Gestion de l'accès au shell sécurisé dans Oracle® Solaris 11.2

Référence: E53962-02
Septembre 2014

Table des matières

Utilisation de la présente documentation

Gestion de l'accès au shell sécurisé dans Oracle® Solaris 11.2 décrit l'administration et l'utilisation et de la fonction de shell sécurisé afin de sécuriser l'accès à distance.

- **Présentation** : décrit les concepts et les tâches liés à l'utilisation du shell sécurisé dans Oracle Solaris.
- **Public visé** : administrateurs système chargé d'implémenter la sécurité dans l'entreprise.
- **Connaissances requises** : être familiarisé avec la terminologie et les concepts de sécurité.

Bibliothèque de la documentation des produits

Les informations de dernière minute et les problèmes connus pour ce produit sont inclus dans la bibliothèque de documentation accessible à l'adresse : http://www.oracle.com/pls/topic/lookup?ctx=E56338.

Accès aux services de support Oracle

Les clients Oracle ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info ou le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs si vous êtes malentendant.

Commentaires

Faites part de vos commentaires sur cette documentation à l'adresse : http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦ CHAPITRE 1

Utilisation du shell sécurisé (Tâches)

La fonction Shell sécurisé d'Oracle Solaris fournit un accès sécurisé à un hôte distant sur un réseau non sécurisé. Le shell fournit des commandes pour une connexion à distance, l'affichage de fenêtres à distance et le transfert de fichiers distant. Ce chapitre comprend les sections suivantes :

Pour obtenir des informations de référence, reportez-vous au Chapitre 2, Référence concernant le shell sécurisé.

Shell sécurisé (Présentation)

Shell sécurisé est le protocole d'accès à distance par défaut sur un système Oracle Solaris que vous venez d'installer. Shell sécurisé dans Oracle Solaris repose sur le kit d'outils Open Source OpenSSL, qui implémente les protocoles SSL (Secure Sockets Layer) et TLS (Transport Layer Security).

Les deux versions distinctes du toolkit sont disponibles dans Oracle Solaris.

- La version 1.0.0 est la version d'exécution par défaut de Shell sécurisé.
- La version 0.9.8 met en oeuvre FIPS-140FIPS 140, norme de sécurité informatique du gouvernement américain destinée aux modules cryptographiques.

  Pour plus d'informations sur l'utilisation de Shell sécurisé en mode FIPS 140, reportez-vous à la section "Secure Shell et FIPS 140" à la page 11.

Dans Shell sécurisé, l'authentification s'effectue par l'utilisation de mots de passe et/ou de clés publiques. Tout le trafic réseau est chiffré. Par conséquent, Shell sécurisé empêche tout intrus potentiel de lire une communication interceptée. Shell sécurisé empêche également un adversaire de mystifier le système.

Shell sécurisé peut également être utilisé comme un virtual private network (VPN) à la demande. Un VPN peut transmettre le trafic système X Window ou connecter des numéros de port individuels compris entre les machines locales et distantes via un lien réseau crypté.

Avec Shell sécurisé, vous pouvez effectuer les actions suivantes :

- Se connecter de manière sécurisée à un autre hôte sur un réseau non sécurisé.
- Copier des fichiers en toute sécurité entre les deux hôtes.
- Exécuter des commandes en toute sécurité sur l'hôte distant.

Sur le côté serveur, Shell sécurisé prend en charge la Version 2 (v2) du protocole Shell sécurisé. Su le côté client, outre v2, le client prend en charge la version 1 (v1).

Authentification du shell sécurisé

Shell sécurisé fournit des méthodes de clé publique et mot de passe pour l'authentification de la connexion à l'hôte distant. L'authentification avec clé publique est un mécanisme d'authentification plus fiable que l'authentification par mot de passe, car la clé privée ne se déplace pas sur le réseau.

Les méthodes d'authentification sont tentées dans l'ordre suivant. Lorsque la configuration ne satisfait pas à une méthode d'authentification, la méthode suivante est tentée.

- **GSS-API –** Fait appel à des informations d'authentification pour les mécanismes GSS-API comme `mech_krb5` (Kerberos V) et `mech_dh` (AUTH_DH) pour authentifier les clients et les serveurs. Pour plus d'informations sur GSS-API, reportez-vous à la section “ Introduction to GSS-API ” du manuel “ Developer’s Guide to Oracle Solaris 11 Security ”.
- **Authentification basée sur l'hôte :** utilise les clés d'hôte et les fichiers `rhosts`. Utilise les clés d'hôte privées/publiques RSA et DSA du client pour authentifier ce dernier. Utilise les fichiers `rhosts` pour autoriser les clients à des utilisateurs.
- **Authentification avec clé publique :** authentifie les utilisateurs avec leurs clés publiques et privées RSA et DSA.
- **Authentification par mot de passe :** utilise PAM pour authentifier les utilisateurs. La méthode d'authentification du clavier dans v2 permet l'invitation arbitraire par PAM. Pour plus d'informations, reportez-vous à la section `SECURITY` de la page de manuel `sshd`(1M).

Le tableau suivant répertorie les conditions requises pour l'authentification d'un utilisateur essayant de se connecter à un hôte distant. L'utilisateur est sur l'hôte local, le client. L'hôte distant, le serveur, exécute le démon `sshd`. Le tableau présente les méthodes d'authentification Shell sécurisé et les conditions requises par l'hôte.

TABLEAU 1-1 Méthodes d'authentification pour le shell sécurisé

Méthode d'authentification	Exigences de l'hôte local (client)	Exigences de l'hôte distant (serveur)
GSS-API	Informations d'identification de l'initiateur pour le mécanisme GSS.	Informations d'identification de l'accepteur pour le mécanisme GSS. Pour plus d'informations, reportez-vous à la section “Acquisition d'informations d'identification GSS dans le shell sécurisé” à la page 35.

Basée sur l'hôte	Compte utilisateur
	
	Clé privée de l'hôte local dans `/etc/ssh/ssh_host_rsa_key` ou `/etc/ssh/ssh_host_dsa_key`
	
	`HostbasedAuthentication yes` dans `/etc/ssh/ssh_config`
		Compte utilisateur
		
		Clé privée de l'hôte local dans `/etc/ssh/known_hosts` ou `~/.ssh/known_hosts`
		
		`HostbasedAuthentication yes` dans `/etc/ssh/sshd_config`
		
		`IgnoreRhosts no` dans `/etc/ssh/sshd_config`
		
		Entrée d'hôte local dans `/etc/ssh/shosts.equiv`, `/etc/hosts.equiv`, `~/.rhosts` ou `~/.shosts`

Par mot de passe	Compte utilisateur	Compte utilisateur Prend en charge PAM.
`.rhosts` avec RSA (v1) sur le serveur uniquement	Compte utilisateur Clé publique de l'hôte local dans `/etc/ssh/ssh_host_rsa1_key`	Compte utilisateur Clé publique de l'hôte local dans `/etc/ssh/ssh_known_hosts` ou `~/.ssh/known_hosts` `IgnoreRhosts no` dans `/etc/ssh/sshd_config` Entrée d'hôte local dans `/etc/ssh/shosts.equiv`, `/etc/hosts.equiv`, `~/.shosts` ou `~/.rhosts`
Clé publique RSA ou DSA	Compte utilisateur Clé privée dans `~/.ssh/id_rsa` ou `~/.ssh/id_dsa` Clé publique d'utilisateur dans `~/.ssh/id_rsa.pub` ou `~/.ssh/id_dsa.pub`	Compte utilisateur Clé publique d'utilisateur en `~/.ssh/authorized_keys`

Shell sécurisé et OpenSSH

Le shell sécurisé est un fork du projet OpenSSH (http://www.openssh.com). Les correctifs de sécurité pour la correction des vulnérabilités découvertes dans les versions ultérieures d'OpenSSH sont intégrés dans le shell sécurisé, tout comme les corrections de bogues et les

fonctionnalités. Depuis septembre 2012, la version du shell sécurisé dans Oracle Solaris est 2.0. La commande `ssh -V`affiche le numéro de version.

Les fonctionnalités suivantes sont mises en oeuvre pour le protocole v2 dans cette version du shell sécurisé :

- Mot-clé `ForceCommand` : force l'exécution de la commande spécifiée indépendamment de ce que l'utilisateur tape sur la ligne de commande. Ce mot-clé est très utile à l'intérieur d'un bloc `Match`. Cette option de configuration `sshd_config` est semblable à l'option `command="..."` dans `$HOME/.ssh/authorized_keys` .
- Protection par phrase de passe `AES-128` : les clés privées qui sont générées par la commande `ssh-keygen` sont protégées avec l'algorithme `AES-128`. Cet algorithme protège les clés qui viennent d'être générées et les clés rechiffrées, comme par exemple lorsqu'une phrase de passe est modifiée.
- Option `-u` de la commande `sftp-server` : permet à l'utilisateur de définir explicitement un `umask` sur des fichiers et des répertoires. Cette option remplace la valeur `umask` par défaut de l'utilisateur. Pour voir un exemple, reportez-vous à la description de `Subsystem` à la page de manuel `sshd_config`(4).
- Mots-clés supplémentaires pour les blocs `Match` : `AuthorizedKeysFile`, `ForceCommand` et `HostbasedUsesNameFromPacketOnly` sont pris en charge à l'intérieur des blocs `Match`. Par défaut, la valeur de `AuthorizedKeysFile` est `$HOME/.ssh/authorized_keys` et celle de `HostbasedUsesNameFromPacketOnly` est `no`. Pour utiliser les blocs `Match`, reportez-vous à la section “Création d'exceptions d'utilisateur et d'hôte aux valeurs par défaut de shell sécurisé” à la page 17.

Les techniciens d'Oracle Solaris proposent des corrections de bogues dans le projet OpenSSH. Ils ont également intégré les fonctionnalités Oracle Solaris suivantes dans le fork du shell sécurisé :

- PAM : le shell sécurisé utilise PAM. L'option de configuration `UsePAM` OpenSSH n'est pas prise en charge.
- Séparation des privilèges : le shell sécurisé n'utilise pas le code de séparation des privilèges du projet OpenSSH. Le shell sécurisé sépare le traitement de l'audit, la conservation des enregistrements et la nouvelle saisie du traitement des protocoles de session.

  Le code de séparation des privilèges du shell sécurisé est toujours actif et ne peut pas être désactivé. L'option OpenSSH `UsePrivilegeSeparation` n'est pas prise en charge.
- Environnement linguistique : le shell sécurisé prend entièrement en charge la négociation de la langue définie dans la norme RFC 4253 relative au *protocole de transfert du shell sécurisé*. Une fois que l'utilisateur est connecté, le profil du shell de connexion de l'utilisateur peut remplacer les paramètres d'environnement linguistique négociés avec le shell sécurisé.
- Audit : le shell sécurisé est totalement intégré dans le service d'audit d'Oracle Solaris. Pour plus d'informations sur le service d'audit, reportez-vous à la section “ Gestion de l’audit dans Oracle Solaris 11.2 ”.

- Prise en charge GSS-API : GSS-API peut s'utiliser pour l'authentification des utilisateurs *et* pour l'échange de clés initiales. La fonction GSS-API est définie dans RFC4462, *Generic Security Service Application Program Interface*.
- Commandes proxy : le shell sécurisé fournit des commandes de proxy pour les protocoles SOCKS5 et HTTP. Pour obtenir un exemple, reportez-vous à la section “Configuration de connexions du shell sécurisé par défaut aux hôtes en-dehors d'un pare-feu” à la page 30.

Dans les versions d'Oracle Solaris, le shell sécurisé resynchronise l'indicateur de compatibilité `SSH_OLD_FORWARD_ADDR` à partir du projet OpenSSH.

Secure Shell et FIPS 140

Shell sécurisé est un consommateur du module OpenSSL FIPS 140. Oracle Solaris fournit une option FIPS 140 pour le côté serveur et le côté client. Pour respecter les exigences de FIPS 140, les administrateurs doivent configurer et utiliser les options FIPS 140.

Mode FIPS, où Shell sécurisé utilise le mode FIPS 140 d'OpenSSL si ce n'est pas le mode par défaut. En tant qu'administrateur, vous devez explicitement autoriser Secure Shell à s'exécuter en mode FIPS 140. Vous pouvez appeler le mode FIPS 140 avec la commande `ssh -o "UseFIPS140 yes"` *remote-host*. Vous pouvez également définir un mot-clé dans les fichiers de configuration.

En résumé, la mise en oeuvre se compose des éléments suivants :

- Les chiffrements certifiés FIPS 140 suivants sont disponibles sur les côtés serveur et client : `aes128-cbc`, `aes192-cbc` et `aes256-cbc`.

  `3des-cbc` est disponible par défaut côté client, mais il ne figure pas dans la liste de chiffrements côté serveur en raison des risques de sécurité potentiels.
- Les codes d'authentification des messages (MAC, Message Authentication Codes) certifiés FIPS 140 suivants sont disponibles :
  - `hmac-sha1`, `hmac-sha1-96`
  - `hmac-sha2-256`, `hmac-sha2-256-96`
  - `hmac-sha2-512`, `hmac-sha2-512-96`
- Quatre configurations serveur-client sont disponibles :
  - Mode FIPS 140 absent sur le côté client ou serveur
  - Mode FIPS 140 disponible sur les côtés client et serveur
  - Mode FIPS 140 disponible sur le côté serveur, pas sur le côté client
  - Mode FIPS 140 absent sur le côté serveur, mais disponible sur le côté client
- La commande `ssh-keygen` comporte une option permettant de générer la clé privée de l'utilisateur au format PKCS #8 requis par les clients Shell sécurisé en mode FIPS. Pour plus d'informations, reportez-vous à la page de manuel `ssh-keygen`(1).

Pour plus d'informations sur FIPS 140, reportez-vous à “ Using a FIPS 140 Enabled System in Oracle Solaris 11.2 ”. Reportez-vous également aux pages de manuel `sshd`(1M), `sshd_config`(4), `ssh`(1) et `ssh_config`(4).

Lorsque vous utilisez une carte Sun Crypto Accelerator 6000 pour les opérations Shell sécurisé, Shell sécurisé s'exécute avec la prise en charge FIPS 140 au niveau 3. Le matériel de niveau 3 est certifié pour résister au sabotage physique, utiliser l'authentification basée sur l'identité et isoler les interfaces qui gèrent les paramètres de sécurité critique à partir des autres interfaces du matériel.

Configuration du shell sécurisé (Tâches)

Shell sécurisé est configuré à l'installation. La modification des paramètres par défaut nécessite l'intervention de l'administrateur. Les tâches suivantes indiquent comment modifier certains paramètres par défaut.

Configuration du shell sécurisé (Liste des tâches)

La liste des tâches suivante présente les procédures de configuration de Shell sécurisé. Pour utiliser Shell sécurisé, reportez-vous à la section “Utilisation du shell sécurisé (Tâches)” à la page 19.

Tâche	Description	Pour obtenir des instructions
Configuration de l'authentification basée sur l'hôte.	Configure l'authentification basée sur l'hôte sur le serveur et sur le client.	“Configuration de l'authentification basée sur l'hôte pour le shell sécurisé” à la page 13
Augmentation de la taille de tampon pour gérer la latence de connexion.	Augmente la valeur de la propriété TCP `recv_buf` pour les réseaux à bande passante et latence élevées.	“ Modification de la taille du tampon de réception TCP ” du manuel “ Administration des réseaux TCP/IP, d’IPMP et des tunnels IP dans Oracle Solaris 11.2 ”
Configuration du transfert de port.	Permet aux utilisateurs d'utiliser le transfert de port.	“Configuration du transfert de port dans le shell sécurisé” à la page 16
Configuration des exceptions des paramètres par défaut du système Shell sécurisé.	Pour les utilisateurs, les hôtes, les groupes et les adresses, spécifie les valeurs Shell sécurisé qui sont différents des valeurs par défaut du système.	“Création d'exceptions d'utilisateur et d'hôte aux valeurs par défaut de shell sécurisé” à la page 17
Isolation d'un environnement `root` pour les transferts `sftp`.	Fournit un répertoire protégé pour les transferts de fichiers.	“Création d'un répertoire isolé pour les fichiers `sftp`” à la page 18

Configuration de l'authentification basée sur l'hôte pour le shell sécurisé

La procédure suivante définit un système de clé publique où la clé publique du client est utilisée pour l'authentification sur le serveur. L'utilisateur doit également créer une paire de clés publiques ou privées.

Dans cette procédure, les termes *client* et *hôte local* désignent la machine sur laquelle un utilisateur saisit la commande `ssh`. Les termes *serveur* et *hôte distant* font tous deux référence au système que le client tente d'atteindre.

Avant de commencer Vous devez prendre le rôle `root`. Pour plus d'informations, reportez-vous à la section " A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

1. **Sur le client, activez l'authentification basée sur l'hôte.**

   Dans le fichier de configuration du client, `/etc/ssh/ssh_config`, tapez l'entrée suivante :

   ```
   HostbasedAuthentication yes
   ```

   Pour connaître la syntaxe du fichier de configuration, reportez-vous à la page de manuel `ssh_config`(4).

2. **Sur le serveur, activez l'authentification basée sur l'hôte.**

   Dans le fichier de configuration du serveur, `/etc/ssh/sshd_config`, saisissez la même entrée :

   ```
   HostbasedAuthentication yes
   ```

   Pour connaître la syntaxe du fichier, reportez-vous à la page de manuel `sshd_config`(4).

3. **Sur le serveur, vous ou l'utilisateur devez configurer un fichier qui permet au client d'être reconnu en tant qu'hôte de confiance.**

   Pour plus d'informations, reportez-vous à la section `FILES` de la page de manuel `sshd`(1M).

   - **Si vous effectuez la configuration, ajoutez le client sous la forme d'une entrée pour le fichier du serveur `/etc/ssh/shosts.equiv`.**

     client-host

   - **Si les utilisateurs effectuent la configuration, ils doivent ajouter une entrée pour le client dans leur fichier `~/.shosts` sur le serveur.**

     client-host

4. **Sur le serveur, vérifiez que le démon `sshd` peut accéder à la liste des hôtes de confiance.**

   Définissez `IgnoreRhosts` sur `no` dans le fichier `/etc/ssh/sshd_config`.

   ```
   ## sshd_config
   IgnoreRhosts no
   ```

5. **Assurez-vous que les utilisateurs de Shell sécurisé sur votre site possèdent des comptes sur les deux hôtes.**

6. **Placez la clé publique du client sur le serveur en suivant l'une des méthodes suivantes :**

   - **Modifiez le fichier `sshd_config` sur le serveur, puis demandez à vos utilisateurs d'ajouter les clés d'hôte publiques du client à leur fichier `~/.ssh/known_hosts`.**

     ```
     ## sshd_config
     IgnoreUserKnownHosts no
     ```

     Pour des instructions d'utilisation, reportez-vous à la section “Génération d'une paire clé publique/clé privée à utiliser avec le shell sécurisé” à la page 20.

   - **Copiez la clé publique du client sur le serveur.**

     Les clés d'hôte sont stockées dans le répertoire `/etc/ssh`. Ces clés sont généralement générées par le démon `sshd` à la première initialisation.

     a. **Ajoutez la clé au fichier `/etc/ssh/ssh_known_hosts` sur le serveur.**

        Sur le client, saisissez la commande suivante sur une seule ligne, sans barre oblique inverse.

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

Remarque - Si des clés d'hôte manquent sur le serveur, l'utilisation du shell sécurisé génère un message d'erreur semblable à celui-ci :

```
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. **Lorsque vous y êtes invité, indiquez votre mot de passe de connexion.**

Lorsque le fichier est copié, le message "Host key copied" (Clé d'hôte copiée) s'affiche.

Chaque ligne du fichier `/etc/ssh/ssh_known_hosts` se compose de champs séparés par des espaces :

hostnames algorithm-name publickey comment

c. Modifiez le fichier `/etc/ssh/ssh_known_hosts` et ajoutez *RemoteHost* comme premier champ de l'entrée copiée.

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

Exemple 1-1 Configuration de l'authentification basée sur l'hôte

Dans l'exemple ci-dessous, chaque hôte est configuré en tant que serveur et en tant que client. Un utilisateur de l'un ou l'autre hôte peut lancer une connexion `ssh` à l'autre hôte. La configuration suivante convertit chaque hôte en serveur et client :

- Sur chaque hôte, les fichiers de configuration Shell sécurisé contiennent les entrées suivantes :

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

- Sur chaque hôte, le fichier `shosts.equiv` contient une entrée pour l'autre hôte :

```
## /etc/ssh/shosts.equiv on machine2
machine1

## /etc/ssh/shosts.equiv on machine1
machine2
```

- La clé publique pour chaque hôte est contenue dans le fichier `/etc/ssh/ssh_known_hosts` sur l'autre hôte :

```
## /etc/ssh/ssh_known_hosts on machine2
… machine1

## /etc/ssh/ssh_known_hosts on machine1
… machine2
```

- Les utilisateurs possèdent un compte sur les deux hôtes. Par exemple, les informations suivantes s'afficheraient pour l'utilisateur John Doe :

```
## /etc/passwd on machine1
jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
```

```
## /etc/passwd on machine2
jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
```

Configuration du transfert de port dans le shell sécurisé

Le transfert de port permet à un port local d'être transmis à un hôte distant. En réalité, un socket est alloué pour écouter le port côté local. De la même façon, un port peut être spécifié côté distant.

Remarque - Le transfert de port Shell sécurisé doit utiliser des connexions TCP. Shell sécurisé ne prend pas en charge les connexions UDP pour le transfert de port.

Avant de commencer

Vous devez prendre le rôle `root`. Pour plus d'informations, reportez-vous à la section " A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

1. **Configurez une définition Shell sécurisé sur le serveur distant pour autoriser le transfert de port.**

   Changez la valeur de `AllowTcpForwarding` en `yes` dans le fichier `/etc/ssh/sshd_config`.

   ```
   # Port forwarding
   AllowTcpForwarding yes
   ```

2. **Redémarrez le service Shell sécurisé.**

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

   Pour plus d'informations sur la gestion des services persistants, reportez-vous au Chapitre 1, " Introduction à l'utilitaire de gestion des services " du manuel " Gestion des services système dans Oracle Solaris 11.2 " et à la page de manuel `svcadm`(1M).

3. **Vérifiez que le transfert de port peut être utilisé.**

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

Création d'exceptions d'utilisateur et d'hôte aux valeurs par défaut de shell sécurisé

Cette procédure permet d'ajouter un bloc `Match` conditionnel après la section globale du fichier `/etc/ssh/sshd_config`. Les paires de valeurs de mot clé qui suivent le bloc `Match` spécifient des exceptions pour l'utilisateur, le groupe, l'hôte ou l'adresse spécifié comme la correspondance.

Avant de commencer Vous devez vous connecter en tant qu'administrateur disposant de l'autorisation `solaris.admin.edit/etc/ssh/sshd_config`. Par défaut, le rôle `root` dispose de cette autorisation. Pour plus d'informations, reportez-vous à la section " A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

1. **Ouvrez le fichier `/etc/ssh/sshd_config` pour le modifier.**

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

2. **Configurez un utilisateur, un groupe, un hôte ou une adresse pour qu'ils utilisent d'autres paramètres Shell sécurisé que les paramètres par défaut.**

   Placez les blocs `Match` après les paramètres globaux.

   Remarque - Il se peut que la section globale du fichier n'affiche pas toujours la liste des paramètres par défaut. Pour connaître les valeurs par défaut, reportez-vous à la page de manuel `sshd_config`(4).

   Vous pouvez, par exemple, avoir des utilisateurs qui ne devraient pas être autorisés à utiliser le transfert TCP. Dans l'exemple ci-dessous, tout utilisateur du groupe `public` et tout nom d'utilisateur commençant par `test` ne peut pas utiliser le transfert TCP :

   ```
   ## 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
```

Pour plus d'informations sur la syntaxe du bloc Match, reportez-vous à la page de manuel sshd_config(4).

▼ Création d'un répertoire isolé pour les fichiers sftp

Cette procédure permet de configurer un répertoire sftponly créé spécifiquement pour les transferts sftp. Les utilisateurs ne peuvent voir aucun fichier ou répertoire se trouvant en dehors du répertoire de transfert.

Avant de commencer Vous devez prendre le rôle root. Pour plus d'informations, reportez-vous à la section " A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

1. **Sur le serveur Shell sécurisé, créez le répertoire isolé en tant qu'environnement chroot.**

```
# 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
```

Dans cette configuration, /export/home/sftonly est le répertoire chroot auquel seul le compte root peut accéder. L'utilisateur dispose de l'autorisation d'écriture sur le sous-répertoire sftponly/WWW.

2. **En restant sur le serveur, configurez un bloc match pour le groupe sftp.**

Dans le fichier /etc/ssh/sshd_config, localisez l'entrée sftp subsystem et modifiez le fichier comme suit :

```
# 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 options
Match Group sftp
```

```
ChrootDirectory %h
ForceCommand internal-sftp
AllowTcpForwarding no
```

Vous pouvez spécifier le chemin `chroot` à l'aide des variables suivantes :

- `%h` : spécifie le répertoire d'accueil.
- `%u` : spécifie le nom d'utilisateur de l'utilisateur authentifié.
- `%%` : séquence d'échappement du signe `%`.

3. Vérifiez que la configuration fonctionne correctement sur le client.

Les fichiers de votre environnement `chroot` sont susceptibles d'être différents.

```
root@client:~# ssh sftponly@server
This service allows sftp connections only.
Connection to server closed.      No shell access, sftp is enforced.
root@client:~# sftp sftponly@server
sftp> pwd      sftp access granted
Remote working directory: /     chroot directory looks like root directory
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     user can read contents
sftp> put /etc/motd
Uploading /etc/motd to /motd
Couldn't get handle: Permission denied     user cannot write to / directory
sftp> cd WWW
sftp> put /etc/motd
Uploading /etc/motd to /WWW/motd
/etc/motd     100%  118     0.1KB/s   00:00     user can write to WWW directory
sftp> ls -l
-rw-r--r--    1 101  10    118 Jul 20 09:07 motd     successful transfer
sftp>
```

Utilisation du shell sécurisé (Tâches)

Cette section présente les procédures permettant aux utilisateurs de se familiariser avec Shell sécurisé.

Utilisation du shell sécurisé (Liste des tâches)

La liste des tâches suivante présente les procédures d'utilisation de Shell sécurisé.

Tâche	Description	Pour obtenir des instructions
Création d'une paire de clés publique/privée	Permet l'accès à Shell sécurisé pour des sites qui nécessitent une authentification avec clé publique.	“Génération d'une paire clé publique/ clé privée à utiliser avec le shell sécurisé” à la page 20
Changez la phrase de passe.	Change la phrase qui authentifie votre clé privée.	“Modification de la phrase de passe pour une clé privée de shell sécurisé” à la page 22
Connexion par le biais de Shell sécurisé	Permet la communication chiffrée Shell sécurisé lors d'une connexion à distance.	“Connexion à un hôte distant avec le shell sécurisé” à la page 23
Connexion à Shell sécurisé sans mot de passe	Permet la connexion par le biais d'un agent qui fournit votre mot de passe à Shell sécurisé.	“Réduction du nombre d'invites de mot de passe dans le shell sécurisé” à la page 24
Connexion à Shell sécurisé en tant qu'utilisateur `root`.	Permet la connexion en tant que `root` pour les commandes `send` et `receive` ZFS.	“Administration à distance de ZFS avec le shell sécurisé” à la page 25
Utilisation du transfert de port dans Shell sécurisé	Spécifie un port local ou un port distant à utiliser pour les connexions Shell sécurisé via TCP.	“Utilisation du transfert de port dans le shell sécurisé” à la page 27
Copie de fichiers avec Shell sécurisé	Copie les fichiers d'un hôte à l'autre en toute sécurité.	“Copie de fichiers avec le shell sécurisé” à la page 28
Connexion sécurisée à partir d'un hôte à l'intérieur d'un pare-feu sur un hôte à l'extérieur du pare-feu.	Utilise les commandes Shell sécurisé compatibles avec le protocole HTTP ou SOCKS5 pour connecter les hôtes séparés par un pare-feu.	“Configuration de connexions du shell sécurisé par défaut aux hôtes en-dehors d'un pare-feu” à la page 30

▼ Génération d'une paire clé publique/clé privée à utiliser avec le shell sécurisé

Les utilisateurs doivent générer une paire de clés publiques ou privées lorsque leur site met en oeuvre l'authentification basée sur l'hôte ou l'authentification avec clé publique de l'utilisateur. Pour plus d'options, reportez-vous à la page de manuel `ssh-keygen`(1).

Avant de commencer Demandez à votre administrateur système si l'authentification basée sur l'hôte est configurée.

1. Démarrez le programme de génération de clés.

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

Où `-t` est le type d'algorithme, `rsa`, `dsa` ou `rsa1`.

2. Spécifiez le chemin vers le fichier qui contiendra la clé.

Par défaut, le nom de fichier `id_rsa`, qui représente une clé RSA v2, s'affiche entre parenthèses. Vous pouvez sélectionner ce fichier en appuyant sur la touche Retour ou indiquer un autre nom de fichier.

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

Le nom de fichier de la clé publique est créé automatiquement par l'ajout de la chaîne `.pub` au nom du fichier de clés privées.

3. **Entrez une phrase de passe pour utiliser votre clé.**

   Cette phrase de passe est utilisée pour chiffrer votre clé privée. Une entrée nulle est *fortement déconseillée*. Notez que la phrase de passe ne s'affiche pas lorsque vous la saisissez.

   ```
   Enter passphrase (empty for no passphrase):     <Type passphrase>
   ```

4. **Entrez de nouveau la phrase de passe pour la confirmer.**

   ```
   Enter same passphrase again:     <Type 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@my

   System
   ```

5. **Vérifiez que le chemin d'accès au fichier de la clé est correct.**

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

   A ce stade, vous avez créé une paire de clés publiques ou privées.

6. **Connectez-vous à l'hôte distant à l'aide de l'option correspondant à la méthode d'authentification de votre réseau.**

   - **Si votre administrateur a configuré l'authentification basée sur l'hôte, vous pouvez être amené à copier la clé publique de l'hôte local sur l'hôte distant.**

     Vous pouvez maintenant vous connecter à l'hôte distant. Pour plus de détails, reportez-vous à la section “Connexion à un hôte distant avec le shell sécurisé” à la page 23.

     a. **Entrez la commande suivante sur une seule ligne, sans barre oblique inverse.**

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

     b. **Lorsque vous y êtes invité, indiquez votre mot de passe de connexion.**

```
Enter password:       <Type password>
Host key copied
%
```

- **Si votre site utilise l'authentification de l'utilisateur avec les clés publiques, remplissez votre fichier `authorized_keys` sur l'hôte distant.**

  a. **Copiez votre clé publique sur l'hôte distant.**

  Entrez la commande suivante sur une seule ligne, sans barre oblique inverse.

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

  Lorsque le fichier est copié, le message "Key copied" (Clé copiée) s'affiche.

  b. **Lorsque vous y êtes invité, indiquez votre mot de passe de connexion.**

```
Enter password:       Type login password
Key copied
mySystem%
```

7. **(Facultatif) Eviter les futures invites de phrases de passe.**

Voir “Réduction du nombre d'invites de mot de passe dans le shell sécurisé” à la page 24.
Pour plus d'informations, reportez-vous aux pages de manuel `ssh-agent`(1) et `ssh-add`(1).

Modification de la phrase de passe pour une clé privée de shell sécurisé

La commande suivante change le mécanisme d'authentification pour la clé privée, la phrase de passe, pas la clé privée. Pour plus d'informations, reportez-vous à la page de manuel `ssh-keygen`(1).

- **Changez la phrase de passe.**

Tapez la commande `ssh-keygen` avec l'option `-p`, et répondez aux invites.

```
mySystem% ssh-keygen -p
Enter file which contains the private key
      (/home/username/.ssh/id_rsa):     <Press Return>
Enter passphrase
      (empty for no passphrase):      <Type passphrase>
Enter same passphrase again:        <Type passphrase>
```

Où `-p` demande la modification de la phrase de passe d'un fichier de clés privées.

▼ Connexion à un hôte distant avec le shell sécurisé

1. **Démarrez une session Shell sécurisé.**

   Tapez la commande `ssh` et spécifiez le nom de l'hôte distant et votre identifiant de connexion.

   ```
   mySystem% ssh myRemoteHost -l username
   ```

2. **A l'invite, vérifiez l'authenticité de la clé de l'hôte distant.**

   Une invite peut s'afficher mettant en doute l'authenticité de l'hôte distant :

   ```
   The authenticity of host 'myRemoteHost' can't be established.
   RSA key fingerprint in md5 is: 04:9f:bd:fc:3d:3e:d2:e7:49:fd:6e:18:4f:9c:26
   Are you sure you want to continue connecting(yes/no)?
   ```

   Cette invite est normale pour les connexions initiales sur des hôtes distants.

   - **Si vous ne pouvez pas confirmer l'authenticité de l'hôte distant, saisissez `no` et contactez votre administrateur système.**

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

     L'administrateur est responsable de la mise à jour du fichier `/etc/ssh/ssh_known_hosts` global. Un fichier `ssh_known_hosts` mis à jour empêche l'affichage de cette invite.

   - **Si vous confirmez l'authenticité de l'hôte distant, répondez à l'invite et passez à l'étape suivante.**

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

3. **Authentifiez-vous sur Shell sécurisé.**

   a. **Lorsque vous y êtes invité, saisissez votre phrase de passe.**

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

   b. **Lorsque vous y êtes invité, saisissez votre mot de passe de compte.**

   ```
   username@myRemoteHost's password:      <Type password>
   Last login: Wed Sep  7 09:07:49 2011 from myLocalHost
   Oracle Corporation      SunOS 5.11       September 2011
   myRemoteHost%
   ```

4. **Effectuez des transactions sur l'hôte distant.**

   Les commandes que vous envoyez sont chiffrées. Les réponses que vous recevez sont chiffrées.

5. **Arrêtez la connexion Shell sécurisé.**

Lorsque vous avez terminé, saisissez **exit** ou utilisez votre méthode habituelle pour quitter votre shell.

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

Exemple 1-2 Affichage d'une interface utilisateur distante dans le shell sécurisé

Dans cet exemple, `jdoe` est l'utilisateur initial sur les deux systèmes et il se voit affecter le profil de droits Software Installation (installation de logiciels). `jdoe` souhaite utiliser l'interface graphique du gestionnaire de packages sur le système distant. La valeur par défaut du mot-clé `X11Forwarding` est toujours `yes` et le package `xauth` est installé sur le système distant.

```
% ssh -l jdoe -X myRemoteHost
jdoe@myRemoteHost's password: password
Last login: Wed Sep  7 09:07:49 2011 from myLocalHost
Oracle Corporation      SunOS 5.11       September 2011
myRemoteHost% packagemanager &
```

▼ Réduction du nombre d'invites de mot de passe dans le shell sécurisé

Si vous ne voulez pas saisir votre phrase de passe et votre mot de passe pour utiliser Shell sécurisé, vous pouvez utiliser le démon de l'agent. Si vous avez des comptes différents sur différents hôtes, ajoutez les clés dont vous avez besoin pour la session.

Vous pouvez démarrer le démon de l'agent manuellement lorsque vous en avez besoin, comme décrit dans la procédure suivante.

1. Démarrez le démon de l'agent.

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

2. Vérifiez que le démon de l'agent a été démarré.

```
mySystem% pgrep ssh-agent
9892
```

3. Ajoutez votre clé privée au démon de l'agent.

```
mySystem% ssh-add
Enter passphrase for /home/username/.ssh/id_rsa:      <Type passphrase>
Identity added: /home/username/.ssh/id_rsa(/home/username/.ssh/id_rsa)
```

```
mySystem%
```

4. **Démarrez une session Shell sécurisé.**

```
mySystem% ssh myRemoteHost -l username
```

Vous n'êtes pas invité à saisir une phrase de passe.

Exemple 1-3 Utilisation des options `ssh-add`

Dans cet exemple, `jdoe` ajoute deux clés pour le démon de l'agent. L'option `-l` sert à répertorier toutes les clés stockées dans le démon. A la fin de la session, l'option `-D` sert à supprimer toutes les clés du démon de l'agent.

```
myLocalHost% ssh-agent
mySystem% ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa:      <Type 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:      <Type passphrase>
Identity added:
/home/jdoe/.ssh/id_dsa(/home/jdoe/.ssh/id_dsa)

mySystem% ssh-add -l
md5 1024 0e:fb:3d:53:71:77:bf:57:b8:eb:f7:a7:aa:df:e0:d1
/home/jdoe/.ssh/id_rsa(RSA)
md5 1024 c1:d3:21:5e:40:60:c5:73:d8:87:09:3a:fa:5f:32:53
/home/jdoe/.ssh/id_dsa(DSA)
```

User conducts Oracle Solaris Secure Shell transactions

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

▼ Administration à distance de ZFS avec le shell sécurisé

Par défaut, le rôle `root` ne peut se connecter à distance à l'aide de Shell sécurisé. Auparavant, root utilisait Shell sécurisé pour des tâches importantes, telles que l'envoi de données de pool ZFS sur un stockage situé sur un système distant. Dans cette procédure, le rôle `root` crée un utilisateur qui peut agir en tant qu'administrateur ZFS distant.

Avant de commencer

Vous devez prendre le rôle `root`. Pour plus d'informations, reportez-vous à la section " A l'aide de vos droits administratifs attribués " du manuel " Sécurisation des utilisateurs et des processus dans Oracle Solaris 11.2 ".

1. **Créez l'utilisateur sur les deux systèmes.**

   Par exemple, créez l'utilisateur `zfsroot` et fournissez un mot de passe.

   ```
   source # useradd -c "Remote ZFS Administrator" -u 1201 -d /home/zfsroot zfsroot
   source # passwd zfsroot
   Enter password:
   Retype password:
   #

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

   L'utilisateur `zfsroot` doit être défini de manière identique sur les deux systèmes.

2. **Créez la paire de clés de l'utilisateur pour l'authentification Shell sécurisé.**

   La paire de clé est créée sur le système source. Ensuite, la clé publique est copiée sur l'utilisateur `zfsroot` sur le système de destination.

   a. **Générez la paire de clés et placez-la dans le fichier `id_migrate`.**

      ```
      # 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:
      3c:7f:40:ef:ec:63:95:b9:23:a2:72:d5:ea:d1:61:f0 root@source
      ```

   b. **Envoyez la partie publique de la paire de clés au système de destination.**

      ```
      # 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
      ```

3. **Sur les deux systèmes, affectez le profil de droits ZFS File Management (gestion des fichiers ZFS) à `zfsroot`.**

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

4. **Vérifiez que le profil de droits est affecté au système de destination.**

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

5. **Sur le système de destination, déplacez la partie publique de la paire de clés dans le répertoire `/home/zfsroot/.ssh` privé.**

```
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. **Vérifiez que la configuration fonctionne.**

```
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. **(Facultatif) Assurez-vous que vous pouvez créer un instantané et répliquez les données.**

```
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. **(Facultatif) Supprimez la possibilité d'utiliser le compte `zfsroot` pour l'administration de ZFS.**

```
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
```

▼ Utilisation du transfert de port dans le shell sécurisé

Vous pouvez spécifier qu'un port local est transmis à un hôte distant. En réalité, un socket est alloué pour écouter le port côté local. La connexion sur l'hôte distant à partir de ce port est effectuée par le biais d'un canal sécurisé. Par exemple, vous pouvez spécifier un port `143` afin de recevoir votre courrier à distance avec IMAP4. De la même façon, un port peut être spécifié côté distant.

Avant de commencer Pour utiliser le transfert de port, l'administrateur doit avoir activé le transfert de port sur le serveur Shell sécurisé distant. Pour plus de détails, reportez-vous à la section “Configuration du transfert de port dans le shell sécurisé” à la page 16.

- **Définissez le transfert de port sécurisé d'un port distant vers un port local ou inversement.**

- **Pour définir un port local pour recevoir une communication sécurisée à partir d'un port distant, spécifiez les deux ports.**
  Spécifiez le port local à l'écoute de la communication à distance. De même, indiquez l'hôte distant et le port distant qui transfèrent la communication.

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

- **Pour définir un port distant de manière à ce qu'il reçoive une connexion sécurisée d'un port local, spécifiez les deux ports.**
  Spécifiez le port distant à l'écoute de la communication à distance. De même, indiquez l'hôte local et le port local qui transfèrent la communication.

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

Exemple 1-4 Utilisation du transfert de port local pour recevoir du courrier

L'exemple suivant illustre l'utilisation du transfert du port local pour recevoir du courrier en toute sécurité à partir d'un serveur distant.

```
myLocalHost% ssh -L 9143:myRemoteHost:143 myRemoteHost
```

Cette commande transmet les connexions à partir du port `9143` sur `myLocalHost` au port `143`. Port `143` est le port du serveur IMAP v2 sur `myRemoteHost`. Lorsque l'utilisateur lance une application de messagerie, celui-ci doit spécifier le numéro de port local pour le serveur IMAP, comme dans `localhost:9143`.

Exemple 1-5 Utilisation du transfert de port distant pour communiquer à l'extérieur d'un pare-feu

Cet exemple montre comment l'utilisateur, dans un environnement d'entreprise, peut transférer vers un hôte à l'intérieur d'un pare-feu d'entreprise des connexions d'un hôte sur un réseau externe.

```
myLocalHost% ssh -R 9022:myLocalHost:22myOutsideHost
```

Cette commande transmet les connexions à partir du port `9022` sur `myOutsideHost` au port `22`, le serveur `sshd`, sur l'hôte local.

```
myOutsideHost% ssh -p 9022 localhost
myLocalHost%
```

▼ Copie de fichiers avec le shell sécurisé

La procédure suivante décrit la façon dont la commande `scp` copie les fichiers chiffrés entre les hôtes. Vous pouvez copier les fichiers chiffrés entre un hôte local et un hôte distant, ou entre

deux hôtes distants. La commande `scp` invite à s'authentifier. Pour plus d'informations, reportez-vous à la section " Copie à distance avec la commande scp " du manuel " Gestion des systèmes distants dans Oracle Solaris 11.2 " et à la page de manuel `scp`(1).

Vous pouvez également utiliser le programme de transfert de fichiers sécurisé `sftp`. Pour plus d'informations, reportez-vous à la page de manuel `sftp`(1). Pour obtenir un exemple, reportez-vous à l'Exemple 1-6, "Spécification d'un port à l'aide de la commande `sftp`" et à la section " Connexion à un système distant pour copier un fichier (sftp) " du manuel " Gestion des systèmes distants dans Oracle Solaris 11.2 ".

Remarque - Le service d'audit permet d'auditer les transactions `sftp` via la classe d'audit `ft`. Pour `scp`, le service d'audit peut auditer l'accès et la sortie pour la session `ssh`. Pour plus d'informations, reportez-vous à la section " Audit des transferts de fichiers FTP et SFTP " du manuel " Gestion de l'audit dans Oracle Solaris 11.2 ".

---

1. **Démarrez le programme de copie sécurisée.**

   Spécifiez le fichier source, le nom d'utilisateur au niveau de la destination distante et le répertoire de destination.

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

2. **Indiquez votre phrase de passe lorsque vous y êtes invité.**

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

   Une fois que vous avez saisi la phrase de passe, un indicateur de progression s'affiche, comme indiqué à la deuxième ligne de la sortie. L'indicateur de progression affiche les données suivantes :

   - Nom du fichier
   - Le pourcentage du fichier qui a été transféré
   - Une série d'astérisques qui indiquent le pourcentage du fichier qui a été transmis
   - La quantité de données transférées
   - L'heure d'arrivée prévue de la totalité du fichier (c'est-à-dire, le temps restant)

Exemple 1-6 Spécification d'un port à l'aide de la commande `sftp`

Dans cet exemple, l'utilisateur souhaite que la commande `sftp` utilise un port spécifique. L'utilisateur utilise l'option `-o` pour spécifier le port.

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

Configuration de connexions du shell sécurisé par défaut aux hôtes en-dehors d'un pare-feu

Vous pouvez utiliser Shell sécurisé pour établir une connexion à partir d'un hôte à l'intérieur d'un pare-feu vers un hôte à l'extérieur du pare-feu. Cette tâche s'effectue en spécifiant une commande proxy pour `ssh` dans un fichier de configuration ou sous forme d'option dans la ligne de commande. Pour l'option de ligne de commande, reportez-vous à l'Exemple 1-7, “Connexion à des hôtes en dehors du pare-feu à partir de la ligne de commande de shell sécurisé”.

Vous pouvez personnaliser vos interactions `ssh` au moyen de votre propre fichier de configuration, `~/.ssh/config`, ou vous pouvez utiliser les paramètres contenus dans le fichier de configuration d'administration, `/etc/ssh/ssh_config`.

Les fichiers peuvent être personnalisés avec deux types de commandes proxy. Une commande proxy sert aux connexions HTTP. L'autre commande proxy sert aux connexions SOCKS5. Pour plus d'informations, reportez-vous à la page de manuel `ssh_config`(4).

1. **Spécifiez les commandes proxy et les hôtes dans un fichier de configuration.**

   Utilisez la syntaxe suivante pour ajouter autant de lignes qu'il est nécessaire :

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

   `Hôte` *outside-host*

   Limite la spécification de la commande du proxy aux instances lorsqu'un nom d'hôte distant est spécifié dans la ligne de commande. Si vous utilisez un caractère générique pour *outside-host*, vous appliquez la spécification de la commande proxy à un ensemble d'hôtes.

   proxy-command

   Spécifie la commande proxy.

   La commande peut avoir l'une des formes suivantes :

   - `/usr/lib/ssh/ssh-http-proxy-connect` pour les connexions HTTP
   - `/usr/lib/ssh/ssh-socks5-proxy-connect` pour les connexions SOCKS5

   `-h` *proxy-server* et `-p` *proxy-port*

   Ces options spécifient un serveur proxy et un port proxy respectivement. Si des proxys sont présents, ils remplacent toutes les variables d'environnement qui spécifient les serveurs et ports proxy, telles que `HTTPPROXY`, `HTTPPROXYPORT`, `SOCKS5_PORT`, `SOCKS5_SERVER` et `http_proxy`. La variable `http_proxy` spécifie une adresse URL. Si ces options ne sont pas utilisées, les variables d'environnement doivent être définies. Pour plus d'informations,

reportez-vous aux pages de manuel `ssh-socks5-proxy-connect`(1) et `ssh-http-proxy-connect`(1).

outside-host

Désigne un hôte spécifique pour la connexion. Utilisez l'argument de substitution `%h` pour spécifier l'hôte sur la ligne de commande.

outside-port

Désigne un port spécifique pour la connexion. Utilisez l'argument de substitution `%p` pour spécifier le port sur la ligne de commande. En spécifiant `%h` et `%p` sans utiliser l'option `Host` *outside-host*, la commande proxy est appliquée à l'argument de l'hôte chaque fois que la commande `ssh` est appelée.

2. **Exécutez Shell sécurisé en indiquant l'hôte externe.**

Par exemple :

```
mySystem% ssh myOutsideHost
```

Cette commande recherche une spécification de commande proxy pour `myOutsideHost` dans votre fichier de configuration. Si la spécification est introuvable, la commande recherche dans le fichier de configuration du système, `/etc/ssh/ssh_config`. La commande proxy remplace la commande `ssh`.

Exemple 1-7 Connexion à des hôtes en dehors du pare-feu à partir de la ligne de commande de shell sécurisé

La section “Configuration de connexions du shell sécurisé par défaut aux hôtes en-dehors d'un pare-feu” à la page 30 explique comment spécifier une commande proxy dans un fichier de configuration. Dans cet exemple, une commande proxy est spécifiée sur la ligne de commande `ssh`.

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

L'option `-o` de la commande `ssh` fournit une méthode de ligne de commande pour spécifier une commande proxy. Cet exemple de commande effectue les opérations suivantes :

- La commande proxy HTTP remplace `ssh`
- Le port `8080` est utilisé et `myProxyServer` défini en tant que serveur proxy
- La connexion a lieu sur le port `22` de `myOutsideHost`

♦ ♦ ♦ CHAPITRE 2

Référence concernant le shell sécurisé

Ce chapitre décrit les options de configuration de la fonction Shell sécurisé d'Oracle Solaris, et traite les sujets suivants :

Pour plus d'informations sur les procédures de configuration de Shell sécurisé, reportez-vous au Chapitre 1, Utilisation du shell sécurisé (Tâches).

Sessions standard du shell sécurisé

Le démon Shell sécurisé (`sshd`) est démarré normalement au moment de l'initialisation lorsque les services réseau sont démarrés. Le démon détecte les connexions des clients. Une session Shell sécurisé commence lorsque l'utilisateur exécute une commande `ssh`, `scp` ou `sftp`. Un nouveau démon `sshd` est cloné pour chaque connexion entrante. Le démon cloné gère l'échange de clés, le chiffrement, l'authentification, l'exécution des commandes et l'échange de données avec le client. Ces caractéristiques de session sont déterminées par les fichiers de configuration côté client et côté serveur. Les arguments de la ligne de commande peuvent remplacer les paramètres des fichiers de configuration.

Le client doit s'authentifier auprès du serveur et vice-versa. Après la réussite de l'authentification, l'utilisateur peut exécuter des commandes à distance et copier des données entre les hôtes.

Caractéristiques des sessions dans le shell sécurisé

Le comportement côté serveur du démon `sshd` est contrôlé par les paramètres de mot-clé dans le fichier `/etc/ssh/sshd_config`. Par exemple, le fichier `sshd_config` détermine les types d'authentification qui sont autorisés pour l'accès au serveur. Le comportement côté serveur peut également être contrôlé par les options de la ligne de commande lorsque le démon `sshd` est démarré.

Le comportement côté client est contrôlé par les mots-clés Shell sécurisé dans l'ordre de priorité suivant :

- Options de ligne de commande
- Fichier de configuration de l'utilisateur, `~/.ssh/config`
- Fichier de configuration à l'échelle du système, `/etc/ssh/ssh_config`

Par exemple, un utilisateur peut remplacer un paramètre `Ciphers` de configuration à l'échelle du système qui préfère `aes128-ctr` en spécifiant `-c aes256-ctr,aes128-ctr,arcfour` sur la ligne de commande. Le premier chiffre, `aes256-ctr`, est désormais préféré.

Authentification et échange de clés dans le shell sécurisé

Le protocole Shell sécurisé prend en charge l'authentification utilisateur/hôte et l'authentification hôte serveur. Les clés cryptographiques sont échangées pour la protection des sessions Shell sécurisé. Shell sécurisé fournit plusieurs méthodes pour l'authentification et l'échange de clés. Certaines de ces méthodes sont facultatives. Les mécanismes d'authentification client sont répertoriés dans le Tableau 1-1, “Méthodes d'authentification pour le shell sécurisé”. Les serveurs sont authentifiés à l'aide de clés publiques d'hôte connu.

Pour l'authentification, Shell sécurisé prend en charge l'authentification de l'utilisateur et l'authentification interactive générique, qui implique généralement des mots de passe. Shell sécurisé prend également en charge l'authentification avec des clés publiques utilisateur et des clés publiques d'hôte de confiance. Il peut s'agir de clés RSA ou DSA. Les échanges de clés de session sont des échanges de clés éphémères Diffie-Hellman qui sont signées à l'étape d'authentification du serveur. En outre, Shell sécurisé peut utiliser des informations d'identification GSS pour l'authentification.

Acquisition d'informations d'identification GSS dans le shell sécurisé

Pour utiliser GSS-API pour l'authentification dans Shell sécurisé, le serveur doit disposer des informations d'identification de l'accepteur GSS-API et le client doit disposer des informations d'identification de l'initiateur GSS-API. La prise en charge est disponible pour `mech_dh` et pour `mech_krb5`.

Pour `mech_dh`, le serveur dispose des informations d'identification de l'accepteur GSS-API si `root` a exécuté la commande `keylogin`.

Pour `mech_krb5`, le serveur dispose des informations d'identification de GSS-API lorsque l'hôte principal qui correspond au serveur possède une valeur correcte dans `/etc/krb5/krb5.keytab`.

Le client dispose des informations d'identification de l'initiateur pour `mech_dh` si l'une des actions ci-après a été effectuée :

- La commande `keylogin` a été exécutée.
- Le module `pam_dhkeys` est utilisé dans le fichier `pam.conf`.

Le client dispose des informations d'identification de l'initiateur pour `mech_krb5` si l'une des actions ci-après a été effectuée :

- La commande `kinit` a été exécutée.
- Le module `pam_krb5` est utilisé dans le fichier `pam.conf`.

Pour plus d'informations sur l'utilisation de `mech_dh` dans le RPC sécurisé, reportez-vous au Chapitre 10, “ Configuration des services réseau d’authentification ” du manuel “ Gestion de Kerberos et d’autres services d’authentification dans Oracle Solaris 11.2 ”. Pour plus d'informations sur l'utilisation de `mech_krb5`, reportez-vous au Chapitre 2, “ A propos du service Kerberos ” du manuel “ Gestion de Kerberos et d’autres services d’authentification dans Oracle Solaris 11.2 ”. Pour plus d'informations sur les mécanismes, reportez-vous aux pages de manuel `mech`(4) et `mech_spnego`(5).

Exécution des commandes et transfert de données dans le shell sécurisé

Une fois l'authentification terminée, l'utilisateur peut utiliser Shell sécurisé, généralement en demandant un shell ou en exécutant une commande. Par l'intermédiaire des options de commande `ssh`, l'utilisateur peut effectuer des demandes. Les demandes peuvent inclure l'allocation d'un pseudo-tty, le transfert des connexions X11 ou TCP/IP ou l'activation d'un programme d'authentification `ssh-agent` via une connexion sécurisée.

Les composants de base d'une session utilisateur sont les suivants :

1. L'utilisateur demande un shell ou l'exécution d'une commande, ce qui lance le mode de session.

   Dans ce mode, les données sont envoyées ou reçues par le biais du terminal sur le côté client. Sur le côté serveur, les données sont envoyées par l'intermédiaire du shell ou d'une commande.
2. Lorsque la transmission des données est terminée, le programme utilisateur s'arrête.
3. L'ensemble du transfert X11 et du transfert TCP/IP est arrêté, sauf pour les connexions qui existent déjà. Les connexions X11 et TCP/IP existantes restent ouvertes.
4. Le serveur envoie un message d'état de sortie au client. Lorsque toutes les connexions sont fermées, telles que les ports transmis qui étaient restés ouverts, le client ferme la connexion au serveur. Ensuite, le client se ferme.

Configuration client et serveur dans le shell sécurisé

Les caractéristiques d'une session de shell sécurisé sont contrôlées par les fichiers de configuration. Les fichiers de configuration peuvent être remplacés dans une certaine mesure par des options de la ligne de commande.

Configuration client dans le shell sécurisé

Dans la plupart des cas, les caractéristiques côté client d'une session Shell sécurisé sont régies par le fichier de configuration à l'échelle du système, `/etc/ssh/ssh_config`. Les paramètres dans le fichier `ssh_config` peuvent être remplacés par le fichier de configuration de l'utilisateur, `~/.ssh/config`. En outre, l'utilisateur peut remplacer les deux fichiers de configuration sur la ligne de commande.

Les paramètres du fichier `/etc/ssh/sshd_config` du serveur déterminent quelles demandes client sont autorisées par le serveur. Pour obtenir la liste des paramètres de configuration de serveur, reportez-vous à la section “Mots-clés dans le shell sécurisé” à la page 37. Pour plus d'informations, reportez-vous à la page de manuel `sshd_config`(4).

Les mots-clés du fichier de configuration du client sont répertoriés dans la section “Mots-clés dans le shell sécurisé” à la page 37. Si le mot-clé a une valeur par défaut, la valeur est donnée. Ces mots-clés sont décrits en détail dans les pages de manuel `ssh`(1), `scp`(1), `sftp`(1) et `ssh_config`(4). Pour obtenir la liste des mots-clés dans l'ordre alphabétique et leurs substituts de ligne de commande équivalents, reportez-vous au Tableau 2-5, “Equivalents de ligne de commande des mots-clés de shell sécurisé”.

Configuration du serveur dans le shell sécurisé

Les caractéristiques côté serveur d'une session Shell sécurisé sont régies par le fichier `/etc/ssh/sshd_config`. Les mots-clés dans le fichier de configuration du serveur sont répertoriés dans “Mots-clés dans le shell sécurisé” à la page 37. Si le mot-clé a une valeur par défaut, la valeur est donnée. Pour une description complète des mots-clés, reportez-vous à la page de manuel `sshd_config`(4).

Mots-clés dans le shell sécurisé

Les tableaux ci-dessous répertorient les mots-clés et leurs valeurs par défaut, le cas échéant. Les mots-clés sont dans l'ordre alphabétique. Les mots-clés qui s'appliquent au client sont dans le fichier `ssh_config`. Les mots-clés qui s'appliquent au serveur sont dans le fichier `sshd_config`. Certains mots-clés sont définis dans les deux fichiers. Les mots-clés pour un serveur Shell sécurisé exécutant le protocole v1 sont marqués.

TABLEAU 2-1 Mots-clés dans les fichiers de configuration du shell sécurisé

Mot-clé	Valeur par défaut	Localisation
`AllowGroups`		Serveur
`AllowTcpForwarding`	`yes`	Serveur
`AllowUsers`		Serveur
`AuthorizedKeysFile`	`~/.ssh/authorized_keys`	Serveur
`Banner`	`/etc/issue`	Serveur
`BatchMode`	`no`	Client
`BindAddress`		Client
`CheckHostIP`	`yes`	Client
`ChrootDirectory`	`no`	Serveur
`Cipher`	`blowfish`, `3des`	Client
`Ciphers`	`aes128-ctr, aes128-cbc, 3des-cbc, blowfish-cbc, arcfour`	Les deux
`ClearAllForwardings`	`no`	Client
`ClientAliveCountMax`	`3`	Serveur
`ClientAliveInterval`	`0`	Serveur
`Compression`	`no`	Les deux
`CompressionLevel`		Client
`ConnectionAttempts`	`1`	Client
`ConnectTimeout`	Délai d'attente TCP système	Client

Mot-clé	Valeur par défaut	Localisation
DenyGroups		Serveur
DenyUsers		Serveur
DisableBanner	no	Client
DynamicForward		Client
EscapeChar	~	Client
FallBackToRsh	no	Client
ForwardAgent	no	Client
ForwardX11	no	Client
ForwardX11Trusted	yes	Client
GatewayPorts	no	Les deux
GlobalKnownHostsFile	/etc/ssh/ssh_known_hosts	Client
GSSAPIAuthentication	yes	Les deux
GSSAPIDelegateCredentials	no	Client
GSSAPIKeyExchange	yes	Les deux
GSSAPIStoreDelegateCredentials	yes	Serveur
HashKnownHosts	no	Client
Host	* Pour plus d'informations, reportez-vous à la section “Paramètres spécifiques de l'hôte dans le shell sécurisé” à la page 40.	Client
HostbasedAuthentication	no	Les deux
HostbasedUsesNameFromPacketOnly	no	Serveur
HostKey (v1)	/etc/ssh/ssh_host_key	Serveur
HostKey (v2)	/etc/ssh/host_rsa_key, /etc/ssh/host_dsa_key	Serveur
HostKeyAlgorithms	ssh-rsa, ssh-dss	Client
HostKeyAlias		Client
HostName		Client
IdentityFile	~/.ssh/id_dsa, ~/.ssh/id_rsa	Client
IgnoreIfUnknown		Client
IgnoreRhosts	yes	Serveur
IgnoreUserKnownHosts	yes	Serveur
KbdInteractiveAuthentication	yes	Les deux
KeepAlive	yes	Les deux
KeyRegenerationInterval	3600 (seconds)	Serveur
ListenAddress		Serveur
LocalForward		Client
LoginGraceTime	120 (seconds)	Serveur
LogLevel	info	Les deux

Mot-clé	Valeur par défaut	Localisation
LookupClientHostnames	yes	Serveur
MACs	Algorithmes hmac-sha1-*, hmac-md5-* et hmac-sha2-*.	Les deux
Match		Serveur
MaxStartups	10:30:60	Serveur
NoHostAuthenticationForLocalHost	no	Client
NumberOfPasswordPrompts	3	Client
PAMServiceName		Serveur
PAMServicePrefix		Serveur
PasswordAuthentication	yes	Les deux
PermitEmptyPasswords	no	Serveur
PermitRootLogin	no	Serveur
PermitUserEnvironment	no	Serveur
PidFile	/system/volatile/sshd.pid	Serveur
Port	22	Les deux
PreferredAuthentications	hostbased,publickey,keyboard-interactive,passwor	Client
PreUserauthHook		Serveur
PrintLastLog	yes	Serveur
PrintMotd	no	Serveur
Protocol	2,1	Les deux
ProxyCommand		Client
PubkeyAuthentication	yes	Les deux
RekeyLimit	1G à 4G	Client
RemoteForward		Client
RhostsAuthentication	no	Serveur, v1
RhostsRSAAuthentication	no	Serveur, v1
RSAAuthentication	no	Serveur, v1
ServerAliveCountMax	3	Client
ServerAliveInterval	0	Client
ServerKeyBits	512 à 768	Serveur, v1
StrictHostKeyChecking	ask	Client
StrictModes	yes	Serveur
Sous-système	sftp/usr/lib/ssh/sftp-server	Serveur
SyslogFacility	auth	Serveur
UseFIPS140	no	Les deux
UseOpenSSLEngine	yes	Les deux
UsePrivilegedPort	no	Les deux

Mot-clé	Valeur par défaut	Localisation
`User`		Client
`UserKnownHostsFile`	`~/.ssh/known_hosts`	Client
`UseRsh`	`no`	Client
`VerifyReverseMapping`	`no`	Serveur
`X11DisplayOffset`	`10`	Serveur
`X11Forwarding`	`yes`	Serveur
`X11UseLocalHost`	`yes`	Serveur
`XAuthLocation`	`/usr/bin/xauth`	Les deux

Paramètres spécifiques de l'hôte dans le shell sécurisé

Il peut quelquefois s'avérer utile d'avoir des caractéristiques de shell sécurisé différentes pour différents hôtes locaux. L'administrateur peut définir différents ensembles de paramètres dans le fichier `/etc/ssh/ssh_config` à appliquer en fonction de l'hôte ou d'une expression régulière en regroupant les entrées dans le fichier selon le mot-clé `Host`. Si le mot-clé `Host` n'est pas utilisé, les entrées dans le fichier de configuration du client s'appliquent à n'importe lequel des hôtes locaux sur lesquels un utilisateur travaille.

Shell sécurisé et variables d'environnement de connexion

Si les mots-clés du shell sécurisé suivants ne sont pas définis dans le fichier `sshd_config`, ils obtiennent leur valeur des entrées équivalentes à partir du fichier `/etc/default/login`.

Entrée dans `/etc/default/login`	Mot-clé et valeur dans `sshd_config`
`CONSOLE=*`	`PermitRootLogin=without-password`
`#CONSOLE=*`	`PermitRootLogin=yes`
`PASSREQ=YES`	`PermitEmptyPasswords=no`
`PASSREQ=NO`	`PermitEmptyPasswords=yes`
`#PASSREQ`	`PermitEmptyPasswords=no`
`TIMEOUT=`*seconds*	`LoginGraceTime=`*seconds*
`#TIMEOUT`	`LoginGraceTime=120`
`RETRIES` et `SYSLOG_FAILED_LOGINS`	S'appliquent uniquement aux méthodes d'authentification `password` et `keyboard-interactive`

Lorsque les variables suivantes sont définies par les scripts d'initialisation du shell de connexion de l'utilisateur, le démon `sshd` utilise ces valeurs. Lorsque les variables ne sont pas définies, le démon utilise la valeur par défaut.

`TIMEZONE`	Contrôle la définition de la variable d'environnement `TZ`. Lorsque cette variable n'est pas définie, le démon `sshd` utilise la valeur de `TZ` telle qu'elle était au moment de son démarrage.
`ALTSHELL`	Contrôle la définition de la variable d'environnement `SHELL`. La valeur par défaut est `ALTSHELL=YES`, où le démon `sshd` utilise la valeur de shell de l'utilisateur. Quand `ALTSHELL=NO`, la valeur de `SHELL` n'est pas définie.
`PATH`	Contrôle la définition de la variable d'environnement `PATH`. Lorsque la valeur n'est pas définie, le chemin d'accès par défaut est `/usr/bin`.
`SUPATH`	Contrôle la définition de la variable d'environnement `SUPATH` pour `root`. Lorsque la valeur n'est pas définie, le chemin d'accès par défaut est `/usr/sbin:/usr/bin`.

Pour plus d'informations, reportez-vous aux pages de manuel `login`(1) et `sshd`(1M).

Gestion des hôtes connus dans le shell sécurisé

Chaque hôte qui doit communiquer de manière sécurisée avec un autre hôte doit avoir la clé publique du serveur stockée dans le fichier `/etc/ssh/ssh_known_hosts` de l'hôte local. Bien qu'un script puisse être utilisé pour mettre à jour les fichiers `/etc/ssh/ssh_known_hosts`, une telle pratique est fortement déconseillée parce qu'un script crée une grande vulnérabilité de la sécurité.

Le fichier `/etc/ssh/ssh_known_hosts` doit uniquement être distribué par un mécanisme sécurisé comme suit :

- Via une connexion sécurisée, comme par exemple le shell sécurisé, IPsec, ou `ftp` utilisant Kerberos à partir d'une machine connue et de confiance.
- Au moment de l'installation

Pour éviter toute possibilité qu'un intrus obtienne l'accès en insérant de fausses clés publiques dans un fichier `known_hosts`, vous devez utiliser une source connue et de confiance du fichier `ssh_known_hosts`. Le fichier `ssh_known_hosts` peut être distribué au cours de l'installation. Plus tard, les scripts utilisant la commande `scp` peuvent être utilisés pour copier la version la plus récente.

Fichiers de shell sécurisé

Le tableau suivant présente les principaux fichiers du shell sécurisé et les autorisations de fichiers suggérées.

TABLEAU 2-2 Fichiers de shell sécurisé

Nom de fichier	Description	Autorisations suggérées et propriétaire
`~/.rhosts`	Contient les paires de noms hôte-utilisateur qui permettent d'indiquer les hôtes auxquels l'utilisateur peut se connecter sans mot de passe. Ce fichier est également utilisé par les démons `rlogind` et `rshd`.	`-rw-r--r--` *username*
`~/.shosts`	Contient les paires de noms hôte-utilisateur qui permettent d'indiquer les hôtes auxquels l'utilisateur peut se connecter sans mot de passe. Ce fichier n'est utilisé par aucun autre utilitaire. Pour plus d'informations, reportez-vous à la page de manuel `sshd`(1M) dans la section `FILES`.	`-rw-r--r--` *username*
`~/.ssh/authorized_keys`	Contient les clés publiques de l'utilisateur qui est autorisé à se connecter au compte utilisateur.	`-rw-r--r--` *username*
`~/.ssh/config`	Permet de configurer les paramètres utilisateur qui remplacent les paramètres système.	`-rw-r--r--` *username*
`~/.ssh/environment`	Contient les affectations initiales au moment de la connexion. Par défaut, ce fichier n'est pas lu. Le mot-clé `PermitUserEnvironment` du fichier `sshd_config` doit être défini sur `yes` pour que ce fichier soit lu.	`-rw-r--r--` *username*
`/etc/hosts.equiv`	Contient les hôtes qui sont utilisés dans l'authentification `.rhosts`. Ce fichier est également utilisé par les démons `rlogind` et `rshd`.	`-rw-r--r-- root`
`~/.ssh/known_hosts`	Contient les clés publiques pour tous les hôtes avec lesquels le client peut communiquer de manière sécurisée. Le fichier est mis à jour automatiquement. Chaque fois que l'utilisateur se connecte à l'aide d'un hôte inconnu, la clé de l'hôte distant est ajoutée au fichier.	`-rw-r--r--` *username*
`/etc/default/login`	Fournit les valeurs par défaut pour le démon `sshd` lorsque les paramètres `sshd_config` correspondants ne sont pas définis.	`-r--r--r-- root`
`/etc/nologin`	Si ce fichier existe, le démon `sshd` n'autorise que `root` à se connecter. Le contenu de ce fichier est affiché pour les utilisateurs qui tentent de se connecter.	`-rw-r--r-- root`
`~/.ssh/rc`	Contient les routines d'initialisation qui sont exécutées avant que le shell utilisateur ne démarre. Pour un exemple de routine d'initialisation, reportez-vous à la page de manuel `sshd`(1M).	`-rw-r--r--` *username*
`/etc/ssh/shosts.equiv`	Contient les hôtes qui sont utilisés dans l'authentification basée sur les hôtes. Ce fichier n'est utilisé par aucun autre utilitaire.	`-rw-r--r-- root`
`/etc/ssh/ssh_config`	Configure les paramètres système sur le système client.	`-rw-r--r-- root`

Nom de fichier	Description	Autorisations suggérées et propriétaire
`/etc/ssh/ssh_host_dsa_key` ou `/etc/ssh/ssh_host_rsa_key`	Contient la clé privée de l'hôte.	`-rw------- root`
`/etc/ssh_host_key.pub` ou `/etc/ssh/ssh_host_dsa_key.pub` ou `/etc/ssh/ssh_host_rsa_key.pub`	Contient la clé publique de l'hôte, par exemple, `/etc/ssh/ssh_host_rsa_key.pub`. Utilisé pour copier la clé d'hôte dans le fichier `known_hosts` local.	`-rw-r--r-- root`
`/etc/ssh/ssh_known_hosts`	Contient les clés publiques pour tous les hôtes avec lesquels le client peut communiquer de manière sécurisée. Le fichier est renseigné par l'administrateur.	`-rw-r--r-- root`
`/etc/ssh/sshd_config`	Contient des données de configuration pour `sshd`, le démon du shell sécurisé.	`-rw-r--r-- root`
`/system/volatile/sshd.pid`	Contient l'ID de processus du démon du shell sécurisé, `sshd`. Si plusieurs démons sont en cours d'exécution, le fichier contient la dernier démon qui a été démarré.	`-rw-r--r-- root`
`/etc/ssh/sshrc`	Contient les routines d'initialisation spécifiques à un hôte qui sont spécifiées par un administrateur.	`-rw-r--r-- root`

Remarque - Le fichier `sshd_config` peut être remplacé par un fichier issu d'un package personnalisé au niveau du site. Pour plus d'informations, reportez-vous à la définition de l'attribut de fichier `overlay` dans la page de manuel `pkg`(5).

Le tableau ci-dessous répertorie les fichiers du shell sécurisé qui peuvent être remplacés par des mots-clés ou des options de commande.

TABLEAU 2-3 Remplacement pour l'emplacement des fichiers du shell sécurisé

Nom de fichier	Remplacement de mot-clé	Remplacement de ligne de commande
`/etc/ssh/ssh_config`		`ssh -F` *config-file* `scp -F` *config-file*
`~/.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`	

Commandes de shell sécurisé

Le tableau suivant récapitule les principales commandes de shell sécurisé.

TABLEAU 2-4 Commandes dans le shell sécurisé

Page de manuel pour les commandes	Description
ssh(1)	Connecte un utilisateur à une machine distante et exécute de manière sécurisée les commandes sur une machine distante. La commande `ssh` permet de protéger les communications chiffrées entre deux hôtes non fiables sur un réseau non sécurisé. Les connexions X11 et les ports TCP/IP arbitraires peuvent également être transmis via le canal sécurisé.
sshd(1M)	Démon du shell sécurisé. Le démon détecte les connexions des clients et sécurise les communications chiffrées entre deux hôtes non fiables sur un réseau non sécurisé.
ssh-add(1)	Ajoute des identités RSA ou DSA à l'agent d'authentification, `ssh-agent`. Les identités sont également appelées *clés*.
ssh-agent(1)	Contient les clés privées utilisées pour l'authentification avec clé publique. Le programme `ssh-agent` est lancé au début d'une session X ou d'une session de connexion. Toutes les autres fenêtres et les autres programmes sont lancés en tant que clients du programme `ssh-agent`. Par le biais de l'utilisation de variables d'environnement, l'agent peut être localisé et utilisé pour l'authentification lorsque les utilisateurs utilisent la commande `ssh` pour se connecter à d'autres systèmes.
ssh-keygen(1)	Génère et gère des clés d'authentification pour le shell sécurisé.
ssh-keyscan(1)	Regroupe les clés publiques d'un certain nombre d'hôtes de shell sécurisé. Facilite la création et la vérification des fichiers `ssh_known_hosts`.
ssh-keysign(1M)	Permet à la commande `ssh` d'accéder aux clés d'hôte sur l'hôte local. Génère la signature numérique requise pendant l'authentification basée sur l'hôte avec le shell sécurisé v2. La commande est appelée par la commande `ssh`, et non par l'utilisateur.
scp(1)	Copie les fichiers de manière sécurisée entre les hôtes d'un réseau via un transport `ssh` chiffré. Contrairement à la commande `rcp`, la commande `scp` invite à saisir les mots de passe ou les phrases de passe si des informations de mot de passe sont nécessaires pour l'authentification.
sftp(1)	Programme de transfert de fichier interactif similaire à la commande `ftp`. Contrairement à la commande `ftp`, la commande `sftp` effectue toutes les opérations sur un transport `ssh` chiffré. La commande établit la connexion, se connecte au nom d'hôte spécifié, puis entre en mode de commande interactif.

Le tableau suivant répertorie les options de commande qui remplacent les mots-clés de shell sécurisé. Les mots-clés sont spécifiés dans les fichiers `ssh_config` et `sshd_config`.

TABLEAU 2-5 Equivalents de ligne de commande des mots-clés de shell sécurisé

Mot-clé	Remplacement de ligne de commande `ssh`	Remplacement de ligne de commande `scp`
`BatchMode`		`scp -B`

Mot-clé	Remplacement de ligne de commande `ssh`	Remplacement de ligne de commande `scp`
`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` pour activer `ssh -a` pour désactiver	
`ForwardX11`	`ssh -X` pour activer `ssh -x` pour désactiver	
`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` pour v2 uniquement	
`RemoteForward`	`ssh -R` *remoteport:localhost:localport*	

Index

Nombres et symboles

A

B

C

D

E

F

G

H

K

L

M

N

P

R

S

T

U

V

X