Oracle® Solaris 11.3 での Secure Shell アクセスの管理

Part No: E62749
2017 年 3 月

Part No: E62749

ドキュメントのアクセシビリティについて

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

Oracle Supportへのアクセス

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

目次

このドキュメントの使用

- **概要** – Oracle Solaris システムで Secure Shell を管理および使用する方法について説明します。
- **対象読者** – エンタープライズのセキュリティーを実装する必要があるシステム管理者。
- **必要な知識** – セキュリティーの概念と用語に関する高度な知識。

製品ドキュメントライブラリ

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

フィードバック

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

♦ ♦ ♦ 第 1 章

1 Secure Shell の使用

Oracle Solaris の Secure Shell 機能は、セキュリティー保護されていないネットワークを介した、リモートホストへのセキュリティー保護されたアクセスを提供します。Solaris Secure Shell には、リモートログイン、リモートウィンドウ表示、およびリモートファイル転送を行うコマンドが組み込まれています。この章で扱う内容は、次のとおりです。

参照情報については、第2章「Secure Shell リファレンス」を参照してください。

Oracle Solaris 11.3 での Secure Shell の新機能

このセクションでは、既存のお客様向けに、このリリースでの Secure Shell の重要な新機能について説明します。

- Oracle Solaris は、Secure Shell の `openssh` 実装を提供しています。この OpenSSH 実装は、OpenSSH 7.2p2 および追加機能に基づいて構築されています。デフォルトは引き続き `sunssh` 実装 (SunSSH) です。詳細は、12 ページの「Secure Shell の OpenSSH 実装」を参照してください。

  2 つの実装間を切り替えるには、`pkg mediator` コマンドを使用します。手順については、24 ページの「Secure Shell の OpenSSH 実装をインストールして切り替える方法」を参照してください。
- Oracle Solaris の最新のリリースには SunSSH 実装の変更が含まれています。詳細は、20 ページの「SunSSH 内の鍵のタイプを制御する新しいキーワード」を参照してください。

- Oracle Solaris の最新のリリースでは、クライアント構成ファイル内の `IgnoreUnknown` および `IgnoreIfUnknown` キーワードがサポートされています。詳細は、21 ページの「複数の Oracle Solaris リリース間での `.ssh/config` ファイルの共有」を参照してください。

Secure Shell について

Secure Shell は、新規にインストールされた Oracle Solaris システム上のデフォルトのリモートアクセスプロトコルです。Secure Shell のデフォルトの実装は `sunssh` 実装 (SunSSH) です。`openssh` 実装 (OpenSSH) も利用できます。

Oracle Solaris の Secure Shell では、OpenSSL `libcrypto` ライブラリの低レベルの暗号化 API が使用されます。OpenSSL ツールキットは、Secure Sockets Layer および Transport Layer Security を実装します。

注記 - OpenSSL のバージョンを表示するには、端末ウィンドウで `openssl version` コマンドを実行します。OpenSSL は、暗号化モジュールのための米国政府のコンピュータセキュリティー標準である FIPS 140-2 を実装できます。

FIPS 140-2 モードで Secure Shell を使用する方法の詳細については、19 ページの「SunSSH と FIPS 140-2」を参照してください。

Secure Shell では、認証は、パスワード、公開鍵、またはその両方の使用によって提供されます。すべてのネットワークトラフィックは暗号化されます。このため、Secure Shell では、悪意を持つ侵入者が傍受した通信を読むことはできません。また、攻撃者が偽装することもできません。

Secure Shell は、オンデマンドタイプの仮想プライベートネットワーク (VPN)として使用することもできます。VPN では、暗号化されたネットワークリンクを介して、ローカルシステムとリモートシステム間で、X ウィンドウシステムのトラフィックを転送したり個々のポート番号を接続したりできます。

Secure Shell では、次の操作を行うことができます。

- セキュリティー保護されていないネットワークを介して、別のホストに安全にログインします。
- 2 つのホスト間でファイルを安全にコピーします。
- リモートシステム上でコマンドをセキュアに実行します。

サーバー側では、Secure Shell が Secure Shell プロトコルのバージョン 2 (v2) をサポートしています。クライアント側では、v2 に加えて、クライアントがバージョン 1 (v1) をサポートしています。

Secure Shell 認証

Secure Shell は、リモートシステムへの接続を認証するために、公開鍵とパスワードの方式を提供します。公開鍵認証は、非公開鍵がネットワーク上を移動しないためより強力な認証メカニズムです。

認証方式は、次の順序で試されます。構成が認証方式を満たさないときは、次の方式が試されます。

- **GSS-API 認証** – `mech_krb5` (Kerberos V) などの GSS-API メカニズムの資格を使用して、Secure Shell クライアントとサーバーを認証します。GSS-API 認証の詳細は、『Oracle Solaris 11 セキュリティーサービス開発ガイド』 の 「GSS-API の紹介」を参照してください。
- **ホストに基づく認証** – ホスト鍵と `rhosts` ファイルを使用します。Secure Shell クライアントの RSA または DSA 公開/非公開ホスト鍵を使用してクライアントを認証します。`rhosts` ファイルを使用して、ユーザーに対してクライアントを承認します。
- **公開鍵認証** – RSA または DSA 公開/非公開鍵によってユーザーを認証します。
- **キーボード対話型認証** – PAM を使用してユーザーを認証します。v2 でのキーボード認証方式では、PAM による任意のプロンプトが可能です。詳細は、`sshd(1M)` のマニュアルページの「`SECURITY`」のセクションを参照してください。

次の表では、リモートシステムにログインしようとするユーザーを認証するための要件を示します。ユーザーは、ローカルシステム (クライアントシステム) 上に存在します。リモートシステムである Secure Shell サーバーは `sshd` デーモンを実行しています。次の表は、Secure Shell の認証方式とシステムの要件を示します。

表 1 Secure Shell の認証方式

方法	ローカルホスト (クライアント) の要件	リモートホスト (サーバー) の要件
GSS-API	GSS メカニズムのイニシエータの資格。	GSS メカニズムのアクセプタの資格。詳細は、52 ページの「Secure Shell での GSS 資格の取得」を参照してください。

ホストに基づく	ユーザーアカウント
	
	`/etc/ssh/ssh_host_rsa_key` または `/etc/ssh/ssh_host_dsa_key` にローカルホストの非公開鍵
	
	`/etc/ssh/ssh_config` 内で `HostbasedAuthentication yes`
		ユーザーアカウント
		
		`/etc/ssh/known_hosts` または `~/.ssh/known_hosts` にローカルホストの公開鍵
		
		`/etc/ssh/sshd_config` 内で `HostbasedAuthentication yes`
		
		`/etc/ssh/sshd_config` 内で `IgnoreRhosts no`
		
		`/etc/ssh/shosts.equiv`、`/etc/hosts.equiv`、`~/.rhosts`、または `~/.shosts` にローカルホストのエントリ

方法	ローカルホスト (クライアント) の要件	リモートホスト (サーバー) の要件
パスワードベース	ユーザーアカウント	ユーザーアカウント PAM をサポートします。
RSA または DSA 公開鍵	ユーザーアカウント `~/.ssh/id_rsa` または `~/.ssh/id_dsa` に非公開鍵 `~/.ssh/id_rsa.pub` または `~/.ssh/id_dsa.pub` にユーザーの公開鍵	ユーザーアカウント `~/.ssh/authorized_keys` にユーザーの公開鍵

Secure Shell の OpenSSH 実装

Oracle Solaris では Secure Shell の 2 つの実装が提供されており、`sunssh` パッケージのレガシーの SunSSH 実装と、オプションの `openssh` パッケージの OpenSSH 実装です。Oracle Solaris の OpenSSH は、バージョン 7.2p2 に加えて Oracle Solaris 環境に特有の追加機能に基づいて構築されています。

SunSSH がデフォルトですが、新しい OpenSSH 実装に切り替えることができます。一度に使用できる実装は 1 つだけです。

注意 - Oracle Solaris 11.3 SRU 5 からは、OpenSSH は、システム管理者からの積極的な注意が必要なバージョンにアップグレードされています。12 ページの「OpenSSH への Oracle Solaris の変更」および 16 ページの「OpenSSH への Oracle Solaris の追加機能」を参照してください。

OpenSSH への Oracle Solaris の変更

OpenSSH 7.1p1 への前回のアップグレード後、Oracle Solaris 11.3 では Secure Shell の `openssh` 実装が OpenSSH 7.2p2 にアップグレードされています。

注記 - システムがまだ SunSSH を使用している場合、OpenSSH の変更は、システムの動作に影響を与えません。

このセクションの変更は、バージョン 7.1p1 に含まれています。

- 13 ページの「Secure Shell プロトコル 1 サポートが削除される」
- 13 ページの「安全でないアルゴリズムが OpenSSH から削除される」
- 14 ページの「`diffie-hellman-group1-sha1` がデフォルトで無効になる」
- 14 ページの「`ssh-dss` 鍵がデフォルトで無効になる」

- 15 ページの「OpenSSH の UseDNS のデフォルト値が No」
- 16 ページの「OpenSSH で TCP ラッパーがサポートされていない」

Secure Shell プロトコル 1 サポートが削除される

Oracle Solaris 11.3 SRU 5 からは、Secure Shell プロトコル 1 (v1) サポートは、サーバー側とクライアント側の両方で OpenSSH から削除されました。v1 のみをサポートするネットワークエンティティーは、ほとんどが古いネットワークルーターです。このようなデバイスには OpenSSH の Oracle Solaris 11.3 実装を使用して接続できなくなりました。ただし、Oracle Solaris 10 ユーザーは SunSSH を使用して、v1 を使用するシステムに引き続きアクセスできます。さらに、Oracle Solaris 11.3 ユーザーも SunSSH を使用して、v1 を使用するシステムに引き続きアクセスできます。

安全でないアルゴリズムが OpenSSH から削除される

安全でないアルゴリズムを削除するように、暗号化規格および MAC のデフォルトセットが変更されました。Oracle Solaris 11.3 SRU 5 リリース以降、デフォルト構成で次の暗号化規格および MAC のみが有効になります。

```
Ciphers
----
chacha20-poly1305@openssh.com
aes128-ctr
aes192-ctr
aes256-ctr
aes128-gcm@openssh.com
aes256-gcm@openssh.com

MACs
----
umac-64-etm@openssh.com
umac-128-etm@openssh.com
hmac-sha2-256-etm@openssh.com
hmac-sha2-512-etm@openssh.com
hmac-sha1-etm@openssh.com
umac-64@openssh.com
umac-128@openssh.com
hmac-sha2-256
hmac-sha2-512
hmac-sha1
```

サポートされているすべての暗号化規格および MAC を一覧表示するには、次のコマンドを使用してください。

```
root@source# ssh -Q cipher
root@source# ssh -Q MACs
```

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

diffie-hellman-group1-sha1 がデフォルトで無効になる

diffie-hellman-group1-sha1 鍵交換は、もうセキュアでないと見なされているため、Secure Shell のクライアント側およびサーバー側の両方で無効になります。

サーバーが diffie-hellman-group1-sha1 のみをサポートする場合は、diffie-hellman-group-exchange-sha256 をサポートするようにそれらをアップグレードしてください。または、2 番目の選択肢として、diffie-hellman-group14-sha1 をサポートするバージョンに Oracle Solaris をアップグレードしてください。

ピアをアップグレードすることを選択できない場合、diffie-hellman-group-exchange-sha256、diffie-hellman-group14-sha1、または diffie-hellman-group-exchange-sha1 をサポートしないシステムに接続しているユーザーは、次のように diffie-hellman-group1-sha1 を有効にできます。

```
root@source# ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 user@system1
```

Secure Shell の OpenSSH 実装の場合、サーバー管理者は、セキュアでない鍵交換方式を明示的に有効化することで、セキュアな鍵交換方式をサポートしないシステムからのログインを許可できます。この行を /etc/ssh/sshd_config ファイルに追加してください。

```
KexAlgorithms
diffie-hellman-group1-sha1,diffie-hellman-group14-sha1,diffie-hellman-group-exchange-sha1,
diffie-hellman-group-exchange-sha256
```

それから、Secure Shell サーバーを再起動します。

注記 - KexAlgorithms キーワードは OpenSSH 実装に固有であるため、SunSSH 実装によって認識されません。

詳細については、Using OpenSSH with Legacy SSH Implementationsを参照してください。

ssh-dss 鍵がデフォルトで無効になる

ssh-dss および ssh-dss-cert-* ホストおよびユーザー鍵タイプは本質的に弱いため、実行時にデフォルトで無効になります。

公開鍵認証で ssh-dss 鍵を使用していた場合は、新しい ssh-rsa 鍵を作成し、既存の ssh-dss 鍵をすべての authorized_keys ファイルから削除してください。新しい鍵の作成手順については、40 ページの「Secure Shell で使用する公開鍵と非公開鍵のペアを生成する方法」を参照してください。

ssh-rsa および ssh-dss ホスト鍵がまだ存在しない場合、svc:/network/ssh:default が両方を作成します。したがって、Oracle Solaris サーバーは通常 ssh-dss ホ

スト鍵と ssh-rsa 鍵を持ちます。まれに、サーバーに ssh-dss ホスト鍵だけがプロビジョニングされていた場合には、ssh_rsa ホスト鍵を追加してください。鍵を作成できない場合、ユーザーはこれらのサーバーに接続して ssh-dss 鍵タイプを有効にできます。

```
root@source# ssh -oHostKeyAlgorithms=+ssh-dss user@somehost
```

詳細については、Using OpenSSH with Legacy SSH Implementationsを参照してください。

OpenSSH の UseDNS のデフォルト値が No

sshd_config ファイルで UseDNS 値が指定されていない場合、UseDNS のデフォルト値は No です。以前のデフォルト値にはセキュリティー利点がありませんでした。

UseDNS 値が No であることは、ssh サービスを構成するときにホスト名を使用できないことを意味します。

2 つの選択肢があります。

- sshd_config ファイルで明示的に UseDNS yes を指定できます。
- 次の例のように、sshd_config ファイルでホスト名の代わりに IP アドレスを使用できます。
  - sshd_config ファイルの Match ブロックセクションで、Host 条件の代わりに Address 条件を使用します。

    たとえば、Match Host somehost.domain を Match Address 192.168.0.10 に置き換えます。
  - AllowUsers、AllowGroups、DenyUsers、および DenyGroups の sshd_config エントリで、ホスト名の代わりに IP アドレスを使用します。

    たとえば、AllowUsers jsmith@somehost.domain を AllowUsers jsmith@192.168.0.10 に置き換えます。
  - /etc/ssh/shosts.equiv または ~/.shosts エントリで、ホスト名の代わりに IP アドレスを使用します。

    たとえば、somehost.domain を 192.168.0.10 に置き換えます。
  - ~/.ssh/authorized_keys エントリで、from オプションを指定するときにホスト名の代わりに IP アドレスを使用します。

    たとえば、これを

    ```
    from="somehost.domain" ssh-rsa AAAAB3...Q== jsmith@work
    ```

    次と置き換えます

    ```
    from="192.168.0.10" ssh-rsa AAAAB3...Q== jsmith@work
    ```

OpenSSH で TCP ラッパーがサポートされていない

OpenSSH は TCP ラッパーをサポートしません。以前に TCP ラッパーによって適用されていた構成を保持するには、`sshd_config` ファイルを変更するかファイアウォールを使用する必要があります。

注記 - Secure Shell の `openssh` 実装は引き続き TCP 接続を使用します。TCP ラッパー関数 `libwrap` のみがサポートされなくなります。

TCP ラッパーを使用していた場合は、ログインを許可または拒否するために `/etc/hosts.allow` または `/etc/hosts.deny` を使用していました。`sshd_config` ファイルで `Match` ブロックを使用して同等の構成を設定できます。

たとえば、`10.163.0.0/16` サブネットからのログインのみを許可するため、TCP ラッパーを次のように設定していたとします。

```
root@jsmith-cz:~# cat /etc/hosts.allow
    sshd : 10.163.
    root@jsmith-cz:~# cat /etc/hosts.deny
    ALL : ALL
```

`sshd_config` ファイル内の次のエントリは、同等の制約を設定します。

```
Match Address *,!10.163.0.0/16
           MaxAuthTries 0
```

別の選択肢は、ファイアウォールを構成してアクセスを制御することです。これらの例に似た設定をファイアウォールで適用できます。ファイアウォールアクセス制御は、ネットワーク接続がカーネル内で確立される前に発生します。

OpenSSH への Oracle Solaris の追加機能

OpenSSH は、システムにインストール可能なオプションパッケージ `openssh` です。機能が `openssh` 実装に追加されていますが、OpenSSH プロジェクトとの互換性は維持されています。

次の Oracle Solaris 機能が `openssh` 実装に追加されました。

- Secure Shell クライアントからのバナーメッセージの表示を無効にするために使用できる `DisabledBanner` キーワード。詳細は、`ssh_config(4)` のマニュアルページを参照してください。
- PAM のサポート。

  `sunssh` 実装内にある `PAMServiceName` および `PAMServicePrefix` キーワードは、`openSSH` 実装に移行されました。

- Oracle Solaris 監査サポート。
- 書き込み不可のホームディレクトリで Xforwarding が機能します。
- RFC 4462 で規定されている GSS-API 認証 Diffie-Hellman 鍵交換。
- ホストに基づく認証のための役割のログインが可能です (PAM および `sshd` で適切に構成されている場合)。

OpenSSH の委任された資格は、資格がほかのプラットフォーム上に格納される方法とは異なる方法で Oracle Solaris に格納されます。

- 委任された資格を `/tmp/krb5cc_101_WO4082` などのデフォルト以外の資格キャッシュに格納する OpenSSH プロジェクトとは異なり、Secure Shell の `openssh` 実装では、`/tmp/krb5cc_101` などのデフォルトの資格キャッシュが使用されます。
- デフォルトの資格キャッシュ内の資格は、Kerberos によって保護される NFS ファイルシステムにアクセスするために使用できます。

  詳細は、『Oracle Solaris 11.3 での Kerberos およびその他の認証サービスの管理』を参照してください。

注記 - Secure Shell の `openssh` 実装は FIPS 140-2 モードで実行するように構成できません。また、OpenSSH の `sftp` 転送は、Oracle Solaris 監査サービスに統合されていません。

Secure Shell パッケージおよび構成ファイル

`openssh` IPS パッケージが導入されたため、Oracle Solaris では 4 つの Secure Shell パッケージが提供されます。

`network/ssh`	`sunssh` 実装のコンポーネント
`network/openssh`	`openssh` 実装のコンポーネント
`net/work/ssh/ssh-utilities`	2 つの実装が共有する Secure Shell ユーティリティー
`/service/network/ssh-common`	SMF サービスおよび 2 つの実装が共有する Secure Shell システム構成ファイル

SunSSH および OpenSSH は次の Secure Shell システム構成ファイルを共有します。

- `/etc/ssh/ssh_config`
- `/etc/ssh/sshd_config`
- `/etc/ssh/moduli`

Secure Shell の SunSSH 実装

Secure Shell の SunSSH 実装は、OpenSSH (http://www.openssh.com) プロジェクトのフォークです。

OpenSSH の最近のバージョンに見つかった脆弱性に対するセキュリティー関連の修正が、個別のバグ修正および機能として Secure Shell (SunSSH) の sunssh 実装に組み込まれています。

次の機能が、現在のリリースの SunSSH で実装されました。

- ForceCommand キーワード – ユーザーがコマンド行に入力した内容に関係なく、指定されたコマンドを強制的に実行します。このキーワードは、Match ブロックの内側で非常に役立ちます。この sshd_config 構成キーワードは、$HOME/.ssh/authorized_keys 内の command="..." キーワードに似ています。
- AES-128 パスフレーズ保護 – ssh-keygen コマンドで生成される非公開鍵は AES-128 アルゴリズムで保護されます。このアルゴリズムでは、新しく生成された鍵や再暗号化された鍵 (パスフレーズが変更された場合など) を保護します。
- sftp-server コマンドの -u オプション – ユーザーがファイルやディレクトリに対して明示的な umask を設定できるようにします。このオプションは、ユーザーのデフォルトの umask をオーバーライドします。例については、sshd_config(4) のマニュアルページの「Subsystem」の説明を参照してください。
- Match ブロック用のその他のキーワード – Match ブロックの内側では AuthorizedKeysFile、ForceCommand、および HostbasedUsesNameFromPacketOnly がサポートされています。デフォルトでは、AuthorizedKeysFile の値は $HOME/.ssh/authorized_keys で、HostbasedUsesNameFromPacketOnly の値は no です。Match ブロックを使用するには、36 ページの「Secure Shell デフォルトのユーザーおよびホスト例外を作成する方法」を参照してください。

Oracle Solaris エンジニアは次の Oracle Solaris 機能を SunSSH に組み込みました。

- PAM – SunSSH は PAM を使用します。OpenSSH UsePAM 構成キーワードはサポートされていません。
- 特権分離 – SunSSH の特権分離コードは常にオンに設定されており、オフに切り替えることはできません。

  この実装では、監査、記録管理、および再キーイングの処理がセッションプロトコルの処理から分離されます。OpenSSH UsePrivilegeSeparation キーワードはサポートされていません。
- ロケール – SunSSH では、RFC 4253「*Secure Shell Transfer Protocol*」で定義されている言語ネゴシエーションが完全にサポートされています。ユーザーがログインしたあと、ユーザーのログインシェルプロファイルは Secure Shell でネゴシエーションを行なったロケール設定をオーバーライドできます。

- 監査 – SunSSH は Oracle Solaris 監査サービスに完全に統合されています。監査サービスについては、『Oracle Solaris 11.3 での監査の管理』を参照してください。
- GSS-API のサポート – SunSSH では、GSS-API 認証はユーザー認証と初期鍵交換の両方に使用できます。GSS-API 認証は、RFC4462「*Generic Security Service Application Program Interface*」で定義されています。
- プロキシコマンド – SunSSH では、SOCKS5 プロトコルと HTTP プロトコルのプロキシコマンドが提供されています。例については、49 ページの「ファイアウォール外部のホストへのデフォルトの Secure Shell 接続を設定する方法」を参照してください。

SunSSH は `SSH_OLD_FORWARD_ADDR` 互換性フラグを OpenSSH プロジェクトから再同期します。

SunSSH と FIPS 140-2

SunSSH は OpenSSL FIPS 140-2 モジュールのコンシューマです。Oracle Solaris は、SunSSH サーバー側とクライアント側に FIPS 140-2 オプションを提供します。FIPS 140-2 の要件に準拠するには、管理者は FIPS 140-2 オプションを構成および使用するようにしてください。

注記 - Oracle Solaris では、Secure Shell (OpenSSH) の `openssh` 実装は FIPS 140-2 をサポートしません。

FIPS 140-2 モードの SunSSH はデフォルトではありません。管理者は、明示的に SunSSH を FIPS 140-2 モードで動作できるようにする必要があります。コマンド `ssh -o "UseFIPS140 yes"` *remote-host* を使用して FIPS 140-2 モードを起動できます。構成ファイル内にキーワードを設定することもできます。

簡単に説明すると、この実装は次のもので構成されます。

- SunSSH のサーバー側とクライアント側で、次の FIPS 140-2 承認の暗号化方式を使用できます: `aes128-cbc`、`aes192-cbc`、および `aes256-cbc`。

  `3des-cbc` は、クライアント側ではデフォルトで使用できますが、セキュリティーリスクの可能性があるため、SunSSH サーバー側の暗号化方式のリストには含まれていません。
- 次の FIPS 140-2 承認のメッセージ認証コード (MAC) を使用できます。
  - `hmac-sha1`、`hmac-sha1-96`
  - `hmac-sha2-256`、`hmac-sha2-256-96`
  - `hmac-sha2-512`、`hmac-sha2-512-96`
- 4 つの SunSSH サーバークライアント構成がサポートされています。
  - クライアントとサーバーの両方で FIPS 140-2 モードを使用しない

  - クライアントとサーバーの両方で FIPS 140-2 モードを使用する
  - サーバーでは FIPS 140-2 モードを使用するがクライアントでは FIPS 140-2 モードを使用しない
  - サーバーでは FIPS 140-2 モードを使用しないがクライアントでは FIPS 140-2 モードを使用する
- `ssh-keygen` コマンドには、FIPS 140-2 モードの SunSSH クライアントに必要な PKCS #8 形式でユーザーの非公開鍵を生成するためのオプションがあります。詳細は、`ssh-keygen(1)` のマニュアルページを参照してください。

FIPS 140-2 および SunSSH の詳細は、『Oracle Solaris 11.3 での FIPS 140-2 対応システムの使用』、『Oracle Solaris 11.3 での暗号化と証明書の管理』 の 「FIPS 140-2 が有効になったブート環境の作成」および `sshd(1M)` のマニュアルページを参照してください。

Secure Shell の操作に Sun Crypto Accelerator 6000 カードを使用すると、SunSSH はレベル 3 の FIPS 140-2 サポートを受けて実行されます。レベル 3 のハードウェアは、物理的な改ざんへの耐性があり、ID ベースの認証を使用し、重要なセキュリティーパラメータを処理するインタフェースをハードウェアのほかのインタフェースから分離するものとして認定されています。

SunSSH 内の鍵のタイプを制御する新しいキーワード

SunSSH では、受け入れた公開鍵タイプを制御して、弱い鍵タイプを無効にできるようにする新しいキーワードが追加されました。デフォルトは、すべての鍵のタイプを受け入れることです。

SunSSH サーバー構成には次の新しいキーワードが追加されました。

- `HostKeyAlgorithms`
- `HostbasedAcceptedKeyTypes`
- `PubkeyAcceptedKeyTypes`
- `Kexalgorithms`

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

SunSSH クライアント構成には次の新しいキーワードが追加されました。

- `HostbasedAcceptedKeyTypes`
- `PubkeyAcceptedKeyTypes`
- `Kexalgorithms`

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

SunSSH による X.509 証明書の使用

X.509 証明書は SunSSH 認証に適しています。これらはリモートスクリプトを実行する場合など、ユーザーとの対話が許可されていないリモートログインではもっとも安全なオプションです。また、ユーザーにホスト ID を受け入れることを求めるプロンプトが表示されず、ユーザーの公開鍵がリモートサーバー上に存在する必要がありません。

ユーザー (SunSSH クライアント) が SunSSH サーバーへの接続を試みたときに、サーバーはホスト証明書をクライアントに渡します。CA 証明書内の認証局 (CA) の公開鍵を使用することにより、クライアントは CA に関連付けられたデジタル署名に対してサーバー上のホスト証明書を検証します。

X.509 証明書の構成には次のステップが必要になります。

- 管理者はユーザーがリモートログインするサーバー上にサーバーの X.509 証明書を生成します
- サーバーにリモートでログインする予定のユーザーは、自分の X.509 証明書を生成します
- 管理者はサーバーのルート証明書の公開部分を、ユーザーを構成する管理者に送信します
- すべてのユーザーは、ルート証明書の公開部分 (SunSSH 構成ファイル内で「トラストアンカー」または TA と呼ばれます) をリモートサーバーの管理者に送信します
- サーバー管理者はユーザーの TA 証明書を、ssh デーモンが読み取りできる場所に保管します
- ユーザー管理者はサーバーの TA 証明書を、ssh デーモンが読み取りできる場所に保管します
- その後、ユーザーは SunSSH を使用して、リモートサーバーにログインできます

ユーザーは、自己署名付きトラストアンカー (TA) 証明書を生成して署名することもできます。自己署名付き証明書はあまりセキュアではありません。証明書に自己署名するユーザーは、証明書に関する技術上およびセキュリティー上の問題について精通している必要があります。

手順については、23 ページの「Secure Shell の構成」を参照してください。

複数の Oracle Solaris リリース間での .ssh/config ファイルの共有

ホームディレクトリがネットワーク上にある場合は、複数のシステムが別の Oracle Solaris リリースまたは別の Secure Shell 実装を実行している場合でも、それらのシス

テム間で ~/.ssh/config ファイルを共有できます。ただし、Secure Shell 実装は、異なる Secure Shell 実装からの一部の構成オプションを認識しないことがあります。場合によっては、Secure Shell 実装は、同じ Secure Shell 実装の異なるバージョンからの構成オプションを認識しないことがあります。

Oracle Solaris 10 Update 11 以降のリリースでは、ネットワーク上の別のシステムによって Secure Shell 構成オプションを認識できない場合は、認識できないオプションを無視するように ~/.ssh/config ファイルを変更できるため、複数のシステム間で共有 ssh_config ファイルを使用できます。

Secure Shell 実装と Ignore キーワード

IgnoreIfUnknown と IgnoreUnknown という 2 つのキーワードを使用して、複数のシステム間で認識できない Secure Shell 構成キーワードを無視できます。IgnoreIfUnknown キーワードが SunSSH 内で使用でき、IgnoreUnknown キーワードが OpenSSH 内で使用できます。

IgnoreIfUnknown と IgnoreUnknown は両方とも、コンマ区切りの ssh_config キーワードのリストを指定し、これは ssh プログラムに認識されない場合は Secure Shell によって無視されます。ただし、IgnoreIfUnknown は構成ファイル全体に適用されますが、IgnoreUnknown は構成ファイル内でその後に続く認識できないキーワードにのみ適用されます。

次の表は、Oracle Solaris の各リリースの Secure Shell 実装と、各実装で使用可能な Ignore キーワードを示しています。

表 2 Secure Shell の Ignore キーワード

リリース	Secure Shell 実装	サポートされている Ignore キーワード
Oracle Solaris 11.3	SunSSH	IgnoreIfUnknown および IgnoreUnknown
Oracle Solaris 11.3	OpenSSH	IgnoreUnknown
Oracle Solaris 11.3 より前の Oracle Solaris 11 リリース	SunSSH	IgnoreIfUnknown
Oracle Solaris 10 Update 11	SunSSH	IgnoreIfUnknown

次のリリースは Ignore キーワードをサポートせず、ネットワーク上での共有 Secure Shell 構成の一部として含めることはできません。

- Oracle Solaris 9
- Update 11 より前の Oracle Solaris 10
- OpenSSH 6.2 および古い OpenSSH バージョン

相互運用性を有効にするための Secure Shell キーワードの無視

Secure Shell の異なる実装を実行しているシステムを持つネットワーク上に `~/.ssh/config` ファイルがある場合、ファイルに `IgnoreUnknown` および `IgnoreIfUnknown` キーワードを追加することによって Secure Shell 構成キーワードを有効にできます。

注記 - 表2に示すように、すべてのシステムで少なくとも 1 つの `Ignore` キーワードが使用できるようにする必要があります。

例 1 異なるキーワードをサポートするリリース間で Secure Shell 構成を共有する

この例は、OpenSSH 6.8 で導入された `HostBasedKeyTypes` キーワードを使用する方法を示しています。このキーワードをサポートしない Secure Shell のリリースを一部のシステムが実行しているネットワーク上にユーザーが存在します。

`ssh_config` ファイルに次のエントリを追加します。

```
---
IgnoreUnknown HostBasedKeyTypes,IgnoreIfUnknown
IgnoreIfUnknown HostBasedKeyTypes,IgnoreUnknown

HostBasedKeyTypes ssh-rsa-cert-v01@openssh.com, ssh-rsa
---
```

通信するすべての Secure Shell 実装を有効にするには、両方の `Ignore` キーワードを追加します。詳細は、`ssh_config(4)` のマニュアルページを参照してください。

Secure Shell の構成

Secure Shell は、`sunssh` 実装をデフォルトの実装として設定したインストール時に構成されます。Secure Shell でデフォルトを変更するには、管理者の介入が必要です。次のタスクでは、サイトで Secure Shell を構成する方法を示しています。

Secure Shell の構成 (タスクマップ)

次のタスクマップでは、Secure Shell を構成するための管理上の手順を示します。通常のユーザーが実行できる手順は、39 ページの「Secure Shell の使用」にあります。

タスク	説明	手順
OpenSSH を実行します。	デフォルトの SunSSH から Secure Shell の最新の OpenSSH 実装に切り替えます。	24 ページの「Secure Shell の OpenSSH 実装をインストールして切り替える方法」
Secure Shell を FIPS 140-2 モードで実行します。	SunSSH で OpenSSL からの FIPS 140-2 暗号化方式を使用できます。	19 ページの「SunSSH と FIPS 140-2」および『Oracle Solaris 11.3 での暗号化と証明書の管理』 の 「FIPS 140-2 が有効になったブート環境の作成」
SunSSH クライアントの X.509 証明書を構成します。	証明書ベースの認証のユーザー側を構成します。	29 ページの「SunSSH クライアントの X.509 証明書を生成する方法」
SunSSH クライアントの X.509 証明書を構成します。	証明書ベースの認証のユーザー側を構成します。	例3「`openssl` を使用して SunSSH クライアント用の X.509 証明書を生成する」
SunSSH サーバーの X.509 証明書を構成します。	証明書ベースの認証のサーバー側を構成します。	32 ページの「SunSSH サーバー用の X.509 証明書を生成する方法」
ホストに基づく認証を構成します。	クライアントと Secure Shell サーバーでのホストに基づく認証を構成します。	26 ページの「ホストに基づく認証を Secure Shell に設定する方法」
接続待ち時間を処理するためのバッファーサイズを増やします。	待ち時間の長い高帯域幅ネットワークのために、TCP プロパティー `recv_buf` の値を増やします。	『Oracle Solaris 11.3 での TCP/IP ネットワーク、IPMP、および IP トンネルの管理』 の 「TCP 受信バッファーサイズの変更」
ポート転送を構成します。	ユーザーがポート転送を使用できるようにします。	35 ページの「Secure Shell のポート転送を構成する方法」
Secure Shell デフォルトの例外を構成します。	ユーザー、ホスト、グループ、およびアドレスに対して、デフォルトとは異なる Secure Shell の値を指定します。	36 ページの「Secure Shell デフォルトのユーザーおよびホスト例外を作成する方法」
`sftp` 転送のために `root` 環境を切り離します。	ファイル転送のための保護されたディレクトリを提供します。	37 ページの「`sftp` ファイルのための切り離されたディレクトリを作成する方法」

▼ Secure Shell の OpenSSH 実装をインストールして切り替える方法

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

注意 - Oracle Solaris 11.3 SRU 5 リリースからは、Secure Shell の OpenSSH 実装は、システム管理者からの積極的な注意が必要なバージョンにアップグレードされています。12 ページの「Secure Shell の OpenSSH 実装」を参照してください。

1. **openssh パッケージがインストールされているかどうかを確認します。**

```
# pkg list openssh
pkg list: no packages matching the following patterns are installed:
  openssh
```

2. **openssh パッケージが一覧表示されない場合、パッケージをインストールします。**

```
# pkg install network/openssh
```

3. **システム上の Secure Shell の実装をすべて表示します。**

```
# pkg mediator -a ssh
MEDIATOR     VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
ssh          vendor            vendor     sunssh
ssh          system            system     openssh
```

出力の vendor は、このリリースでは sunssh であるデフォルトの実装を示しています。

4. **openssh 実装に切り替えます。**

```
# pkg set-mediator -I openssh ssh
            Packages to change:   3
           Mediators to change:   1
            Services to change:   1
      Create boot environment:  No
Create backup boot environment: Yes
PHASE                                        ITEMS
Removing old actions                         34/34
Updating modified actions                    25/25
Updating package state database               Done
Updating package cache                         0/0
Updating image state                          Done
Creating fast lookup database                 Done
Updating package cache                         1/1
```

注記 - 変更には、選択した実装に適したすべてのマニュアルページが含まれています。

このコマンドにより Secure Shell サーバーが再起動します。既存の Secure Shell 接続は機能し続けます。サーバーを現在使用しているユーザーは、前の実装を引き続き使用することも、新しい実装を使用するためにログアウトしてログインすることもできます。

5. **(オプション) 有効な Secure Shell の実装を表示します。**

```
$ pkg mediator ssh
MEDIATOR   VER. SRC.   VERSION   IMPL. SRC.   IMPLEMENTATION
```

```
ssh        system                local        openssh
```

この例では、openssh 実装が有効です。

pkg mediator コマンドの使用の詳細は、『Oracle Solaris 11.3 ソフトウェアの追加と更新』 の 「優先アプリケーションの変更」および pkg(1) のマニュアルページを参照してください。

6. **(オプション) SunSSH に戻します。**

```
# pkg set-mediator -I sunssh ssh
```

このコマンドにより Secure Shell サーバーが再起動します。既存の Secure Shell 接続は機能し続けます。サーバーを現在使用しているユーザーは、前の実装を引き続き使用することも、新しい実装を使用するためにログアウトしてログインすることもできます。

▼ ホストに基づく認証を Secure Shell に設定する方法

次の手順によって公開鍵システムが設定され、クライアントの公開鍵が Secure Shell サーバー上での認証に使用できるようになります。また、ユーザーは、公開鍵と非公開鍵のペアを作成する必要があります。

この手順での「クライアント」および「ローカルホスト」という用語は、ユーザーが ssh コマンドを入力するシステムを指しています。「サーバー」および「リモートホスト」という用語は、クライアントが到達しようとするシステムを指しています。

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

1. **クライアントで、ホストに基づく認証を有効にします。**
   クライアントの構成ファイル /etc/ssh/ssh_config で、次のエントリを追加します。

```
HostbasedAuthentication yes
```

このファイルの構文については、ssh_config(4) のマニュアルページを参照してください。

2. **Secure Shell サーバーで、ホストに基づく認証を有効にします。**
   サーバーの構成ファイル /etc/ssh/sshd_config で、同じエントリを追加します。

```
HostbasedAuthentication yes
```

3. **お客様またはユーザーは、クライアントが信頼されるホストとして認識できるようにするファイルをサーバーに構成します。**

詳細については、sshd(1M) のマニュアルページの「FILES」のセクションを参照してください。

- **お客様が構成を行う場合、エントリとしてクライアントをサーバーの /etc/ssh/shosts.equiv ファイルに追加します。**

  client-host

- **ユーザーが構成を行う場合は、クライアントのエントリをサーバーの ~/.shosts ファイルに追加する必要があります。**

  client-host

4. **サーバーで、sshdデーモンが信頼されるホストのリストにアクセスできるようにします。**

   /etc/ssh/sshd_config ファイルで、IgnoreRhosts を no に設定します。

```
## sshd_config
IgnoreRhosts no
```

5. **使用するサイトの Secure Shell のユーザーが両方のホストでアカウントを持つようにします。**

6. **次のいずれかの方法を使用してクライアントの公開鍵をサーバー上に配置します。**

   - **サーバー上の sshd_config ファイルを変更後、クライアントの公開ホスト鍵を ~/.ssh/known_hosts ファイルに追加するようにユーザーに指示します。**

```
## sshd_config
IgnoreUserKnownHosts no
```

     ユーザーへの指示については、40 ページの「Secure Shell で使用する公開鍵と非公開鍵のペアを生成する方法」を参照してください。

   - **クライアントの公開鍵をサーバーにコピーします。**

     ホスト鍵は、/etc/ssh ディレクトリに格納されます。鍵は、通常、最初のブート時に sshd デーモンによって生成されます。

     a. **サーバー上の /etc/ssh/ssh_known_hosts ファイルに鍵を追加します。**

        クライアントで、バックスラッシュなしで 1 行に次のコマンドを入力します。

```
# cat /etc/ssh/ssh_host_dsa_key.pub | ssh RemoteHost \
'cat >> /etc/ssh/ssh_known_hosts && echo "Host key copied"'
```

注記 - ホスト鍵がサーバーに存在しない場合、Secure Shell を使用すると次のようなエラーメッセージが生成されます。

```
Client and server could not agree on a key exchange algorithm:
client "diffie-hellman-group-exchange-sha256,diffie-hellman-group-
exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1",
server "gss-group1-sha1-toWM5Slw5Ew8Mqkay+al2g==". Make sure host keys
are present and accessible by the server process. See sshd_config(4)
description of "HostKey" option.
```

b. プロンプトが表示されたら、ログインパスワードを入力します。

ファイルがコピーされると、「Host key copied」というメッセージが表示されます。

c. `ssh_known_hosts` ファイル内のコピーされたエントリの前に *RemoteHost* を付加します。

`/etc/ssh/ssh_known_hosts` ファイルの各行は、スペースで区切られたフィールドで構成されています。

hostnames algorithm-name publickey comment

hostnames フィールドに *RemoteHost* を配置します。

```
## /etc/ssh/ssh_known_hosts File
RemoteHost  <copied entry>
```

例 2 ホストに基づく認証を設定する

次の例では、各ホストが Secure Shell サーバーおよびクライアントとして構成されます。一方のホストのユーザーが、他方のホストへの `ssh` 接続を開始できます。次の構成によって、各ホストがサーバーおよびクライアントの両方になります。

- ホストごとに、 Secure Shell 構成ファイルに次のエントリを入力します。

```
## /etc/ssh/ssh_config
HostBasedAuthentication yes
#
## /etc/ssh/sshd_config
HostBasedAuthentication yes
IgnoreRhosts no
```

- ホストごとに、`shosts.equiv` ファイルに他方のホストに対するエントリを入力します。

```
## /etc/ssh/shosts.equiv on system2
machine1

## /etc/ssh/shosts.equiv on system1
machine2
```

- 各ホストの公開鍵を、他方のホストの `/etc/ssh/ssh_known_hosts` ファイルに入力します。

```
## /etc/ssh/ssh_known_hosts on system2
… machine1

## /etc/ssh/ssh_known_hosts on system1
… machine2
```

- ユーザーは、両方のホストにアカウントを持ちます。たとえば、ユーザー Jane Doe の次の情報が表示されます。

```
## /etc/passwd on system1
jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh

## /etc/passwd on system2
jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
```

▼ SunSSH クライアントの X.509 証明書を生成する方法

ユーザーは認証局 (CA) から証明書を要求したり、独自の自己署名付き TA を生成したりできます。どちらの方法でも、ユーザーは公開鍵と非公開鍵のペアを生成し、非公開鍵を安全な場所に格納します。サードパーティーからの証明書を得るために、ユーザーは CA に要求を送信します。CA は要求に基づいてその非公開鍵を使用して証明書に署名し、署名される証明書にデジタル署名を付加します。CA は署名されたホスト証明書および独自の CA 証明書を元の要求者に送信します。これらの証明書を受け取ったら、ユーザーは、SunSSH サーバーで使用されるキーストア内にホスト証明書をインストールします。最後のステップで、ユーザーは SunSSH サーバーの管理者に TA 証明書を送信します。

鍵および証明書は、OpenSSL 証明書ディレクトリまたは PKCS #11 ソフトトークンキーストアに格納できます。この手順では、PKCS #11 ソフトトークンキーストアおよび自己署名付き TA を使用します。OpenSSL の使用方法については、例3「`openssl` を使用して SunSSH クライアント用の X.509 証明書を生成する」を参照してください。

1. **通常のユーザーとして、PIN (パスフレーズ) を初期化して PKCS #11 ソフトトークンキーストアに設定します。**

```
# pktool setpin
     Enter token passphrase: changeme
     Create new passphrase: xxxxxxxx
     Re-enter new passphrase:xxxxxxxx
     Passphrase changed.
```

PKCS #11 ソフトトークンキーストアは `/var/user/`*username* に証明書および鍵を格納するために使用されます。指定したパスフレーズを忘れないでください。

2. **`pktool gencert` コマンドを使用して、ソフトトークンキーストア内の自己署名付き証明書および鍵のペアを生成します。**

   結果として得られる証明書はルート証明書です。この証明書をサーバー管理者に送信し、これがユーザーの証明書に署名するために使用されます。

```
# pktool gencert keystore=pkcs11 \
   label=ExampleUserTAcert subject="CN=ExUserTA" serial=0x01
     Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

   各情報の意味は次のとおりです。

   - `keystore` はキーストアのタイプを指定し、ここでは `pkcs11` です。
   - `label` は、鍵のペアとトラストアンカー証明書のラベルを指定します。ラベル *label* の値を使用して、キーストア内の証明書を見つけることができます。
   - `subject` は、トラストアンカー証明書の識別名を指定します。これは認証中に証明書を識別するために使用できます。
   - `serial` は、トラストアンカー証明書の一意のシリアル番号を指定します。

3. **証明書署名要求 (CSR) を生成します。**

   前のステップで作成したルート証明書の非公開鍵を使用してこの CSR に署名します。

```
#  pktool gencsr keystore=pkcs11 label=ExampleUserOwnCert \
  outcsr=ExampleUserOwnCert.csr subject="CN=ExampleUserOwnCert"
     Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

   各情報の意味は次のとおりです。

   - `label` は、キーストア内の非公開鍵を検索するために使用できるラベルを指定します
   - `outcsr` は、書き込み先の出力 CSR ファイル名を指定します。
   - `subject` は、証明書要求の識別名を指定します。この名前は、このユーザーの証明書を識別するために使用できます。

4. **ルート証明書の非公開鍵を使用して CSR に署名し、ユーザーの証明書を作成します。**

```
# pktool signcsr keystore=pkcs11 signkey=ExampleUserOwnCert csr=ExampleUserOwnCert.csr \
  outcert=ExampleUserOwnCert.cert serial=0x02 issuer="CN=ExampleUserOwnCert"
     Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

   各情報の意味は次のとおりです。

   - `signkey` は、署名される証明書のラベルを指定します。
   - `csr` は、署名される入力 CSR を指定します。
   - `outcert` は、署名付きユーザー証明書の出力ファイル名を指定し、ここでは `Exampleupriv.cert` です
   - `serial` は、ユーザー証明書の一意のシリアル番号を指定します。
   - `issuer` は、TA の識別名が付いた発行者名をユーザー証明書に追加します

5. **ユーザー証明書をソフトトークンキーストアに追加します。**

```
# pktool import keystore=pkcs11 \
  infile=ExampleUserOwnCert.cert label=SignedExampleUserOwnCert
```

各情報の意味は次のとおりです。

- `infile` は、インポートされるユーザー証明書のファイル名を指定します。
- `label` は、このシステム上のこのユーザー証明書の一意のラベルを指定します。

6. **ソフトトークンキーストアからルート証明書をエクスポートします。**
この証明書は、次のステップでサーバー管理者に電子メールで送信されます。

```
# pktool export keystore=pkcs11 outfile=ExampleUserta.cert \
  objtype=cert label=ExampleUserSigned outformat=pem
     Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

各情報の意味は次のとおりです。

- `outfile` は、ルート証明書の出力ファイル名を指定し、ここでは `ExampleUserta.cert` です
- `objtype` は、オブジェクトのクラス、つまり証明書を指定します。
- `label` は、キーストア内のルート証明書の `label` 属性の値を指定します。
- `outformat` は、ルート証明書の出力形式 `pem` を指定します。

7. **ルート証明書をサーバー管理者に電子メールで送信します。**
証明書の内容を電子メールにコピー&ペーストでき、またはファイルを電子メールに添付することもできます。これは公開証明書であるため、非公開情報は含まれていません。
これで操作は完了で、サーバー管理者がサーバーのルート証明書をシステムの管理者に送信するまで待機します。使用しているシステム上で、管理者はサーバーのルート証明書をインストールし、X.509 証明書を認証に使用できるように `sshd` デーモンをふたたび有効化します。

8. **通常のユーザーとして、クライアントからサーバーにログインします。**
ユーザーはサーバーによって認証される識別情報としてユーザー証明書ラベルを使用します。

```
# ssh -o IdentityFile="pkcs11:object=SignedExampleUserOwnCert;token=Sun Software PKCS#11
 softtoken" username@hostname
   Enter PIN for 'Sun Software PKCS#11 softtoken': xxxxxxxx
```

各情報の意味は次のとおりです。

- `IdentityFile` は、PKCS #11 URI スキームによってソフトトークンキーストア内のユーザー証明書を指定します。ssh(1) のマニュアルページを参照してください。
- `object` は、ソフトトークンキーストア内の証明書オブジェクトのラベルを指定します

- token は、そのトークンラベルによって PKCS #11 トークンを指定します

9. **サーバー上の root として、/etc/ssh/cert ディレクトリを作成します。**
次に、その中に Exampleta.cert 証明書ファイルを配置します。このトラストアンカー証明書は、クライアント側からユーザー証明書を検証するために使用されます。

```
#  cd ; mkdir /etc/ssh/cert
  mv Exampleta.cert /etc/ssh/cert
```

例 3 openssl を使用して SunSSH クライアント用の X.509 証明書を生成する

この例では、root として openssl コマンドを使用して、自己署名付き証明書と鍵のペアを生成します。

```
# cd $HOME
# openssl req \
> -x509 -new \
> -subj "/C=CZ/ST=Here region/L=Here /CN=ExampleTAkey" \
> -newkey rsa:2048 -keyout ExampleTAkey.pem \
> -out exampleTAkey.pem
Generating a 2048 bit RSA private key
.+++
........+++
writing new private key to 'exampleTAkey.pem'
Enter PEM pass phrase: xxxxxxxx

Verifying - Enter PEM pass phrase:  xxxxxxxx
```

▼ SunSSH サーバー用の X.509 証明書を生成する方法

始める前に クライアント上で、.ssh ディレクトリ内の known_hosts ファイルには、クライアントが認証するホストの公開鍵を識別するエントリが含まれていないことを確認します。

1. **サーバー上で pktool setpin コマンドを使用して、PIN (パスフレーズ) を初期化して PKCS #11 ソフトトークンキーストアに設定します。**

```
#  pktool setpin
     Enter token passphrase: changeme
     Create new passphrase:
     Re-enter new passphrase:
     Passphrase changed.
```

PKCS #11 ソフトトークンキーストアは証明書および鍵を保存するために使用されます。ソフトトークンキーストアのデフォルトのパスフレーズは changeme です。

2. **サーバー上で、pktool gencert コマンドを使用してソフトトークンキーストア内に自己署名付き証明書と鍵のペアを生成します。**
結果として生成された証明書はこの手順でトラストアンカー証明書として使用されます。gencert サブコマンドを実行すると、ソフトトークンキーストア用の PIN を入力するように求めるプロンプトが表示されます。

```
#  pktool gencert keystore=pkcs11 label=authority subject="CN=authority" serial=0x01
     Enter PIN for Sun Software PKCS#11 softtoken:
```

各情報の意味は次のとおりです。

- keystore はキーストアのタイプを指定し、ここでは pkcs11 です
- label は、鍵のペアとトラストアンカー証明書のラベルを指定します
- subject は、トラストアンカー証明書の識別名を指定します
- serial は、トラストアンカー証明書の一意のシリアル番号を指定します

3. サーバー上で、pktool export コマンドを使用して、ソフトトークンキーストアからトラストアンカー証明書をエクスポートします。

export サブコマンドを実行すると、ソフトトークンキーストア用の PIN を入力するように求めるプロンプトが表示されます。

```
#  pktool export keystore=pkcs11 outfile=ta.cert \
  objtype=cert label=authority outformat=pem
     Enter PIN for Sun Software PKCS#11 softtoken:
```

各情報の意味は次のとおりです。

- outfile は、トラストアンカー証明書の出力ファイル名を指定し、ここでは ta.cert です
- objtype は、トラストアンカー証明書のクラスを指定します
- label は、トラストアンカー証明書のラベルを指定します
- outformat は、トラストアンカー証明書の出力形式を指定します

4. サーバー上の root として、/etc/ssh/cert ディレクトリを作成します。

次に、その中に Exampleta.cert 証明書ファイルを配置します。このトラストアンカー証明書は、クライアント側からユーザー証明書を検証するために使用されます。

```
#  mv Exampleta.cert /etc/ssh/cert
```

5. サーバー上で、pktool gencsr コマンドを使用して CSR を生成します。

gencsr サブコマンドを実行すると、ソフトトークンキーストア用の PIN を入力するように求めるプロンプトが表示されます。

```
#  pktool gencsr keystore=pkcs11 label=host outcsr=host.csr subject="CN=<HOSTNAME>"
     Enter PIN for Sun Software PKCS#11 softtoken:
```

各情報の意味は次のとおりです。

- label は、ソフトトークンキーストア内で作成される非公開鍵のラベルを指定します
- outcsr は、出力 CSR ファイルの名前を指定します
- subject は、証明書要求の識別名を指定します

6. **サーバー上で `pktool signcsr` コマンドを使用して、ホスト証明書を作成するためにトラストアンカー非公開鍵を使用して CSR に署名します。**

   `signcsr` サブコマンドを実行すると、ソフトトークンキーストア用の PIN を入力するように求めるプロンプトが表示されます。

   ```
   #  pktool signcsr keystore=pkcs11 signkey=authority csr=host.csr \
     outcert=host.cert serial=0x03 issuer="CN=authority"
        Enter PIN for Sun Software PKCS#11 softtoken:
   ```

   各情報の意味は次のとおりです。

   - `signkey` は、トラストアンカー非公開鍵のラベルを指定します
   - `csr` は、署名される入力 CSR を指定します
   - `outcert` は、ホスト証明書の出力ファイル名を指定し、ここでは `host.cert` です
   - `serial` は、ホスト証明書の一意のシリアル番号を指定します
   - `issuer` は、トラストアンカーの識別名が付いた発行者名をホスト証明書に指定します

7. **サーバー上で `pktool import` コマンドを使用して、ホスト証明書をソフトトークンキーストアにインポートします。**

   ```
   #  pktool import keystore=pkcs11 infile=host.cert label=host
   ```

   各情報の意味は次のとおりです。

   - `infile` は、インポートされるホスト証明書のファイル名を指定します
   - `label` は、ホスト証明書のラベルを指定します

8. **`root` 役割のクライアント上で、鍵管理フレームワーク (KMF) ポリシーデータベースファイルを作成します。**

   ```
   #  kmfcfg create dbfile=/etc/ssh/policy.xml policy=ssh ta-name=search mapper-name=cn
   ```

   各情報の意味は次のとおりです。

   - `dbfile` は、ポリシーデータベースファイルの名前を指定します
   - `policy` は、ポリシーデータベースファイルに作成されるポリシーレコードの名前を指定します
   - `ta-name` は、ここでは `search` 値を指定し、これは KMF ポリシーが、証明書の発行者名と一致する証明書を検証用に使用することを示します
   - `mapper` は、その共通名 (cn) 属性を使用して、証明書を名前にマッピングすることを指定します

9. **サーバー上で、root として、ソフトトークンキーストアのパスフレーズを含む PIN ファイルを作成します。**

   ```
   #  printf "keystore-passphrase" > /etc/ssh/pinfile
   ```

10. **サーバー上で `root` 役割として `HostKey` エントリを `sshd_config` ファイルに追加します。**

```
#  pfedit /etc/ssh/sshd_config
     ...
     HostKey       pkcs11:object=host;token=Sun Metaslot;pinfile=/etc/ssh/pinfile
```

ここで、PKCS #11 URI スキームの属性は次のとおりです。

- `object` は、ソフトトークンキーストア内の証明書オブジェクトのラベルを指定します
- `token` は、そのトークンラベルによって PKCS #11 トークンを指定します
- `pinfile` は、ソフトトークンキーストアのパスフレーズを指定します

`HostKey` キーワードは、PKCS #11 URI スキームのホスト証明書を指定します。sshd(1M) のマニュアルページを参照してください。

11. **サーバー上で、`ssh` サービスインスタンスを無効にして有効にします。**

SSH デーモンはホスト証明書をホスト鍵として使用します。

```
#  svcadm disable svc:/network/ssh
#  svcadm enable svc:/network/ssh
```

12. **クライアントから、ユーザーは `TrustedAnchorKeystore`、`KMFPolicyDatabase`、および `KMFPolicyName` オプションを使用してサーバーにログインします。**

これらのキーワードは、ホスト証明書を検証するために使用するトラストアンカー証明書を指定します。

```
#  ssh -o TrustedAnchorKeystore=/etc/ssh/cert -o KMFPolicyDatabase=/etc/ssh/policy.xml
    -o KMFPolicyName=ssh <USERNAME>@<HOSTNAME>
```

各情報の意味は次のとおりです。

- `TrustedAnchorKeystore` は、トラストアンカーの証明書を含むディレクトリを指定します
- `KMFPolicyDatabase` は、ポリシーデータベースファイルのファイル名を指定します
- `KMFPolicyName` は、使用するポリシーの名前を指定します

▼ Secure Shell のポート転送を構成する方法

ポート転送によって、ローカルポートをリモートシステムに転送できるようになります。指定すると、ソケットはローカル側で、そのポートを待機します。また、リモート側のポートを指定することもできます。

注記 - Secure Shell のポート転送では、TCP 接続を使用する必要があります。Secure Shell は、ポート転送のための UDP 接続をサポートしていません。

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

1. **ポート転送ができるようにリモートサーバーで Secure Shell の設定を構成します。**
   `/etc/ssh/sshd_config` ファイルで `AllowTcpForwarding` の値を `yes` に変更します。

   ```
   # Port forwarding
   AllowTcpForwarding yes
   ```

2. **Secure Shell サービスを再起動します。**

   ```
   remoteHost# svcadm restart network/ssh:default
   ```

   永続的なサービスの管理については、『Oracle Solaris 11.3 でのシステムサービスの管理』 の 第 1 章, 「サービス管理機能の概要」および svcadm(1M) のマニュアルページを参照してください。

3. **ポート転送が使用できることを確認します。**

   ```
   remoteHost# /usr/bin/pgrep -lf sshd
   1296 ssh -L 2001:remoteHost:23 remoteHost
   ```

▼ Secure Shell デフォルトのユーザーおよびホスト例外を作成する方法

この手順では、条件付きの `Match` ブロックを `/etc/ssh/sshd_config` ファイルのグローバルセクションのあとに追加します。`Match` ブロックのあとに続くキーワードと値のペアは、一致するものとして指定されたユーザー、グループ、ホスト、またはアドレスの例外を示しています。

始める前に `solaris.admin.edit/etc/ssh/sshd_config` 承認が割り当てられている管理者になる必要があります。デフォルトでは、`root` 役割がこの承認を持っています。詳細は、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 「割り当てられている管理権利の使用」を参照してください。

1. **編集のために `/etc/ssh/sshd_config` ファイルを開きます。**

   ```
   # pfedit /etc/ssh/sshd_config
   ```

2. **デフォルト設定とは異なる Secure Shell 設定を使用するようにユーザー、グループ、ホスト、またはアドレスを構成します。**

それらの Match ブロックをグローバル設定のあとに配置します。

注記 - このファイルのグローバルセクションには、デフォルト設定が常に一覧表示されるわけではありません。それらのデフォルトについては、sshd_config(4) のマニュアルページを参照してください。

たとえば、TCP 転送の使用を許可すべきでないユーザーが含まれている場合があります。この構成では、グループ public のユーザーと、名前が test で始まるユーザーはすべて TCP 転送を使用できません。

```
## sshd_config file
## Global settings

# Example (reflects default settings):
#
# Host *
#   ForwardAgent no
#   ForwardX11 no
#   PubkeyAuthentication yes
#   PasswordAuthentication yes
#   FallBackToRsh no
#   UseRsh no
#   BatchMode no
#   CheckHostIP yes
#   StrictHostKeyChecking ask
#   EscapeChar ~
Match Group public
AllowTcpForwarding no
Match User test*
AllowTcpForwarding no
```

Match ブロックの構文については、sshd_config(4) のマニュアルページを参照してください。

▼ sftp ファイルのための切り離されたディレクトリを作成する方法

この手順では、sftp 転送専用に作成した sftponly ディレクトリを構成します。ユーザーは、このディレクトリの外部のどのファイルまたはディレクトリも表示できません。

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

1. **Secure Shell サーバー上で、chroot 環境として切り離されたディレクトリを作成します。**

```
# groupadd sftp
```

```
# useradd -m -G sftp -s /bin/false sftponly
# chown root:root /export/home/sftponly
# mkdir /export/home/sftponly/WWW
# chown sftponly:staff /export/home/sftponly/WWW
```

この構成で、/export/home/sftonly は、root アカウントのみがアクセスできる chroot ディレクトリです。ユーザーには、sftponly/WWW サブディレクトリへの書き込み権があります。

2. **そのままサーバー上で、sftp グループのための Match ブロックを構成します。**

   /etc/ssh/sshd_config ファイル内で sftp subsystem エントリを見つけ、このファイルを次のように変更します。

```
# pfedit /etc/ssh/sshd_config
...
# sftp subsystem
#Subsystem       sftp    /usr/lib/ssh/sftp-server
Subsystem       sftp    internal-sftp
...
## Match Group for Subsystem
## At end of file, to follow all global keywords
Match Group sftp
ChrootDirectory %h
ForceCommand internal-sftp
AllowTcpForwarding no
```

   次の変数を使用して chroot のパスを指定できます。

   - %h – ホームディレクトリを指定します。
   - %u – 認証されたユーザーのユーザー名を指定します。
   - %% – % 記号をエスケープします。

3. **クライアント上で、構成が正しく機能することを確認します。**

   実際の chroot 環境内のファイルは異なる可能性があります。

```
root@client:~# ssh sftponly@server
This service allows sftp connections only.
Connection to server closed.      シェルアクセスなし、sftp が適用される。
root@client:~# sftp sftponly@server
sftp> pwd      sftp アクセスが付与される
Remote working directory: /     chroot ディレクトリがルートディレクトリのように見える
sftp> ls
WWW             local.cshrc     local.login     local.profile
sftp> get local.cshrc
Fetching /local.cshrc to local.cshrc
/local.cshrc    100%  166     0.2KB/s   00:00     ユーザーは内容を読み取れる
sftp> put /etc/motd
Uploading /etc/motd to /motd
Couldn't get handle: Permission denied     ユーザーは / directory に書き込めない
sftp> cd WWW
sftp> put /etc/motd
Uploading /etc/motd to /WWW/motd
/etc/motd     100%  118     0.1KB/s   00:00     ユーザーは WWW ディレクトリに書き込める
sftp> ls -l
-rw-r--r--    1 101  10    118 Jul 20 09:07 motd     転送成功
sftp>
```

Secure Shell の使用

このセクションでは、ユーザーが Secure Shell に習熟するための手順について説明します。

Secure Shell の使用 (タスクマップ)

次のタスクマップでは、ユーザーが Secure Shell を使用する際の手順を示します。

タスク	説明	手順
公開鍵と非公開鍵のペアを作成します。	公開鍵認証を必要とするサイトの Secure Shell にアクセスできるようにします。	40 ページの「Secure Shell で使用する公開鍵と非公開鍵のペアを生成する方法」
パスフレーズを変更します。	非公開鍵を認証するフレーズを変更します。	41 ページの「Secure Shell の公開鍵のパスフレーズを変更する方法」
Secure Shell を使用してログインします。	リモートログイン時に、暗号化された Secure Shell 通信を行うことができるようにします。	42 ページの「Secure Shell を使用してリモートホストにログインする方法」
パスワードのプロンプトを表示せずに Secure Shell にログインします。	パスワードを Secure Shell に提供するエージェントを使用してログインできるようにします。	43 ページの「Secure Shell でのパスワードのプロンプトを減らす方法」
`root` として Secure Shell にログインします。	ZFS の `send` および `receive` コマンドのための `root` としてのログインを有効にします。	44 ページの「Secure Shell を使用して ZFS をリモートで管理する方法」
Secure Shell のポート転送を使用します。	TCP 経由の Secure Shell 接続で使用するローカルポートまたはリモートポートを指定します。	46 ページの「Secure Shell のポート転送を使用する方法」
Secure Shell を使用してファイルをコピーします。	ホスト間で安全にファイルをコピーします。	47 ページの「Secure Shell を使用してファイルをコピーする方法」
ファイアウォールの内側のホストからファイアウォールの外側のホストに安全に接続します。	HTTP または SOCKS5 と互換性のある Secure Shell のコマンドを使用して、ファイアウォールで分断されているホスト間を接続します。	49 ページの「ファイアウォール外部のホストへのデフォルトの Secure Shell 接続を設定する方法」

▼ Secure Shell で使用する公開鍵と非公開鍵のペアを生成する方法

使用するサイトがホストに基づく認証またはユーザーの公開鍵認証を実装しているときは、ユーザーは公開鍵と非公開鍵のペアを生成する必要があります。追加のオプションについては、ssh-keygen(1) のマニュアルページを参照してください。

始める前に ホストに基づく認証が構成されているかどうかをシステム管理者に確認します。

1. **鍵の生成プログラムを起動します。**

```
mySystem$ ssh-keygen -t rsa
Generating public/private rsa key pair.
…
```

`-t` はアルゴリズムの種類で、`rsa`、`dsa`、`rsa1` のいずれかです。

2. **鍵が格納されるファイルのパスを指定します。**
デフォルトでは、RSA v2 の鍵を表すファイル名 `id_rsa` がカッコ内に表示されます。このファイルを選択するときは、Return キーを押します。代わりのファイル名を入力することもできます。

```
Enter file in which to save the key (/home/username/.ssh/id_rsa):     <Return キーを押す>
```

文字列 `.pub` を非公開鍵のファイル名に追加すると、自動的に公開鍵のファイル名になります。

3. **鍵に使用するパスフレーズを入力します。**
このパスフレーズは、非公開鍵を暗号化するときに使用されます。空文字列入力は極力避けてください。入力したパスフレーズは表示されません。

```
Enter passphrase (empty for no passphrase): passphrase
```

4. **確認のためにパスフレーズを再入力します。**

```
Enter same passphrase again: passphrase
Your identification has been saved in /home/username/.ssh/id_rsa
Your public key has been saved in /home/username/.ssh/id_rsa.pub
The key fingerprint is:
0e:fb:3d:57:71:73:bf:58:b8:eb:f3:a3:aa:df:e0:d1 username@mySystem
```

5. **鍵ファイルへのパスが正しいことを確認します。**

```
$ ls ~/.ssh
id_rsa
id_rsa.pub
```

この時点で公開鍵と非公開鍵のペアが作成されました。

6. **リモートホストにログインします。**
サイトの認証方式に基づいて次のいずれかのログインステップを選択します。

- **ホストに基づく認証の場合、ローカルホストの公開鍵をリモートホストにコピーします。**

  a. **バックスラッシュなしで 1 行に次のコマンドを入力します。**

  ```
  $ cat /etc/ssh/ssh_host_dsa_key.pub | ssh RemoteHost \
  'cat >> ~./ssh/known_hosts && echo "Host key copied"'
  ```

  b. **プロンプトが表示されたら、ログインパスワードを入力します。**

  ```
  Enter password: password
  Host key copied
  $
  ```

  リモートホストにログインできるようになっています。詳細については、42 ページの「Secure Shell を使用してリモートホストにログインする方法」を参照してください。

- **公開鍵によるユーザー認証の場合、リモートホストの `authorized_keys` ファイルに情報を入力します。**

  a. **公開鍵をリモートホストにコピーします。**

  バックスラッシュなしで 1 行に次のコマンドを入力します。

  ```
  mySystem$ cat $HOME/.ssh/id_rsa.pub | ssh myRemoteHost \
  'cat >> .ssh/authorized_keys && echo "Key copied"'
  ```

  b. **プロンプトが表示されたら、ログインパスワードを入力します。**

  ```
  Enter password: password
  Key copied
  mySystem$
  ```

7. **(オプション) パスフレーズを要求するプロンプトを表示しないようにします。**

   43 ページの「Secure Shell でのパスワードのプロンプトを減らす方法」を参照してください。詳細は、`ssh-agent(1)` および `ssh-add(1)` のマニュアルページを参照してください。

▼ Secure Shell の公開鍵のパスフレーズを変更する方法

次のコマンドは、実際の非公開鍵ではなく、非公開鍵の認証メカニズム (パスフレーズ) を変更するものです。詳細は、`ssh-keygen(1)` のマニュアルページを参照してください。

- **パスフレーズを変更します。**
  ssh-keygen コマンドを-p オプションを指定して入力し、プロンプトに答えます。

  ```
  mySystem$ ssh-keygen -p
  Enter file which contains the private key (/home/username/.ssh/id_rsa):     <Return キーを押す>
  Enter passphrase (empty for no passphrase): passphrase
  Enter same passphrase again: passphrase
  ```

  -p は、非公開鍵ファイルのパスフレーズの変更を要求します。

▼ Secure Shell を使用してリモートホストにログインする方法

1. **Secure Shell セッションを開始します。**
   ssh コマンドを入力して、リモートホストとログインの名前を指定します。

   ```
   mySystem$ ssh myRemoteHost -l username
   ```

2. **プロンプトが表示されたら、リモートホスト鍵の信頼性を確認します。**
   リモートホストの信頼性を尋ねるプロンプトが表示される場合があります。

   ```
   The authenticity of host 'myRemoteHost' can't be established....Are you sure you want to
    continue connecting(yes/no)?
   ```

   このプロンプトは、リモートホストに初めて接続する場合には正常なプロンプトです。

   - **リモートホストの信頼性を確認できない場合は、no と入力してシステム管理者に連絡します。**

     ```
     Are you sure you want to continue connecting(yes/no)? no
     ```

     システム管理者は、大域の /etc/ssh/ssh_known_hosts ファイルを更新する責任があります。更新された ssh_known_hosts ファイルでは、このようなプロンプトは表示されません。

   - **リモートホストの信頼性を確認したら、プロンプトに答えて次の手順に進みます。**

     ```
     Are you sure you want to continue connecting(yes/no)? yes
     ```

3. **Secure Shell に対して自分を認証します。**

   a. **プロンプトが表示されたら、自分のパスフレーズを入力します。**

      ```
      Enter passphrase for key '/home/username/.ssh/id_rsa': passphrase
      ```

   b. **プロンプトが表示されたら、アカウントのパスワードを入力します。**

```
username@myRemoteHost's password: password
Last login: Wed Sep  7 09:07:49 2016 from mySystem
Oracle Corporation      SunOS 5.11  11.3     September 2016
myRemoteHost$
```

4. **リモートホストでトランザクションを実行します。**

   ユーザーが送信するコマンドはすべて暗号化されます。ユーザーが受信する応答はすべて暗号化されます。

5. **Secure Shell の接続を切断します。**

   終了したら、**exit** を入力するか、通常の方法でシェルを終了します。

```
myRemoteHost$ exit
myRemoteHost$ logout
Connection to myRemoteHost closed
mySystem$
```

例 4 Secure Shell でのリモート GUI の表示

この例では、jdoe は両方のシステム上の初期ユーザーであり、ソフトウェアインストールに関連する権利プロファイルが割り当てられています。X11Forwarding キーワードのデフォルト値は引き続き yes であり、xauth パッケージはリモートシステム上にインストールされています。

```
$ ssh -l jdoe -X myRemoteHost
jdoe@myRemoteHost's password: password
Last login: Wed Sep  7 09:07:49 2016 from myLocalHost
Oracle Corporation      SunOS 5.11 11.3       September 2016
myRemoteHost$ useful-app-with-GUI &
```

▼ Secure Shell でのパスワードのプロンプトを減らす方法

Secure Shell の使用に際してパスフレーズやパスワードを入力しない場合は、エージェントデーモンを使用できます。ホストごとにアカウントが異なる場合は、セッションに必要な非公開鍵を追加します。

エージェントデーモンの起動は、次の手順で説明するように、必要に応じて手動で行うことができます。

1. **エージェントデーモンを起動します。**

```
mySystem$ eval `ssh-agent`
Agent pid 9892
```

2. **エージェントデーモンが起動していることを確認します。**

```
mySystem$ pgrep ssh-agent
9892
```

3. **使用する非公開鍵をエージェントデーモンに追加します。**

```
mySystem$ ssh-add
Enter passphrase for /home/username/.ssh/id_rsa: passphrase
Identity added: /home/username/.ssh/id_rsa(/home/username/.ssh/id_rsa)
mySystem$
```

4. **Secure Shell セッションを開始します。**

```
mySystem$ ssh myRemoteHost -l username
```

パスフレーズを要求するプロンプトは表示されません。

例 5 ssh-add オプションを使用する

この例では、jdoe が 2 つの鍵をエージェントデーモンに追加します。セッションの最後に、jdoe は、エージェントデーモンからすべての鍵を削除します。

```
mySystem$ ssh-agent
mySystem$ ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa: passphrase
Identity added: /home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa)
mySystem$ ssh-add /home/jdoe/.ssh/id_dsa
Enter passphrase for /home/jdoe/.ssh/id_dsa: passphrase
Identity added:
/home/jdoe/.ssh/id_dsa(/home/jdoe/.ssh/id_dsa)

mySystem$ ssh-add -l
SHA256:OX5V4xxoVozwqdZfAbykwawMuwVM+sfc+ThMeai8r9
/home/jdoe/.ssh/id_rsa(RSA)
SHA256:OX5V4xxoVozwqdZfAbykwawMuwVM+sfc+ThMeai8r9
/home/jdoe/.ssh/id_dsa(DSA)
```

ユーザーが *Secure Shell* トランザクションを実行

```
mySystem$ ssh-add -D
Identity removed:
/home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa.pub)
/home/jdoe/.ssh/id_dsa(DSA)
```

▼ Secure Shell を使用して ZFS をリモートで管理する方法

デフォルトでは、root 役割は Secure Shell を使用してリモートログインできません。従来、root は、重要なタスク (ZFS プールデータのリモートシステム上のストレージへの送信など) に Secure Shell を使用してきました。この手順では、root 役割は、リモートの ZFS 管理者として行動できるユーザーを作成します。

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

1. **両方のシステム上にユーザーを作成します。**

   たとえば、zfsroot ユーザーを作成し、パスワードを提供します。

   ```
   source # useradd -c "Remote ZFS Administrator" -u 1201 -d /home/zfsroot zfsroot
   source # passwd zfsroot
   New Password: password
   Re-enter new password: password
   passwd: password successfully changed for zfsroot
   #

   dest # useradd -c "Remote ZFS Administrator" -u 1201 -d /home/zfsroot zfsroot
   dest # passwd zfsroot
   ...
   ```

   zfsroot ユーザーは、両方のシステム上で同様に定義されている必要があります。

2. **両方のシステム上で、zfsroot に ZFS File Management 権利プロファイルを割り当てます。**

   ```
   source # usermod -P +'ZFS File System Management' -S files zfsroot
   dest # usermod -P +'ZFS File System Management' -S files zfsroot
   ```

3. **この権利プロファイルが宛先システムの zfsroot に割り当てられていることを確認します。**

   ```
   dest # profiles zfsroot
   zfsroot:
   ZFS File System Management
   Basic Solaris User
   All
   ```

4. **Secure Shell 認証のためのユーザーの鍵のペアを作成します。**

   この鍵のペアは、ソースシステム上で作成されます。次に、公開鍵が、宛先システム上の zfsroot ユーザーにコピーされます。

   a. **鍵のペアを生成し、それをファイル id_migrate 内に格納します。**

      ```
      # ssh-keygen -t rsa -P "" -f ~/id_migrate
      Generating public/private rsa key pair.
      Your identification has been saved in /root/id_migrate.
      Your public key has been saved in /root/id_migrate.pub.
      The key fingerprint is:
      SHA256:BLNj0v9...izsQ cpltester@Local
      The key's randomart image is:
      +---[RSA 2048]----+
      |      o       .=B|
      ...
      ```

   b. **鍵のペアの公開の部分を宛先システムに送信します。**

      ```
      # scp ~/id_migrate.pub zfsroot@dest:
      ```

```
The authenticity of host 'dest (10.134.76.126)' can't be established.
RSA key fingerprint is 44:37:ab:4e:b7:2f:2f:b8:5f:98:9d:e9:ed:6d:46:80.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'dest,10.134.76.126' (RSA) to the list of known hosts.
Password:
id_migrate.pub 100% |*****************************| 399 00:00
```

5. **宛先システム上で、鍵のペアの公開の部分を非公開の `/home/zfsroot/.ssh` ディレクトリに移動します。**

```
root@dest # su - zfsroot
Oracle Corporation      SunOS 5.11      11.1    May 2012
zfsroot@dest $ mkdir -m 700 .ssh
zfsroot@dest $ cat id_migrate.pub >> .ssh/authorized_keys
```

6. **構成が正しく機能することを確認します。**

```
root@source# ssh -l zfsroot -i ~/id_migrate dest \
pfexec /usr/sbin/zfs snapshot zones@test
root@source# ssh -l zfsroot -i ~/id_migrate dest \
pfexec /usr/sbin/zfs destroy zones@test
```

7. **(オプション) スナップショットを作成し、そのデータを複製できることを確認します。**

```
root@source# zfs snapshot -r rpool/zones@migrate-all
root@source#  zfs send -rc rpool/zones@migrate-all | \
ssh -l zfsroot -i ~/id_migrate dest pfexec /usr/sbin/zfs recv -F zones
```

8. **(オプション) `zfsroot` アカウントを ZFS の管理に使用する機能を削除します。**

```
root@dest# usermod -P -'ZFS File System Management' zfsroot
root@dest#  su - zfsroot
zfsroot@dest#  cp .ssh/authorized_keys .ssh/authorized_keys.bak
zfsroot@dest#  grep -v root@source .ssh/authorized_keys.bak> .ssh/authorized_keys
```

▼ Secure Shell のポート転送を使用する方法

リモートホストに転送されるローカルポートを指定することができます。指定すると、ソケットはローカル側で、そのポートを待機します。このポートからリモートホストへの接続は、セキュアなチャネルを介して行われます。たとえば、ポート 143 を指定すれば、IMAP4 で電子メールをリモートで取得できます。また、リモート側のポートを指定することもできます。

始める前に ポート転送を使用するために、管理者は、リモートの Secure Shell サーバーでのポート転送を有効にする必要があります。詳細については、35 ページの「Secure Shell のポート転送を構成する方法」を参照してください。

- **セキュアなポート転送をリモートポートからローカルポートに設定するか、ローカルポートからリモートポートに設定します。**

- **ローカルポートをリモートポートからのセキュアな通信の受信側として設定するには、両方のポートを指定します。**

  リモートからの通信を待機するローカルポートを指定します。また、通信を転送するリモートホストとリモートポートを指定します。

  ```
  mySystem$ ssh -L localPort:remoteHost:remotePort
  ```

- **リモートポートをローカルポートからのセキュアな接続の受信側として設定するには、両方のポートを指定します。**

  リモートからの通信を待機するリモートポートを指定します。また、通信を転送するローカルホストとローカルポートを指定します。

  ```
  mySystem$ ssh -R remotePort:localhost:localPort
  ```

例 6 ローカルポート転送を使用してメールを受信する

次の例は、ローカルポート転送を使用して、リモートサーバーからのメールを安全に受信する方法を示しています。

```
mySystem$ ssh -L 9143:myRemoteHost:143 myRemoteHost
```

このコマンドは、`myLocalHost` のポート `9143` からポート `143` に接続を転送します。ポート `143` は、`myRemoteHost` の IMAP v2 のサーバーポートです。ユーザーがメールアプリケーションを起動するときは、`localhost:9143` にあるように、ユーザーは IMAP サーバーのローカルポート番号を指定します。

例 7 リモートポート転送を使用してファイアウォールの外部と通信する

この例では、エンタープライズ環境のユーザーが、外部ネットワーク上のホストから企業のファイアウォール内部のホストに接続を転送する方法を示しています。

```
mySystem$ ssh -R 9022:mySystem:22myOutsideHost
```

このコマンドは、`myOutsideHost` 上のポート `9022` から、`sshd` デーモンがローカルホスト上で待機するポートに接続を転送します。通常、待機ポートはポート 22 です。

```
myOutsideHost$ ssh -p 9022 localhost
mySystem$
```

▼ Secure Shell を使用してファイルをコピーする方法

次の手順では、`scp` コマンドを使用して、暗号化されたファイルをホスト間でコピーする方法を示します。暗号化されたファイルは、ローカルホストとリモートホストと

の間、または 2 つのリモートホスト間でコピーできます。scp コマンドは、認証を求めるプロンプトを表示します。詳細は、『Oracle Solaris 11.3 でのリモートシステムの管理』 の 「scp コマンドによるリモートコピー」および scp(1) のマニュアルページを参照してください。

セキュアなファイル転送プログラム sftp も使用できます。詳細は、sftp(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、例8「sftp コマンドを使用するときにポートを指定する」および『Oracle Solaris 11.3 でのリモートシステムの管理』 の 「ファイルをコピーするためのリモートシステムへのログイン (sftp)」を参照してください。

注記 - 監査サービスでは、ft 監査クラスを通じて SunSSH sftp トランザクションを監査できます。scp については、監査サービスで ssh セッションのアクセスおよび終了を監査できます。詳細は、『Oracle Solaris 11.3 での監査の管理』 の 「FTP および SFTP ファイル転送を監査する方法」を参照してください。

---

1. **セキュアなコピープログラムを起動します。**

   ソースファイル、リモートコピー先のユーザー名、およびコピー先ディレクトリを指定します。

   ```
   mySystem$ scp myfile.1 username@myRemoteHost:~
   ```

2. **プロンプトが表示されたら、パスフレーズを入力します。**

   ```
   Enter passphrase for key '/home/username/.ssh/id_rsa': passphrase
   myfile.1       25% |*******                     |    640 KB  0:20 ETA
   myfile.1
   ```

   パスフレーズを入力すると、出力の 2 行目のように進行状況インジケータが表示されます。進行状況インジケータには、次の項目が表示されます。

   - ファイル名
   - そのファイル全体に対する、転送が完了した量の割合 (%)
   - そのファイル全体に対する、転送が完了した量の割合を示すアスタリスク(*)
   - 転送が完了したデータの量
   - ファイル全体が転送されるまでの推定時間 (ETA)。推定残り時間

例 8 sftp コマンドを使用するときにポートを指定する

この例では、ユーザーは sftp コマンドで特定のポートを使用しようとしています。ユーザーは -o オプションを使用してポートを指定します。

```
$ sftp -o port=2222 guest@RemoteFileServer
```

▼ ファイアウォール外部のホストへのデフォルトの Secure Shell 接続を設定する方法

SunSSH を使用して、ファイアウォールの内側のホストからファイアウォールの外側のホストに接続できます。接続するには、構成ファイル内またはコマンド行オプションに ssh のプロキシコマンドを指定します。コマンド行オプションについては、例 9「Secure Shell コマンド行からのファイアウォール外部のホストへの接続」を参照してください。

個人用構成ファイル `~/.ssh/config` を使用して ssh の対話操作をカスタマイズできます。または、管理構成ファイル `/etc/ssh/ssh_config` の設定を使用できます。

ファイルは、2 種類のプロキシコマンドでカスタマイズできます。一方が HTTP 接続用、もう一方が SOCKS5 接続用です。詳細は、`ssh_config(4)` のマニュアルページを参照してください。

1. **構成ファイルにプロキシコマンドとホストを指定します。**

   次の構文を使用して、必要なプロキシコマンドとホストの数に応じて行を追加します。

   ```
   [Host outside-host]
   ProxyCommand proxy-command [-h proxy-server] \
   [-p proxy-port] outside-host|%h outside-port|%p
   ```

   `Host` *outside-host*

   プロキシコマンド指定を、コマンド行でリモートホスト名が指定された場合に限定します。*outside-host* でワイルドカードを使用した場合、一連のホストに対してプロキシコマンド指定が適用されます。

   proxy-command

   プロキシコマンドを指定します。

   次のいずれかを指定できます。

   - HTTP 接続の場合は、`/usr/lib/ssh/ssh-http-proxy-connect`
   - SOCKS5 接続の場合は、`/usr/lib/ssh/ssh-socks5-proxy-connect`

   -h *proxy-server* と -p *proxy-port*

   これらのオプションは、プロキシサーバーとプロキシポートをそれぞれ指定します。これらのオプションは、`HTTPPROXY`、`HTTPPROXYPORT`、`SOCKS5_PORT`、`SOCKS5_SERVER`、`http_proxy` などの、プロキシサーバーとプロキシポートを指定するどのような環境変数もオーバーライドします。`http_proxy` 変数は URL を指定します。これらのオプションを指定しない場合、適切な環境変数を設定する必要があります。詳細

は、ssh-socks5-proxy-connect(1) および ssh-http-proxy-connect(1) のマニュアルページを参照してください。

outside-host

接続先のホストを指定します。%h 代入引数を使うとコマンド行からホストを指定できます。

outside-port

接続先のポートを指定します。%p 代入引数を使うとコマンド行からポートを指定できます。Host *outside-host* オプションを使わずに %h と %p を指定した場合、ssh コマンドが呼び出されるたびに、引数に指定されたホストにプロキシコマンドが適用されます。

2. **外部のホストを指定して、Secure Shell を実行します。**

例:

```
mySystem$ ssh myOutsideHost
```

このコマンドは、個人用構成ファイル内で myOutsideHost のプロキシコマンド指定を検索します。指定が検出されない場合、このコマンドは、システム全体の構成ファイル /etc/ssh/ssh_config から検索します。プロキシコマンドが ssh コマンドに置き換わります。

例 9 Secure Shell コマンド行からのファイアウォール外部のホストへの接続

この例では、プロキシコマンドが Secure Shell 構成ファイルではなく ssh コマンド行で指定されます。

```
$ ssh -o'Proxycommand=/usr/lib/ssh/ssh-http-proxy-connect \
-h myProxyServer -p 8080 myOutsideHost 22' myOutsideHost
```

ssh コマンドの -o オプションには、プロキシコマンドを指定するコマンド行を入力できます。この例のコマンドは次のことを行います。

- ssh を HTTP プロキシコマンドに置き換える
- プロキシサーバーとして、ポート 8080 および myProxyServer を使用する
- myOutsideHost のポート 22 に接続する

♦ ♦ ♦ 第 2 章

2 Secure Shell リファレンス

この章では、Oracle Solaris の Secure Shell 機能に含まれる構成オプションについて説明します。この章の内容は次のとおりです。

Secure Shell を構成する手順については、第1章「Secure Shell の使用」を参照してください。

標準的な Secure Shell セッション

Secure Shell デーモン (`sshd`) は通常、ネットワークサービスが開始されるブート時にブートされます。デーモンは、SSH クライアントからの接続を待機します。Secure Shell セッションは、`ssh`、`scp`、または `sftp` コマンドが実行されると開始します。接続を受信するたびに、新しい `sshd` デーモンがフォークされます。フォークされたデーモンは、鍵の交換、暗号化、認証、コマンドの実行、およびクライアントとのデータ交換を行います。Secure Shell セッションの特性は、クライアント構成ファイルとサーバー構成ファイルによって決定されます。コマンド行引数は、構成ファイルの設定をオーバーライドできます。

SSH クライアントおよびサーバーは、相互に認証する必要があります。認証に成功したあと、ユーザーはコマンドをリモートで実行でき、システム間でデータをコピーできます。

Secure Shell でのセッションの特性

Secure Shell サーバー側の sshd デーモンの動作は、/etc/ssh/sshd_config ファイルのキーワードを設定することで変更できます。たとえば、sshd_config ファイルにより、サーバーにアクセスするときに使用する認証タイプを変更できます。サーバー側の動作は、sshd デーモンを起動するときに、コマンド行オプションによって変更することもできます。

クライアント側の動作は、Secure Shell のキーワードで変更できます。キーワードの優先順位は次のとおりです。

- コマンド行オプション
- ユーザーの構成ファイル (~/.ssh/config)
- システム全体の構成ファイル (/etc/ssh/ssh_config)

たとえば、ユーザーはコマンド行を使用して aes128-ctr を優先するシステム全体の構成 Ciphers の設定をオーバーライドできます。

```
$ ssh -c aes256-ctr,aes128-ctr,arcfour
```

これで、最初の暗号化方式 aes256-ctr が優先されるようになります。

Secure Shell での認証と鍵の交換

Secure Shell プロトコルは、SSH クライアントユーザー/ホストの認証、およびサーバーホストの認証をサポートしています。Secure Shell セッションを保護するために、暗号化鍵が交換されます。Secure Shell は、認証と鍵交換のためのさまざまな方法を提供します。その中には、任意のものもあります。クライアント認証メカニズムは、表 1に示されています。サーバーは、既知のホスト公開鍵を使用して認証されます。

認証については、Secure Shell はユーザー認証と、通常はパスワードを必要とする汎用対話型認証をサポートしています。Secure Shell はまた、ユーザー公開鍵と信頼できるホスト公開鍵による認証もサポートしています。鍵は、RSA の場合と DSA の場合があります。セッション鍵の交換は、サーバー認証手順で署名される Diffie-Hellman 短期鍵の交換で構成されます。さらに、Secure Shell は認証に GSS 資格を使用することもできます。

Secure Shell での GSS 資格の取得

Secure Shell で GSS-API 認証を使用するには、サーバーが GSS-API アダプタの資格を持ち、クライアントが GSS-API のイニシエータの資格を持つ必要があります。mech_krb5 のサポートを使用できます。

mech_krb5 では、サーバーは、サーバーに対応するホスト主体が /etc/krb5/krb5.keytab で有効なエントリを持つと、GSS-API アダプタの資格を持ちます。

クライアントは、次のどちらかによって、mech_krb5 に対するイニシエータの資格を持ちます。

- kinit コマンドが実行された。
- pam_krb5 モジュールが pam.conf ファイルで使用されている。

GSS-API および Kerberos については、『Oracle Solaris 11.3 での Kerberos およびその他の認証サービスの管理』 の 「Kerberos ユーティリティー」を参照してください。メカニズムの詳細は、mech(4) および mech_spnego(5) のマニュアルページを参照してください。

Secure Shell でのコマンドの実行とデータの転送

認証が完了すると、ユーザーは通常、シェルまたはコマンド実行を要求して Secure Shell を使用します。ユーザーは、ssh のオプションを使用して、要求を行うことができます。要求には、擬似端末の配置、X11 または TCP/IP の接続の転送、セキュリティー保護された接続上での ssh-agent 認証プログラムの有効化などがあります。

ユーザーセッションの基本手順は次のとおりです。

1. ユーザーがシェルまたはコマンドの実行を要求し、セッションモードを開始します。

   セッションモードでは、SSH クライアント側では、データは端末を通して送受信されます。また、サーバー側ではシェルまたはコマンドを介して送受信されます。
2. データの転送が完了すると、ユーザープログラムは終了します。
3. 既存の接続を除いて、すべての X11 接続と TCP/IP 接続の転送を停止します。既存の X11 接続と TCP/IP 接続は、開いたままです。
4. SSH サーバーは、終了ステータスのメッセージをクライアントに送信します。開いたままになっていた転送先のポートなど、すべての接続が切断されると、クライアントはサーバーへの接続を切断します。その後、クライアントが終了します。

Secure Shell でのクライアントとサーバーの構成

Secure Shell セッションの特性は、構成ファイルで変更できます。構成ファイルに対しては、コマンド行オプションを使用することで、ある程度オーバーライドできます。

Secure Shell でのクライアントの構成

ほとんどの場合、Secure Shell セッションのクライアント側の特性は、システム全体の構成ファイル (`/etc/ssh/ssh_config`) によって決定されます。`ssh_config` ファイル内の設定は、ユーザーの構成ファイル (`~/.ssh/config`) にオーバーライドされます。さらに、ユーザーはこれらの構成ファイルをコマンド行でオーバーライドできます。

サーバーの `/etc/ssh/sshd_config` ファイル内の設定によって、クライアント側のどの要求がサーバーによって許可されるかが決まります。サーバー構成の設定の一覧については、54 ページの「Secure Shell でのキーワード」を参照してください。詳細は、`sshd_config(4)` のマニュアルページを参照してください。

クライアントの構成ファイルのキーワードは、54 ページの「Secure Shell でのキーワード」 に一覧表示されています。キーワードにデフォルト値がある場合は、その値が指定されます。これらのキーワードについては、`ssh(1)`、`scp(1)`、`sftp(1)`、および `ssh_config(4)` のマニュアルページに詳細を記載しています。アルファベット順のキーワードとそれと同等のコマンド行オーバーライドの一覧については、表7を参照してください。

Secure Shell でのサーバーの構成

サーバー側の Secure Shell セッションの特性は、`/etc/ssh/sshd_config` ファイルによって管理されます。サーバーの構成ファイルのキーワードは、54 ページの「Secure Shell でのキーワード」 に一覧表示されています。キーワードにデフォルト値がある場合は、その値が指定されます。キーワードの詳細については、`sshd_config(4)` のマニュアルページを参照してください。

Secure Shell でのキーワード

次の表は、キーワードおよびそのデフォルト値 (存在する場合) の一覧です。キーワードはアルファベット順になっています。クライアントに適用されるキーワードは、`ssh_config` ファイルにあります。サーバーに適用されるキーワードは、`sshd_config` ファイルにあります。両方のファイルで設定されているキーワードもあります。v1 プロトコルが動作している Secure Shell サーバーのキーワードにはマークが付いています。

表 3 Secure Shell 構成ファイルでのキーワード

キーワード	デフォルト値	場所
`AllowGroups`		サーバー

キーワード	デフォルト値	場所
AllowTcpForwarding	yes	サーバー
AllowUsers		サーバー
AuthorizedKeysFile	~/.ssh/authorized_keys	サーバー
Banner	/etc/issue	サーバー
Batchmode	no	クライアント
BindAddress		クライアント
CheckHostIP	yes	クライアント
ChrootDirectory	no	サーバー
Cipher	blowfish、3des	クライアント
Ciphers	aes128-ctr、aes128-cbc、3des-cbc、blowfish-cbc、arcfour	両方
ClearAllForwardings	no	クライアント
ClientAliveCountMax	3	サーバー
ClientAliveInterval	0	サーバー
Compression	no	両方
CompressionLevel		クライアント
ConnectionAttempts	1	クライアント
ConnectTimeout	システムの TCP タイムアウト	クライアント
DenyGroups		サーバー
DenyUsers		サーバー
DisableBanner	no	クライアント
DynamicForward		クライアント
EscapeChar	~	クライアント
FallBackToRsh	no	クライアント
ForwardAgent	no	クライアント
ForwardX11	no	クライアント
ForwardX11Trusted	yes	クライアント
GatewayPorts	no	両方

キーワード	デフォルト値	場所
GlobalKnownHostsFile	/etc/ssh/ssh_known_hosts	クライアント
GSSAPIAuthentication	yes	両方
GSSAPIDelegateCredentials	no	クライアント
GSSAPIKeyExchange	yes	両方
GSSAPIStoreDelegateCredentials	yes	サーバー
HashKnownHosts	no	クライアント
Host	* 詳細については、58 ページの「Secure Shell でのホスト固有のパラメータ」を参照してください。	クライアント
HostbasedAuthentication	no	両方
HostbasedUsesNameFromPacketOnly	no	サーバー
HostKey (v1)	/etc/ssh/ssh_host_key	サーバー
HostKey (v2)	/etc/ssh/host_rsa_key、/etc/ssh/host_dsa_key	サーバー
HostKeyAlgorithms	ssh-rsa、ssh-dss	クライアント
HostKeyAlias		クライアント
HostName		クライアント
IdentityFile	~/.ssh/id_dsa、~/.ssh/id_rsa	クライアント
IgnoreIfUnknown		クライアント
IgnoreRhosts	yes	サーバー
IgnoreUserKnownHosts	yes	サーバー
KbdInteractiveAuthentication	yes	両方
KeepAlive	yes	両方
KeyRegenerationInterval	3600 (秒)	サーバー
ListenAddress		サーバー
LocalForward		クライアント
LoginGraceTime	120 (秒)	サーバー
LogLevel	info	両方
LookupClientHostnames	yes	サーバー
MACs	hmac-sha1-*、および hmac-sha2-* アルゴリズム。	両方
Match		サーバー
MaxStartups	10:30:60	サーバー

キーワード	デフォルト値	場所
NoHostAuthenticationForLocalHost	no	クライアント
NumberOfPasswordPrompts	3	クライアント
PAMServiceName		サーバー
PAMServicePrefix		サーバー
PasswordAuthentication	yes	両方
PermitEmptyPasswords	no	サーバー
PermitRootLogin	no	サーバー
PermitUserEnvironment	no	サーバー
PidFile	/system/volatile/sshd.pid	サーバー
Port	22	両方
PreferredAuthentications	hostbased,publickey,keyboard-interactive,passwor	クライアント
PreUserauthHook		サーバー
PrintLastLog	yes	サーバー
PrintMotd	no	サーバー
Protocol	2,1	両方
ProxyCommand		クライアント
PubkeyAuthentication	yes	両方
RekeyLimit	1G から 4G	クライアント
RemoteForward		クライアント
RhostsAuthentication	no	サーバー、v1
RhostsRSAAuthentication	no	サーバー、v1
RSAAuthentication	no	サーバー、v1
ServerAliveCountMax	3	クライアント
ServerAliveInterval	0	クライアント
ServerKeyBits	512 から 768	サーバー、v1
StrictHostKeyChecking	ask	クライアント
StrictModes	yes	サーバー
Subsystem	sftp /usr/lib/ssh/sftp-server	サーバー
SyslogFacility	auth	サーバー

キーワード	デフォルト値	場所
UseFIPS140	no	両方
UseOpenSSLEngine	yes **注記 -** SunSSH 実装を使用する場合は、次の例外に注意してください。x86 および T4 シリーズ以降の SPARC システムでは、プラットフォーム固有の命令がすでに OpenSSL 内部暗号化実装に組み込まれているため、UseOpenSSLEngine キーワードはデフォルトで無効になっています。	両方
UsePrivilegedPort	no	両方
User		クライアント
UserKnownHostsFile	~/.ssh/known_hosts	クライアント
UseRsh	no	クライアント
VerifyReverseMapping	no	サーバー
X11DisplayOffset	10	サーバー
X11Forwarding	yes	サーバー
X11UseLocalHost	yes	サーバー
XAuthLocation	/usr/bin/xauth	両方

Secure Shell でのホスト固有のパラメータ

異なるローカルホストシステムに対して別の Secure Shell 特性を使用すると役立つことがあります。システム管理者は、ファイル内のエントリを Host キーワードでグループ化することにより、ホストまたは正規表現に従って別々のパラメータセットを適用する /etc/ssh/ssh_config ファイルに定義できます。Host キーワードを使用しない場合、クライアント構成ファイル内のエントリは、ユーザーが使用しているローカルホストに適用されます。

Secure Shell およびログインの環境変数

次の Secure Shell キーワードが sshd_config ファイル内に設定されていない場合は、/etc/default/login ファイル内の同等のエントリから値が取得されます。

/etc/default/login のエントリ	sshd_config のキーワードと値
CONSOLE=*	PermitRootLogin=without-password

/etc/default/login のエントリ	sshd_config のキーワードと値
#CONSOLE=*	PermitRootLogin=yes
PASSREQ=YES	PermitEmptyPasswords=no
PASSREQ=NO	PermitEmptyPasswords=yes
#PASSREQ	PermitEmptyPasswords=no
TIMEOUT=*seconds*	LoginGraceTime=*seconds*
#TIMEOUT	LoginGraceTime=120
RETRIES と SYSLOG_FAILED_LOGINS	password および keyboard-interactive 認証方式にのみ適用される

次の変数は、ユーザーのログインシェルから初期化スクリプトで設定され、sshd デーモンによってその値が使用されます。変数が設定されていない場合には、デーモンはデフォルト値を使用します。

TIMEZONE　TZ 環境変数の設定を制御します。設定されていない場合、sshd デーモンの起動時のTZ の値を使用します。

ALTSHELL　SHELL 環境変数の設定を制御します。デフォルトは ALTSHELL=YES で、sshd デーモンはユーザーのシェルの値を使用します。ALTSHELL=NO の場合、SHELL の値は設定されません。

PATH　PATH 環境変数の設定を制御します。値が設定されていない場合、デフォルトのパスは /usr/bin になります。

SUPATH　root に対する PATH 環境変数の設定を制御します。値が設定されていない場合、デフォルトのパスは /usr/sbin:/usr/bin になります。

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

Secure Shell での既知のホストの保守

別のホストとの通信をセキュアに行う必要のあるホストシステムのそれぞれに、ローカルホストの /etc/ssh/ssh_known_hosts ファイルにサーバーの公開鍵が格納されている必要があります。/etc/ssh/ssh_known_hosts ファイルを更新するときに、スクリプトを使用することもできますが、セキュリティーが大幅に低下するため、使用しないことを強くお勧めします。

/etc/ssh/ssh_known_hosts ファイルを配布するときは、次のようなセキュアなメカニズムのみで行う必要があります。

- Secure Shell、IPsec、または Kerberos を使用した ftp などのセキュアな接続を使用して、既知の信頼できるシステムから配布する

- システムインストール時に配布する

侵入者が known_hosts ファイルに偽の公開鍵を挿入してアクセス権を取得する可能性を避けるため、ssh_known_hosts ファイルの信頼できる既知のソースを使用するようにしてください。ssh_known_hosts ファイルは、インストール中に配布できます。あとで、scp コマンドを使用するスクリプトを使用して、最新バージョンをコピーすることもできます。

Secure Shell ファイル

次の表に、主な Secure Shell ファイルと推奨されるファイルアクセス権を示します。

表 4 Secure Shell ファイル

ファイル名	説明	推奨アクセス権と所有者
~/.rhosts	ホスト名とユーザー名のペアを含みます。ユーザーは、対応するホストシステムにパスワードを使用しないでログインできます。このファイルは、rlogind デーモンおよび rshd デーモンでも使用されます。	-rw-r--r-- *username*
~/.shosts	ホスト名とユーザー名のペアを含みます。ユーザーは、対応するホストシステムにパスワードを使用しないでログインできます。このファイルは、ほかのユーティリティーでは使用されません。詳細は、sshd(1M) のマニュアルページの「FILES」セクションを参照してください。	-rw-r--r-- *username*
~/.ssh/authorized_keys	ユーザーアカウントへのログインが許可されているユーザーの公開鍵を保持します。	-rw-r--r-- *username*
~/.ssh/config	システム設定をオーバーライドするユーザー設定を構成します。	-rw-r--r-- *username*
~/.ssh/environment	ログイン時の初期割り当てを含みます。デフォルトでは、このファイルは読み取られません。このファイルを読み取るには、sshd_config ファイルの PermitUserEnvironment キーワードが yes に設定されている必要があります。	-rw-r--r-- *username*
/etc/hosts.equiv	.rhosts 認証で使用されるホストシステムを含みます。このファイルは、rlogind デーモンおよび rshd デーモンでも使用されます。	-rw-r--r-- root
~/.ssh/known_hosts	この SSH クライアントがセキュアな通信を行うことのできるすべてのホストシステムのホスト公開鍵を含みます。このファイルは自動的に管理されます。ユーザーが未知のホストに接続すると、リモートホスト鍵がファイルに追加されます。	-rw-r--r-- *username*
/etc/default/login	対応する sshd_config パラメータが設定されていないときの、sshd デーモンのデフォルトを指定します。	-r--r--r-- root

ファイル名	説明	推奨アクセス権と所有者
/etc/nologin	このファイルが存在する場合、sshd デーモンは root のログインのみを許可します。このファイルの内容は、ログインしようとするユーザーに対して表示されます。	-rw-r--r-- root
~/.ssh/rc	ユーザーシェルが起動する前に実行される初期化ルーチンを含みます。初期化ルーチンの例については、sshd(1M) のマニュアルページを参照してください。	-rw-r--r-- *username*
/etc/ssh/shosts.equiv	ホストに基づく認証で使用されるホストシステムを含みます。このファイルは、ほかのユーティリティーでは使用されません。	-rw-r--r-- root
/etc/ssh/ssh_config	SSH クライアントシステムでのシステム設定を構成します。	-rw-r--r-- root
/etc/ssh/ssh_host_dsa_key または /etc/ssh/ssh_host_rsa_key	ホスト非公開鍵を含みます。	-rw------- root
/etc/ssh_host_key.pub または /etc/ssh/ssh_host_dsa_key.pub または /etc/ssh/ssh_host_rsa_key.pub	ホスト公開鍵を含みます。/etc/ssh/ssh_host_rsa_key.pub など。ホスト鍵をローカル known_hosts ファイルにコピーするときに使用します。	-rw-r--r-- root
/etc/ssh/ssh_known_hosts	この SSH クライアントがセキュアな通信を行うことのできるすべてのホストシステムのホスト公開鍵を含みます。このファイルはシステム管理者が管理します。	-rw-r--r-- root
/etc/ssh/sshd_config	sshd (Secure Shell デーモン) の構成データを含みます。	-rw-r--r-- root
/system/volatile/sshd.pid	Secure Shell デーモン sshd のプロセス ID を含みます。複数のデーモンが実行されている場合は、起動された最後のデーモンを含みます。	-rw-r--r-- root
/etc/ssh/sshrc	システム管理者が用意したホスト固有の初期化ルーチンを含みます。	-rw-r--r-- root

注記 - sshd_config ファイルは、サイトでカスタマイズされたパッケージのファイルでオーバーライドできます。詳細は、pkg(5) のマニュアルページにある overlay ファイル属性の定義を参照してください。

次の表は、キーワードまたはコマンドオプションにオーバーライドされる Secure Shell ファイルの一覧です。

表 5 オーバーライドされる Secure Shell ファイルの場所

ファイル名	キーワードのオーバーライド	コマンド行のオーバーライド
/etc/ssh/ssh_config		ssh -F *config-file* scp -F *config-file*

ファイル名	キーワードのオーバーライド	コマンド行のオーバーライド
~/.ssh/config		ssh -F *config-file*
/etc/ssh/host_rsa_key /etc/ssh/host_dsa_key	HostKey	
~/.ssh/identity ~/.ssh/id_dsa, ~/.ssh/id_rsa	IdentityFile	ssh -i *ID-file* scp -i *ID-file*
~/.ssh/authorized_keys	AuthorizedKeysFile	
/etc/ssh/ssh_known_hosts	GlobalKnownHostsFile	
~/.ssh/known_hosts	UserKnownHostsFile IgnoreUserKnownHosts	

Secure Shell コマンド

次の表は、主な Secure Shell コマンドのサマリーです。

表 6 Secure Shell でのコマンド

コマンドのマニュアルページ	説明
ssh(1)	ユーザーをリモートシステムにログインさせ、リモートシステム上でコマンドをセキュアに実行します。ssh コマンドは、セキュアでないネットワークを介して 2 つの信頼できないホストシステム間でセキュアな暗号化通信を行うことを可能にします。X11 接続と任意の TCP/IP ポートも、セキュアなチャネルを介して転送されます。
sshd(1M)	Secure Shell 用のデーモン。このデーモンは、クライアントからの接続を待機します。セキュアでないネットワークを介して 2 つの信頼できないホストシステム間でセキュアな暗号化通信を行うことを可能にします。
ssh-add(1)	RSA または DSA ID を認証エージェント ssh-agent に追加します。ID は「鍵」とも呼ばれます。
ssh-agent(1)	公開鍵認証時に使用される非公開鍵を保持します。ssh-agent プログラムは、X セッションまたはログインセッションの開始時に起動します。ほかのすべてのウィンドウおよびプログラムは、ssh-agent プログラムのクライアントとして起動します。環境変数を使用すれば、ユーザーが ssh コマンドを使用してほかのシステムにログインするときに、エージェントを検出して認証に使用することができます。
ssh-keygen(1)	Secure Shell の認証鍵を生成および管理します。
ssh-keyscan(1)	多数の Secure Shell ホストの公開鍵を収集します。ssh_known_hosts ファイルの作成および検証時に役立ちます。
ssh-keysign(1M)	ssh コマンドがローカルホスト上のホスト鍵にアクセスするときに使用します。Secure Shell v2 によるホストに基づく認証中に必要となるデジタル署名を生成します。このコマンドは、ユーザーではなく ssh コマンドによって呼び出されます。
scp(1)	暗号化された ssh トランスポートを介して、ネットワーク上のホスト間でファイルを安全にコピーします。rcp コマンドと異なり、scp コマンドは、パスワード情報が認証に必要な場合、パスワードまたはパスフレーズを要求します。

コマンドのマニュアルページ	説明
sftp(1)	ftp コマンドと同様の対話型ファイル転送プログラムです。ftp コマンドと異なり、sftp コマンドは、暗号化された ssh トランスポートを介してすべての操作を実行します。このコマンドは、指定したホスト名に接続してログインし、対話型コマンドモードに入ります。

次の表は、Secure Shell キーワードをオーバーライドするコマンドオプションを示しています。キーワードは、ssh_config ファイルおよび sshd_config ファイルで指定します。

表 7 Secure Shell のキーワードに相当するコマンド行

キーワード	ssh コマンド行のオーバーライド	scp コマンド行のオーバーライド
BatchMode		scp -B
BindAddress	ssh -b *bind-addr*	scp -a *bind-addr*
Cipher	ssh -c *cipher*	scp -c *cipher*
Ciphers	ssh -c *cipher-spec*	scp -c *cipher-spec*
Compression	ssh -C	scp -C
DynamicForward	ssh -D *SOCKS4-port*	
EscapeChar	ssh -e *escape-char*	
ForwardAgent	ssh -A (有効) ssh -a (無効)	
ForwardX11	ssh -X (有効) ssh -x (無効)	
GatewayPorts	ssh -g	
IPv4	ssh -4	scp -4
IPv6	ssh -6	scp -6
LocalForward	ssh -L *localport:remotehost:remoteport*	
MACS	ssh -m *MAC-spec*	
Port	ssh -p *port*	scp -P *port*
Protocol	ssh -2 (v2 のみ)	
RemoteForward	ssh -R *remoteport:localhost:localport*	

索引

T

U

V

X