き監査トレールを作成する方法」からのものです。この権利プロファイルをユーザーに直接割り当てたり、ユーザーが引き受ける役割に割り当てたりできます。

   - **ユーザーに権利プロファイルを割り当てるには、`profiles+=` または `auth_profiles+=` キーワードを使用します。**

     ```
     # usermod -K profiles+="Labeled Audit Review" username
     # usermod -K auth_profiles+="Labeled Audit Review" username
     ```

注記 - ユーザーに Audit Review 権利プロファイルも割り当てられている場合、Labeled Audit Review プロファイルをその前に付ける必要があります。

- **権利プロファイルを役割に追加し、その役割をユーザーに割り当てるには、次のようにします。**

  a. **`profiles` キーワードを使用します。**

  ```
  # rolemod -K profiles+="Labeled Audit Review" rolename
  # rolemod -K auth_profiles+="Labeled Audit Review" rolename
  ```

  b. **役割をユーザーに割り当てます。**

  ```
  # usermod -R +rolename username
  ```

▼ ラベル付きファイルへのユーザーアクセスを検証する方法

ユーザーに認可上限を割り当てたあと、この構成によって、認可上限を持つユーザーは、認可されたファイルにアクセスできること、および認可上限を持たないユーザーは、ファイルの表示やバックアップ、またはこれらのファイルの監査トレールの表示ができないことを検証します。

1. **認可上限が割り当てられたユーザーになります。**

```
# su - cleared-user
cleared-user$ plabel
user's explicit clearance
```

2. **ラベル付きデータセットディレクトリに変更します。**

```
$ cd labeled-dataset
```

ゾーン内のラベル付きデータセットをテストするには、『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』 の 「ラベル付きファイルシステムをゾーン内に分離する方法」を参照してください。

3. **ユーザーが実行するタスクを実行します。**

次に例を示します。

- ディレクトリ内のファイルをリストします。
- ディレクトリにファイルを追加し、ファイルのラベルを表示します。
- ディレクトリからファイルを削除します。
- ディレクトリ内のファイルを変更します。
- ユーザーの認可上限内にある別のラベルのディレクトリに変更します。
- 同様のラベルが付いたファイルシステムにファイルを送信します。
- 別のユーザーに変更し、元のユーザーのファイルをラベルのないファイルシステムに送信してみます。

  このテストは失敗するはずです。

注記 - 高い認可上限で実行するコマンドを格納する権利プロファイルを割り当てられた場合、ユーザーは (`pfexec praudit` のように) プロファイルシェルでこれらのコマンドを実行する必要があります。

4. **`root` 役割で、`/usr/demo/tsol` ディレクトリから `auditfiles.ksh` スクリプトを実行して監査トレールを検査し、次にブラウザ内で出力を読み取ります。**

   監査トレールがラベル付きファイルシステムの一部である場合、`ADMIN_HIGH` ファイルを読み取るための認可上限が必要です。『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』 の 「ラベル付き監査トレールを作成する方法」を参照してください。次の例では、Labeled Audit Review 権利プロファイルが割り当てられているユーザーがコマンドを実行します。

   ```
   $ pfexec /usr/demo/tsol/auditfiles.ksh audit-html-file
   ```

例 - ラベルによる FTP サービスの保護

この例では、組織用の FTP サービスにラベルを付けます。FTP サーバーには、「Confidential - Internal」のラベルが付いている社内ファイルを含むラベル付きデータセットが格納されています。「Confidential - Internal」ファイルを認可されているユーザーは、`ftp` を使用してこれらのファイルを転送できます。認可されていないユーザーは、ファイルの取得や表示ができません。

1. FTP サーバー上で、管理者は FTP サービスを実行するラベルの 16 進数を判別し、`network/ftp` パッケージをインストールします。

   ```
   # atohexlabel "Confidential - Internal"
   0x0002-08-20

   # pkg install network/ftp
   ```

2. 管理者は「Confidential - Internal」認可上限の 16 進数を `svc:/network/ftp` サービスの開始メソッドに割り当て、サービスを再開します。

   ```
   # svccfg -s ftp
   svc:/network/ftp> set start/clearance = astring: 0x0002-08-20
   svc:/network/ftp> refresh
   svc:/network/ftp> exit

   # svcadm restart ftp
   ```

3. 管理者はマルチレベルのデータセットを作成してマウントします。

   ```
   # zfs -o multilevel=on rpool/ftp-files
   # zfs set mountpoint=/ftpsource rpool/ftp-files
   ```

4. 管理者は、「Confidential - Internal」のラベルが付いたデータセットを新しいサーバーに転送します。

```
rs-sys # zfs send -r rpool/research-intern | ssh ftp1 zfs receive -d rpool/ftp-files
hr-sys # zfs send -r rpool/hr-intern | ssh ftp1 zfs receive -d rpool/ftp-files
tr-sys # zfs send -r rpool/training-intern | ssh ftp1 zfs receive -d rpool/ftp-files
```

5. 配備の前に、管理者は「Confidential - Internal」の認可上限を持つユーザーがサーバーからファイルを取得できることをテストします。

プロジェクトの分離のためのサンドボックスの構成

サンドボックスは、承認されたユーザーが、システムのほかのプロセスから保護されたアプリケーションを実行できる分離された環境です。サンドボックスは、ファイル、プロセス、および共有メモリーに対してセキュリティー上の分離を提供し、ファイル割り当て、メモリー割り当て、および CPU に対してリソース上の分離を提供します。

サンドボックスは軽量の環境です。 では、これらはラベルを使用して実装されます。サンドボックスは、その認可上限や特権などのアプリケーションのプロセス属性を制限することによってアプリケーションを分離します。標準ユーザーは `sandbox` コマンドを使用して、一時的なサンドボックスを作成できます。管理者は `sandboxadm` コマンドを使用して、特定のセキュリティー属性を持つ永続的な名前付きサンドボックスを作成できます。承認されたユーザーは `sandbox` コマンドを使用して、名前付きサンドボックスを作成できます。

名前付きサンドボックスは、それらの認可上限に対応する、階層化された無関係なプロパティーを持っています。たとえば、親サンドボックスは最大 4096 個の子サンドボックスを持つことができ、互いに分離されています。最大 8 個の無関係な親サンドボックスをシステムに作成できます。さらに、ほかのすべてのサンドボックスよりも優位な最上位の親サンドボックスを作成できます。名前付きサンドボックスは、関連付けられたユーザー ID、プライマリグループ ID および補助グループ ID、プロジェクトを持つこともできます。管理者には、名前付きサンドボックスを作成および構成するための Sandbox Management 権利プロファイルを割り当てる必要があります。

次の階層例では、CDBall-SandboxAll が最上位の親サンドボックスです。これには CDB1-SandboxAll および CDB2-SandboxAll という 2 つの無関係な子サンドボックスがあります。省略記号は最上位の親サンドボックスがさらに多くの子を持つことができることを示しています。それぞれの子サンドボックスは名前付きの子を持つことができ、最大 4096 個まで増やせます。最上位の親サンドボックスから子サンドボックスに移るにつれて、プロセス特権および認可上限が減少します。子サンドボックスは、最上位サンドボックスに付与された特権よりも少ない特権しか必要としない操作に使用されます。それぞれのサンドボックスはサンドボックスに関連付けられた固有のプロジェクト名を持つことができ、リソースを割り当てることができます。

図 4 サンドボックス階層の例

sandbox コマンドを使用すると、一時的なサンドボックスまたは名前付きサンドボックスのいずれかに入ることができます。このコマンドに特権はありませんが、呼び出しプロセスのセキュリティー属性はターゲットサンドボックスのセキュリティー属性と一貫性がなければなりません。たとえば、現在のプロセスの認可上限は、ターゲットサンドボックスの認可上限より優位である必要があります。サンドボックスに入る前に、ターゲットサンドボックスの方が優位となるラベルを持つディレクトリに現在の作業ディレクトリを設定するようにします。詳細は、sandbox(1) のマニュアルページを参照してください。

名前付きの親サンドボックスに入るには、現在の有効な UID が、ターゲットサンドボックスに割り当てられた UID と一致する必要があります。名前付きの子サンドボックスに入る前に、呼び出し側はその親サンドボックスに入ったことがなければなりません。サンドボックスに入ったあと、プロセスプロジェクトはそのサンドボックスのプロセスプロジェクトに設定され、認可上限はサンドボックスの認可上限に設定されます。詳細は、sandbox_create(3SANDBOX) のマニュアルページを参照してください。

名前付きサンドボックスを作成するには sandboxadm コマンドを使用します。このコマンドは、特殊バージョンのエンコーディングファイル「Sandbox Labels v1.0」と一緒に機能します。このファイルのインストールおよび使用については、sandboxadm(8) のマニュアルページを参照してください。

You use the sandboxadm command to create named sandboxes. This command works with a special version of the encodings file, "Sandbox Labels v1.0". See 131 ページの「How to Configure Persistent Sandboxes」 and the sandboxadm(8) man page for information about installing and using this file.

sandboxadm コマンドを使用して名前付きサンドボックスを作成する場合は、コマンドで指定する関係によって、適切な認可上限をサンドボックスに自動的に割り当てることができます。親サンドボックスは、そのいずれかの子サンドボックスを作成する前に作成される必要があります。親サンドボックスが指定された場合、新しいサンドボックスは、親サンドボックスの方が優位となる認可上限が割り当てられ、最上位サンドボックスを除くほかのすべてのサンドボックスと無関係になります。それぞれの子サンドボックスは固有のユーザー名を持つ必要がありますが、複数の親サンドボックスを同じユーザーが所有できます。この場合、そのユーザーの認可上限は、そのユーザーが所有するすべてのサンドボックスより優位となる認可上限に自動的に設定されます。

すべての名前付きサンドボックスには、サンドボックスに入ったときに自動的に適用される、同じ名前を持つ対応するプロジェクトがあります。名前付きサンドボッ

クスに関連付けられたユーザーには、そのプロジェクトが自動的に割り当てられます。`projmod` コマンドによってリソース管理属性をサンドボックスに割り当てることができます。

子サンドボックス内のプロセスは、そのサンドボックス外のプロセスを監視できません。親サンドボックス内のプロセスは、自身のサンドボックスおよびその子のサンドボックス内のプロセスのみ監視できます。ファイルアクセスについても同様に、サンドボックスの認可上限によって優位となるファイルとディレクトリのみが認識されるように制限されています。`shmctl` システム呼び出しによってラベルが付けられた共有メモリーも、個々のサンドボックスに分離できます。

永続的なサンドボックスの準備

名前付きサンドボックスは、承認されたユーザーにデフォルトで割り当てられる特権レベルが不要な操作を実行するとき、これらのユーザーがログインできる永続的なサンドボックスを提供します。サンドボックス用ではない名前付き環境では、承認されたユーザーは `sandbox` コマンドを使用してサンドボックスを作成し、低い認可上限でアプリケーションまたはプロセスを実行できますが、これらのサンドボックスはセッションを閉じたあとに残りません。

永続的な名前付きサンドボックスの場合、特殊なラベルエンコーディングファイルを構成する必要があります。そのあと、管理者またはサンドボックスへのアクセスが承認されたユーザーが、サンドボックスのラベルでファイルシステム上のディレクトリを作成する必要があります。サンドボックスには 1 人のユーザーのみがアクセスでき、サンドボックスはそのユーザーのための分離された環境です。ラベル付きのディレクトリおよびファイルシステムの作成については、『Oracle Solaris 12 でのファイルのセキュリティー保護とファイル整合性の検証』 の 第 2 章,「データ損失保護のためのファイルのラベル付け」を参照してください。

▼ How to Configure Persistent Sandboxes

始める前に You must become an administrator who is authorized to install packages and to manage sandboxes. The `root` role is assigned these authorizations. Administrators who are assigned the Software Management rights profile can install packages. Administrators who are assigned Sandbox Management can administer sandboxes. For more information, see 110 ページの「割り当てられている管理権利の使用」.

1. **Install the `sandboxing` package.**

```
$ pkg install security/sandboxing
```

2. **Configure the sandbox encodings file.**

```
$ sandboxadm init
```

This command creates a label encodings file that conforms to a convention called Sandbox Labels v1.0. This version property is verified by the `labeld:clearance` service.

3. **Enable the `labeld:clearance` service.**

```
# svcadm enable clearance
```

4. **(オプション) Create directories for each sandbox at the label of the sandbox.**

5. **Assign the sandboxes to their authorized users.**

6. **Modify the encodings file, install it, and confirm the new file is in use.**

```
$ cd /etc/security/tsol
$ cp current-encodings orig.current-encodings
$ labelcfg current-encodings
$ more /etc/security/tsol/label_encodings.sandboxes
```

♦ ♦ ♦ 第 7 章

7 の権利の一覧表示

この章では、システムのすべての権利、特定ユーザーに割り当てられている権利、およびユーザー自身の権利を一覧表示する方法を説明します。

権利の概要については、19 ページの「ユーザー権管理」を参照してください。参照情報については、第9章「 権利リファレンス」を参照してください。

権利とその定義の一覧表示

このセクションのコマンドを使用すると、システムで定義されている権利を検索し、ユーザーのプロセスに対して有効な権利を一覧表示できます。

このセクションのコマンドの詳細については、次に示すマニュアルページを参照してください。

- `auths`(1)
- Unresolved link to " getent8"
- `ppriv`(1)
- `profiles`(1)
- `privileges`(7)
- `roles`(1)
- Unresolved link to " useradm8"

ユーザーに割り当てられたすべての権利の一覧表示

注記 - `useradm` はユーザーマネージャー GUI の CLI です。このコマンドにアクセスするには、`useradm` パッケージをロードします。

- `useradm list` *username* – *username* の直接権利割り当てを一覧表示します
- `useradm list -S [files|ldap]` *username* – 指定されたネームサービスで *username* の直接権利割り当てを一覧表示します。
- `useradm list -q` *qualifier username* – *username* の修飾属性を一覧表示します

例 56 LDAP でのユーザーの権利の一覧表示

このユーザーの権利は LDAP とファイルとで異なります。比較については、例 57「ローカルユーザーの権利の一覧表示」を参照してください。

```
$ useradm list -S ldap jdoe
inactive = 0
userID = 1234
groupName = staff
defaultShell = /bin/bash
username = jdoe
description = Jane Doe
groups  =
        docusers
        test_src
        web_publish
accountStatus = Unknown
homeDirectory = /home/jdoe
groupID = 123
```

例 57 ローカルユーザーの権利の一覧表示

このユーザーの権利はファイルと LDAP とで異なります。比較については、例 56「LDAP でのユーザーの権利の一覧表示」を参照してください。

```
$ useradm -S files jdoe
inactive = 0
Profiles  =
        Compliance Assessor
userID = 1234
groupName = staff
defaultShell = /bin/bash
username = jdoe
description = Jane Doe
accountStatus = Unknown
homeDirectory = /home/jdoe
groupID = 123
```

承認の一覧表示

- `auths` – 現在のユーザーの承認を一覧表示します
- `auths list` – 現在のユーザーの承認を、1 行あたり 1 つの承認で一覧表示します
- `auths list -u` *username* – *username* の承認を一覧表示します
- `auths list -x` – 認証が必要な現在のユーザーの承認を一覧表示します
- `auths list -xu` *username* – 認証が必要な *username* の承認を一覧表示します
- `auths info` – ネームサービス内のすべての承認名を一覧表示します
- `getent auth_attr` – ネームサービス内のすべての承認の完全な定義を一覧表示します

例 58 すべての承認の一覧表示

```
$ auths info
    solaris.account.activate
    solaris.account.setpolicy
    solaris.admin.edit
...
    solaris.zone.login
    solaris.zone.manage
```

例 59 承認データベースの内容の一覧表示

```
$ getent auth_attr | more
solaris.:::All Solaris Authorizations::
solaris.account.:::Account Management::
...
solaris.zone.login:::Zone Login::
solaris.zone.manage:::Zone Deployment::
```

例 60 ユーザーのデフォルト承認の一覧表示

次に示す承認は、デフォルトですべてのユーザーに割り当てられる権利プロファイルに含まれています。

```
$ auths
solaris.device.cdrw,solaris.device.mount.removable,solaris.mail.mailq
solaris.network.autoconf.read,solaris.admin.wusb.read
solaris.smf.manage.vbiosd,solaris.smf.value.vbiosd
```

権利プロファイルの一覧表示

- `profiles` – 現在のユーザーの権利プロファイルを一覧表示します

- `profiles -a` – すべての権利プロファイル名を一覧表示します
- `profiles -l` – 現在のユーザーの権利プロファイルの完全な定義を一覧表示します
- `profiles` *username* – *username* の権利プロファイルを一覧表示します
- `profiles -x` – 認証が必要な現在のユーザーの権利プロファイルを一覧表示します
- `profiles -x` *username* – 認証が必要な *username* の権利プロファイルを一覧表示します
- `profiles -p` *profile-name* `info` – 指定された権利プロファイルの内容をプリティプリントで出力します
- `getent prof_attr` – ネームサービス内のすべての権利プロファイルの完全な定義を一覧表示します

例 61 すべての権利プロファイル名の一覧表示

```
$ profiles -a
        Console User
        CUPS Administration
        Desktop Removable Media User
...
        VSCAN Management
        WUSB Management
```

例 62 権利プロファイルデータベースの内容の一覧表示

```
$ getent prof_attr | more
All:::Execute any command as the user or role:
Audit Configuration:::Configure Solaris Audit:auths=solaris.smf.value.audit;
...
Zone Management:::Zones Virtual Application Environment Administration:
Zone Security:::Zones Virtual Application Environment Security:auths=solaris.zone.*,
solaris.auth.delegate;...
```

例 63 ユーザーのデフォルト権利プロファイルの一覧表示

権利プロファイルを一覧表示します。デフォルトでは次の権利プロファイルがすべてのユーザーに割り当てられます。

```
$ profiles
Basic Solaris User
All
```

例 64 初期ユーザーの権利プロファイルの一覧表示

初期ユーザーにはさまざまな権利プロファイルが割り当てられます。

```
$ profiles Initial user
System Administrator
Audit Review
...
CPU Power Management
```

```
Basic Solaris User
All
```

初期ユーザーのプロファイルに割り当てられているすべてのセキュリティー属性を表示するには、-l オプションを使用します。

```
$ profiles -l Initial user | more
Initial user:
System Administrator
 profiles=Install Service Management,Audit Review,Extended Accounting
Flow Management,Extended Accounting Net Management,Extended Accounting Process
Management,Extended Accounting Task Management,Printer Management,Cron Management,
Device Management,File System Management,Log Management,Mail Management,
Maintenance and Repair,Media Catalog,Name Service Management,Network Management,
Project Management,RAD Management,Service Operator,Shadow Migration Monitor,
Software Installation,System Configuration,User Management,ZFS Storage Management
          /usr/sbin/gparted          uid=0
Install Service Management
 auths=solaris.autoinstall.service
 profiles=Install Manifest Management,Install Profile Management,
Install Client Management
...
```

例 65 割り当てられている権利プロファイルの内容の一覧表示

初期ユーザーが、Audit Review プロファイルにより付与された権利を一覧表示します。

```
$ profiles -l
Audit Review
  solaris.audit.read

  /usr/sbin/auditreduce  euid=0
  /usr/sbin/auditstat    privs=proc_audit
  /usr/sbin/praudit      privs=file_dac_read
```

例 66 権利プロファイルのコマンドのセキュリティー属性の一覧表示

profiles コマンドのこのバリアントは、ユーザー自身に割り当てられていない権利プロファイル内のコマンドのセキュリティー属性を表示する場合に役立ちます。

最初に、プロファイル内のコマンドを一覧表示します。

```
$ profiles -p "Audit Review" info
 name=Audit Review
 desc=Review Solaris Auditing logs
 cmd=/usr/sbin/auditreduce
 cmd=/usr/sbin/auditstat
 cmd=/usr/sbin/praudit
```

次に、プロファイル内の 1 つのコマンドのセキュリティー属性を一覧表示します。

```
$ profiles -p "Audit Review" "select cmd=/usr/sbin/praudit ; info; end;"
select: command is read-only
 id=/usr/sbin/praudit
 privs=file_dac_read
end: command is read-only
```

例 67 最近作成された権利プロファイルの内容の一覧表示

`less` オプションは、最近追加された権利プロファイルを最初に表示します。`profiles` コマンドのこのバリアントは、サイトで権利プロファイルを作成または変更する際に役立ちます。次の出力は、例38「レガシーアプリケーションへのセキュリティー属性の割り当て」で追加されたプロファイルの内容を示しています。通常のユーザーがこのコマンドを実行できます。

```
$ profiles -la | less
LegacyApp
         /opt/legacy-app/bin/legacy-cmd
                                    euid=0
OpenLDAP...
```

役割の一覧表示

- `roles` – 現在のユーザーの役割を一覧表示します
- `roles` *username* – *username* の役割を一覧表示します
- `logins -r` – 使用可能なすべての役割を一覧表示します

例 68 割り当てられている役割の一覧表示

デフォルトでは、`root` 役割が初期ユーザーに割り当てられます。`No roles` は、役割が割り当てられないことを示します。

```
$ roles
root
```

特権の一覧表示

- `man privileges` – 開発者により使用される特権の定義と特権の名前を一覧表示します
- `ppriv -vl` – 開発者により使用される特権の定義と特権の名前を一覧表示します
- `ppriv -vl basic` – 特権の基本セット内の特権の名前と定義を一覧表示します
- `ppriv $$` – 現在のシェル (`$$`) 内の特権を一覧表示します
- `getent exec_attr` – 権利プロファイル名別にセキュリティー属性 (`setuid` または特権) を持つすべてのコマンドを一覧表示します

```
$ getent exec_attr | more
All:solaris:cmd:::*:
Audit Configuration:solaris:cmd:::/usr/sbin/auditconfig:privs=sys_audit
```

```
...
Zone Security:solaris:cmd:::/usr/sbin/txzonemgr:uid=0
Zone Security:solaris:cmd:::/usr/sbin/zonecfg:uid=0 ...
```

例 69 すべての特権とその定義の一覧表示

privileges(7) のマニュアルページで説明されている特権フォーマットが開発者によって使用されます。

```
$ man privileges
Standards, Environments, and Macros                 privileges(7)

NAME
     privileges - process privilege model
...
     The defined privileges are:

     PRIV_CONTRACT_EVENT

         Allow a process to request reliable delivery  of  events
         to an event endpoint.

         Allow a process to include events in the critical  event
         set  term  of  a  template  which  could be generated in
         volume by the user.
...
```

例 70 特権割り当てで使用される特権の一覧表示

ppriv コマンドは、すべての特権の名前を一覧表示します。定義を表示するには -v オプションを使用します。

この特権フォーマットは、useradd, roleadd、usermod、および rolemod コマンドを使用してユーザーと役割に特権を割り当てる場合、および profiles コマンドを使用して権利プロファイルに特権を割り当てる場合に使用されます。

```
$ ppriv -lv | more
contract_event
 Allows a process to request critical events without limitation.
 Allows a process to request reliable delivery of all events on
 any event queue.
...
win_upgrade_sl
        Allows a process to set the sensitivity label of a window
        resource to a sensitivity label that dominates the existing
        sensitivity label.
        This privilege is interpreted only if the system is configured
        with Trusted Extensions.
```

例 71 現在のシェル内の特権の一覧表示

デフォルトでは、どのユーザーにも基本特権セットが割り当てられます。デフォルトの制限セットはすべての特権です。

出力の最初の文字は、次の特権セットを指しています。

E　　有効特権セット

I　　継承可能な特権セット

P　　許可された特権セット

L　　制限特権セット

```
$ ppriv $$
1200:   -bash
flags = <none>
        E: basic
        I: basic
        P: basic
        L: all
$ ppriv -v $$
1200:   -bash
flags = <none>
 E: file_link_any,file_read,file_write,net_access,proc_exec,proc_fork,
           proc_info,proc_session,sys_ib_info
 I: file_link_any,file_read,...,sys_ib_info
 P: file_link_any,file_read,...,sys_ib_info
 L: contract_event,contract_identity,...,sys_time
```

2 つのドル記号 ($$) により、親シェルのプロセス番号がコマンドに渡されます。この一覧表示には、割り当てられている権利プロファイル内のコマンドに制限されている特権は含まれません。

例 72　　基本特権とその定義の一覧表示

```
$ ppriv -vl basic
file_link_any
 Allows a process to create hardlinks to files owned by a uid
 different from the process' effective uid.
file_read
 Allows a process to read objects in the filesystem.
file_write
 Allows a process to modify objects in the filesystem.
net_access
 Allows a process to open a TCP, UDP, SDP or SCTP network endpoint.
proc_exec
 Allows a process to call execve().
proc_fork
 Allows a process to call fork1()/forkall()/vfork()
proc_info
 Allows a process to examine the status of processes other
 than those it can send signals to.  Processes which cannot
 be examined cannot be seen in /proc and appear not to exist.
proc_session
 Allows a process to send signals or trace processes outside its
 session.
sys_ib_info
 Allows a process to perform read InfiniBand MAD (Management Datagram)
 operations.
```

例 73 ユーザーの権利プロファイル内のセキュリティー属性を持つコマンドの一覧表示

Basic Solaris User プロファイルには、ユーザーが CD-ROM への読み取りと書き込みを行えるコマンドが含まれています。

```
$ profiles -l
  Basic Solaris User
...
   /usr/bin/cdrecord.bin   privs=file_dac_read,sys_devices,
     proc_lock_memory,proc_priocntl,net_privaddr
   /usr/bin/readcd.bin     privs=file_dac_read,sys_devices,net_privaddr
   /usr/bin/cdda2wav.bin   privs=file_dac_read,sys_devices,
     proc_priocntl,net_privaddr
  All
   *
```

修飾属性の一覧表示

- `man user_attr` – セキュリティー属性の修飾子を定義します
- `getent` – コマンドが実行されるシステム上の特定のユーザーまたは役割の修飾セキュリティー属性を一覧表示します
- `ldapaddent` – 特定のユーザーまたは役割の修飾セキュリティー属性をすべて一覧表示します

例 74 このシステム上のユーザーの修飾属性の一覧表示

```
system1$ getent user_attr | grep jdoe:
jdoe:system1:::lock_after_retries=no;profiles=System Administrator
```

例 75 LDAP 内のユーザーの全修飾属性の一覧表示

```
system1$ ldapaddent -d user_attr | grep ^jdoe:
jdoe:system1:::lock_after_retries=no;profiles=System Administrator
jdoe:sysopgroup:::lock_after_retries=no;profiles=System Operator
```

♦ ♦ ♦ 第 8 章

8 での権利のトラブルシューティング

この章では、 で管理権利を管理および使用する際のトラブルシューティングについて提案します。

- 143 ページの「RBAC と特権のトラブルシューティング」
- 151 ページの「パスワードのトラブルシューティング」

権利の使用については、次の情報を参照してください。

- 第3章「 での権利の割り当て」
- 50 ページの「役割を割り当てることができるユーザー」
- 19 ページの「ユーザー権管理」
- 28 ページの「プロセス権管理」

パスワードについては、『Oracle Solaris 11.4 でのシステムおよび接続されたデバイスのセキュリティー保護』 の 「特殊なシステムアカウント」、および `passwd(1)` と `user_attr(5)` のマニュアルページを参照してください。

RBAC と特権のトラブルシューティング

このセクションのタスクと例では、権利の割り当てに関する問題の解決策を提案します。バックグラウンド情報については、39 ページの「権利の検証」を参照してください。

権利を割り当てるには、コマンド行インタフェースを使用します。次のコマンドは、権利データベースを変更します。

- `passwd`
- `useradd`、`usermod`、および `userdel`
- `roleadd`、`rolemod`、および `roledel`
- `profiles`
- `auths`

注意 - エディタを使用して権利データベースを変更しないでください。エディタは構文の有効性をチェックしたり、カーネルプロセスを更新したりできません。

▼ 権利割り当てをトラブルシューティングする方法

権利が評価されず正しく適用されない原因には、さまざまな要素の影響があります。この手順は、割り当てられている権利がユーザー、役割、プロセスに対して使用できない原因のデバッグを支援します。いくつかのステップは 40 ページの「割り当てられた権利の検索順序」に基づいています。

始める前に root 役割になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **ネームサービスを検証して再起動します。**

   a. **ユーザーまたは役割のセキュリティー割り当てが、システムで有効になっているネームサービス内にあることを確認します。**

   ```
   # svccfg -s name-service/switch

   svc:/system/name-service/switch>
   listprop config

   config                      application
   config/value_authorization  astring  solaris.smf.value.name-service.switch
   config/default              astring  files ldap
   config/host                 astring  "files dns mdns ldap"
   config/netgroup             astring  ldap
   config/printer              astring  "user files"
   ```

   この出力では、明示的に示されていないサービスはすべて、デフォルトの値 files ldap を継承します。そのため、passwd とその関連属性データベース user_attr、auth_attr、および prof_attr は、まずファイル内で検索され、次に LDAP 内で検索されます。

   b. **ネームサービスキャッシュ svc:/system/name-service/cache を再起動します。**

   nscd デーモンには長い有効期間を設定することができます。デーモンを再起動して、現在のデータでネームサービスを更新します。

   ```
   # svcadm restart name-service/cache
   ```

2. **userattr -v コマンドを実行することによってユーザーに権利が割り当てられているかどうかを判定します。**

   すべての属性についてコマンドを 1 回実行します。たとえば次のコマンドは、割り当てられている権利と、その割り当てがユーザー jdoe に対して行われた場所を示します。出力がない場合、jdoe がデフォルトを使用していることを示します。

   ```
   $ userattr -v access_times jdoe
   $ userattr -v access_tz jdoe
   $ userattr -v annotation jdoe
   $ userattr -v auth_profiles jdoe
   $ userattr -v defaultpriv jdoe
   $ userattr -v limitpriv jdoe
   ```

```
$ userattr -v idlecmd jdoe
$ userattr -v idletime jdoe
$ userattr -v lock_after_retries jdoe
$ userattr -v pam_policy jdoe
$ userattr -v unlock_after jdoe

$ userattr -v auths jdoe        Output indicates authorizations from rights profiles
Basic Solaris User :solaris.mail.mailq,solaris.network.autoconf.read,
solaris.admin.wusb.read
Console User :solaris.system.shutdown,solaris.device.cdrw,
solaris.device.mount.removable,solaris.smf.manage.vbiosd,solaris.smf.value.vbiosd
$ userattr -v audit_flags jdoe
user_attr: fw:no        Output indicates jdoe is individually assigned audit flags
$ userattr -v profiles jdoe
user_attr: Audit Review,Stop        Output indicates two assigned rights profiles
$ userattr roles jdoe
user_attr : cryptomgt,infosec        Output indicates two assigned roles
```

この出力は、jdoe に監査フラグ、2 つの権利プロファイル、および 2 つの役割が直接割り当てられていることを示しています。割り当てられている承認は、account-policy SMF ステンシルまたは /etc ディレクトリ内のファイルのいずれかに設定されているデフォルトの権利プロファイルからのものです。システムのデフォルトの権利プロファイルのソースを判別するには、82 ページの「新機能 – account-policy サービスの有効化」を参照してください。

- jdoe には監査フラグが直接割り当てられるため、権利プロファイルの監査フラグ値は使用されません。
- 権利プロファイルは、Audit Review 権利プロファイルから始まり、次に Stop プロファイルという順序で評価されます。
- そのほかのすべての権利は、役割 cryptomgt と infosec で jdoe に割り当てられています。これらの権利を表示するには、jdoe が各役割を引き受けてから、権利を一覧表示する必要があります。

ヒント - useradm パッケージがインストールされている場合、useradm list jdoe コマンドを実行して直接割り当てられている権利を表示できます。134 ページの「ユーザーに割り当てられたすべての権利の一覧表示」を参照してください。

権利がユーザーに直接割り当てられていない場合は、次の確認作業に進みます。

3. **割り当てられている承認のスペルが正しいことを確認します。**
   承認はユーザーに対して累積されるため、承認の割り当てのソースは重要ではありません。ただし、スペルが正しくない承認は暗黙のうちに失敗します。

4. **作成した権利プロファイルについて、そのプロファイル内のコマンドに適切なセキュリティー属性を割り当てたことを確認します。**
   たとえば、成功するには euid=0 ではなく uid=0 が必要なコマンドもあります。コマンドのマニュアルページを参照し、コマンドとそのオプションに承認が必要であるかどうかをか確認してください。

5. **ユーザーの権利プロファイルの権利を確認します。**

a. **認証権利プロファイルのリストで権利を順に確認します。**

リスト内のもっとも古い権利プロファイルに含まれる属性の値は、カーネル内の値です。この値が正しくない場合は、その権利プロファイル内の値を変更するか、またはプロファイルを正しい順序で再割り当てします。148 ページの「割り当てられている権利を並べ替える方法」を参照してください。

特権コマンドの場合、`defaultpriv` または `limitpriv` キーワードから特権が削除されていないことを確認します。

b. **標準権利プロファイルのリストで権利を順に確認します。**

認証権利プロファイルで実行したものと同じ検査を行います。

c. **検索する権利がリストにない場合は、ユーザーに割り当てられている役割を調べます。**

権利が 1 つの役割に割り当てられている場合、ユーザーはその役割を引き受けて権利を取得する必要があります。

6. **失敗したコマンドが成功するには承認が必要であるかどうかを確認します。**

a. **既存の権利プロファイルに必要な承認が含まれているかどうかを確認します。**

プロファイルが存在する場合は、そのプロファイルを使用します。それを、認証権利プロファイルまたは通常の権利プロファイルとしてユーザーに割り当てます。この承認が成功する必要があるコマンドを含むそのほかの権利プロファイルよりも前に、このプロファイルを配置します。

b. **コマンドのオプションに承認が必要であるかどうかを確認します。**

特権が必要なコマンドにその特権を割り当て、必要な承認を追加し、そのコマンドと承認を権利プロファイル内に配置してから、そのプロファイルをユーザーに割り当てます。

7. **あるユーザーに対してコマンドが引き続き失敗する場合は、そのユーザーがプロファイルシェルでコマンドを実行していることを確認します。**

管理コマンドは、プロファイルシェルで実行する必要があります。例76「プロファイルシェルを使用しているかどうかの判断」は、プロファイルシェルのテスト方法を示しています。

ユーザーエラーが発生する可能性を削減するには、次を試してみることができます。

- プロファイルシェルをユーザーのログインシェルとして割り当てます。
- ユーザーに対し、すべての特権付きコマンドより前に`pfexec` コマンドを配置するように指示します。
- 管理コマンドをプロファイルシェルで実行するようユーザーに注意を促します。

- サイトで役割が使用されている場合は、ユーザーに対し、管理コマンドを実行する前にその役割を引き受けるように注意を促します。ユーザーではなく役割としてコマンドを正常に実行する例については、例78「役割での特権付きコマンドの実行」を参照してください。

8. **役割でコマンドが失敗する場合は、その役割を引き受け、ユーザーの権利を確認する場合と同じ手順を実行します。**

例 76 プロファイルシェルを使用しているかどうかの判断

特権付きコマンドが機能しない場合、ユーザーは PRIV_PFEXEC フラグをテストしてからそのコマンドを実行します。エラーメッセージに、この問題が特権の問題であると示されない場合があります。

```
$ praudit 20120814200247.20120912213421.example-system
praudit: Cannot associate stdin with 20120814200247.20120912213421.example-system:
Permission denied

$ ppriv $$
107219: bash
flags = <none>
...

$ pfbash
$ ppriv $$
1072232: bash
flags = PRIV_PFEXEC
...

$ praudit 20120814200247.20120912213421.example-system
     /** Command succeeds **/
```

例 77 役割の特権付きコマンドの判断

この例では、ユーザーは割り当てられた役割を引き受け、いずれかの権利プロファイルに含まれている権利を一覧表示します。コマンドを強調するため、権利は切り捨てられています。

```
$ roles
devadmin

$ su - devadmin
Password: xxxxxxxx

$ profiles -l
Device Security
        ...
 profiles=Service Configuration
          /usr/sbin/add_drv          uid=0
          /usr/sbin/devfsadm         uid=0
                                     privs=sys_devices,sys_config,
                                           sys_resource,file_owner,
                                           file_chown,file_chown_self,
                                           file_dac_read
```

```
          /usr/sbin/eeprom           uid=0
          /usr/bin/kbd
          /usr/sbin/list_devices     euid=0
          /usr/sbin/rem_drv          uid=0
          /usr/sbin/strace           euid=0
          /usr/sbin/update_drv       uid=0
          /usr/sbin/add_allocatable  euid=0
          /usr/sbin/remove_allocatable   euid=0
Service Configuration
          /usr/sbin/svcadm
          /usr/sbin/svccfg
```

例 78 役割での特権付きコマンドの実行

次の例で、admin 役割は、 useful.script ファイルに関する権利を変更することができます。

```
$ whoami
jdoe
$ ls -l useful.script
-rwxr-xr-- 1 elsee eng 262 Apr 2 10:52 useful.script

$ chgrp admin useful.script
chgrp: useful.script: Not owner

$ su - admin
Password: xxxxxxxx

$ chgrp admin useful.script
$ chown admin useful.script
$ ls -l useful.script
-rwxr-xr-- 1 admin admin 262 Apr 2 10:53 useful.script
```

▼ 割り当てられている権利を並べ替える方法

特権付きコマンドではなく特権なしのコマンドがユーザーに対して有効である場合は、ユーザーの権利プロファイルの割り当てを並べ替える必要があります。詳細は、40 ページの「割り当てられた権利の検索順序」を参照してください。

始める前に User Security 権利プロファイルが割り当てられている管理者になる必要があります。詳細は、110 ページの「割り当てられている管理権利の使用」を参照してください。

1. **現在ユーザーまたは役割に割り当てられている権利プロファイルのリストを表示します。**

   リストが順番に表示されます。

   ```
   $ profiles username | rolename
   ```

2. **権利プロファイルを正しい順序で割り当てます。**

   ```
   # usermod | rolemod -K profiles="list-of-profiles"
   ```

例 79　権利プロファイルの特定の順序での割り当て

この例では、管理者は、特権付きコマンドを含む権利プロファイルが、役割 devadmin のすべての権利プロファイルのあとに一覧表示されるように決定します。

```
# profiles devadmin

    Basic Solaris User
    All
    Device Management
```

そのため、devadmin 役割は、その役割に割り当てられている特権でデバイス管理コマンドを実行できません。

管理者は、devadmin に権利プロファイルを再割り当てします。新しい割り当て順序では、割り当てられている特権を使用してデバイス管理コマンドが実行されます。

```
# rolemod -K profiles="Device Management,Basic Solaris User,All"

# profiles devadmin

    Device Management
    Basic Solaris User
    All
```

▼ プログラムが必要とする特権を判断する方法

このデバッグ手順は、コマンドまたはプロセスが失敗した場合に使用します。最初の特権失敗を検出して修正したあとで、ppriv -eD *command* コマンドを再度実行し、追加の特権要件を見つける必要がある場合があります。

1. **失敗したコマンドを ppriv デバイスコマンドの引数として入力します。**

```
$ ppriv -eD touch /etc/acct/yearly

touch[5245]: missing privilege "file_dac_write"
     (euid = 130, syscall = 224) needed at zfs_zaccess+0x258
touch: cannot create /etc/acct/yearly: Permission denied
```

2. **デバッグ出力の syscall 番号を使用して、どのシステム呼び出しが失敗したかを特定します。**

/etc/name_to_sysnum ファイルで syscall 番号の名前を見つけます。

```
$ grep 224 /etc/name_to_sysnum

creat64                 224
```

この例ではcreat64() 呼び出しが失敗しています。正常に完了するには、/etc/acct/yearly ディレクトリ内にファイルを作成できる権利がプロセスに割り当てられている必要があります。

例 80 特権の使用を検査するための truss コマンドの使用

truss コマンドは、通常のシェルで特権の使用をデバッグすることができます。たとえば、次のコマンドは、失敗した touch プロセスをデバッグします。

```
$ truss -t creat touch /etc/acct/yearly

creat64("/etc/acct/yearly", 0666)
                      Err#13 EACCES [file_dac_write
]
touch: /etc/acct/yearly cannot create
```

拡張された /proc インタフェースで、truss 出力のエラーコードの後に欠如している file_dac_write 特権がレポートされます。

例 81 プロファイルシェルで特権の使用を検査するための ppriv コマンドの使用

次の例で、jdoe ユーザーは、役割 objadminを引き受けることができます。objadmin 役割には、Object Access Management 権利プロファイルが含まれます。この権利プロファイルによって、objadmin 役割は objadmin が所有しないファイルに関するアクセス権を変更することができます。

次の例で、jdoe は、useful.script ファイルに関するアクセス権を変更することができません。

```
jdoe$ ls -l useful.script

-rw-r--r--  1 aloe  staff  2303 Apr 10 10:10 useful.script
jdoe$
chown objadmin useful.script

chown: useful.script: Not owner
jdoe$
ppriv -eD chown objadmin useful.script

chown[11444]: missing privilege "file_chown"
            (euid = 130, syscall = 16) needed at zfs_zaccess+0x258
chown: useful.script: Not owner
```

jdoe が objadmin 役割を引き受けると、ファイルに関するアクセス権が変更されます。

```
jdoe$ su - objadmin
Password: xxxxxxxx

$ ls -l useful.script
-rw-r--r--  1 aloe  staff  2303 Apr 10 10:10 useful.script

$ chown objadmin useful.script
$ ls -l useful.script
-rw-r--r--  1 objadmin  staff  2303 Apr 10 10:10 useful.script
$ chgrp admin useful.script

$ ls -l objadmin.script
-rw-r--r--  1 objadmin  admin  2303 Apr 10 10:11 useful.script
```

例 82 root ユーザーが所有するファイルの変更

この例は、特権エスカレーションに対する保護を示しています。説明については、38 ページの「特権エスカレーションとカーネル特権」を参照してください。ファイルは、root ユーザーが所有します。権限の弱い objadmin 役割ではファイルの所有を変更するためにはすべての特権が必要なので、処理は失敗します。

```
jdoe$ su - objadmin
Password: xxxxxxxx

$ cd /etc; ls -l system
-rw-r--r--  1 root  sys   1883 Oct 10 10:20 system

$ chown objadmin system
chown: system: Not owner
$ ppriv -eD chown objadmin system
chown[11481]: missing privilege "ALL"
     (euid = 101, syscall = 16) needed at zfs_zaccess+0x258
chown: system: Not owner
```

パスワードのトラブルシューティング

次の例は、パスワードの問題をデバッグする方法を示しています。背景情報については、passwd(1) および user_attr(5) のマニュアルページを参照してください。

例 83 openldap システムアカウントを使用した cron ジョブの実行

この例では、管理者は から提供されているシステムアカウントのパスワード文字列を *LK* から NP に変更しています。このアカウントは、openldap ユーザーのみにロックダウンされている UNIX ドメインソケットを介して、外部データソースを使用してディレクトリ内のデータを定期的に更新する cron ジョブを実行します。

最初は openldap アカウントはロックされています。パスワードがロックされた時間は 12111 です。

```
# grep openldap /etc/shadow
openldap:*LK*:12111::::::
# passwd -N openldap
WARNING: changing account in reserved uid range: openldap.
passwd: password information changed for openldap
```

-N は、パスワードエントリを、UNIX 認証でのログインに使用できない値にします。

```
# passwd -u openldap
WARNING: changing account in reserved uid range: openldap.
passwd: password information changed for openldap
```

-u オプションはアカウントをロック解除します。アカウントは非 UNIX の認証アカウントのままです。

```
# grep openldap /etc/shadow
```

```
openldap:NP:13222::::::
```

最後のエントリでは、openldap という非 UNIX 認証アカウントのパスワードはありません。アカウントはロックダウンされた UNIX ドメインソケットによって保護されます。

例 84 ユーザーのパスワードを必要とする役割の作成

この例では、sudo ユーザーが構成される方法に似た、ユーザーのパスワードを使用して認証される管理アカウントを構成する方法を示します。root 役割で、ユーザーおよび役割を作成し、役割をユーザーパスワードが必要な役割にして、役割をユーザーに割り当てます。

```
# useradd [ ... ] jdoe
# roleadd [ ... ] administrator
# rolemod -K roleauth=user administrator
# usermod -K type=role administrator jdoe
```

例 85 アカウントのパスワード要件のオーバーライド

この例で、root 役割はパスワードを数回変更し、パスワード制約をオーバーライドします。

```
# passwd -n 14 -x 25 user1
```

/etc/shadow エントリは次のようになります。

```
user1:$5$Cuz1WCgx$4CFN...:last-changed-date:14:25::::
```

root 役割の管理者はパスワードを同じ日に数回変更できます。

```
# passwd user1
Enter user1's password: password
# grep user1 /etc/shadow
user1:$5$Cuz1WCgx$4CFN...:11333:14:25::::
# passwd user1
Enter user1's password: password
user1:$5$Cuz1WCgx$4CFN...:11444:14:25::::
```

root 役割は solaris.passwd.assign 承認を持つため、root は、次のパスワード変更までの最低日数が 14 日 (-n 14)、最大日数が 25 日 (-x 25) というパスワードの制約をオーバーライドできます。solaris.passwd.assign 承認を持たない user1 は、割り当てられたパスワードを最低 2 週間は変更できません。次に示すように、ユーザーは別のユーザーのパスワードを変更することもできません。

```
$ passwd user2
Enter user1's password: password
Permission denied: Missing authorization: solaris.passwd.assign
```

♦ ♦ ♦ 第 9 章

9 権利リファレンス

この章は、 での管理権利の使用に関する参照情報を提供します。

特権を含む権利の使用については、第3章「 での権利の割り当て」を参照してください。概要については、19 ページの「ユーザー権管理」および 28 ページの「プロセス権管理」を参照してください。

account-policy SMF ステンシル

account-policy サービスおよび特定のセキュリティー属性を有効化することが、システムのセキュリティー属性を管理する上で優先される方法です。account-policy サービスが有効な場合、この章で説明されているデータベースは現在のセキュリティーポリシーを反映しないことがあります。

はブート時に account-policy SMF ステンシルをロードしますが、有効化しません。account-policy サービスを有効化し、サイトのセキュリティーポリシーが求めるセキュリティー属性をデフォルトのものとは異なる属性に有効化すると、すべてのシステムセキュリティー属性は、svcprop コマンドで値を表示できる SMF プロパティーとなります。account-policy サービスで有効化されるセキュリティー属性は変更できます。

注記 - `/etc/policy.conf` や `/etc/default/login` などのセキュリティー属性を一覧表示するファイルは、既存のセキュリティーポリシーを反映していないことがあります。また、これらのファイルの内容を変更してもセキュリティーポリシーに影響はありません。

次のコマンドは、管理者が `account-policy` サービスを有効化したかどうか、および特定のプロパティーを変更できるかどうかを示しています。

```
$ svcs account-policy
$ svcprop -p config/ -s account-policy
```

`online` はサービスが有効であることを示します。

セキュリティー属性の値を表示するには、次の構文を使用します。

```
$ svcprop -p property account-policy:default
```

SMF のセキュリティー属性、および `/etc` ファイル内のそれらに対応する名前のリストについては、166 ページの「ファイルおよび対応する SMF プロパティー内のセキュリティー属性」 のマニュアルページを参照してください。

権利プロファイルのリファレンス

このセクションでは、一般的な権利プロファイルについて説明します。権利プロファイルとは、承認やその他のセキュリティー属性、セキュリティー属性を持つコマンド、および補助権利プロファイルを使いやすく集めたものです。 には、多くの権利プロファイルが用意されています。それらがニーズを満たさない場合は、既存のものを変更して、新しいものを作成できます。

権利プロファイルは、もっとも権限のあるものからもっとも権限のないものへと順番に割り当てられる必要があります。詳細は、40 ページの「割り当てられた権利の検索順序」を参照してください。

次に示す権利プロファイルの内容を表示するには、155 ページの「権利プロファイルの内容の表示」を参照してください。

- **System Administrator 権利プロファイル** – セキュリティーで保護された状態で接続していないほとんどのタスクにアクセスできるようにします。このプロファイルには、権限のある役割を作成するためにいくつかのほかのプロファイルが含まれます。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。
- **Operator 権利プロファイル** – ファイルおよびオフラインメディアを管理するための限られた権利を提供します。このプロファイルには、単純な役割を作成するための補助権利プロファイルが含まれます。

- **Printer Management 権利プロファイル –** 出力を処理するための限られた数のコマンドと承認を提供します。このプロファイルは、単一の管理領域を対象とする複数のプロファイルのうちの 1 つです。
- **Basic Solaris User 権利プロファイル –** ユーザーがセキュリティーポリシーの範囲内でシステムを使用できるようにします。このプロファイルは、デフォルトユーザーの権利プロファイルです。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ー要件を考慮する必要があります。高いセキュリティーを必要とするサイトでは、このプロファイルを削除するか、または Stop 権利プロファイルを割り当てることをお勧めします。Basic Solaris User 権利プロファイルの実装については、例73「ユーザーの権利プロファイル内のセキュリティー属性を持つコマンドの一覧表示」を参照してください。
- **Console User 権利プロファイル –** ワークステーション所有者の場合、コンピュータの前に着席しているユーザーが、承認、コマンド、およびアクションにアクセスできるようにします。
- **All 権利プロファイル –** 役割に対して、セキュリティー属性を持たないコマンドにアクセスできるようにします。このプロファイルは、限られた権限を持つユーザーに適している場合があります。
- **Stop 権利プロファイル –** それ以降のプロファイルの評価を停止する特殊な権利プロファイルです。また、このプロファイルは `AUTHS_GRANTED`、`PROFS_GRANTED`、および `CONSOLE_USER` セキュリティー属性の評価を中止します。Stop プロファイルを使用すると、役割とユーザーに制限付きプロファイルシェルを提供できます。

注記 - Stop プロファイルは、特権の割り当てに間接的な影響を及ぼします。Stop プロファイルのあとにリストされた権利プロファイルは評価されません。したがって、そのようなプロファイル内に特権が指定されているコマンドは有効ではありません。例31「明示的に割り当てられた権利への管理者の制限」を参照してください。

権利プロファイルの内容の表示

権利プロファイルの内容を表示できる 3 つのビューがあります。

- `getent` コマンドを使用すると、システム上のすべての権利プロファイルの内容を表示できます。出力例については、第7章「 の権利の一覧表示」を参照してください。
- `profiles -p "`*Profile Name*`" info` コマンドを使用すると、特定の権利プロファイルの内容を表示できます。
- `profiles -l` *account-name* コマンドを使用すると、特定のユーザーまたは役割に割り当てられている権利プロファイルの内容を表示できます。

詳細は、第7章「 の権利の一覧表示」と、Unresolved link to " getent8" および profiles(1) のマニュアルページを参照してください。

承認のリファレンス

承認は、役割またはユーザーに付与できる個別の権利です。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。

承認はユーザーレベルであり、したがって拡張可能です。承認を必要とするプログラムを作成し、承認をシステムに追加し、これらの承認の権利プロファイルを作成して、その権利プロファイルを、プログラムの使用が許可されているユーザーまたは役割に割り当てることができます。

承認の命名規則

承認には、内部で使用される名前があります。たとえば `solaris.system.date` は承認の名前です。また、承認にはグラフィカルユーザーインタフェース (GUI) に表示される短い説明もあります。たとえば、`Set Date & Time` は `solaris.system.date` 承認の説明です。

慣例上、承認名はインターネット名とは逆の順序になり、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。承認名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。この規則の例外として、インターネット名の代わりに接頭辞 `solaris` を使用する、 からの承認があります。システム管理者は、承認を階層方式で適用することができます。ワイルドカード (*) は、ドットの右側の任意の文字列を表すことができます。

承認の使用方法の例として、Network Link Security 権利プロファイルには `solaris.network.link.security` 承認のみが含まれるのに対して、Network Security 権利プロファイルには補助プロファイルとしての Network Link Security プロファイルに加え、`solaris.network.*` および `solaris.smf.manage.ssh` 承認が含まれます。

承認での委託権限

接尾辞が `delegate` 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で始まるものを、ほかのユーザーに委任できます。

solaris auth.delegate 承認では、ユーザーまたは役割が、これらの委任ユーザーまたは役割が割り当てられている任意の承認をほかのユーザーに委任できます。たとえば、solaris auth.delegate と solaris.network.wifi.wep の承認を持つ役割は、solaris.network.wifi.wep 承認をほかのユーザーまたは役割に委任できます。

権利データベース

次のデータベースには、 の権利のデータが格納されます。

- **拡張ユーザー属性のデータベース** (user_attr) – ユーザーと役割を、ほかのキーワードの中でも承認、特権、権利プロファイルに関連付けます。
- **権利プロファイル属性のデータベース** (prof_attr) – 権利プロファイルを定義し、プロファイルの割り当てられた承認、特権、およびキーワードを一覧表示します
- **承認属性のデータベース** (auth_attr) – 承認およびその属性を定義します
- **実行属性のデータベース** (exec_attr) – 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ー属性を持つコマンドを識別します

policy.conf データベースには、すべてのユーザーに適用される承認、特権、および権利プロファイルが含まれます。詳細については、161 ページの「policy.conf ファイル」を参照してください。82 ページの「新機能 – account-policy サービスの有効化」も参照してください。

権利データベースおよびネームサービス

権利データベースのネームサービススコープは、ネームサービススイッチ svc:/system/name-service/switch の SMF サービスで定義されます。このサービス内での権利データベースのプロパティーは、auth_attr、password、および prof_attr です。password プロパティーは、passwd および user_attr データベースに対するネームサービスの優先順位を設定します。prof_attr プロパティーは、prof_attr および exec_attr データベースに対するネームサービスの優先順位を設定します。

次の出力では、auth_attr、password、および prof_attr エントリは一覧表示されていません。したがって、権利データベースは files ネームサービスを使用しています。

```
# svccfg -s name-service/switch listprop config
config                          application
config/value_authorization      astring       solaris.smf.value.name-service.switch
config/default                  astring       files
config/host                     astring       "files ldap dns"
config/printer                  astring       "user files ldap"
```

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。*attr* フィールドにはセキュリティー属性が含まれ、*qualifier* フィールドには、セキュリティー属性の効果を 1 つのシステムまたはシステムのグループに限定または制限する属性が含まれています。

attr フィールドのセキュリティー属性
は、roleadd、rolemod、useradd、usermod、および profiles コマンドを使用して設定できます。ローカルに設定するか、または LDAP ネーミングスコープ内で設定できます。

- ユーザーの場合、roles キーワードで 1 つ以上の定義された役割を割り当てます。
- 役割の場合、roleauth キーワードの値を user にすると、その役割は役割のパスワードではなくユーザーのパスワードを使用して認証できるようになります。デフォルトでは、この値は role です。
- ユーザーまたは役割の場合、次の属性を設定できます。
  - access_times キーワード – 指定されているアプリケーションとサービスにアクセスできる日と時間を指定します。詳細は、getaccess_times(3C) のマニュアルページを参照してください。
  - access_tz キーワード – access_times エントリの時間を解釈するときに使用する時間帯を指定します。詳細は、Unresolved link to " pam_unix_account7" のマニュアルページを参照してください。
  - annotation キーワード – 監査レコードのためにユーザーのログインを注釈として示す目的でユーザーにプロンプトを表示するかどうかを指定します。デフォルトでは、ユーザーにプロンプトが表示されません。詳細は、『Managing Auditing in Oracle Solaris 11.4』 の 「New Feature – Annotating Reason for Access in the Audit Record」 を参照してください。
  - audit_flags キーワード – 監査マスクを変更します。詳細は、audit_flags(7) のマニュアルページを参照してください。
  - auths キーワード – 承認を割り当てます。詳細は、auths(1) のマニュアルページを参照してください。
  - auth_profiles キーワード – 認証権利プロファイルを割り当てます。詳細は、profiles(1) のマニュアルページを参照してください。
  - defaultpriv キーワード – デフォルトの特権の基本セットに特権を追加するか、またはそこから特権を削除します。
  - limitpriv キーワード – デフォルトの特権の制限セットに特権を追加するか、または特権を削除します。

    defaultpriv および limitpriv 特権は、ユーザーの初期プロセスに割り当てられているため常に有効です。詳細は、privileges(7) のマニュアルページと 31 ページの「特権の実装方法」を参照してください。

- idlecmd キーワード – idletime に達したあとでユーザーをログアウトし、画面をロックします。
- idletime キーワード – キーボードアクティビティーが行われなかったあとでシステムが使用可能である時間を設定します。idlecmd の値を指定するときには idletime を設定します。
- lock_after_retries キーワード – この値が yes の場合は、再試行回数が /etc/default/login ファイルで許可されている回数を超えると、システムはロックされます。詳細は、login(1) のマニュアルページを参照してください。ロックされたアカウントのロックを解除するには、passwd(1) のマニュアルページを参照してください。
- pam_policy キーワード – ユーザー単位の PAM ポリシーを指定します。Unresolved link to " pam_user_policy7" のマニュアルページを参照してください。
- project キーワード - デフォルトのプロジェクトを追加します。詳細は、project(5) のマニュアルページを参照してください。
- profilesキーワード – 権利プロファイルを割り当てます。詳細は、profiles(1) のマニュアルページを参照してください。
- unlock_after キーワード – 認証の成功によりロック済みアカウントをロック解除できるまでの時間を指定します。時間は、分数、時間数、日数、または週数として指定できます。この属性の時間が指定されていない場合、管理者はアカウントを明示的にロック解除する必要があります。ロックされたアカウントのロックを解除するには、passwd(1) のマニュアルページを参照してください。

注記 - access_times および access_tz 属性は PAM 属性であるため、認証中に検査されます。したがって、ユーザーまたは役割に直接割り当てるか、または認証権利プロファイルに含める必要があります。これらは通常の権利プロファイルでは無視されます。

修飾属性は、LDAP ネーミングスコープ内のユーザーと役割に対してのみ設定できます。これらの修飾子により、権利プロファイルなどのユーザーと役割の属性割り当てが 1 つまたは複数のシステムに限定されます。例については、useradd(8) および user_attr(5) のマニュアルページを参照してください。

修飾子は host と netgroup です。

- host 修飾子 – ユーザーまたは役割が指定されたアクションを実行できるシステムを識別します。
- netgroup 修飾子 – ユーザーまたは役割が指定されたアクションを実行できるシステムを一覧表示します。host 割り当ては、netgroup よりも優先されます。

詳細は、user_attr(5) のマニュアルページを参照してください。このデータベースの内容を表示するには、getent user_attr コマンドを使用します。詳細は、Unresolved link to " getent8" のマニュアルページおよび 第7章「 の権利の一覧表示」を参照してください。

auth_attr データベース

auth_attr データベースには、承認定義が格納されます。承認は、ユーザー、役割、権利プロファイルに割り当てることができます。承認を権利プロファイルに配置してから、その権利プロファイルを役割またはユーザーに割り当てる方法が推奨されます。

このデータベースの内容を表示するには、getent auth_attr コマンドを使用します。詳細は、Unresolved link to " getent8" のマニュアルページおよび 第7章「 の権利の一覧表示」を参照してください。

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、特権、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られたコマンドとセキュリティー属性は、exec_attr データベースに格納されます。詳細については、160 ページの「exec_attr データベース」を参照してください。

詳細は、prof_attr(5) のマニュアルページを参照してください。このデータベースの内容を表示するには、getent exec_attr コマンドを使用します。詳細は、Unresolved link to " getent8" のマニュアルページおよび 第7章「 の権利の一覧表示」を参照してください。

exec_attr データベース

exec_attr データベースでは、成功するためにセキュリティー属性を必要とするコマンドが定義されます。このコマンドは、権利プロファイルの一部です。セキュリティー属性を指定したコマンドは、プロファイルが割り当てられている役割またはユーザーが実行できます。

詳細は、exec_attr(5) のマニュアルページを参照してください。このデータベースの内容を表示するには、getent コマンドを使用します。詳細は、Unresolved link to " getent8" のマニュアルページおよび 第7章「 の権利の一覧表示」を参照してください。

policy.conf ファイル

注記 - このファイルは SMF account-policy サービスで置き換えられました。詳細は、82 ページの「新機能 – account-policy サービスの有効化」および account-policy(8S) のマニュアルページを参照してください。

/etc/security/policy.conf ファイルは、特定の権利プロファイル、特定の承認、および特定の特権をすべてのユーザーに付与する方法を定義します。ファイル内の関連するエントリは、*key*=*value* のペアから構成されます。

- AUTHS_GRANTED=*authorizations* – 1 つまたは複数の承認を示します。
- AUTH_PROFS_GRANTED=*rights profiles* – 1 つまたは複数の認証権利プロファイルを示します。
- PROFS_GRANTED=*rights profiles* – 1 つまたは複数の未認証の権利プロファイルを示します。
- CONSOLE_USER=Console User – Console User 権利プロファイルを示します。このプロファイルは、便利な承認セットとともにコンソールユーザーに提供されます。このプロファイルはカスタマイズできます。
- PRIV_DEFAULT=*privileges* – 1 つまたは複数の特権を示します。
- PRIV_LIMIT=*privileges* – すべての特権を示します。

次の例では、policy.conf データベースの権利値をいくつか示します。

```
##
AUTHS_GRANTED=
AUTH_PROFS_GRANTED=
CONSOLE_USER=Console User
PROFS_GRANTED=Basic Solaris User
#PRIV_DEFAULT=basic
#PRIV_LIMIT=all
```

権利管理コマンド

このセクションには、権利の管理に使用するコマンドのリストを示します。承認を使用してアクセス権を制御できるコマンドの表も含まれています。

承認、権利プロファイル、および役割を管理するコマンド

次の表に示すコマンドは、ユーザープロセスに対する権利を取得および設定します。

表 3 権利管理コマンド

コマンド	説明
account-policy(8S)	システムセキュリティーポリシーのための SMF ステンシル。
auths(1)	ユーザーの承認を表示します。新しい承認を作成します。
Unresolved link to "getent8"	権利データベースの内容を一覧表示します。
nscd(8)	権利データベースをキャッシュする場合に有用なネームサービスキャッシュデーモン。svcadm コマンドを使用してデーモンを再起動します。
Unresolved link to "pam_roles7"	PAM 用の役割アカウント管理モジュール。役割を引き受けるための承認を確認します。
Unresolved link to "pam_unix_account7"	PAM 用 の UNIX アカウント管理モジュール。時間制約や非アクティブな状態などのアカウントの制限を調べます。
pfbash(1)	権利を評価できるプロファイルシェルプロセスの作成に使用されます。
pfedit(8)	管理ファイルの編集に使用します。
pfexec(1)	セキュリティー属性を持つコマンドの実行に使用されます。
profiles(1)	指定したユーザーの権利プロファイルを表示します。権利プロファイルを作成または変更します。
roles(1)	指定されたユーザーが引き受けられる役割を表示します。
roleadd(8)	役割をローカルシステムまたは LDAP ネットワークに追加します。
roleadd(8)	役割をローカルシステムまたは LDAP ネットワークに追加します。
rolemod(8)	ローカルシステムまたは LDAP ネットワーク上で役割のプロパティーを変更します。
userattr(1)	ユーザーまたは役割アカウントに割り当てられている特定の権利の値を表示します。
Unresolved link to "useradm8"	ユーザーまたは役割アカウントに直接割り当てられるすべての権利を表示します。useradm パッケージのインストールが必要です。
useradd(8)	ユーザーアカウントをシステムまたは LDAP ネットワークに追加します。ユーザーのアカウントに役割を割り当てるには、-R オプションを使用します。
userdel(8)	ユーザーのログインをシステムまたは LDAP ネットワークから削除します。
usermod(8)	システム上のユーザーのアカウントプロパティーを変更します。

承認を必要とする特別なコマンド

次の表では、承認を使用して システムのコマンドオプションを制限する方法を示します。承認の詳細は、156 ページの「承認のリファレンス」を参照してください。

表 4 コマンドおよび関連する承認

コマンド	承認の要件
at(1)	すべてのオプションで solaris.jobs.user が必要です (at.allow ファイルおよび at.deny ファイルがない場合)
atq(1)	すべてのオプションで solaris.jobs.admin が必要です
cdrw(1)	すべてのオプションで solaris.device.cdrw が必要であり、これは policy.conf ファイルでデフォルトで付与されます
crontab(1)	ジョブを送信するオプションの場合は solaris.jobs.user が必要です (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 ファイルを一覧表示または変更する場合は、solaris.jobs.admin が必要です
allocate(8)	デバイスを割り当てる場合は、solaris.device.allocate (または、device_allocate ファイルに指定されている別承認) が必要です 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の承認) が必要です
deallocate(8)	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 ファイルに指定されている別の承認) が必要です 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 に指定されている別承認) が必要です
list_devices(1)	ほかのユーザーのデバイスを一覧表示する場合 (-U オプション) は、solaris.device.revoke が必要です
roleadd(8)	役割を作成するには、solaris.user.manage が必要です。初期パスワードを設定するには、solaris.account.activate が必要です。アカウントロックやパスワードの有効期限などのパスワードポリシーを設定するには、solaris.account.setpolicy が必要です。
roledel(8)	パスワードを削除するには、solaris.passwd.assign 承認が必要です。
rolemod(8)	パスワードを変更するには、solaris.passwd.assign 承認が必要です。アカウントロックやパスワードの有効期限などのパスワードポリシーを変更するには、solaris.account.setpolicy が必要です。
sendmail(8)	メールサブシステム機能にアクセスするには、solaris.mail が必要です。メールキューを表示するには、solaris.mail.mailq が必要です
useradd(8)	ユーザーを作成するには、solaris.user.manage が必要です。初期パスワードを設定するには、solaris.account.activate が必要です。アカウントロックやパスワードの有効期限などのパスワードポリシーを設定するには、solaris.account.setpolicy が必要です。
userdel(8)	パスワードを削除するには、solaris.passwd.assign 承認が必要です。
usermod(8)	パスワードを変更するには、solaris.passwd.assign 承認が必要です。アカウントロックやパスワードの有効期限などのパスワードポリシーを変更するには、solaris.account.setpolicy が必要です。

特権のリファレンス

プロセスを制限する特権は、カーネル内に実装され、コマンド、ユーザー、役割、またはシステムレベルでプロセスを制限できます。

特権処理のためのコマンド

次の表に、特権の処理に使用できるコマンドのリストを示します。

表 5 特権処理のためのコマンド

目的	コマンド	マニュアルページ
システムのデフォルト特権および制限特権を設定および一覧表示します。	`svccfg -s account-policy`	`account-policy(8S)`
特権の失敗をデバッグします	`ppriv -eD` *failed-operation*	`ppriv(1)`
システム上の特権を一覧表示します	`ppriv -l`	`ppriv(1)`
特権とその説明を一覧表示します	`ppriv -lv` *priv*	`ppriv(1)`
UID、プロセス、またはポートに関する拡張特権ポリシーを一覧表示します	`ppriv -lv` *extended-policy*	`ppriv(1)`
プロセスの特権を検査します	`ppriv -v` *pid*	`ppriv(1)`
拡張特権ポリシーを UID、プロセス、またはポートに追加します	`ppriv -r` *rule*	`privileges(7)`
プロセスの特権を設定します	`ppriv -s` *spec*	`ppriv(1)`
拡張特権ポリシールールを削除します	`ppriv -X` *rule*	`privileges(7)`
特権を権利プロファイルに割り当てます	`profiles -p` *profile-name*	`profiles(1)`
特権を新しい役割に割り当てます	`roleadd -K defaultpriv=`	`roleadd(8)`
特権を既存の役割に追加します	`rolemod -K defaultpriv+=`	`rolemod(8)`
特権を新しいユーザーに割り当てます	`useradd -K defaultpriv=`	`useradd(8)`
特権を既存のユーザーに追加します	`usermod -K defaultpriv+=`	`usermod(8)`
デバイスポリシーをデバイスに追加します	`add_drv -p` *policy driver*	`add_drv(8)`
デバイスポリシーを設定します	`devfsadm`	`devfsadm(8)`
デバイスポリシーを表示します	`getdevpolicy`	Unresolved link to " getdevpolicy8"
オープンデバイスでのデバイスポリシーを更新します	`update_drv -p` *policy driver*	`update_drv(8)`

特権情報が含まれる SMF ステンシル

account-policy SMF ステンシルには次の特権情報が格納および設定されています。

- default_privileges – システムに対する特権の継承可能セット
- limit_privileges – システムに対する特権の制限セット
- syslog – システムロギングファイル
  デバッグメッセージのパスは priv.debug エントリに設定されています。

監査レコードの特権アクション

特権の使用は監査することができます。プロセスで特権が使用される場合は常に、upriv 監査トークン内の監査トレールに特権の使用が記録されます。特権の名前がレコードに含まれる場合、テキスト形式が使用されます。次の監査イベントにより、特権の使用が記録されます。

- AUE_SETPPRIV **監査イベント** – 特権セットが変更されたときに監査レコードを生成します。AUE_SETPPRIV 監査イベントは pm クラスにあります。
- AUE_MODALLOCPRIV **監査イベント** – カーネルの外部から特権が追加されたときに監査レコードを生成します。AUE_MODALLOCPRIV 監査イベントは ad クラスにあります。
- AUE_MODDEVPLCY **監査イベント** – デバイスポリシーが変更されたときに監査レコードを生成します。AUE_MODDEVPLCY 監査イベントは ad クラスにあります。
- AUE_PFEXEC **監査イベント** – pfexec() が有効になっている execve() の呼び出しが行われたときに監査レコードを生成します。AUE_PFEXEC 監査イベントは、as、ex、ps、および ua 監査クラスにあります。特権の名前は、監査レコードに含まれます。

基本セット内に含まれている特権の正常な使用は監査されません。ユーザーの基本セットから削除された基本特権の使用を試みる場合、監査されます。

特権単位の監査については、『Managing Auditing in Oracle Solaris 11.4』 の 「What's New in the Audit Service in Oracle Solaris 11.4」を参照してください。

ファイルおよび対応する SMF プロパティー内のセキュリティー属性

次の表は、/etc ディレクトリ内のファイルのセキュリティー属性の変数名と、account-policy サービス内の対応する SMF プロパティーを一覧表示しています。

- 表6
- 表7
- 表8
- 表9
- 表10

表6の SMF プロパティーは、account-policy サービスの config/etc_default_login ステンシルが有効な場合に変更できます。

表 6 ファイルおよび SMF 内のログインセキュリティー属性

変数名	レガシーファイル	SMF プロパティー
ANNOTATION	/etc/security/policy.conf	login_policy/annotation
CLEARANCE	/etc/security/policy.conf	login_policy/clearance
CONSOLE	/etc/default/login	login_policy/root_login_device
DISABLETIME	/etc/default/login	login_policy/disabletime
LOCK_AFTER_RETRIES	/etc/security/policy.conf	login_policy/lock_after_retries
PAM_POLICY	/etc/security/policy.conf	login_policy/pam_policy
PASSREQ	/etc/default/login	login_policy/password_required
RETRIES	/etc/default/login	login_policy/retries
SLEEPTIME	/etc/default/login	login_policy/sleeptime
TIMEOUT	/etc/default/login	login_policy/timeout
UNLOCK_AFTER	/etc/security/policy.conf	login_policy/auto_unlock_time

表7 の SMF プロパティーは、account-policy サービスの config/etc_default_passwd ステンシルが有効な場合に変更できます。

表 7 ファイルおよび SMF 内のパスワードセキュリティー属性

変数名	レガシーファイル	SMF プロパティー
CRYPT_DEFAULT	/etc/security/policy.conf	password/crypt/default
CRYPT_ALGORITHMS_ALLOW	/etc/security/policy.conf	password/crypt/algorithms_allow
CRYPT_ALGORITHMS_DEPRECATE	/etc/security/policy.conf	password/crypt/algorithms_deprecate
DICTIONDBDIR	/etc/default/passwd	password/dictionary/db_dir

変数名	レガシーファイル	SMF プロパティー
DICTIONLIST	/etc/default/passwd	password/dictionary/word_list
DICTIONMINWORDLENGTH	/etc/default/passwd	password/dictionary/ min_word_length
HISTORY	/etc/default/passwd	password/history
MAXDAYS	/etc/default/passwd	password/aging_defaults/max_days
MAXREPEATS	/etc/default/passwd	password/complexity/max_repeats
MAXWEEKS	/etc/default/passwd	password/aging_defaults/ max_weeks
MINALPHA	/etc/default/passwd	password/complexity/min_alpha
MINDAYS	/etc/default/passwd	password/aging_defaults/min_days
MINDIFF	/etc/default/passwd	password/complexity/min_diff
MINDIGIT	/etc/default/passwd	password/complexity/min_digit
MINLOWER	/etc/default/passwd	password/complexity/min_lower
MINNONALPHA	/etc/default/passwd	password/complexity/min_nonalpha
MINSPECIAL	/etc/default/passwd	password/complexity/min_special
MINUPPER	/etc/default/passwd	password/complexity/min_upper
MINWEEKS	/etc/default/passwd	password/aging_defaults/ min_weeks
NAMECHECK	/etc/default/passwd	password/complexity/namecheck
PASSLENGTH	/etc/default/passwd	password/complexity/passlength
WARNDAYS	/etc/default/passwd	password/aging_defaults/ warn_days
WARNWEEKS	/etc/default/passwd	password/aging_defaults/ warn_weeks
WHITESPACE	/etc/default/passwd	password/complexity/whitespace

表8の SMF プロパティーは、account-policy サービスの config/ etc_security_policyconf ステンシルが有効な場合に変更できます。

表 8 ファイルおよび SMF 内のユーザーアカウントセキュリティー属性

変数名	レガシーファイル	SMF プロパティー
AUTH_PROFS_GRANTED	/etc/security/policy.conf	rbac/default_auth_profiles
AUTHS_GRANTED	/etc/security/policy.conf	rbac/default_authorizations
CONSOLE_USER	/etc/security/policy.conf	rbac/console_user_profiles
PRIV_DEFAULT	/etc/security/policy.conf	rbac/default_privileges
PRIV_LIMIT	/etc/security/policy.conf	rbac/default_limit_privileges
PROFS_GRANTED	/etc/security/policy.conf	rbac/default_profiles

表9の SMF プロパティーは、account-policy サービスの config/ etc_default_login ステンシルが有効な場合に変更できます。

表 9 ファイルおよび SMF 内のユーザー環境セキュリティー属性

変数名	レガシーファイル	SMF プロパティー
ALTSHELL	/etc/default/login	login/environment/set_shell
HZ	/etc/default/login	login/environment/hz
PATH	/etc/default/login	login/environment/path
SUPATH	/etc/default/login	login/environment/root_path
TIMEZONE	/etc/default/login	login/environment/timezone
ULIMIT	/etc/default/login	login/environment/ulimit
UMASK	/etc/default/login	login/environment/umask

表10の SMF プロパティーは、account-policy サービスの config/etc_default_login ステンシルおよび config/etc_default_su ステンシルが有効な場合に変更できます。

表 10 ファイルおよび SMF 内のロギングおよび su セキュリティー属性

変数名	レガシーファイル	SMF プロパティー
SYSLOG	/etc/default/login	login/log/syslog
SYSLOG_FAILED_LOGINS	/etc/default/login	login/log/syslog_failed_attempts
CONSOLE	/etc/default/su	su/log/device
PATH	/etc/default/su	su/environment/path
SULOG	/etc/default/su	su/log/logfile
SUPATH	/etc/default/su	su/environment/path
SYSLOG	/etc/default/su	su/log/syslog

ユーザーおよびプロセスのセキュリティー保護に関する用語集

基本セット ログイン時にユーザーのプロセスに割り当てられる一連の特権。変更されていないシステムの場合、各ユーザーの初期の継承可能セットはログイン時の基本セットと同じです。

許可されたセット プロセスによって使用できる一連の特権。

継承可能セット プロセスが exec の呼び出しを通して継承できる一連の特権。

権利 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。ユーザー権利の管理およびプロセス権利の管理で、組織はスーパーユーザーの特権を分割して、ユーザーまたは役割に割り当てることができます。 の権利は、カーネル特権、承認、または特定の UID や GID としてプロセスを実行する機能として実装されています。権利は、権利プロファイルおよび役割に集めることができます。

権利プロファイル プロファイルとも呼ばれます。役割またはユーザーに割り当てることができるセキュリティーオーバーライドの集合。権利プロファイルには、承認、特権、セキュリティー属性が割り当てられたコマンド、および補足プロファイルと呼ばれるその他の権利プロファイルを含めることができます。

権利ポリシー コマンドに関連付けられるセキュリティーポリシー。現在、 で有効なポリシーは solaris です。solaris ポリシーでは、特権と拡張特権ポリシー、承認、および setuid セキュリティー属性が認識されます。

権利モデル コンピュータシステムにおいてスーパーユーザーモデルより厳密なセキュリティーモデル。権利モデルでは、実行するプロセスには特権が必要です。システムの管理は、管理者が各自のプロセスで与えられている特権に基づいて複数の個別部分に分割できます。特権は、管理者のログインプロセスに割り当てることも、特定のコマンドだけで有効なように割り当てることも可能です。

最小特権 指定されたプロセスにスーパーユーザー権限のサブセットのみを提供するセキュリティーモデル。最小特権モデルでは、通常のユーザーに、ファイルシステムのマウントやファイルの所有権の変更などの個人の管理タスクを実行できる十分な特権を割

り当てます。これに対して、プロセスは、スーパーユーザーの完全な権限 (つまり、すべての特権) ではなく、タスクを完了するために必要な特権のみで実行されます。バッファーオーバーフローなどのプログラミングエラーによる損害を、保護されたシステムファイルの読み取りまたは書き込みやシステムの停止などの重要な機能にはアクセスできない root 以外のユーザーに封じ込めることができます。

再認証 コンピュータ操作を実行するためにパスワードを指定する際の要件。通常、sudo 操作では再認証が必要です。認証済み権利プロファイルには、再認証が必要なコマンドを含めることができます。認証権利プロファイルを参照してください。

承認 通常であればセキュリティーポリシーによって禁止されている操作のクラスを実行するために役割またはユーザーに割り当てる (または権利プロファイルに埋め込む) ことができる権利。承認はカーネルではなく、ユーザーアプリケーションレベルで適用されます。

信頼できるユーザー 一定の信頼レベルで管理タスクを実行できるように決定されたユーザー。一般に、管理者は最初に信頼できるユーザーのログインを作成してから、ユーザーの信頼および能力レベルに合致した管理者権利を割り当てます。その後、これらのユーザーはシステムの構成および保守を支援します。特権ユーザーとも呼ばれます。

スーパーユーザーモデル コンピュータシステムにおける典型的な UNIX セキュリティーモデル。スーパーユーザーモデルでは、管理者は絶対的なシステム制御権を持ちます。一般に、システム管理のために 1 人のユーザーがスーパーユーザー (root) になり、すべての管理作業を行える状態となります。

制限セット プロセスとその子プロセスでどの特権が利用できるかを示す上限。

責務分離 最小特権の概念の一部。責務分離により、1 人のユーザーが、トランザクションを完了するためのすべての操作を実行または承認することが回避されます。たとえば、RBAC では、セキュリティーオーバーライドの割り当てからログインユーザーの作成を分離できます。1 つの役割がユーザーを作成します。個別の役割により、権利プロファイル、役割、特権などのセキュリティー属性を既存のユーザーに割り当てることができます。

セキュリティー属性 セキュリティーポリシーをオーバーライドし、スーパーユーザー以外のユーザーによって実行されても成功する管理コマンドを有効にします。スーパーユーザーモデルでは、`setuid root` プログラムと `setgid` プログラムがセキュリティー属性です。これらの属性がコマンドで指定されると、そのコマンドがどのようなユーザーによって実行されているかにかかわらず、コマンドは正常に処理されます。特権モデルでは、セキュリティー属性として 権利 プログラムがカーネル特権およびその他の`rights`によって置き換えられます。特権モデルは、スーパーユーザーモデルと互換性があります。このため、特権モデルは `setuid` プログラムと `setgid` プログラムをセキュリティー属性として認識します。

セキュリティーポリシー ポリシーを参照してください。

特権

1. 一般に、コンピュータシステム上で通常のユーザーの能力を超える操作を実行する能力または機能。スーパーユーザー特権は、スーパーユーザーに付与されているすべての権利です。特権ユーザーまたは特権アプリケーションは、追加の権利が付与されているユーザーまたはアプリケーションです。

2. システムにおいてプロセスに対する個々の権利。特権を使用すると、root を使用するよりもきめ細かなプロセス制御が可能です。特権の定義と適用はカーネルで行われます。特権は、プロセス特権やカーネル特権とも呼ばれます。特権の詳細は、privileges(7) のマニュアルページを参照してください。

特権エスカレーション

権利 (デフォルトをオーバーライドして許可する権利を含む) を割り当てられたリソース範囲の外部のリソースへのアクセス権を取得すること。その結果、プロセスは未承認の操作を実行できます。

特権セット

一連の特権。各プロセスには、プロセスが特定の特権を使用できるかどうかを判断する 4 セットの特権があります。詳細は、制限セット、有効セット、許可されたセット、および 継承可能セットを参照してください。

基本セットも、ユーザーのログインプロセスに割り当てられる特権セットです。

特権付きアプリケーション

システム制御をオーバーライドできるアプリケーション。このようなアプリケーションは、セキュリティー属性 (特定の UID、GID、承認、特権など) をチェックします。

特権モデル

権利モデルを参照してください。

特権ユーザー

コンピュータシステム上で通常のユーザーの権利を超えるrightsが割り当てられているユーザー。信頼できるユーザーも参照してください。

特権を認識する

自身のコードでの特権の使用を有効および無効にするプログラム、スクリプト、およびコマンド。本稼動環境では、たとえば、プログラムのユーザーに、その特権をプログラムに追加する権利プロファイルの使用を要求することによって、有効になった特権をプロセスに提供する必要があります。特権の詳細は、privileges(7) のマニュアルページを参照してください。

認証

ログインまたはプロセスの請求された識別情報を検証するプロセス。

認証権利プロファイル

権利プロファイル の 1 つ。割り当てられたユーザーまたは役割は、プロファイルから操作を実行する前に、パスワードを入力する必要があります。この動作は、sudo の動作に似ています。パスワードが有効である時間の長さは構成可能です。

パスワードポリシー

パスワードの生成に使用できる暗号化アルゴリズム。パスワードをどれぐらいの頻度で変更すべきか、パスワードの試行を何回まで認めるかといったセキュリティー上の考慮事項など、パスワードに関連した一般的な事柄を指すこともあります。セキュリティーポリシーにはパスワードが必要です。パスワードポリシーでは、AES アルゴリズムを使用してパスワードを暗号化することを要求したり、パスワードの強度に関連したそれ以上の要件を設定したりすることもできます。

プロファイル

権利プロファイルを参照してください。

プロファイルシェル 権利の管理で、役割 (またはユーザー) がコマンド行から、その役割の権利プロファイルに割り当てられた任意の特権付きアプリケーションを実行できるようにするシェル。プロファイルシェルのバージョンは、システム上で使用可能なシェルのバージョン (`bash` の `pfbash` バージョンなど) に対応します。

ポリシー 一般には、意思やアクションに影響を与えたり、これらを決定したりする計画や手続き。コンピュータシステムでは、多くの場合セキュリティーポリシーを指します。実際のサイトのセキュリティーポリシーは、処理される情報の重要度や未承認アクセスから情報を保護する手段を定義する規則セットです。たとえば、セキュリティーポリシーでパスワードを 6 週ごとに変更することを要求される可能性があります。パスワードポリシーおよび権利ポリシーも参照してください。

役割 管理タスクを実行するために作成し、信頼できるユーザーに割り当てる権利を持つアカウント。`armor` パッケージには、7 つの事前定義された役割が含まれています。

有効セット プロセスにおいて現在有効である一連の特権。

RBAC のユーザー権利管理機能である、役割に基づくアクセス制御。権利を参照してください。

RBAC ポリシー 権利ポリシーを参照してください。

索引

さ

た

な

は

ら

わ

A

B

C

D

E

F

S

T

U

V

W

Z