Oracle® Solaris 11.2 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

Part No: E53949
2014 年 7 月

ORACLE®

目次

このドキュメントの使用

- **概要** - 正当なファイルの保護、非表示のファイルアクセス権の表示、および不正なファイルの実行防止の方法について説明します。また、Oracle Solaris システム上で一定期間にわたってファイルの整合性を検証する方法についても説明します。
- **対象読者** - システム管理者。
- **必要な知識** - 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

この製品の最新情報や既知の問題は、ドキュメントライブラリ (http://www.oracle.com/pls/topic/lookup?ctx=E56342) に含まれています。

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info を参照してください。聴覚に障害をお持ちの場合は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs を参照してください。

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 ファイルアクセスの制御

この章では、Oracle Solaris でファイルを保護する方法について説明します。また、システムに悪影響を与える可能性のあるアクセス権が設定されたファイルからシステムを保護する方法についても説明します。

注記 - ZFS は、デフォルトの OS ファイルシステムです。アクセス制御リスト (ACL) で ZFS ファイルを保護するには、『Oracle Solaris 11.2 での ZFS ファイルシステムの管理 』の第 7 章「ACL および属性を使用した Oracle Solaris ZFS ファイルの保護」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。

UNIX アクセス権によるファイル保護

UNIX ファイルアクセス権と ACL を使用して、ファイルをセキュリティー保護できます。スティッキービットが設定されたファイルと、実行可能なファイルは、特殊なセキュリティー対策が必要です。

ファイルの監視と保護を行うコマンド

この表は、ファイルとディレクトリのモニタリングと保護を行うコマンドについて説明したものです。

表 1-1　ファイルとディレクトリの保護を行うコマンド

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

ファイルとディレクトリの所有権

従来の UNIX ファイルアクセス権では、次に示す 3 つのユーザークラスに所有権を割り当てることができます。

- **ユーザー** – ファイルまたはディレクトリの所有者。通常はファイルを作成したユーザーです。ファイルの所有者は、そのファイルの読み取り、書き込み (ファイルを変更する)、または実行 (コマンドの場合) が行えるユーザーを決定できます。
- **グループ** – ユーザーグループのメンバー。
- **その他** – ファイル所有者ではなくグループメンバーでもないその他の全ユーザー。

ファイルアクセス権の割り当てや変更が行えるのは、通常、そのファイルの所有者だけです。さらに、root アカウントでもファイルの所有権を変更できます。システムポリシーをオーバーライドする方法については、例1-2「自分のファイルの所有権を変更するためのユーザーの有効化」を参照してください。

ファイルは、7 つの形式のいずれかになります。各形式は、次のように記号で示されます。

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

UNIX ファイルアクセス権

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

表 1-2 ファイルとディレクトリのアクセス権

記号	アクセス権	オブジェクト	説明
r	読み取り	ファイル	ファイルの内容を開いて読み込むことができます。
		ディレクトリ	ディレクトリ内のファイルを一覧表示できます。
w	書き込み	ファイル	ファイルの内容の変更またはファイルの削除を行えます。
		ディレクトリ	ディレクトリに対してファイルまたはリンクを追加できます。また、ディレクトリ内のファイルまたはリンクの削除も行えます。
x	実行	ファイル	ファイルがプログラムまたはシェルスクリプトの場合、そのファイルを実行できます。また、exec(2) システム呼び出しの 1 つを使用してそのプログラムを実行することもできます。
		ディレクトリ	ディレクトリ内のファイルを開いたり、実行したりできます。また、ディレクトリを作成し、その下にサブディレクトリを作成できます。
-	拒否	ファイルとディレクトリ	ファイルの読み込み、書き込み、または実行を行うことができません。

これらのファイルアクセス権は、通常のファイルと特殊ファイル (デバイス、ソケット、名前付きパイプ (FIFO) など) の両方に適用されます。

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

ディレクトリとそのサブディレクトリ内のファイルは、そのディレクトリに対するファイルアクセス権を制限することで保護できます。ただし、root 役割は、システム上のすべてのファイルとディレクトリにアクセスできることに注意してください。

setuid、setgid、およびスティッキービットを使用する特殊なファイルアクセス権

実行可能ファイルと公開ディレクトリには、3 種類の特殊なアクセス権 (setuid、setgid、およびスティッキービット) を設定できます。これらのアクセス権を設定すると、その実行可能ファイ

ルを実行するユーザーは、そのファイルの所有者 (またはグループ) の ID を持つことができます。

特殊なアクセス権はセキュリティー上の問題を引き起こすため、設定するときは十分な注意が必要です。たとえば、ユーザーは、ユーザー ID (UID) を `0` (`root` の UID) に設定するプログラムを実行することによって `root` 権限を取得できます。また、すべてのユーザーは、所有するファイルに対して特殊なアクセス権を設定できるため、これもセキュリティー上の問題の原因となります。

`root` 権限を取得するための `setuid` アクセス権や `setgid` アクセス権の承認されていない使用がないかどうか、システムをモニターするようにしてください。疑わしいアクセス権によって、管理プログラムの所有権が `root` や `bin` ではなく一般ユーザーに付与されていることが考えられます。この特殊なアクセス権を使用しているファイルをすべて検索し、リストする方法は、23 ページの「特殊なファイルアクセス権が設定されたファイルを見つける方法」を参照してください。

`setuid` アクセス権

`setuid` アクセス権を実行可能ファイルに設定すると、このファイルを実行するプロセスにはその実行可能ファイルを実行しているユーザーではなく、ファイルの所有者に基づいてアクセス権が与えられます。この特殊なアクセス権を使用すると、通常は所有者しか利用できないファイルやディレクトリにアクセスできます。

たとえば、`passwd` コマンドの `setuid` アクセス権によってユーザーはパスワードを変更できます。次に、`setuid` アクセス権のある `passwd` コマンドの例を示します。

```
-r-sr-sr-x   1 root     sys       56808 Jun 17 12:02 /usr/bin/passwd
```

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

注記 - プログラムから、予約済みの UID (0 から 100) で `setuid` アクセス権を使用しても、実効 UID が正しく設定されないことがあります。シェルスクリプトを使用するか、`setuid` アクセス権では予約済み UID を使用しないようにしてください。

---

setgid アクセス権

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

```
-r-x--s--x   1 root    mail     71212 Jun 17 12:01 /usr/bin/mail
```

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

root 権限を取得するための setgid アクセス権の承認されていない使用がないかどうか、システムをモニターするようにしてください。疑わしいアクセス権によって、このようなプログラムへのグループアクセス権が、root や bin ではなく、意外なグループに与えられることがあります。このようなアクセス権を使用しているファイルをすべて検索し、一覧表示する方法は、23 ページの「特殊なファイルアクセス権が設定されたファイルを見つける方法」を参照してください。

スティッキービット

*「スティッキービット」*は、ディレクトリ内のファイルを保護するアクセス権ビットです。ディレクトリにスティッキービットが設定されている場合、そのファイルを削除できるのはその所有者、ディレクトリの所有者、または特権ユーザーだけです。特権ユーザーの例として root ユーザーが挙げられます。スティッキービットにより、ユーザーは /tmp などの公開ディレクトリからほかのユーザーのファイルを削除できなくなります。

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

TMPFS ファイルシステム上で公開ディレクトリを設定するときには、スティッキービットを手動で設定してください。手順については、例1-5「絶対モードによる特殊なファイルアクセス権の設定」を参照してください。

umask のデフォルト値

ファイルやディレクトリを作成するときには、デフォルトのアクセス権が設定されます。このシステムデフォルトは変更が可能です。テキストファイルのアクセス権は 666 であり、すべてのユーザーに読み取り権と書き込み権を与えます。ディレクトリと実行可能ファイルのアクセス権は 777 で、すべてのユーザーに読み取り権、書き込み権、および実行権を与えます。一般に、ユーザーは .bashrc や .kshrc.user などの自分のシェル初期化ファイルでシステムデフォルトをオーバーライドします。管理者は、/etc/profile ファイルでデフォルトを設定することもできます。

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

次の表に、umask の標準的な値とその値が実行可能ファイルに与える影響を示します。

表 1-3 各セキュリティーレベルの umask 設定

セキュリティーレベル	umask 設定	許可されないアクセス権
緩やか (744)	022	グループとその他のユーザーによる w
中程度 (751)	026	グループによる w、その他のユーザーによる rw
厳密 (740)	027	グループによる w、その他のユーザーによる rwx
厳しい (700)	077	グループとその他のユーザーによる rwx

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

ファイルアクセス権を設定するモード

chmod コマンドを使用すると、ファイルのアクセス権を変更できます。ファイルやディレクトリのアクセス権を変更するには、root またはファイルまたはディレクトリの所有者になる必要があります。

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

- **絶対モード** – 数値を使用してファイルアクセス権を表します。絶対モードを使用してアクセス権を変更するときは、3 つ 1 組のアクセス権を 8 進数で表します。絶対モードは、アクセス権の設定に一番多く使用されている方法です。
- **記号モード** – 文字と記号の組み合わせを使用して、アクセス権を追加したりアクセス権を削除したりします。

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

表 1-4 絶対モードによるファイルのアクセス権の設定

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

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

表 1-5 記号モードによるファイルのアクセス権の設定

記号	機能	説明
u	*<対象ユーザー>*	ユーザー (所有者)
g	*<対象ユーザー>*	グループ
o	*<対象ユーザー>*	その他のユーザー
a	*<対象ユーザー>*	すべて
=	*オペレータ*	割り当て
+	*オペレータ*	追加

記号	機能	説明
-	*オペレータ*	削除
r	*アクセス権 (アクセス権ビット)*	読み取り
w	*アクセス権 (アクセス権ビット)*	書き込み
x	*アクセス権 (アクセス権ビット)*	実行
l	*アクセス権 (アクセス権ビット)*	強制ロック、setgid ビットはオン、グループ実行ビットはオフ
s	*アクセス権 (アクセス権ビット)*	setuid または setgid ビットはオン
t	*アクセス権 (アクセス権ビット)*	スティッキービットはオン、その他の実行ビットはオン

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

who アクセス権を変更する対象となるユーザーを指定します。

operator 実行する操作を指定します。

permissions 変更するアクセス権を指定します。

絶対モードまたは記号モードで、ファイルに特殊なアクセス権を設定できます。しかし、ディレクトリに setuid アクセス権を設定する場合と、ディレクトリからこのアクセス権を削除する場合は、記号モードを使用する必要があります。絶対モードでは、3 つ 1 組のアクセス権の左端に新しい 8 進数値を追加して、特殊なアクセス権を設定します。例1-5「絶対モードによる特殊なファイルアクセス権の設定」を参照してください。次の表に、ファイルに特殊なアクセス権を設定する 8 進数値を示します。

表 1-6 絶対モードによる特殊なファイルアクセス権の設定

8 進数値	特殊なファイルアクセス権
1	スティッキービット
2	setgid
4	setuid

アクセス制御リストによる UFS ファイルの保護

従来の UNIX ファイル保護機能は、ファイルの所有者、ファイルグループ、その他のユーザーという 3 つのユーザークラスに 読み取り権、書き込み権、実行権を提供します。UFS ファイルシステムでは、アクセス制御リスト (ACL) により次のことが可能となり、ファイルセキュリティーを管理するレベルがさらに詳細になります。

- ファイル所有者、グループ、その他のユーザー、特定のユーザーおよびグループにファイルアクセス権を定義します
- これらの各カテゴリにデフォルトのアクセス権を定義します

注記 - ZFS ファイルシステム内の ACL と NFSv4 ファイルに対する ACL については、『Oracle Solaris 11.2 での ZFS ファイルシステムの管理 』の第 7 章「ACL および属性を使用した Oracle Solaris ZFS ファイルの保護」を参照してください。

たとえば、グループ内のすべてのユーザーがファイルを読み取れるようにする場合は、そのファイルにグループの読み取り権を設定すればすみます。ただし、グループ内の 1 人のユーザーだけがそのファイルに書き込めるようにしたい場合は、ACL を使用できます。

UFS ファイルシステムの ACL の詳細は、Oracle Solaris 10 リリース 用の*Solaris のシステム管理: セキュリティーサービス*を参照してください。

実行可能ファイルを原因とするセキュリティーへの悪影響を防止する

プログラムは、スタック上のデータの読み取りと書き込みを行います。通常、それらはコード用に特別に指定されたメモリーの読み取り専用部分から実行されます。スタック上のバッファーをオーバーフローさせる一部の攻撃では、新しいコードをそのスタックに挿入し、プログラムにそれを実行させようとします。スタックメモリーから実行権を削除すると、これらの攻撃が成功するのを防ぐことができます。つまり、ほとんどのプログラムは、実行可能スタックを使用せずに正しく機能できます。

64 ビットプロセスには通常、非実行可能スタックがあります。デフォルトでは、32 ビット SPARC プロセスには実行可能スタックがあります。`noexec_user_stack` 変数を使用すると、32 ビットプロセスのスタックを実行可能にするかどうかを指定できます。

この変数が設定されている場合、プログラムがスタック上でコードを実行しようとすると `SIGSEGV` シグナルが送信されます。このシグナルが送信されると、通常、プログラムはコアダン

プして終了します。このようなプログラムは、違反しているプログラム名、プロセス ID、およびプログラムを実行した実ユーザー ID を含む警告メッセージも生成します。例:

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

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

syslog メッセージは、セキュリティー上の潜在的な問題を観察するために役立ちます。また、このメッセージで、noexec_user_stack 変数を設定したために正しく動作しなくなった、実行可能スタックに依存する有効なプログラムを特定することもできます。メッセージを記録しない場合は、/etc/system ファイルで noexec_user_stack_log 変数を 0 に設定します。メッセージは記録されなくなりますが、SIGSEGV シグナルは引き続き送られるので、実行中のプログラムはコアダンプで終了します。

プログラムは、スタック実行を明示的にマークまたは防止することができます。プログラム内の mprotect() 関数は、スタックを実行可能として明示的にマークします。詳細は、mprotect(2) のマニュアルページを参照してください。-M /usr/lib/ld/map.noexstk でコンパイルされたプログラムは、システム全体の設定には関係なく、スタックを非実行可能にします。

ファイルの保護

次の手順では、UNIX アクセス権を持つファイルの保護、セキュリティーリスクのあるファイルの検出、およびそれらのファイルによるシステムの危殆化の防止を行います。

UNIX アクセス権によるファイルの保護

次のタスクマップは、ファイルアクセス権の一覧表示、ファイルアクセス権の変更、特殊なファイルアクセス権によるファイルの保護などのタスク操作について説明した箇所を示しています。

タスク	手順
ファイル情報を表示します。	17 ページの「ファイル情報を表示する方法」

タスク	手順
ローカルファイルの所有権を変更します。	18 ページの「ファイルの所有者を変更する方法」 19 ページの「ファイルのグループ所有権を変更する方法」
ローカルファイルのアクセス権を変更します。	20 ページの「ファイルアクセス権を記号モードで変更する方法」 21 ページの「ファイルアクセス権を絶対モードで変更する方法」 22 ページの「特殊なファイルアクセス権を絶対モードで変更する方法」

▼ ファイル情報を表示する方法

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

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

```
% ls -la
```

`-l`	ユーザー所有権、グループ所有権、ファイルのアクセス権などを長形式で表示します。
`-a`	ドット (`.`) で始まる隠しファイルを含め、すべてのファイルを表示します。

`ls` コマンドのすべてのオプションについては、`ls(1)` のマニュアルページを参照してください。

例 1-1 ファイル情報の表示

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

```
% cd /sbin
% ls -l
total 4960
-r-xr-xr-x   1 root   bin      12756 Dec 19  2013 6to4relay
lrwxrwxrwx   1 root   root        10 Dec 19  2013 accept -> cupsaccept
-r-xr-xr-x   1 root   bin      38420 Dec 19  2013 acctadm
-r-xr-xr-x   2 root   sys      70512 Dec 19  2013 add_drv
-r-xr-xr-x   1 root   bin       3126 Dec 19  2013 addgnupghome
drwxr-xr-x   2 root   bin         37 Dec 19  2013 amd64
-r-xr-xr-x   1 root   bin       2264 Dec 19  2013 applygnupgdefaults
-r-xr-xr-x   1 root   bin        153 Dec 19  2013 archiveadm
-r-xr-xr-x   1 root   bin      12644 Dec 19  2013 arp
.
.
.
```

それぞれの行には、ファイルについての情報が次の順で表示されています。

- ファイルの形式 – たとえば、d。ファイル形式の一覧は、8 ページの「ファイルとディレクトリの所有権」を参照してください。
- アクセス権 – たとえば、r-xr-xr-x。詳細は、8 ページの「ファイルとディレクトリの所有権」を参照してください。
- ハードリンクの数 – たとえば、2。
- ファイルの所有者 – たとえば、root。
- ファイルのグループ – たとえば、bin。
- バイト数でのファイルサイズ – たとえば、12644。
- ファイルの作成日時、またはファイルが最後に変更された日時 - たとえば、Dec 19 2013。
- ファイルの名前 - たとえば、arp。

▼ ファイルの所有者を変更する方法

始める前に ファイルまたはディレクトリの所有者でない場合は、Object Access Management 権利プロファイルが割り当てられている必要があります。公開オブジェクト であるファイルを変更するには、root 役割になる必要があります。

詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **ローカルファイルのアクセス権を表示します。**

```
% ls -l example-file
-rw-r--r--   1 janedoe   staff   112640 May 24 10:49 example-file
```

2. **ファイルの所有者を変更します。**

```
# chown stacey example-file
```

3. **ファイルの所有者が変更されていることを確認します。**

```
# ls -l example-file
-rw-r--r--   1 stacey   staff   112640 May 26 08:50 example-file
```

NFS マウントしたファイルのアクセス権を変更するには、『Oracle Solaris 11.2 でのネットワークファイルシステムの管理 』の第 5 章「ネットワークファイルシステムを管理するためのコマンド」を参照してください。

例 1-2 自分のファイルの所有権を変更するためのユーザーの有効化

セキュリティー上の考慮事項 – rstchown 変数の設定をゼロに変更するにはそれ相応の理由が必要です。デフォルト設定では、容量の割り当て制限を回避するために、ユーザーは自分のファイルがほかのユーザーの所有になっているときはそれを一覧表示できません。

この例では、rstchown 変数の値は、/etc/system ファイル内でゼロに設定されます。この設定によりファイルの所有者は、chown コマンドを使用してファイルの所有権をほかのユーザーに変更できます。所有者は chgrp コマンドを使用し、ファイルのグループ所有権を所有者自身が属していないグループに設定することもできます。変更は、システムのリブート時に適用されます。

```
set rstchown = 0
```

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

▼ ファイルのグループ所有権を変更する方法

始める前に ファイルまたはディレクトリの所有者でない場合は、Object Access Management 権利プロファイルが割り当てられている必要があります。公開オブジェクト であるファイルを変更するには、root 役割になる必要があります。

詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **ファイルのグループ所有権を変更します。**

```
% chgrp scifi example-file
```

グループの設定については、『Oracle Solaris 11.2 のユーザーアカウントとユーザー環境の管理 』の第 1 章「ユーザーアカウントとユーザー環境について」を参照してください。

2. **ファイルのグループ所有権が変更されていることを確認します。**

```
% ls -l example-file
-rw-r--r--   1 stacey   scifi   112640 June 20 08:55  example-file
```

また、例1-2「自分のファイルの所有権を変更するためのユーザーの有効化」も参照してください。

▼ ファイルアクセス権を記号モードで変更する方法

次の手順では、ユーザーはそのユーザーが所有するファイルのアクセス権を変更します。

1. **アクセス権を記号モードで変更します。**

```
% chmod who operator permissions filename
```

who	アクセス権を変更する対象となるユーザーを指定します。
operator	実行する操作を指定します。
permissions	変更するアクセス権を指定します。有効な記号の一覧は、表1-5「記号モードによるファイルのアクセス権の設定」を参照してください。
filename	ファイルまたはディレクトリを指定します。

2. **ファイルのアクセス権が変更されていることを確認します。**

```
% ls -l filename
```

注記 - ファイルまたはディレクトリの所有者でない場合は、Object Access Management 権利プロファイルが割り当てられている必要があります。公開オブジェクト であるファイルを変更するには、`root` 役割になる必要があります。

例 1-3 アクセス権を記号モードで変更する

次の例では、所有者がその他のユーザーから読み取り権を削除します。

```
% chmod o-r example-file1
```

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

```
% chmod a+rx example-file2
```

次の例では、所有者がグループのメンバーに読み取り権、書き込み権、および実行権を追加します。

```
% chmod g=rwx example-file3
```

▼ ファイルアクセス権を絶対モードで変更する方法

次の手順では、ユーザーはそのユーザーが所有するファイルのアクセス権を変更します。

1. **アクセス権を絶対モードで変更します。**

```
% chmod nnn filename
```

nnn	所有者、グループ、その他のユーザーのアクセス権をこの順序で表す 8 進数値を指定します。有効な 8 進数値の一覧は、表1-4「絶対モードによるファイルのアクセス権の設定」を参照してください。
filename	ファイルまたはディレクトリを指定します。

注記 - chmod コマンドを使用して、既存の ACL エントリを持つオブジェクトのファイルまたはディレクトリアクセス権を変更すると、それらの ACL エントリも変更される可能性があります。正確な変更は、chmod のアクセス権操作の変更と、ファイルシステムの aclmode および aclinherit プロパティー値に依存します。

詳細は、『Oracle Solaris 11.2 での ZFS ファイルシステムの管理 』の第 7 章「ACL および属性を使用した Oracle Solaris ZFS ファイルの保護」を参照してください。

---

2. **ファイルのアクセス権が変更されていることを確認します。**

```
% ls -l filename
```

注記 - ファイルまたはディレクトリの所有者でない場合は、Object Access Management 権利プロファイルが割り当てられている必要があります。公開オブジェクト であるファイルを変更するには、root 役割になる必要があります。

例 1-4 アクセス権を絶対モードで変更する

次の例では、管理者が公開されているディレクトリのアクセス権を 744 (読み取り/書き込み/実行、読み取り専用、読み取り専用) から 755 (読み取り/書き込み/実行、読み取り/実行、読み取り/実行) に変更します。

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

次の例では、所有者が実行可能シェルスクリプトのアクセス権を読み取り/書き込みから読み取り/書き込み/実行に変更します。

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

▼ 特殊なファイルアクセス権を絶対モードで変更する方法

始める前に ファイルまたはディレクトリの所有者でない場合は、Object Access Management 権利プロファイルが割り当てられている必要があります。公開オブジェクト であるファイルを変更するには、root 役割になる必要があります。

詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **特殊なアクセス権を絶対モードで変更します。**

   ```
   % chmod nnnn filename
   ```

   nnnn ファイルまたはディレクトリのアクセス権を変更する 8 進数値を指定します。一番左端の 8 進数値で、ファイルに特殊なアクセス権を設定します。特殊なアクセス権に有効な 8 進数値の一覧は、表1-6「絶対モードによる特殊なファイルアクセス権の設定」を参照してください。

   filename ファイルまたはディレクトリを指定します。

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

2. **ファイルのアクセス権が変更されていることを確認します。**

   ```
   % ls -l filename
   ```

例 1-5 絶対モードによる特殊なファイルアクセス権の設定

次の例では、管理者が dbprog ファイルに setuid アクセス権を設定します。

```
# chmod 4555 dbprog
# ls -l dbprog
```

```
-r-sr-xr-x   1 db     staff        12095 May  6 09:29 dbprog
```

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

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

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

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

セキュリティーリスクのあるプログラムからの保護

次のタスクマップは、リスクのある実行ファイルをシステム内に見つけるタスクや、プログラムが実行可能スタックを不正利用しないように防ぐタスクなどの操作を説明した箇所を示しています。

タスク	説明	手順
特殊なアクセス権を持つファイルを見つけます。	setuid ビットがセットされているが、root ユーザーによって所有されていないというファイルを見つけます。	23 ページの「特殊なファイルアクセス権が設定されたファイルを見つける方法」
実行可能スタックがオーバーフローするのを防ぎます。	プログラムが実行可能スタックを不正に利用しないように防ぎます。	25 ページの「プログラムが実行可能スタックを使用できないようにする方法」
実行可能スタックのメッセージがログに記録されるのを防ぎます。	実行可能スタックのメッセージのログ記録を無効にします。	例1-7「実行可能スタックメッセージのログ記録を無効にする」

▼ 特殊なファイルアクセス権が設定されたファイルを見つける方法

この手順では、プログラム上での承認されていない可能性のある、setuid および setgid アクセス権の使用を見つけます。疑わしい実行可能ファイルによって、所有権が root または bin ではなく、通常のユーザーに与えられることがあります。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **find コマンドを使用して setuid アクセス権が設定されているファイルを検索します。**

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

find *directory*	指定したディレクトリから始めて、マウントされているすべてのパスを検査します。*ディレクトリ*としてルート (/)、usr、opt などを指定できます。
-user root	root が所有するファイルを表示します。
-perm -4000	アクセス権が 4000 に設定されているファイルだけを表示します。
-exec ls -ldb	find コマンドの出力を ls -ldb 形式で表示します。ls(1) のマニュアルページを参照してください。
/tmp/*filename*	find コマンドの結果が書き込まれるファイルです。

   詳細は、find(1) を参照してください。

2. **結果を /tmp/*filename* に出力します。**

   ```
   # more /tmp/filename
   ```

   背景知識については、10 ページの「setuid アクセス権」を参照してください。

例 1-6 setuid アクセス権が設定されているファイルを検索する

次の例の出力は、rar というグループのユーザーが /usr/bin/rlogin のコピーを作成し、そのアクセス権を root への setuid に設定したことを示しています。この結果、/usr/rar/bin/rlogin プログラムは root アクセス権で実行されます。

/usr/rar ディレクトリを調べ、/usr/rar/bin/rlogin コマンドを削除したあと、管理者は find コマンドの出力をアーカイブします。

```
# find /usr -user root -perm -4000 -exec ls -ldb {} \; > /var/tmp/ckprm
# cat /var/tmp/ckprm
-rwsr-xr-x  1 root   sys   28000  Jul 14 14:14  /usr/bin/atq
-rwsr-xr-x  1 root   sys   32364  Jul 14 14:14  /usr/bin/atrm
-r-sr-xr-x  1 root   sys   41432  Jul 14 14:14  /usr/bin/chkey
-rwsr-xr-x  1 root   bin   82804  Jul 14 14:14  /usr/bin/cdrw
-r-sr-xr-x  1 root   bin    8008  Jul 14 14:14  /usr/bin/mailq
-r-sr-sr-x  1 root   sys   45348  Jul 14 14:14  /usr/bin/passwd
-rwsr-xr-x  1 root   bin   37724  Jul 14 14:14  /usr/bin/pfedit
-r-sr-xr-x  1 root   bin   51440  Jul 14 14:14  /usr/bin/rcp
---s--x---  1 root   rar   41592  Jul 24 16:14  /usr/rar/bin/rlogin
-r-s--x--x  1 root   bin  166908  Jul 14 14:14  /usr/bin/sudo
-r-sr-xr-x  4 root   bin   24024  Jul 14 14:14  /usr/bin/uptime
-r-sr-xr-x  1 root   bin   79488  Jul 14 14:14  /usr/bin/xlock
```

```
# mv /var/tmp/ckprm /var/share/sysreports/ckprm
```

▼ プログラムが実行可能スタックを使用できないようにする方法

32 ビットの実行可能スタックのセキュリティーリスクについては、15 ページの「実行可能ファイルを原因とするセキュリティーへの悪影響を防止する」を参照してください。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **/etc/system ファイルを編集し、次の行を追加します。**

```
# pfedit /etc/system
...
set noexec_user_stack=1
set noexec_user_stack_log=1
```

2. **システムをリブートします。**

```
# reboot
```

例 1-7 実行可能スタックメッセージのログ記録を無効にする

この例では、管理者が実行可能スタックメッセージのロギングを無効にしてから、システムをリブートします。

```
# cat /etc/system
set noexec_user_stack=1
set noexec_user_stack_log=0
# reboot
```

参照 詳細は、次を参照してください。

- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_overview
- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_continued
- https://blogs.oracle.com/gbrunett/entry/solaris_non_executable_stack_concluded

♦ ♦ ♦ 第 2 章

2 BART を使用したファイル整合性の検証

この章では、ファイル整合性ツール BART について説明します。BART は、一定期間にわたってシステム上のファイルの整合性を検証できるコマンド行ツールです。この章で扱う内容は、次のとおりです。

BART について

BART は、暗号化強度チェックサムとファイルシステムメタデータを使用して変更を判定する、ファイル整合性の走査および報告ツールです。BART は、壊れたファイルや異常なファイルを識別することによって、セキュリティー違反を検出したり、システム上のパフォーマンスの問題をトラブルシューティングしたりするのに役立ちます。BART を使用すると、配備されたシステム上にインストールされているファイルの相違が容易に、かつ確実に報告されるため、システムのネットワークの管理コストを削減できます。

BART を使用することで管理者は、既知のベースラインと比較し、ファイルレベルで見てどのような変化がシステムに起きたかを確認できます。BART を使用して、完全にインストールおよび構成されたシステムからベースラインまたは*制御目録*を作成します。作成したこのベースラインをあとでシステムのスナップショットと比較すれば、システムのインストール後にシステムで発生したファイルレベルの変化を示すレポートが生成されます。

BART の機能

BART では、強力で、かつ柔軟性の高い単純な構文が使用されます。このツールを使用すると、一定期間にわたって特定のシステム上のファイル変更を追跡できます。また、類似したシス

テム間のファイルの違いを追跡することもできます。このような比較は、壊れたファイルや異常なファイル、またはソフトウェアが期限切れになったシステムを見つけるのに役立ちます。

BART には、ほかにも次のようなメリットや利用法があります。

- どのファイルをモニターするかを指定できます。たとえば、ローカルなカスタマイズをモニターできます。これにより、ソフトウェアを容易に、かつ効率的に再構成できるようになります。
- システム性能の問題をトラブルシューティングできます。

BART コンポーネント

BART は、2 つの主なファイルである*目録* と比較ファイル (つまり、*レポート*) を作成します。オプションの*規則ファイル*を使用すると、目録やレポートをカスタマイズできます。

BART 目録

*目録*は、特定の時点でのシステムのファイルレベルのスナップショットです。目録にはファイルの属性に関する情報が含まれており、これには、いくつかの一意に識別する情報 (チェックサムなど) を含めることができます。`bart create` コマンドのオプションは、特定のファイルやディレクトリをターゲットにすることができます。29 ページの「BART 規則ファイル」で説明されているように、規則ファイルは、よりきめ細かなフィルタリングを提供できます。

注記 - デフォルトでは、BART は、ルート (/) ディレクトリの下にあるすべての ZFS ファイルシステムをカタログ化します。その他のファイルシステムタイプ (NFS ファイルシステムや TMPFS ファイルシステムなど)、およびマウントされた CD-ROM がカタログ化されます。

システムの目録は、Oracle Solaris の初期インストールの直後に作成できます。また、サイトのセキュリティーポリシーを満たすようにシステムを構成したあとに目録を作成することもできます。このタイプの制御目録によって、あとの比較のためのベースラインが提供されます。

ベースライン目録を使用すると、一定期間にわたって同じシステム上のファイル整合性を追跡できます。これはまた、ほかのシステムと比較するための基礎としても使用できます。たとえば、ネットワーク上のほかのシステムのスナップショットを作成したあと、これらの目録をベースライン目録と比較することができます。報告されたファイルの相違は、ほかのシステムをベースラインシステムと同期化するために何を行う必要があるかを示しています。

目録の形式については、42 ページの「BART 目録のファイル形式」を参照してください。目録を作成するには、`How to Create a Control Manifest`の説明に従って、31 ページの「制御目録を作成する方法」コマンドを使用します。

BART レポート

BART レポートは、2 つの目録間のファイルごとの相違を一覧表示します。*相違*とは、両方の目録に対してカタログ化されている特定のファイルの任意の属性への変更のことです。また、ファイルエントリの追加または削除も相違と見なされます。

有効な比較のためには、2 つの目録が同じファイルシステムをターゲットにする必要があります。また、同じオプションと規則ファイルを使用して目録を作成および比較することも必要です。

レポートの形式については、44 ページの「BART レポート」を参照してください。レポートを作成するには、`How to Compare Manifests for the Same System Over Time`の説明に従って、34 ページの「一定期間内で同一システムの目録を比較する方法」コマンドを使用します。

BART 規則ファイル

BART 規則ファイルは、追加または除外のために特定のファイルやファイル属性をフィルタリングしたり、ターゲットにしたりするために作成するファイルです。そのあと、BART 目録およびレポートを作成するときにこのファイルを使用します。目録を比較する場合、目録間の相違を報告するには規則ファイルが役立ちます。

注記 - 規則ファイルを使用して目録を作成する場合は、同じ規則ファイルを使用して比較目録を作成する必要があります。また、目録を比較するときにも、この規則ファイルを使用する必要があります。そうしないと、レポートによって、多くの無効な相違が一覧表示されます。

規則ファイルを使用してシステム上の特定のファイルやファイル属性をモニターする場合は、計画が必要です。規則ファイルを作成する前に、システム上のどのファイルおよびファイル属性をモニターするかを決定してください。

ユーザーエラーの結果として、規則ファイルには、構文エラーやその他のあいまいな情報が含まれている場合もあります。規則ファイルにエラーが含まれている場合は、これらのエラーも報告されます。

規則ファイルの形式については、43 ページの「BART 規則ファイルの書式」および bart_rules(4) のマニュアルページを参照してください。規則ファイルを作成するには、40 ページの「規則ファイルを使用して BART レポートをカスタマイズする方法」を参照してください。

BART の使用について

bart コマンドは、目録を作成および比較するために使用されます。このコマンドは、どのユーザーでも実行できます。ただし、ユーザーがファイルのカタログ化やモニターを行うことができるのは、自分がアクセス権を持っているファイルだけです。そのため、ユーザーやほとんどの役割は自分のホームディレクトリ内のファイルを有効にカタログ化できますが、root アカウントは、システムファイルを含むすべてのファイルをカタログ化できます。

BART のセキュリティー上の考慮事項

BART 目録およびレポートは、だれでも読み取ることができます。BART の出力に機密情報が含まれている可能性がある場合は、出力を保護するための適切な対策を取ってください。たとえば、アクセス権が制限された出力ファイルを生成するオプションや、出力ファイルを保護されたディレクトリ内に配置するオプションを使用します。

BART の使用

タスク	説明	手順
BART 目録を作成します。	システムにインストールされているファイルごとに情報の一覧表示を生成します。	31 ページの「制御目録を作成する方法」
カスタム BART 目録を作成します。	システムにインストールされている特定のファイルに関する情報の一覧を生成します。	33 ページの「目録をカスタマイズする方法」
BART 目録を比較します。	一定期間における同一システムの変化を比較するレポートを生成します。 または、1 つまたは複数のシステムを制御システムと比較するレポートを生成します。	34 ページの「一定期間内で同一システムの目録を比較する方法」 37 ページの「異なるシステムからの目録の比較方法」

タスク	説明	手順
(オプション) BART レポートをカスタマイズします。	次に示す方法のいずれかでカスタム BART レポートを生成します。 ■ 属性を指定します ■ 規則ファイルを使用する	39 ページの「ファイル属性を指定して BART レポートをカスタマイズする方法」 40 ページの「規則ファイルを使用して BART レポートをカスタマイズする方法」

▼ 制御目録を作成する方法

この手順では、比較用のベースライン (制御) 目録の作成方法について説明します。このタイプの目録は、中央のイメージから多数のシステムをインストールするときに使用してください。または、インストールが同一であることを確認したい場合は、このタイプの目録を使用して比較を実行してください。制御目録の詳細は、28 ページの「BART 目録」を参照してください。形式の規則を理解するには、例2-1「BART 目録の形式の説明」を参照してください。

注記 - ネットワーク化されたファイルシステムをカタログ化しようとしないでください。BART を使用して、ネットワーク化されたファイルシステムをモニターすると、ほとんど価値のない目録を生成するために大量のリソースが消費されます。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **サイトのセキュリティー要件を満たすように Oracle Solaris システムをカスタマイズしたあと、制御目録を作成し、その出力をファイルにリダイレクトします。**

```
# bart create options > control-manifest
```

`-R` 目録の検査対象としてルートディレクトリを指定します。規則で指定されるパスはすべて、このディレクトリからの相対パスとなるように変換されます。目録で報告されるパスはすべて、このディレクトリからの相対パスとなります。

`-I` カタログ化される個々のファイルの一覧 (コマンド行上で指定されるか、あるいは標準入力から読み取られる) を受け入れます。

`-r` この目録の規則ファイルの名前です。- 引数を指定すると、規則ファイルが標準入力から読み取られます。

-n　　ファイルリストに挙げられたすべての通常ファイルの署名を無効にします。このオプションは、パフォーマンスを上げる目的で使用できます。また、(システムログファイルの場合のように) ファイルリストの内容が変わる予定がある場合にも使用できます。

2. **目録の内容を確認します。**

   形式については、例2-1「BART 目録の形式の説明」を参照してください。

3. **(オプション) 目録を保護します。**

   システム目録を保護するための 1 つの方法として、システム目録を root アカウントだけがアクセスできるディレクトリ内に配置します。

```
# mkdir /var/adm/log/bartlogs
# chmod 700 /var/adm/log/bartlogs
# mv control-manifest /var/adm/log/bartlogs
```

   目録にわかりやすい名前をつけてください。たとえば、mach1-120313 のように、目録が作成されたシステム名と日付を使用します。

例 2-1　BART 目録の形式の説明

この例では、出力例のあとに目録の形式の説明が表示されています。

```
# bart create
! Version 1.1
! HASH SHA256
! Saturday, September 07, 2013 (22:22:27)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 1024 40755 user::rwx,group::r-x,mask:r-x,other:r-x
3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090 0 0
.
.
.
/zone D 512 40755 user::rwx group::r-x,mask:r-x,other:r-x 3f81e892
154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334 0 0
.
.
.
```

各目録は、ヘッダーとファイルエントリで構成されています。各ファイルエントリは、ファイルタイプに応じて 1 行になります。たとえば、前の出力にある各ファイルエントリの場合、タイプ F はファイルを指定し、タイプ D はディレクトリを指定します。また、サイズ、内容、ユーザー ID、グループ ID、およびアクセス権も表示されます。特殊文字を正しく処理するため、出力内のファイルエントリはエンコードされたバージョンのファイル名でソートされます。この結果、すべてのエントリがファイル名をキーにして昇順に並べられます。標準でないファイル名 (改行文字やタブ文字が埋め込まれたファイル名など) の場合はすべて、ソート前に非標準文字が引用符で囲まれます。

! で始まる行には、目録についてのメタデータが示されてます。目録バージョン行には、目録の仕様バージョンが示されます。ハッシュ行には、使用されたハッシュメカニズムが示されます。チェックサムとして使用される SHA256 ハッシュの詳細は、sha2(3EXT) のマニュアルページを参照してください。

日付行には、目録が作成された日付が日付形式で示されます。date(1) のマニュアルページを参照してください。一部の行は、目録比較ツールによって無視されます。無視される行には、メタデータ、空行、空白のみで構成された行、および # で始まるコメントが含まれます。

▼ 目録をカスタマイズする方法

目録は、次に示す方法のいずれかでカスタマイズできます。

- サブツリーを指定する

  個々のサブツリーの指定は、選択された重要なファイル (/etc ディレクトリ内のすべてのファイルなど) への変更をモニターするための効率的な方法です。

- ファイル名を指定する

  ファイル名の指定は、特に重要なファイル (データベースアプリケーションを構成および実行するファイルなど) をモニターするための効率的な方法です。

- 規則ファイルを使用する

  規則ファイルを使用して目録の作成と比較を行うと、複数のファイルまたはサブツリーの複数の属性を指定するという柔軟性が得られます。コマンド行からは、目録またはレポート内のすべてのファイルに適用されるグローバルな属性定義を指定できます。規則ファイルからは、グローバルには適用されない属性を指定できます。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **どのファイルのカタログ化とモニターを行うかを決定します。**

2. **次のオプションのいずれかを使用して、カスタムマニフェストを作成します。**

   - サブツリーを指定する。

     ```
     # bart create -R subtree
     ```

   - ファイル名 (1 つまたは複数) を指定する。

     ```
     # bart create -I filename...
     ```

     例:

     ```
     # bart create -I /etc/system /etc/passwd /etc/shadow
     ```

   - 規則ファイルを使用する。

     ```
     # bart create -r rules-file
     ```

3. **目録の内容を確認します。**

4. **(オプション) 目録を将来の使用のために保護されたディレクトリ内に保存します。**

   例については、ステップ 3の31 ページの「制御目録を作成する方法」を参照してください。

ヒント - 規則ファイルを使用した場合は、目録とともに規則ファイルを保存します。有効な比較のためには、この規則ファイルを使用して比較を実行する必要があります。

▼ 一定期間内で同一システムの目録を比較する方法

一定期間にわたって目録を比較することによって、壊れたファイルや異常なファイルを見つけたり、セキュリティー違反を検出したり、システム上のパフォーマンスの問題をトラブルシューティングしたりすることができます。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **システム上でモニターするファイルの制御目録を作成します。**

   ```
   # bart create -R /etc > control-manifest
   ```

2. **(オプション) 目録を将来の使用のために保護されたディレクトリ内に保存します。**

例については、ステップ 3の31 ページの「制御目録を作成する方法」を参照してください。

3. **あとで、制御目録と同一の目録を準備します。**

```
# bart create -R /etc > test-manifest
```

4. **2 番目の目録を保護します。**

```
# mv test-manifest /var/adm/log/bartlogs
```

5. **2 つの目録を比較します。**

同じコマンド行オプションと規則ファイルを使用して、それらを作成するために使用した目録を比較します。

```
# bart compare options control-manifest test-manifest > bart-report
```

6. **BART レポートを調べ、異常がないかを確認します。**

例 2-2 一定期間にわたる同じシステムに対するファイル変更の追跡

この例は、一定期間にわたって /etc ディレクトリの変更を追跡する方法を示しています。このタイプの比較を使用すると、攻撃されたシステム上の重要なファイルを見つけることができます。

■ 制御目録を作成します。

```
# cd /var/adm/logs/manifests
# bart create -R /etc > system1.control.090713
! Version 1.1
! HASH SHA256
! Saturday, September 07, 2013 (11:11:17)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/.cpr_config F 2236 100644 owner@:read_data/write_data/append_data/read_xattr/wr
ite_xattr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchr
```

```
onize:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:all
ow,everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4e271c59 0 0 3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
/.login F 1429 100644 owner@:read_data/write_data/append_data/read_xattr/write_x
attr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchronize
:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow,ev
eryone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4bf9d6d7 0 3 ff6251a473a53de68ce8b4036d0f569838cff107caf1dd9fd04701c48f09242e
.
.
.
```

■ あとで、同じコマンド行オプションを使用して、テスト目録を作成します。

```
# bart create -R /etc > system1.test.101013
Version 1.1
! HASH SHA256
! Monday, October 10, 2013 (10:10:17)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/.cpr_config F 2236 100644 owner@:read_data/write_data/append_data/read_xattr/wr
ite_xattr/read_attributes/write_attributes/read_acl/write_acl/write_owner/synchr
onize:allow,group@:read_data/read_xattr/read_attributes/read_acl/synchronize:all
ow,everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
4e271c59 0 0 3ebc418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
.
.
.
```

■ 目録を比較します。

```
# bart compare system1.control.090713 system1.test.101013
/security/audit_class
```

```
mtime  4f272f59
```

この出力は、audit_class ファイル上の変更時間が、制御目録が作成されたあとに変化したことを示しています。この変化が予期しないものである場合は、さらに調査を行うことができます。

▼ 異なるシステムからの目録の比較方法

異なるシステムの目録を比較することによって、システムが同様にインストールされているかどうか、または同期してアップグレードされているかどうかを判定できます。たとえば、システムを特定のセキュリティーターゲットにカスタマイズした場合は、この比較によって、そのセキュリティーターゲットを表す目録と、その他のシステムの目録の間の相違がすべて見つかります。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **制御目録を作成します。**

   ```
   # bart create options > control-manifest
   ```

   これらのオプションについては、bart(1M) のマニュアルページを参照してください。

2. **(オプション) 目録を将来の使用のために保護されたディレクトリ内に保存します。**

   例については、ステップ 3の31 ページの「制御目録を作成する方法」を参照してください。

3. **テストシステム上で、同じ bart オプションを使用して目録を作成します。**

   ```
   # bart create options > test1-manifest
   ```

4. **(オプション) 目録を将来の使用のために保護されたディレクトリ内に保存します。**

5. **比較を実行するには、目録を中央の場所にコピーします。**

   例:

   ```
   # cp control-manifest /net/test-server/var/adm/logs/bartlogs
   ```

   テストシステムが NFS マウントしたシステムでない場合は、sftp または別の信頼できる手段を使用して、目録を中央の場所にコピーします。

6. **目録を比較し、その出力をファイルにリダイレクトします。**

   ```
   # bart compare control-manifest test1-manifest > test1.report
   ```

7. **BART レポートを調べ、異常がないかを確認します。**

例 2-3 /usr/bin ディレクトリ内の疑わしいファイルの識別

この例では、2 つのシステム上の /usr/bin ディレクトリの内容を比較します。

- 制御目録を作成します。

```
# bart create -R /usr/bin > control-manifest.090713
! Version 1.1
! HASH SHA256
! Saturday, September 07, 2013 (11:11:17)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/2to3 F 105 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribut
es/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:read
_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:re
ad_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4bf9d261 0
2 154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334
/7z F 509220 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribu
tes/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:rea
d_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:r
ead_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4dadc48a 0
2 3ecd418eb5be3729ffe7e54053be2d33ee884205502c81ae9689cd8cca5b0090
...
```

- 制御システムと比較する各システムで同一の目録を作成します。

```
# bart create -R /usr/bin > system2-manifest.101013
! Version 1.1
! HASH SHA256
! Monday, October 10, 2013 (10:10:22)
# Format:
#fname D size mode acl dirmtime uid gid
```

```
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/2to3 F 105 100555 owner@:read_data/read_xattr/write_xattr/execute/read_attribut
es/write_attributes/read_acl/write_acl/write_owner/synchronize:allow,group@:read
_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow,everyone@:re
ad_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow 4bf9d261 0
2 154de3e7bdfd0d57a074c9fae0896a9e2e04bebfe5e872d273b063319e57f334
...
```

- これらの目録を同じ場所にコピーします。

```
# cp control-manifest.090713 /net/system2.central/bart/manifests
```

- 目録を比較します。

```
# bart compare control-manifest.090713 system2.test.101013 > system2.report
/su:
gid  control:3  test:1
/ypcat:
mtime  control:3fd72511  test:3fd9eb23
```

この出力は、/usr/bin ディレクトリ内の su ファイルのグループ ID が、制御システムのグループ ID と同じでないことを示しています。この情報は、テストシステム上に別のバージョンのソフトウェアがインストールされたことを示している可能性があります。GID が変更されているため、だれかがファイルを改ざんしたことが可能性として考えられます。

▼ ファイル属性を指定して BART レポートをカスタマイズする方法

この手順は、既存の目録の出力を特定のファイル属性に関してフィルタリングするために役立ちます。

始める前に root 役割になる必要があります。詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **どのファイル属性をチェックするかを決定します。**

2. **チェックされるファイル属性を含む 2 つの目録を比較します。**

   例:

   ```
   # bart compare -i lnmtime,mtime control-manifest.121513 \
     test-manifest.010514 > bart.report.010514
   ```

   各ファイル属性を区切るには、コマンド行構文でコンマを使用します。

3. **BART レポートを調べ、異常がないかを確認します。**

▼ 規則ファイルを使用して BART レポートをカスタマイズする方法

規則ファイルを使用すると、目的とする特定のファイルやファイル属性に関して BART 目録をカスタマイズできます。デフォルトの BART 目録に対して異なる規則ファイルを使用すると、同じ目録に対して異なる比較を実行できます。

始める前に `root` 役割になる必要があります。詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「割り当てられている管理権利の使用」を参照してください。

1. **どのファイルおよびファイル属性をモニターするかを決定します。**

2. **適切な指示語を含む規則ファイルを作成します。**

3. **作成した規則ファイルを使用して制御目録を作成します。**

   ```
   # bart create -r myrules1-file > control-manifest
   ```

4. **(オプション) 目録を将来の使用のために保護されたディレクトリ内に保存します。**

   例については、ステップ 3の31 ページの「制御目録を作成する方法」を参照してください。

5. **別のシステム上で同一の目録を作成します。これをあとで、またはその両方で行います。**

   ```
   # bart create -r myrules1-file > test-manifest
   ```

6. **同じ規則ファイルを使用して目録を比較します。**

```
# bart compare -r myrules1-file control-manifest test-manifest > bart.report
```

7. **BART レポートを調べ、異常がないかを確認します。**

例 2-4 規則ファイルを使用した BART 目録および比較レポートのカスタマイズ

次の規則ファイルは、bart create コマンドに、/usr/bin ディレクトリ内のファイルのすべての属性を一覧表示するよう指示します。さらに、この規則ファイルは bart compare コマンドに、同じディレクトリ内のサイズと内容の変更のみを報告するよう指示します。

```
# Check size and content changes in the /usr/bin directory.
# This rules file only checks size and content changes.
# See rules file example.

IGNORE all
CHECK size contents
/usr/bin
```

- 作成した規則ファイルを使用して制御目録を作成します。

```
# bart create -r usrbinrules.txt > usr_bin.control-manifest.121013
```

- /usr/bin ディレクトリへの変更をモニターする場合は常に、同一の目録を準備します。

```
# bart create -r usrbinrules.txt > usr_bin.test-manifest.121113
```

- 同じ規則ファイルを使用して目録を比較します。

```
# bart compare -r usrbinrules.txt usr_bin.control-manifest.121013 \
usr_bin.test-manifest.121113
```

- bart compare コマンドの出力を調べます。

```
 /usr/bin/gunzip:  add
/usr/bin/ypcat:
delete
```

前の出力は、/usr/bin/ypcat ファイルが削除されたこと、および /usr/bin/gunzip ファイルが追加されたことを示しています。

BART 目録、規則ファイル、およびレポート

このセクションでは、BART が使用して作成するファイルの形式について説明します。

BART 目録のファイル形式

目録ファイルの各エントリは、ファイルタイプに応じて単一の行に示されます。各エントリは、ファイルの名前である *fname* で始まります。ファイル名に使用された特殊文字の解析によって起きる問題を防ぐため、ファイル名はエンコードされます。詳細は、43 ページの「BART 規則ファイルの書式」を参照してください。

後続のフィールドは、次のファイル属性を表します。

type
ファイルの種類であり、次のような値となります。
- B: ブロックデバイスノード
- C: キャラクタデバイスノード
- D: ディレクトリ
- F: ファイル
- L: シンボリックリンク
- P: パイプ
- S: ソケット

size
ファイルサイズ (バイト)。

mode
ファイルのアクセス権を示す 8 進の数値。

acl
ファイルの ACL 属性。ACL 属性を持つファイルの場合、`acltotext()` の出力が入ります。

uid
このエントリの所有者の、数値で示したユーザー ID。

gid
このエントリの所有者の、数値で示したグループ ID。

dirmtime
ディレクトリに最後に変化が起きた時点。1970 年 1 月 1 日の 00:00:00 UTC から数えた秒数で示されます。

lnmtime
リンクに最後に変化が起きた時点。1970 年 1 月 1 日の 00:00:00 UTC から数えた秒数で示されます。

mtime
ファイルに最後に変化が起きた時点。1970 年 1 月 1 日の 00:00:00 UTC から数えた秒数で示されます。

contents
ファイルのチェックサム値。この属性が指定されるのは通常ファイルのみです。コンテキストのチェックを無効にした場合や、チェックサムを計算できない場合、このフィールドの値は - になります。

dest シンボリックリンクのリンク先。

devnode デバイスノードの値。この属性が使用されるのは、キャラクタデバイスファイルとブロックデバイスファイルのみです。

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

BART 規則ファイルの書式

規則ファイルは、どのファイルを目録に含めるかや、どのファイル属性を目録またはレポートに含めるかを指定する行で構成されたテキストファイルです。#、空の行、空白を含む行は、ツールが無視します。

入力ファイルには、次に示す 3 種類の指示語が指定されます。

- オプションのパターンマッチング修飾子を持つ subtree 指示語
- CHECK 指示語
- IGNORE 指示語

例 2-5 規則ファイルの書式

```
<Global CHECK/IGNORE Directives>
<subtree1> [pattern1..]
<IGNORE/CHECK Directives for subtree1>

<subtree2> [pattern2..]
<subtree3> [pattern3..]
<subtree4> [pattern4..]
<IGNORE/CHECK Directives for subtree2, subtree3, subtree4>
```

注記 - すべての指示語が順番に読み取られます。あとの指示語で前の指示語をオーバーライドすることができます。

subtree 指示語は絶対パス名で始まり、そのあとに 0 個以上のパターンマッチング文が続く*必要があります*。

BART 規則ファイルの属性

CHECK 文と IGNORE 文は、どのファイル属性を追跡または無視するかを定義します。各目録を開始するメタデータによって、ファイルタイプごとの属性の*キーワード*が一覧表示されます。例 2-1「BART 目録の形式の説明」を参照してください。

all キーワードは、すべてのファイル属性を示します。

BART 引用構文

BART が規則ファイルに使用する記述言語は、標準に準拠していないファイル名を表現する標準の UNIX 引用構文です。埋め込まれたタブ、スペース、改行、特殊文字は、ツールがファイル名を読み取ることができるようにそれらの 8 進形式にエンコードされます。この変動的な引用構文では、埋め込みのキャリッジリターンを含むファイル名などがコマンドパイプラインで正しく処理されません。規則記述言語を使用することで、シェル構文だけでは表現が難しい効率の悪い複雑なファイル名フィルタリング基準を表現できます。

詳細は、bart_rules(4) のマニュアルページを参照してください。

BART レポート

デフォルトモードでは、BART レポートは、変更されたディレクトリのタイムスタンプ (dirmtime) を除く、システム上にインストールされたすべてのファイルをチェックします。

```
CHECK all
IGNORE dirmtime
```

規則ファイルを指定すると、汎用指示語である CHECK all と IGNORE dirmtime がこの順で規則ファイルの先頭に自動的に付けられます。

BART の出力

次の終了ステータスが返されます。

0	成功
1	ファイル処理時の致命的でないエラー (アクセス権問題など)
>1	致命的なエラー (無効なコマンド行オプションなど)

レポーティングメカニズムとして、詳細出力と、プログラムを考慮した出力の 2 種類を利用できます。

- 詳細出力はデフォルトの出力であり、地域対応化され、複数の行にわたって表示されます。詳細出力は国際化されたもので、ユーザーが内容を読み取ることができます。bart

compare コマンドは、2 つのシステム目録を比較する時に、ファイル間の相違を示すリストを生成します。

出力の構造は次のとおりです。

filename attribute control:control-val test:test-val

filename	制御目録とテスト目録で相違があるファイルの名前。
attribute	比較される目録間で異なるファイル属性の名前。*test-val* の前に *control-val* が来ます。同じファイルの複数の属性に相違が見つかった場合、別々の行にそれぞれの相違が示されます。

次に、/etc/passwd ファイルの属性の違いの例を示します。この出力から、size、mtime、および contents 属性に変化があったことがわかります。

```
/etc/passwd:
size control:74 test:81
mtime control:3c165879 test:3c165979
contents control:daca28ae0de97afd7a6b91fde8d57afa
test:84b2b32c4165887355317207b48a6ec7
```

- bart compare コマンドの -p オプションを使用すると、プログラムを考慮した出力が生成されます。この出力は、プログラムでの操作に適しています。

  出力の構造は次のとおりです。

  filename attribute control-val test-val [*attribute control-val test-val*]*

filename	デフォルト形式における *filename* 属性に同じ
attribute control-val test-val	各ファイルの制御目録とテスト目録間で異なるファイル属性の説明

bart コマンドでサポートされる属性の一覧については、43 ページの「BART 規則ファイルの属性」を参照してください。

詳細は、bart(1M) のマニュアルページを参照してください。

セキュリティー用語集

アクセス制御リスト (ACL) アクセス制御リスト (ACL) を使用すると、従来の UNIX ファイル保護よりもきめ細かな方法でファイルセキュリティーを確立できます。たとえば、特定のファイルにグループ読み取り権を設定し、そのグループ内の 1 人のメンバーだけにそのファイルへの書き込み権を与えることが可能です。

アプリケーションサーバー ネットワークアプリケーションサーバーを参照してください。

アルゴリズム 暗号化アルゴリズム。これは、入力を暗号化 (ハッシング) する既成の再帰的な計算手続きです。

暗号化アルゴリズム アルゴリズムを参照してください。

暗号化フレームワークにおけるポリシー Oracle Solaris の暗号化フレームワーク機能では、ポリシーは既存の暗号化メカニズムの無効化です。無効に設定されたメカニズムは使用できなくなります。暗号化フレームワークにおけるポリシーにより、プロバイダ (DES など) からの特定のメカニズム (`CKM_DES_CBC`など) を使用できなくなることがあります。

インスタンス 主体名の 2 番目の部分。インスタンスは、主体の主ノード指定します。サービス主体の場合、インスタンスは必ず指定する必要があります。`host/central.example.com` にあるように、インスタンスはホストの完全修飾ドメイン名です。ユーザー主体の場合、インスタンスは省略することができます。ただし、`jdoe` と `jdoe/admin` は、一意の主体です。プライマリ、主体名、サービス主体、ユーザー主体も参照してください。

オーセンティケータ オーセンティケータは、KDC にチケットを要求するときおよびサーバーにサービスを要求するときに、クライアントから渡されます。オーセンティケータには、クライアントとサーバーだけが知っているセッション鍵を使用して生成された情報が含まれます。オーセンティケータは、最新の識別として検査され、そのトランザクションが安全であることを示します。これをチケットとともに使用すると、ユーザー主体を認証できます。オーセンティケータには、ユーザーの主体名、ユーザーのホストの IP アドレス、タイムスタンプが含まれます。チケットとは異なり、オーセンティケータは一度しか使用できません。通常、サービスへのアクセスが要求されたときに使用されます。オーセンティケータは、そのクライアントとそのサーバーのセッション鍵を使用して暗号化されます。

鍵 1. 一般には、次に示す 2 種類の主要鍵のどちらか一方です。

- *対称鍵* – 復号化鍵とまったく同じ暗号化鍵。対称鍵はファイルの暗号化に使用されます。

- *非対称鍵*または*公開鍵* – Diffie-Hellman や RSA などの公開鍵アルゴリズムで使用される鍵。公開鍵には、1 人のユーザーしか知らない非公開鍵、サーバーまたは一般リソースによって使用される公開鍵、およびこれらの 2 つを組み合わせた公開鍵と非公開鍵のペアがあります。非公開鍵は、「秘密鍵」とも呼ばれます。公開鍵は、「共有鍵」や「共通鍵」とも呼ばれます。

2. キータブファイルのエントリ (主体名)。キータブファイルも参照してください。
3. Kerberos では暗号化鍵であり、次の 3 種類があります。

- 「非公開鍵」– 主体と KDC によって共有される暗号化鍵。システムの外部に配布されます。非公開鍵も参照してください。
- 「サービス鍵」– 非公開鍵と同じ目的で使用されますが、この鍵はサーバーとサービスによって使用されます。サービス鍵も参照してください。
- 「セッション鍵」 – 一時的な暗号化鍵。2 つの主体の間で使用され、その有効期限は 1 つのログインセッションの期間に制限されます。セッション鍵も参照してください。

仮想プライベートネットワーク (VPN) 暗号化とトンネルを使用して、セキュアな通信を提供するネットワーク。公開ネットワークを通してユーザーを接続します。

関係 `kdc.conf` または `krb5.conf` ファイルに定義される構成変数または関係の 1 つ。

監査トレール すべてのホストから収集した一連の監査ファイル。

監査ファイル バイナリ形式の監査ログ。監査ファイルは、監査ファイルシステム内に個別に格納されます。

キーストア キーストアは、アプリケーションによる取得のために、パスワード、パスフレーズ、証明書、およびその他の認証オブジェクトを保持します。キーストアはテクノロジ固有にすることも、複数のアプリケーションで使用される場所にすることもできます。

キータブファイル 1 つまたは複数の鍵 (主体) が含まれるキーテーブル。ホストまたはサービスとキータブファイルとの関係は、ユーザーとパスワードの関係と似ています。

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。

機密性 プライバシを参照してください。

強化 ホストが本来抱えるセキュリティー上の脆弱性を解決するためにオペレーティングシステムのデフォルト構成を変更すること。

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。

クライアント

狭義では、rlogin を使用するアプリケーションなど、ユーザーの代わりにネットワークサービスを使用するプロセスを指します。サーバー自身が他のサーバーやサービスのクライアントになる場合もあります。

広義では、a) Kerberos 資格を受け取り、b) サーバーから提供されたサービスを利用するホストを指します。

広義では、サービスを使用する主体を指します。

クライアント主体

(RPCSEC_GSS API) RPCSEC_GSS で保護されたネットワークサービスを使用するクライアント (ユーザーまたはアプリケーション)。クライアント主体名は、rpc_gss_principal_t 構造体の形式で格納されます。

クロックスキュー

Kerberos 認証システムに参加しているすべてのホスト上の内部システムクロックに許容できる最大時間。参加しているホスト間でクロックスキューを超過すると、要求が拒否されます。クロックスキューは、krb5.conf ファイルに指定できます。

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。ユーザー権利管理とプロセス権利管理を使用することで、組織はスーパーユーザーの特権を分割し、それらをユーザーまたは役割に割り当てることができます。Oracle Solaris の権利は、カーネル特権、承認、および特定の UID または GID で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で収集されます。

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を指定したコマンド、および補助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、Oracle Solaris の有効なポリシーは solaris です。solaris ポリシーは、特権と拡張された特権ポリシー、承認、および setuid セキュリティー属性を認識します。

公開オブジェクト

root ユーザーによって所有され、すべてのユーザーが読み取ることのできるファイル。たとえば、/etc ディレクトリ内のファイルです。

公開鍵技術のポリシー

鍵管理フレームワーク (KMF) におけるポリシーは、証明書の使用を管理します。KMF ポリシーデータベースを使えば、KMF ライブラリによって管理される鍵や証明書の使用に、制約を設けることができます。

公開鍵の暗号化

暗号化スキームの 1 つ。各ユーザーが 1 つの公開鍵と 1 つの非公開鍵を所有します。公開鍵の暗号化では、送信者は受信者の公開鍵を使用してメッセージを暗号化し、受信者は非公開鍵を使用してそれを復号化します。Kerberos サービスは非公開鍵システムです。非公開鍵の暗号化も参照してください。

更新可能チケット

有効期限の長いチケットは、セキュリティーを低下させることがあるため、「更新可能」チケットに指定することができます。更新可能チケットには 2 つの有効期限があります。a) チケットの

現在のインスタンスの有効期限と、b) 任意のチケットの最長有効期限です。クライアントがチケットの使用を継続するときは、最初の有効期限が切れる前にチケットの有効期限を更新します。たとえば、すべてのチケットの最長有効期限が 10 時間のときに、あるチケットが 1 時間だけ有効だとします。このチケットを保持するクライアントが 1 時間を超えて使用する場合は、チケットの有効期限を更新する必要があります。チケットが最長有効期限に達すると、チケットの有効期限が自動的に切れ、それ以上更新できなくなります。

コンシューマ Oracle Solaris の暗号化フレームワーク機能では、コンシューマはプロバイダが提供する暗号化サービスのユーザー。コンシューマになりえるものとして、アプリケーション、エンドユーザー、カーネル処理などが挙げられます。Kerberos、IKE、IPsec などはコンシューマの例です。プロバイダの例は、プロバイダを参照してください。

サーバー ネットワーククライアントにリソースを提供する主体。たとえば、システム `central.example.com` に `ssh` で接続する場合、そのシステムが `ssh` サービスを提供するサーバーになります。サービス主体も参照してください。

サーバー主体 (RPCSEC_GSS API) サービスを提供する主体。サーバー主体は、*service*@*host* という書式で ASCII 文字列として格納されます。クライアント主体も参照してください。

サービス 1. ネットワーククライアントに提供されるリソース。多くの場合、複数のサーバーから提供されます。たとえば、マシン `central.example.com` に `rlogin` で接続する場合、そのマシンが `rlogin` サービスを提供するサーバーになります。

2. 認証以外の保護レベルを提供するセキュリティーサービス (整合性またはプライバシ)。整合性とプライバシも参照してください。

サービス鍵 サービス主体と KDC によって共有される暗号化鍵。システムの外部に配布されます。鍵も参照してください。

サービス主体 1 つまたは複数のサービスに Kerberos 認証を提供する主体。サービス主体では、プライマリ名はサービス名 (`ftp` など) で、インスタンスはサービスを提供するシステムの完全指定ホスト名になります。ホスト主体、ユーザー主体も参照してください。

最小化 サーバーを稼働させる上で必要な最小限のオペレーティングシステムをインストールすること。サーバーの処理に直接影関係がないソフトウェアはすべて、インストールされないか、あるいはインストール後削除されます。

最少特権 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マシン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。

最少特権の原則 最少特権を参照してください。

再認証 あるコンピュータ操作を実行する際にパスワードの提供を求めること。一般に、sudo 操作では再認証が要求されます。認証権利プロファイルに、再認証を要求する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。

シード 乱数生成のスターター (元になる値)。この値から生成が開始されます。このスターターがランダムソースから生じる場合、このシードは「ランダムシード」と呼ばれます。

資格 チケットと照合セッション鍵を含む情報パッケージ。主体の識別情報を認証するときに使用します。チケットとセッション鍵も参照してください。

資格キャッシュ KDC から受信した資格を含むストレージ領域。通常はファイルです。

主体 1. ネットワーク通信に参加する、一意の名前を持つ「クライアントまたはユーザー」あるいは「サーバーまたはサービス」のインスタンス。Kerberos トランザクションでは、主体 (サービス主体とユーザー主体) 間、または主体と KDC の間で対話が行われます。つまり、主体とは、Kerberos がチケットを割り当てることができる一意のエンティティーのことです。主体名、サービス主体、ユーザー主体も参照してください。

2. (RPCSEC_GSS API) クライアント主体、サーバー主体を参照してください。

主体名 1. 主体の名前。書式は、*primary/instance@REALM*。インスタンス、プライマリ、レルムも参照してください。

2. (RPCSEC_GSS API) クライアント主体、サーバー主体を参照してください。

承認 1. Kerberos では、主体がサービスを使用できるかどうか、主体がアクセスできるオブジェクト、各オブジェクトに許可するアクセスの種類を決定するプロセス。

2. ユーザー権利管理で、役割またはユーザーに割り当てることができる権利。権利プロファイルに埋め込まれていることもあります。承認が与えられていない操作クラスは、セキュリティーポリシーによって拒否されます。承認は、カーネル内ではなく、ユーザーアプリケーションのレベルで適用されます。

初期チケット 直接発行されるチケット (既存のチケット許可チケットは使用されない)。パスワードを変更するアプリケーションなどの一部のサービスでは、クライアントが非公開鍵を知っていることを確認するために、「初期」と指定されたチケットを要求することができます。初期チケットを使用した検査は、クライアントが最近認証されたことを証明するときに重要になります。チケット許可チケットの場合は、取得してから時間が経過していることがあります。

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と判定されたユーザー。一般に、管理者は信頼できるユーザー用のログインを最初に作成し、それらのユーザーの信頼と機能のレベルに一致する管理者権限を割り当てます。これらのユーザーは、システムの構成と管理を支援します。*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、マシン管理のために 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。

スキャンエンジン 既知のウイルスがないかどうかファイルを検査する、外部ホスト上に存在するサードパーティーのアプリケーション。

スレーブ KDC マスター KDC のコピー。マスター KDC のほとんどの機能を実行できます。各レルムには通常、いくつかのスレーブ KDC (と 1 つのマスター KDC) を配置します。KDC、マスター KDCも参照してください。

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。

整合性 ユーザー認証に加えて、暗号チェックサムを使用して、転送されたデータの有効性を提供するセキュリティーサービス。認証、プライバシも参照してください。

責務分離 最少特権の概念の一部。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。

セキュリティーサービス サービスを参照してください。

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ます。セキュリティー属性を使用すると、スーパーユーザー以外のユーザーでも管理コマンドを実行できます。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 `setuid root` プログラムの代わりにカーネル特権とその他の権利が使用されます。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。

セキュリティーフレーバ フレーバを参照してください。

セキュリティーポリシー ポリシーを参照してください。

セキュリティーメカニズム メカニズムを参照してください。

セッション鍵 認証サービスまたはチケット認可サービスによって生成される鍵。セッション鍵は、クライアントとサービス間のトランザクションのセキュリティーを保護するために生成されます。セッション鍵の有効期限は、単一のログインセッションに制限されます。鍵も参照してください。

ソフトウェアプロバイダ Oracle Solaris の暗号化フレームワーク機能では、暗号化サービスを提供するカーネルソフトウェアモジュールまたは PKCS #11 ライブラリ。プロバイダも参照してください。

ダイジェスト メッセージダイジェストを参照してください。

単一システムイメージ 単一システムイメージは、同じネームサービスを使用する一連の検査対象システムを記述するために、Oracle Solaris 監査で使用されます。これらのシステムは監査レコードを中央の監査

サーバーに送信しますが、その監査サーバー上では、それらのレコードがまるで 1 つのシステムからやってきたかのように、レコードの比較を行えます。

遅延チケット 遅延チケットは、作成されても指定された時点まで有効になりません。このようなチケットは、夜遅く実行するバッチジョブなどのために効果的です。そのチケットは盗まれても、バッチジョブが実行されるまで使用できないためです。遅延チケットは、無効チケットとして発行され、a) 開始時間を過ぎて、b) クライアントが KDC による検査を要求したときに有効になります。遅延チケットは通常、チケット認可チケットの有効期限まで有効です。ただし、その遅延チケットが「更新可能」と指定されている場合、その有効期限は通常、チケット認可チケットの有効期限に設定されます。無効チケット、更新可能チケットも参照してください。

チケット ユーザーの識別情報をサーバーやサービスに安全に渡すために使用される情報パケット。チケットは、単一クライアントと特定サーバー上の特定サービスだけに有効です。チケットには、サービスの主体名、ユーザーの主体名、ユーザーのホストの IP アドレス、タイムスタンプ、チケットの有効期限を定義する値などが含まれます。チケットは、クライアントとサービスによって使用されるランダムセッション鍵を使用して作成されます。チケットは、作成されてから有効期限まで再使用できます。チケットは、最新のオーセンティケータとともに提示されなければ、クライアントの認証に使用することができません。オーセンティケータ、資格、サービス、セッション鍵も参照してください。

チケットファイル 資格キャッシュを参照してください。

デバイスの割り当て ユーザーレベルでのデバイス保護。デバイス割り当ては、一度に 1 人のユーザーだけが使用できるようにデバイスを設定する作業です。デバイスデータは、デバイスが再使用される前に消去されます。誰にデバイス割り当てを許可するかは、承認を使用して制限できます。

デバイスポリシー カーネルレベルでのデバイス保護。デバイスポリシーは、2 つの特権セットとしてデバイスに実装されます。この 1 つはデバイスに対する読み取り権を制御し、もう 1 つはデバイスに対する書き込み権を制御します。ポリシーも参照してください。

転送可能チケット チケットの 1 つ。クライアントがリモートホスト上のチケットを要求するときに使用できます。このチケットを使用すれば、リモートホスト上で完全な認証プロセスを実行する必要がありません。たとえば、ユーザー `david` がユーザー `jennifer` のマシンで転送可能チケットを取得した場合、`david` は自分のマシンにログインするときに新しいチケットを取得する必要はありません (ふたたび認証を受けることもない)。プロキシ可能チケットも参照してください。

同期監査イベント 監査イベントの大半を占めます。これらのイベントは、システムのプロセスに関連付けられています。失敗したログインなど、あるプロセスに関連付けられた、ユーザーに起因しないイベントは、同期イベントです。

特権 1. 一般に、コンピュータシステム上で標準ユーザーの権限を超える操作を実行するための権限または機能。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たユーザーまたはアプリケーションです。

2. Oracle Solaris システム内の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*または*カーネル特権*とも呼ばれます。特権の詳細は、`privileges`(5) 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 自分に割り当てられた権利 (デフォルトをオーバーライドする権利を含む) によって許可されるリソースの範囲外にあるリソースにアクセスすること。その結果、プロセスは未承認の操作を実行できることになります。

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 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。特権モデルでは、プロセスの実行に特権が必要です。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。

特権ユーザー コンピュータシステム上で標準ユーザーの権利を超える権利を割り当てられたユーザー。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`(5) のマニュアルページを参照してください。

認証 特定の主体の識別情報を検証するプロセス。

認証権利プロファイル 割り当てられたユーザーまたは役割に対して、プロファイルの操作を実行する前にパスワードの入力を要求する権利プロファイル。この動作は、`sudo` の動作に似ています。パスワードの有効期間は構成可能です。

ネームサービススコープ 特定の役割が操作を許可されている適用範囲。つまり、NIS LDAP などの指定されたネームサービスからサービスを受ける個々のホストまたはすべてのホスト。

ネットワークアプリケーションサーバー ネットワークアプリケーションを提供するサーバー (`ftp` など)。レルムは、複数のネットワークアプリケーションサーバーで構成されます。

ネットワークポリシー ネットワークトラフィックを保護するためにネットワークユーティリティーで行われる設定。ネットワークセキュリティーについては、『Oracle Solaris 11.2 でのネットワークのセキュリティー保護 』を参照してください。

ハードウェアプロバイダ Oracle Solaris の暗号化フレームワーク機能では、デバイスドライバとそのハードウェアアクセラレータを指します。ハードウェアプロバイダを使用すると、コンピュータシステムから負荷の高

い暗号化処理を解放され、その分 CPU リソースをほかの用途に充てることができます。プロバイダも参照してください。

パスフレーズ 非公開鍵がパスフレーズユーザーによって作成されたことを検証するために使用されるフレーズ。望ましいパスフレーズは、10 - 30 文字の長さで英数字が混在しており、単純な文や名前を避けたものです。通信の暗号化と復号化を行う非公開鍵の使用を認証するため、パスフレーズの入力を求めるメッセージが表示されます。

非公開鍵 各ユーザー (主体) に与えられ、主体のユーザーと KDC だけが知っている鍵。ユーザー主体の場合、鍵はユーザーのパスワードに基づいています。鍵も参照してください。

非公開鍵の暗号化 非公開鍵の暗号化では、送信者と受信者は同じ暗号化鍵を使用します。公開鍵の暗号化も参照してください。

非同期監査イベント 非同期イベントは、システムイベントの内の少数です。これらのイベントは、プロセスに関連付けられていないため、ブロックした後に起動できるプロセスはありません。たとえば、システムの初期ブートや PROM の開始および終了のイベントは、非同期イベントです。

秘密鍵 非公開鍵を参照してください。

プライバシ セキュリティーサービスの 1 つ。送信データを送信前に暗号化します。プライバシには、データの整合性とユーザー認証も含まれます。認証、整合性、サービスも参照してください。

プライマリ 主体名の最初の部分。インスタンス、主体名、レルムも参照してください。

フレーバ 従来は、「セキュリティーフレーバ」と「認証フレーバ」は、認証のタイプ (AUTH_UNIX、AUTH_DES、AUTH_KERB) を指すフレーバとして、同じ意味を持っていました。RPCSEC_GSS もセキュリティーフレーバですが、これは認証に加えて、整合性とプライバシのサービスも提供します。

プロキシ可能チケット クライアントに代わってクライアント操作を行うためにサービスによって使用されるチケット。このことを、サービスがクライアントのプロキシとして動作するといいます。サービスは、チケットを使用して、クライアントの識別情報を所有できます。このサービスは、プロキシ可能チケットを使用して、ほかのサービスへのサービスチケットを取得できますが、チケット認可チケットは取得できません。転送可能チケットと異なり、プロキシ可能チケットは単一の操作に対してだけ有効です。転送可能チケットも参照してください。

プロバイダ Oracle Solaris の暗号化フレームワーク機能では、コンシューマに提供される暗号化サービス。プロバイダには、PKCS #11 ライブラリ、カーネル暗号化モジュール、ハードウェアアクセラレータなどがあります。プロバイダは暗号化フレームワークに結合 (プラグイン) されるため、*プラグイン*とも呼ばれます。コンシューマの例は、コンシューマを参照してください。

プロファイルシェル 権利管理で、役割 (またはユーザー) が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コマンド行から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に対応しています (`bash` の `pfbash` バージョンなど)。

ホスト ネットワークを通じてアクセス可能なシステム。

ホスト主体 サービス主体のインスタンスの 1 つ (プライマリ名は `host`)。さまざまなネットワークサービス (`ftp`、`rcp`、`rlogin` など) を提供するために設定します。`host/central.example.com@EXAMPLE.COM` はホスト主体の例です。サーバー主体も参照してください。

ポリシー 一般には、意思やアクションに影響を与えたり、これらを決定したりする計画や手続き。コンピュータシステムでは、多くの場合セキュリティーポリシーを指します。実際のサイトのセキュリティーポリシーは、処理される情報の重要度や未承認アクセスから情報を保護する手段を定義する規則セットです。たとえば、セキュリティーポリシーで、システムの監査、使用するデバイスの割り当て、6 週ごとのパスワード変更といったことを設定できます。

Oracle Solaris OS の特定の領域におけるポリシーの実装については、audit policy、暗号化フレームワークにおけるポリシー、デバイスポリシー、Kerberos ポリシー、password policy、および権利ポリシーを参照してください。

マスター KDC 各レルムのメイン KDC。Kerberos 管理サーバー `kadmind` と、認証とチケット認可デーモン `krb5kdc` で構成されます。レルムごとに、1 つ以上のマスター KDC を割り当てる必要があります。また、クライアントに認証サービスを提供する複製 (スレーブ) KDC を任意の数だけ割り当てることができます。

無効チケット まだ使用可能になっていない遅延チケット。無効チケットは、有効になるまでアプリケーションサーバーから拒否されます。これを有効にするには、開始時期が過ぎたあと、TGS 要求で `VALIDATE` フラグをオンにしてクライアントがこのチケットを KDC に提示する必要があります。遅延チケットも参照してください。

メカニズム 1. データの認証や機密性を実現するための暗号化技術を指定するソフトウェアパッケージ。たとえば、Kerberos V5、Diffie-Hellman 公開鍵など。

2. Oracle Solaris の暗号化フレームワーク機能では、特定の目的のためのアルゴリズムの実装。たとえば、認証に適用される DES メカニズム (CKM_DES_MAC など) は、暗号化に適用されるメカニズム (CKM_DES_CBC_PAD) とは別です。

メッセージダイジェスト メッセージダイジェストは、メッセージから計算されるハッシュ値です。ハッシュ値によってメッセージはほぼ一意に識別されます。ダイジェストは、ファイルの整合性を検証するのに便利です。

メッセージ認証コード (MAC) データの整合性を保証し、データの出所を明らかにするコード。MAC は盗聴行為には対応できません。

役割 特権付きアプリケーションを実行するための特別な ID。割り当てられたユーザーだけが引き受けられます。

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。

ユーザー主体 特定のユーザーに属する主体。ユーザー主体のプライマリ名はユーザー名であり、その省略可能なインスタンスは対応する資格の使用目的を説明するために使われる名前です (`jdoe`、`jdoe/admin` など)。「ユーザーインスタンス」とも呼ばれます。サービス主体も参照してください。

ユーザーに起因しない監査イベント
開始した人を特定できない監査イベント。AUE_BOOT イベントなど。

レルム
1. 1 つの Kerberos データベースといくつかの鍵配布センター (KDC) を配置した論理ネットワーク。

2. 主体名の 3 番目の部分。主体名が `jdoe/admin@CORP.EXAMPLE.COM` の場合、レルムは `CORP.EXAMPLE.COM` です。主体名も参照してください。

admin 主体
username`/admin` という形式 (`jdoe/admin` など) の名前を持つユーザー主体。通常のユーザー主体より多くの特権 (ポリシーの変更など) を持つことができます。主体名とユーザー主体も参照してください。

AES
Advanced Encryption Standard。対称 128 ビットブロックのデータ暗号技術。2000 年の 10 月、米国政府は暗号化標準としてこのアルゴリズムの Rijndael 方式を採用しました。ユーザー主体の暗号化に代わる米国政府の標準として、AES が採用されています。

audit policy
どの監査イベントが記録されるかを決定する設定であり、大域の設定とユーザーごとの設定があります。大域の設定は監査サービスに適用され、一般にどのオプション情報を監査トレールに含めるかを決定します。2 つの設定 `cnt` と `ahlt` は、監査キューがいっぱいになった時点でのシステムの処理を決定します。たとえば、各監査レコードにシーケンス番号を含めるように監査ポリシーを設定できます。

Blowfish
32 ビットから 448 ビットまでの可変長鍵の対称ブロックの暗号化アルゴリズム。その作成者である Bruce Schneier 氏は、鍵を頻繁に変更しないアプリケーションに効果的であると述べています。

DES
Data Encryption Standard。1975 年に開発され、1981 年に ANSI X.3.92 として ANSI で標準化された対称鍵の暗号化方式。DES では 56 ビットの鍵を使用します。

Diffie-Hellman プロトコル
公開鍵暗号化としても知られています。1976 年に Diffie 氏と Hellman 氏が開発した非対称暗号鍵協定プロトコルです。このプロトコルを使用すると、セキュアでない伝達手段で、事前の秘密情報がなくても 2 人のユーザーが秘密鍵を交換できます。Diffie-Hellman は Kerberos で使用されます。

DSA
デジタル署名アルゴリズム。512 ビットから 4096 ビットまでの可変長鍵の公開鍵アルゴリズム。米国政府標準である DSS は最大 1024 ビットです。DSA は入力に SHA1 を使用します。

ECDSA
Elliptic Curve Digital Signature Algorithm。楕円曲線数学に基づく公開鍵アルゴリズム。ECDSA 鍵サイズは、同じ長さの署名の生成に必要な DSA 公開鍵のサイズより大幅に小さくなります。

FQDN
完全指定形式のドメイン名。`central.example.com` など (単なる `denver` は FQDN ではない)。

GSS-API
Generic Security Service Application Programming Interface の略。さまざまなモジュールセキュリティーサービス (Kerberos サービスなど) をサポートするネットワーク層。GSS-

API は、セキュリティー認証、整合性、およびプライバシサービスを提供します。認証、整合性、プライバシも参照してください。

KDC 鍵配布センター (Key Distribution Center)。次の 3 つの Kerberos V5 要素で構成されるマシン。

- 主体と鍵データベース
- 認証サービス
- チケット許可サービス

レルムごとに、1 つのマスター KDC と、1 つ以上のスレーブ KDC を配置する必要があります。

Kerberos 認証サービス、Kerberos サービスが使用するプロトコル、または Kerberos サービスの実装に使用されるコード。

Oracle Solaris の Kerberos は、Kerberos V5 認証にほぼ準拠して実装されています。

「Kerberos」と「Kerberos V5」は技術的には異なりますが、Kerberos のドキュメントでは多くの場合、同じ意味で使用されます。

Kerberos (または Cerberus) は、ギリシャ神話において、ハデスの門を警護する 3 つの頭を持つどう猛な番犬のことです。

Kerberos ポリシー Kerberos サービスでのパスワードの使用方法を管理する一連の規則。ポリシーは、主体のアクセスやチケットのパラメータ (有効期限など) を制限できます。

kvno 鍵バージョン番号。特定の鍵に対して、生成順に付けられたシーケンス番号。もっとも大きい kvno が、最新の鍵を示します。

MAC 1. メッセージ認証コード (MAC)を参照してください。

2. 「ラベル付け」とも呼ばれます。政府のセキュリティー用語規定では、MAC は「Mandatory Access Control」の略です。「Top Secret」や「Confidential」というラベルは MAC の例です。MAC と対称をなすものに DAC (Discretionary Access Control) があります。UNIX アクセス権は DAC の 1 例です。

3. ハードウェアにおいては、LAN における一意のシステムアドレス。システムが Ethernet 上に存在する場合は、Ethernet アドレスが MAC に相当します。

MD5 デジタル署名などのメッセージ認証に使用する繰り返し暗号化のハッシュ関数。1991 年に Rivest 氏によって開発されました。その使用は非推奨です。

NTP Network Time Protocol (NTP)。デラウェア大学で開発されたソフトウェア。ネットワーク環境で、正確な時間またはネットワーククロックの同期化を管理します。NTP を使用して、Kerberos 環境のクロックスキューを管理できます。「クロックスキュー」も参照してください。

PAM プラグイン可能認証モジュール (Pluggable Authentication Module)。複数の認証メカニズムを使用できるフレームワーク。認証メカニズムを使用するサービスはコンパイルし直す必要がありません。PAM は、ログイン時に Kerberos セッションを初期化できます。

password policy 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。

QOP 保護の質 (Quality of Protection)。整合性サービスまたはプライバシサービスで使用する暗号化アルゴリズムを選択するときに使用されるパラメータの 1 つ。

RBAC Oracle Solaris 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 policy 権利ポリシーを参照してください。

RSA デジタル署名と公開鍵暗号化システムを取得するための方法。その開発者である Rivest 氏、Shamir 氏、Adleman 氏によって 1978 年に最初に公開されました。

SEAM Solaris システム上の Kerberos の初期バージョンの製品名。この製品は、マサチューセッツ工科大学 (MIT) で開発された Kerberos V5 技術に基づいています。SEAM は、現在 Kerberos サービスと呼ばれています。MIT バージョンとは、今でも若干の違いがあります。

Secure Shell セキュリティー保護されていないネットワークを通して、セキュアなリモートログインおよびその他のセキュアなネットワークサービスを使用するための特別なプロトコル。

SHA1 セキュアなハッシュアルゴリズム。メッセージ要約を作成するために 2^{64} 文字以下の長さを入力するときに操作します。SHA1 アルゴリズムは DSA に入力されます。

stash ファイル stash ファイルには、KDC のマスター鍵を暗号化したコピーが含まれます。サーバーがリブートされると、このマスター鍵を使用して KDC が自動的に認証されてから、`kadmind` プロセスと `krb5kdc` プロセスがブートされます。stash ファイルにはマスター鍵が入っているため、このファイルやこのファイルのバックアップは安全な場所に保管する必要があります。暗号が破られると、この鍵を使用して KDC データベースのアクセスや変更が可能になります。

TGS チケット認可サービス (Ticket-Granting Service)。KDC のコンポーネントの 1 つ。チケットを発行します。

TGT チケット認可チケット (Ticket-Granting Ticket)。KDC によって発行されるチケット。クライアントは、このチケットを使用して、ほかのサービスのチケットを要求することができます。

索引

さ

た

は

ま

R

S

T

U