SSH サーバセキュリティ設定ガイド

Ver 1.0

日本コンピュータセキュリティインシデント対応チーム協議会

－目次－

更新履歴

バージョン	日付	内容
1.0	2015-03-06	初版
1.0 Rev.1	2015-03-15	改訂（例題を 1024bit の DSA から 2048bit の RSA に変更）

本ガイドは、下記 URL からダウンロードできます。

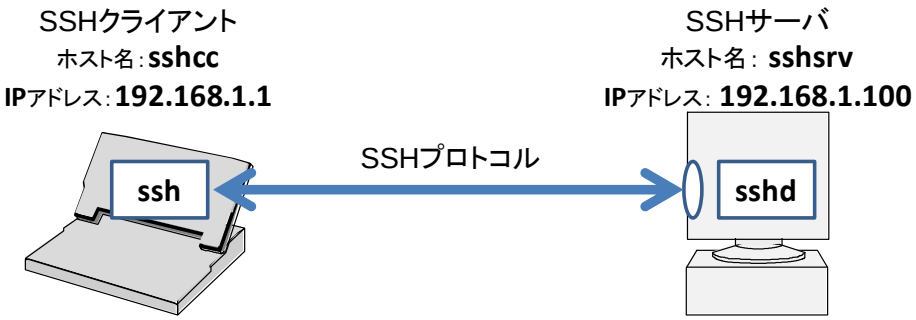
SSH サーバセキュリティ設定ガイド Ver 1.0
http://www.nca.gr.jp/activity/sshconfig-wg.html

1　はじめに

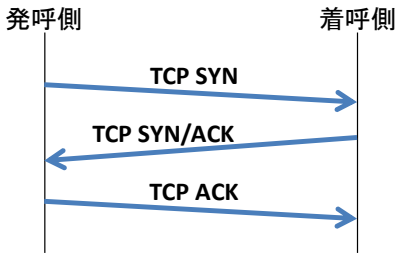
SSH サービスは、サーバのリモート保守やファイル転送などで利用できます。その反面、サイバー攻撃活動の攻撃対象となり、踏み台として悪用され、第三者への不正アクセスに加担してしまう可能性があります。本ガイドでは、SSH サーバをサイバー攻撃から守るためのセキュリティ設定について解説します。本ガイドが、SSH サーバのセキュリティ対策を改善する一助となれば幸いです。

2　前提条件

- 対象読者は、企業や個人を問わず、SSH サーバの構築や運用に関わる方で、TCP/IP などのネットワークの基本的な仕組みを理解している方を想定しています。
- 本ガイドは、CentOS 6.5、OpenSSH 5.3p1 のクライアントとサーバ環境（openssh-5.3p1-94.el6.i686、openssh-clients-5.3p1-94.el6.i686、openssh-server-5.3p1-94.el6.i686）で動作確認を行っています。Mac、Linux、Cygwin 環境であれば、本ガイドの設定は参考となりますが、他のプラットフォーム上で OpenSSH を設定する際には、マニュアル等を参照し、動作確認を行ってください。
- 第 4 章は、該当する設定項目のみを変更して動作確認をしています。ただし、動作を保証するものではありません。また、本ガイドに示す設定内容は、あくまで一例ですので、自分の環境に置き換えて読み進めてください。
- 用語説明



SSH サーバ	sshd の稼働するホスト
SSH サービス	sshd が提供するサービス
sshd	SSH デーモン（/usr/sbin/sshd）
SSH クライアント	SSH コマンドを実行するホスト
ssh	SSH コマンド（/usr/bin/ssh）
発信元 IP アドレス	送信元 IP アドレス、ソース IP アドレスと呼ばれることもあります。
送信先 IP アドレス	宛先 IP アドレスと呼ばれることもあります。



発呼	通信開始のトリガをかけたホストが発呼側です。 TCP 通信の開始を宣言する SYN パケットを送出した側となります。
着呼	TCP 通信の開始を宣言する SYN パケットを受信した側となります。

図 1：用語説明

- ガイドの構成
  第 4 章は、設定方法（設定はどのようにすれば良いのか）、確認方法（設定をどのように確認すれば良いのか）、参考情報から構成されています。また、設定ファイル、SSH クライアントでのコンソール画面操作、SSH サーバでのコンソール画面操作については、例示のタイトル行の背景色を色分けしてあります（表 1）。

表 1：例示の説明

例示の記載例	説明
File: /etc/ssh/sshd_config 設定ファイル 	設定ファイルの場合、背景色：灰色、タイトル：“File: ファイルの位置”で記載しています。本ガイドで使用する設定ファイルは、6 ファイルです。 ● /etc/ssh/sshd_config ● /etc/ssh/ssh_config ● ~/.ssh/config ● /etc/sysconfig/iptables ● /etc/hosts.deny ● /etc/hosts.allow
Host: sshcc SSH クライアントでの コンソール画面操作 	SSH クライアントでのコンソール画面操作の場合、背景色：黄色、タイトル：“Host: ホスト名”で記載しています。本ガイドでは、ホスト名「sshcc」、「sshxx」のコンソール画面を説明に使用しています。
Host: sshsrv SSH サーバでの コンソール画面操作 	SSH サーバでのコンソール画面操作の場合、背景色：青色、タイトル：“Host: ホスト名”で記載しています。本ガイドでは、ホスト名「sshsrv」のコンソール画面を説明に使用しています。 また、[root@sshsrv ~]の記載は、root 権限で SSH サーバ（sshsrv）上で設定ならびに確認作業していることを示しています。

【コラム I:設定ファイルの文法チェック】

設定ファイルの文法が正しいかどうか、事前にチェックしたいことがあります。
ここでは、sshd の設定ファイル（/etc/ssh/sshd_config）の文法チェックについて紹介します。sshd では、設定ファイルの文法チェックだけを行うテストモードとして、-t オプションをサポートしています。このオプションを利用すると、設定ファイル項目が適切ではない場合、メッセージで該当箇所を表示してくれます。

Host: sshsrv

```
[root@sshsrv ~]# /usr/sbin/sshd -t ⏎
/etc/ssh/sshd_config line 143: Deprecated option RhostsAuthentication
/etc/ssh/sshd_config line 144: Bad SSH2 mac spec 'hmac-sha2-512-etm@openssh.com'.
```

3　チェックリスト

本ガイドで推奨する事項をチェックリスト化したものです。詳細は、第 4 章を参照してください。

表 2：チェックリスト

分類	項番	項目	○／×
ネットワーク	4.1.1	不要なサービスを停止／無効化する【推奨】	
ネットワーク	4.1.2	特定のホストからの接続だけを許可する【推奨】	
SSH サービス	4.2.1	SSH プロトコルのバージョン 2 のみを許可する【推奨】	
SSH サービス	4.2.2	公開鍵認証を使用する（パスワード認証を無効化する）【推奨】	
SSH サービス	4.2.3	ポートフォワードを止める【推奨】	
SSH サービス	4.2.4	SSH サービスを 22/TCP 以外で稼働させる【推奨】	
SSH サービス	4.2.5	known_hosts ファイルをハッシュ化する【推奨】 （ログインホストを設置する場合）	
鍵管理	4.3.1	ユーザ秘密鍵ファイルにパスフレーズをつけること【推奨】	
鍵管理	4.3.2	SSH サーバ上にユーザ秘密鍵ファイルを置かない【推奨】	
ネットワーク	4.1.3	一定時間内の SSH サービスへのアクセス数を制限する【オプション】	
ネットワーク	4.1.4	アウトバウンド（発呼）通信に対するアクセスを制限する【オプション】	
SSH サービス	4.2.6	ログインを特定のユーザだけに許可する【オプション】	
SSH サービス	4.2.7	IPv6 を使わない場合には、無効化する【オプション】	
SSH サービス	4.2.8	弱い暗号方式やメッセージ認証コードを利用しない【オプション】	

4　SSH サーバの要塞化

4.1　ネットワークサービス

4.1.1.　不要なサービスを停止／無効化する【推奨】

SSH サーバ上で稼働している SSH サービス以外への攻撃による脅威を低減するために、不要なサービスを停止あるいは、無効化します。SSH サービスは、TELNET、FTP サービスと同等の機能を備えていますので、TELNET、FTP サービスを利用している場合には、これらのサービスを停止あるいは、無効化してください。

■設定方法

サービスを停止する場合には、コマンド（/sbin/service）を使用します。

サービスを無効化する場合（再起動してもサービスが起動しないようにする場合）には、コマンド（/sbin/service）で停止した後、コマンド（/sbin/chkconfig）で無効化します。コマンド（/sbin/chkconfig）で off を指定することにより、すべての項目（0～6）が off となります。

Host: sshsrv

```
■サービスを停止する
[root@sshsrv ~]# /sbin/service avahi-daemon stop ⏎
Shutting down Avahi daemon:                                [  OK  ]
[root@sshsrv ~]# /sbin/service cups stop ⏎
Stopping cups:                                             [  OK  ]

■サービスを無効化する（再起動してもサービスが起動しないようにする）
[root@sshsrv ~]# /sbin/chkconfig avahi-daemon off ⏎
[root@sshsrv ~]# /sbin/chkconfig --list avahi-daemon ⏎
avahi-daemon    0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

■確認方法

コマンド（/bin/netstat）を使って、稼働しているサービスを確認します。不要なサービスが稼働している場合には、コマンド（/sbin/service、/sbin/chkconfig）を使ってサービスを停止／無効化します。

Host: sshsrv

```
[root@sshsrv ~]# /bin/netstat -pantu ⏎
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address       State       PID/Program name
tcp        0      0 0.0.0.0:111         0.0.0.0:*             LISTEN      1449/rpcbind
tcp        0      0 0.0.0.0:22          0.0.0.0:*             LISTEN      3777/sshd
tcp        0      0 0.0.0.0:45014       0.0.0.0:*             LISTEN      1467/rpc.statd
tcp        0      0 127.0.0.1:631       0.0.0.0:*             LISTEN      1810/cupsd
tcp        0      0 127.0.0.1:25        0.0.0.0:*             LISTEN      1762/sendmail
udp        0      0 0.0.0.0:776         0.0.0.0:*                         1449/rpcbind
udp        0      0 0.0.0.0:795         0.0.0.0:*                         1467/rpc.statd
udp        0      0 0.0.0.0:43740       0.0.0.0:*                         1467/rpc.statd
udp        0      0 0.0.0.0:111         0.0.0.0:*                         1449/rpcbind
udp        0      0 0.0.0.0:631         0.0.0.0:*                         1810/cupsd
```

4.1.2. 特定のホストからの接続だけを許可する【推奨】

SSH サーバにアクセスできる発信元 IP アドレスを制限することで、第三者からの不正アクセスを低減できます。SSH サーバをファイアウォールの内側に設置する場合にも、同様です。ここでは、iptables と TCP_wrapper を用いて制限する方法を紹介します。

（1）iptables を使用する場合

■設定方法

iptables の設定ファイル（/etc/sysconfig/iptables）で設定した後、設定を反映します（/sbin/service iptables restart）。

File: /etc/sysconfig/iptables

```
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp -s 192.168.1.0/24 --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp -s 198.51.100.0/24 --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp -s 203.0.113.0/32 --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited

# SSH サービスにアクセスできる発信元 IP アドレスを
# 192.168.1.0～192.168.1.255、198.51.100.0～198.51.100.255、203.0.113.1 に制限する。
```

■確認方法

コマンド（/sbin/iptables）を使って、設定の状態を確認します。

Host: sshsrv

```
[root@sshsrv ~]# /sbin/iptables -L -n ↵
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
ACCEPT     tcp  --  192.168.1.0/24       0.0.0.0/0           state NEW tcp dpt:22
ACCEPT     tcp  --  198.51.100.0/24      0.0.0.0/0           state NEW tcp dpt:22
ACCEPT     tcp  --  203.0.113.1/32       0.0.0.0/0           state NEW tcp dpt:22
REJECT     all  --  0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited
```

■参考情報

reject-with icmp-host-prohibited では、INPUT に定義されていないパケットを受信した場合に、該当するパケットを遮断し、ICMP パケット（type=3：Destination Unreachable、code=10：Host administratively prohibited）を応答します。

（2）TCP_wrapper を使用する場合

■設定方法

TCP_wrapper の設定ファイル（/etc/hosts.deny、/etc/hosts.allow）で設定します。設定を反映するための追加操作はありません。

File: /etc/hosts.deny

```
sshd : ALL
```

File: /etc/hosts.allow

```
sshd : 192.168.1.0/255.255.255.0
sshd : 198.51.100.0/255.255.255.0
sshd : 203.0.113.1/255.255.255.255

# SSH サービスにアクセスできる発信元 IP アドレスを
# 192.168.1.0～192.168.1.255、198.51.100.0～198.51.100.255、203.0.113.1 に制限する。
```

■確認方法

許可した発信元 IP アドレス以外からアクセスを試みた後（[taro@evil]$ /usr/bin/ssh taro@sshsrv）、SSH サーバ（sshsrv）上のログファイル（/var/log/secure）に、『refused connect from・・・』と記録されていることを確認します。

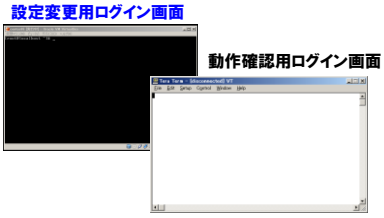
Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/tail -f /var/log/secure ⏎
Apr 29 12:00:04 sshsrv sshd[23071]: refused connect from 172.16.11.10 (172.16.11.10)
Apr 29 12:02:04 sshsrv sshd[23075]: refused connect from 172.16.11.10 (172.16.11.10)
Apr 29 12:02:36 sshsrv sshd[23078]: refused connect from 172.16.11.10 (172.16.11.10)
```

【コラム II:その設定で大丈夫ですか？】

SSH サーバの設定を変更する前に、いま一度、確認をしましょう。設定によっては、自分自身がリモートからログインできなくなり、現地に駆けつけて解決する必要があります。
ここでは、ありがちな失敗例を紹介します。

- 自身の運用管理端末の IP アドレスを、iptables の許可リストに追加するのを忘れる。
- SSH サービスの稼働するポート番号を変更したにも関わらず、iptables の設定変更を忘れる。
- パスワード認証を無効化する前に、公開鍵認証で使用するユーザ公開鍵を登録するのを忘れる。
- 『root でのログインを許可しない』を有効とする前に、一般ユーザアカウントの追加を忘れる。



回避方法として、設定変更用ログイン画面と動作確認用ログイン画面の 2 つを用意します。ログインとログアウトを繰り返す動作確認には、動作確認用ログイン画面のみを使用します。設定変更には、設定変更用ログイン画面のみを使用し、設定変更が完了するまでログアウトしない、画面を終了しないことです。

4.1.3. 一定時間内の SSH サービスへのアクセス数を制限する【オプション】

SSH サービスを対象としたブルートフォース攻撃では、連続してログイン試行を繰り返すため、同一発信元 IP アドレスからのアクセスを抑制することで、不正ログインの成功を低減できます。ここでは、iptables の hashlimit モジュールを使った一定期間内のアクセス数を制限する方法、OpenSSH の MaxStartups 設定で同時アクセス数を制限する方法を紹介します。

（1）iptables の hashlimit モジュールを使用する場合

■設定方法

iptables の設定ファイル（/etc/sysconfig/iptables）で設定した後、設定を反映します（/sbin/service iptables restart）。

File: /etc/sysconfig/iptables

```
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22  -m hashlimit --hashlimit-name ssh --hashlimit-burst
3 --hashlimit 2/m --hashlimit-mode srcip --hashlimit-htable-expire 180000 -j ACCEPT

# SSH サービスに対して、同一ホスト（--hashlimit-mode srcip)からの接続回数が 3 回を越えた場合
#  （--hashlimit-burst 3）、その後は 1 分間に 2 回まで接続を許可する（--hashlimit 2/m）。
# 最後の接続から 3 分間が経過すると（--hashlimit-htable-expire 180000）、接続制限が解除される。
#  --hashlimit-name：名称
#  --hashlimit-burst：接続回数
#  --hashlimit：--hashlimit-burst で指定した接続回数以降の許可レート
#  --hashlimit-mode：制限を掛ける単位（dstip/srcip/dstport/srcport）
#  --hashlimit-htable-expire：接続制限を解除するまでの時間（ミリ秒）
```

■参考情報

iptables の hashlimit モジュールは、SSH サービスに対して、同一ホストからの接続回数が上限を越えた場合、その後、単位時間あたりの接続を制限します。また、最後の接続から一定時間が経過したところで、単位時間あたりの接続制限を解除するという仕組みです。例えば、上記の設定では、5 秒間隔でアクセスした場合、最初の 3 回に制限はありませんが、以降については単位時間あたりの接続制限がかかります。

Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/tail -f /var/log/secure ⏎
Aug  2 10:45:35 sshsrv sshd[1398]: Bad protocol version identification '1 ' from 172.16.11.10
Aug  2 10:45:40 sshsrv sshd[1399]: Bad protocol version identification '2 ' from 172.16.11.10
Aug  2 10:45:45 sshsrv sshd[1400]: Bad protocol version identification '3 ' from 172.16.11.10
Aug  2 10:46:16 sshsrv sshd[1402]: Bad protocol version identification '5 ' from 172.16.11.10
Aug  2 10:46:47 sshsrv sshd[1403]: Bad protocol version identification '7 ' from 172.16.11.10

# 最初の 3 回は制限がないため 5 秒間隔でのアクセス（No.1, 2, 3 で番号も連続）となっています。
# 4 回目以降については単位時間あたりの接続制限がかかり、約 30 秒間隔でのアクセス（No. 5, 7 で
# 番号が飛んでいる）となっています。
```

（2）OpenSSH の MaxStartups 設定を使用する場合

■設定方法

MaxStartups は、認証成功前の同時アクセス数の上限を設定するパラメータです。sshd の設定ファイル（/etc/ssh/sshd_config）で設定した後、設定を反映します（/sbin/service sshd restart）。

File: /etc/ssh/sshd_config

```
MaxStartups 3              # 認証成功前の同時アクセス数を 3 件に設定する
LogLevel DEBUG             # 拒否された接続記録を見る場合
```

■参考情報

MaxStartups は、認証されていない段階の接続を sshd が最大でどれだけ受けつけるかを指定します。この値を超えた（認証されていない段階の）接続を受け付けません。LogLevel DEBUG にすると、拒否された接続の記録を見ることができます。

設定フォーマットは、"start:rate:full"（開始時:確率:最大数）で、デフォルトの設定は 10:30:100 です。この設定の場合、認証されていない段階の接続要求が 10 件（start）を超えると、これ以降の接続要求を 30%（rate）の確率で拒否します。認証されていない段階の接続要求が 100 件（full）を越えると、これ以降の接続要求をすべて拒否します。

Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/tail -f /var/log/secure ⏎
Aug  2 11:16:15 sshsrv sshd[1536]: debug1: Forked child 1544.
Aug  2 11:16:15 sshsrv sshd[1544]: Set /proc/self/oom_score_adj to 0
Aug  2 11:16:15 sshsrv sshd[1544]: debug1: rexec start in 5 out 5 newsock 5 pipe 9 sock 10
Aug  2 11:16:15 sshsrv sshd[1544]: debug1: inetd sockets after dupping: 3, 3
Aug  2 11:16:15 sshsrv sshd[1544]: Connection from 172.16.11.10 port 49784
Aug  2 11:16:18 sshsrv sshd[1536]: debug1: drop connection #3
Aug  2 11:16:18 sshsrv sshd[1536]: debug1: drop connection #3
        :
# debug1: drop connection #3 の#3 は、認証されていない段階の接続要求が 3 件あり、
# 接続を拒否したことを意味します。
```

【コラム III:SSHGuard】

SSHGuard はログを監視し、ID やパスワードを変えて何度も試みる不審な接続を検出すると、iptables などのフィルタに発信元 IP アドレスを自動的に追加することで、一定時間、その接続を遮断する機構です。次のパラメータで、遮断を制御できます。

-a num	num 回の不審な接続を検知すると、その発信元 IP アドレスからの接続を遮断する（デフォルト 4 回）
-p secs	遮断した発信元 IP アドレスのエントリを secs 秒経過後に解放する（デフォルト 420 秒）
-s secs	発信元 IP アドレスのエントリを secs 秒後に解放する。secs 秒以上の間隔での接続は遮断されない（デフォルト 1200 秒）
-w addr/host/block/file	ホワイトリスト（IP、ホスト名、ネットマスク）を指定する。

SSHGuard - Defend from brute force attacks
http://www.sshguard.net/

4.1.4. アウトバウンド（発呼）通信に対するアクセスを制限する【オプション】

SSH サーバからアクセスできる送信先 IP アドレスを制限することで、不正アクセス発生時の二次被害を低減できます。SSH サーバをファイアウォールの内側に設置する場合にも、同様です。ここでは、iptables を用いて制限する方法を紹介します。

■設定方法

iptables の設定ファイル（/etc/sysconfig/iptables）で設定した後、設定を反映します（/sbin/service iptables restart）。

File: /etc/sysconfig/iptables

```
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A OUTPUT -p tcp -d 192.168.1.200 --dport 22 -j ACCEPT
-A OUTPUT -j DROP

# SSH サーバからアクセスできる送信先 IP アドレスを
# データサーバ（192.168.1.200）に制限する。
```

■確認方法

コマンド（/sbin/iptables）を使って、設定の状態を確認します。

Host: sshsrv

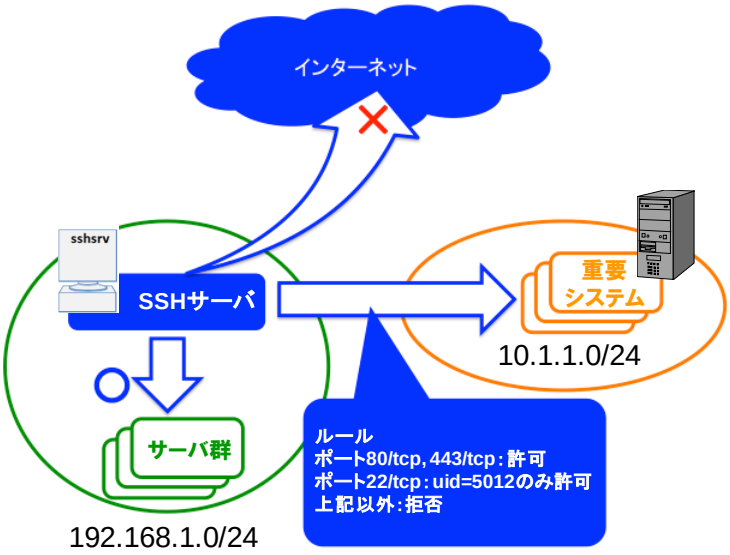
```
[root@sshsrv ~]# /sbin/iptables -L -n ⏎
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
REJECT     all  --  0.0.0.0/0            0.0.0.0/0           reject-with icmp-host-prohibited
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             192.168.1.200
DROP       all  --  anywhere             anywhere
```

【コラム IV:iptables を用いた送信制限】

Linux 系 OS では、iptables を使用してインバウンドとアウトバウンド向けの通信の制限することができます。例えば、次のような要件を iptables で実現することを考えます。

- SSH サーバからインターネットへの通信は制限したい。
- 特定のユーザ（jason, uid=5012）のみに重要システムが設置されたネットワーク（10.1.1.0/24）への ssh 通信を許可したい。
- 全てのユーザに対して、重要システムが設置されたネットワークへの一般的な Web アクセス（http/https）は許可したい。
- 全てのユーザに対して、SSH サーバと同一のサブネットワーク内（192.168.1.0/24）であれば、通信に制限をかけたくない。

この条件を満たす具体的な設定例を見て行きましょう。想定するネットワーク構成は、次の通りです。



iptables に複数のルールが存在する場合、上から順にルールが検査しマッチしないと、次のルールに進みます。このように順番に検査して、最初にマッチしたルールが適用されることになります。このため、条件を満たすためのルールの作り方は、次のようになります。

Host: sshsrv

①コネクション確立しているものは許可
[root@sshsrv ~]# /sbin/iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT ⏎

②特定のユーザのみに許可したい通信に関するルール（jason : uid=5012 は 10.1.1.0/24 に対して ssh でアクセスできる）
[root@sshsrv ~]# /sbin/iptables -A OUTPUT -d 10.1.1.0/24 -m owner --uid-owner 5012 -p tcp --dport 22 -j ACCEPT ⏎

③全てのユーザに許可したい通信に関するルール（全てのユーザは 10.1.1.0/24 に対して http/https でアクセスでき、192.168.1.0/24 内であれば通信に制限はない）
[root@sshsrv ~]# /sbin/iptables -A OUTPUT -d 10.1.1.0/24 -p tcp --dport 80 -j ACCEPT ⏎
[root@sshsrv ~]# /sbin/iptables -A OUTPUT -d 10.1.1.0/24 -p tcp --dport 443 -j ACCEPT ⏎
[root@sshsrv ~]# /sbin/iptables -A OUTPUT -d 192.168.1.0/24 -j ACCEPT ⏎

④①～③を満たさない通信はすべて拒否
[root@sshsrv ~]# /sbin/iptables -A OUTPUT -j DROP ⏎

⑤OS 再起動後も、これらの設定が適用されるようにするために、/etc/sysconfig/iptables に設定を格納する。
[root@sshsrv ~]# /bin/service iptables save ⏎
iptables: Saving firewall rules to /etc/sysconfig/iptable:[OK]

iptables の定義は順序が関係するため、定義を追加したい場合には、(1) 挿入したいルール番号を -I オプションで指定、(2) 一旦 -D オプションや -F オプションでルールを削除して改めて頭から定義し直すなどで対応します。直接/etc/sysconfig/iptables を編集して iptables に読み込ませる方法も可能です。

4.2　SSH サービス

4.2.1.　SSH プロトコルのバージョン 2 のみを許可する【推奨】

SSH のプロトコルには、バージョン 1 と 2 があります。バージョン 1 には既知の脆弱性[a]があるため、各種ガイドラインで無効化することが推奨されています。

■設定方法

sshd の設定ファイル（/etc/ssh/sshd_config）で設定した後、設定を反映します（/sbin/service sshd restart）。

File: /etc/ssh/sshd_config

```
Protocol 2                    # バージョン 2 を指定する
```

■確認方法

ssh コマンドを使って、バージョン 1（オプション：-1）でログインできないことを確認し、バージョン 2（オプション：-2）でログインできることを確認します。

Host: sshcc

```
■バージョン 1 でログインする
[hanako@sshcc ~]$ /usr/bin/ssh -1 hanakosrv@sshsrv ⏎
Protocol major versions differ: 1 vs. 2

■バージョン 2 でログインする
[hanako@sshcc ~]$ /usr/bin/ssh -2 hanakosrv@sshsrv ⏎
hanakosrv@sshsrv's password:＜パスワード＞ ⏎
Last login: Sat Aug  2 23:45:23 2014 from sshcc
[hanakosrv@sshsrv ~]$
```

また、バージョン 1（オプション：-1）で接続した場合には、SSH サーバ（sshsrv）上のログファイル（/var/log/secure）に、次のようなログが記録されます。

Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/tail -f /var/log/secure ⏎
Aug  2 11:51:31 sshsrv sshd[1596]: Did not receive identification string from 192.168.1.1
Aug  2 11:51:52 sshsrv sshd[1598]: Connection closed by 192.168.1.1
```

a) Vulnerability Note VU#13877
Weak CRC allows packet injection into SSH sessions encrypted with block ciphers
http://www.kb.cert.org/vuls/id/13877

4.2.2. 公開鍵認証を使用する（パスワード認証を無効化する）【推奨】

SSH ログインでのパスワード認証を無効化して、公開鍵認証（ユーザ秘密鍵ファイル＋パスフレーズ）を使用することで、パスワード認証に対するブルートフォース攻撃を防ぐことができます。

■設定方法

sshd の設定ファイル（/etc/ssh/sshd_config）で、パスワード認証を無効にし、公開鍵認証を有効にした後、設定を反映します（/sbin/service sshd restart）。

File: /etc/ssh/sshd_config

```
HostbasedAuthentication no              # ホストベースの認証を使用しない
RhostsAuthentication no                 # ホストベースの認証を使用しない [*]
                                        # [*] OpenSSH 5.3p1 では、非推奨オプションになっています。
RhostsRSAAuthentication no              # ホストベースの RSA 認証を使用しない
PasswordAuthentication no               # パスワード認証を許可しない
ChallengeResponseAuthentication no      # パスワード認証を許可しない
PermitEmptyPasswords no                 # 空パスワードでのアクセスを許可しない
RSAAuthentication no                    # RSA 認証を使用しない（バージョン 1 のみのため）
PubkeyAuthentication yes                # 公開鍵認証を使用する
```

なお、各ユーザ毎に、SSH サーバ上のユーザのホームディレクトリにある.ssh/authorized_keys ファイルにユーザ公開鍵を追加登録する必要があります。

Host: sshcc

■ユーザ hanakosrv が sshsrv にログイン後、authorized_keys にユーザ公開鍵を追加する

```
[hanakosrv@sshsrv ~]$ cd /home/hanakosrv/.ssh ⏎
[hanakosrv@sshsrv .ssh]$ cat id_rsa.pub >> authorized_keys ⏎
[hanakosrv@sshsrv .ssh]$ chmod 600 authorized_keys ⏎
[hanakosrv@sshsrv .ssh]$ ls -l authorized_keys ⏎
-rw-------. 1 hanakosrv users 609 Aug  2 12:45 authorized_keys
```

■確認方法

ログイン（[hanako@sshcc ~]$ /usr/bin/ssh hanakosrv@sshsrv -i ~/.ssh/id_rsa）を試みた後、SSH サーバ（sshsrv）上のログファイル（/var/log/secure）に、『Accepted publickey・・・ssh2』が記録されていることを確認します。

Host: sshcc

■公開鍵認証でログインする

```
[hanako@sshcc ~]$ /usr/bin/ssh hanakosrv@sshsrv -i ~/.ssh/id_rsa ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa':＜パスフレーズ＞ ⏎
Last login: Sat Aug  2 21:15:03 2014 from sshcc
[hanakosrv@sshsrv ~]$
```

Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/tail -f /var/log/secure ⏎
Aug  2 12:50:43 sshsrv sshd[1125]: Accepted publickey for hanakosrv from 192.168.1.1 port 41576 ssh2
```

4.2.3. ポートフォワードを止める【推奨】

OpenSSH には、リモートからのコンソールアクセス以外に、任意の TCP 通信をカプセル化し、さらに暗号化して転送するポートフォワードという機能が提供されています。ポートフォワード機能を悪用された場合、不正アクセスが発生した SSH サーバを基点とした踏み台攻撃を誘発する可能性がありますので、禁止することが推奨されます。

■設定方法

sshd の設定ファイル（/etc/ssh/sshd_config）で、ポートフォワードを無効にした後、設定を反映します（/sbin/service sshd restart）。

File: /etc/ssh/sshd_config

```
AllowTcpForwarding no          # TCP ポートフォワードを禁止する
X11Forwarding no               # X11 ポートフォワードを禁止する
```

■確認方法

TCP ポートフォワードには、ローカルポートフォワードとリモートポートフォワードがあります。ここでは、これらポートフォワードが無効化されていることを確認する方法を説明します。

（a）ローカルポートフォワードの確認

ローカルポートフォワードを指定してログイン（[hanako@sshcc ~]$ /usr/bin/ssh hanakosrv@sshsrv -L 2525:localhost:25）した後、SSH クライアント（sshcc）上のポート番号 2525/TCP にアクセスします。SSH サーバ（sshsrv）のログファイル（/var/log/secure）に、“Received request to connect to host ・・・ but the request was denied.”が記録されていることを確認します。

Host: sshcc

■ローカルポートフォワードを指定してログインする

```
[hanako@sshcc ~]$ /usr/bin/ssh hanakosrv@sshsrv -L 2525:localhost:25 ⏎
hanakosrv@sshsrv's password:＜パスワード＞ ⏎
Last login: Sat Aug  2 23:54:03 2014 from sshcc
[hanakosrv@sshsrv ~]$ channel 3: open failed: administratively prohibited: open failed    ←[*]

# [*] SSH クライアント（sshcc）上のポート番号 2525/TCP にアクセスした際に、
# 接続失敗のエラーが表示されます。
```

Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/tail -f /var/log/secure ⏎
Aug  2 20:34:32 sshsrv sshd[1362]: Received request to connect to host localhost port 25, but the request was denied.
```

（b）リモートポートフォワードの確認
リモートポートフォワードを指定してログイン（[hanako@sshcc ~]$ /usr/bin/ssh hanakosrv@sshsrv -R 2525:localhost:25）します。SSH サーバ（sshsrv）のログイン時に、警告 “Warning: remote port forwarding failed for listen port・・・” が表示されることを確認します。

Host: sshcc

■リモートポートフォワードを指定してログインする

```
[hanako@sshcc ~]$ /usr/bin/ssh hanakosrv@sshsrv -R 2525:localhost:25 ⏎
hanakosrv@sshsrv's password: ＜パスワード＞ ⏎
Warning: remote port forwarding failed for listen port 2525
Last login: Sat Aug  2 21:26:24 2014 from sshcc
[hanakosrv@sshsrv ~]$
```

【コラム V:history を削除する】

コマンドの実行履歴を表示する history コマンドをご存知ですか？
history コマンドは、矢印キー「↑」キーを押すと入力したコマンドを呼び出すことができたり、「!」と履歴番号を併用すること（例：!5　履歴番号 5 番目の ls -a を実行）で入力の手間を省くことができます。便利な反面、知らない間に、接続先、アカウント、パスワードなどが.bash_history ファイルに記録されていることもあり、侵害された場合、これらの情報が悪用されてしまう可能性もあります。

Host: sshsrv

```
[hanakosrv@sshsrv ~]$ history ⏎
     5  18:05   ls -a
     6  18:05   vi work/config.txt
     7  18:05   ls -a
     8  18:05   ssh taro@192.168.1.200
     9  18:06   scp taro@192.168.1.20:/work/keys/sec.users work/sec.users
    10  18:07   htpasswd -n -m -b test 20130612
    11  18:10   cd /etc
```

history コマンドは、HISTSIZE（メモリ上に記録する履歴）、HISTFILESIZE（.bash_history ファイル上に記録する履歴）で設定を変更できます。HISTFILESIZE=0 とすれば、そのセッションで使ったコマンド履歴を参照できるが、新しいセッションでは以前の履歴を参照できないことになります。

Host: sshsrv

①ログイン時の設定ファイルに HISTFILESIZE=0 を追記

```
[hanakosrv@sshsrv ~]$ echo "HISTFILESIZE=0" >> /home/hanakosrv/.bash_profile ⏎
```

②設定を反映（ドットの後に、/home/hanakosrv/.bash_profile を入力）

```
[hanakosrv@sshsrv ~]$ . /home/hanakosrv/.bash_profile ⏎
```

③設定の確認

```
[hanakosrv@sshsrv ~]$ set | grep HIST ⏎
HISTFILE=/home/hanakosrv/.bash_history
HISTFILESIZE=0
HISTSIZE=500
```

4.2.4. SSH サービスを 22/TCP 以外で稼働させる【推奨】

SSH サービスのログインに対する攻撃では、インターネットに接続した SSH サーバを無差別に狙ってきます。SSH サービスをデフォルトの 22/TCP で稼動させている場合、インターネットに接続後には、SSH サービスが稼働していることを見つけられ、攻撃対象になるということを十分留意してください。システム要件や運用上の問題がなければ、攻撃対象になりにくくするために、ポート番号の変更を検討してください。

なお、ポート番号を変更する際には、iptables(4.1.2 節：特定のホストからの接続だけを許可する【推奨】、4.1.3 節：一定時間内の SSH サービスへのアクセス数を制限する【オプション】)の設定変更もあわせて検討する必要があります。

■設定方法

sshd の設定ファイル（/etc/ssh/sshd_config）で設定した後、設定を反映します（/sbin/service sshd restart）。なお、ファイアウォールや iptables で 22/TCP を許可する設定を行っている場合には、22/TCP の通信を拒否し、22222/TCP を許可する設定に変更してください。iptables については、4.1.2 節を参照してください。

File: /etc/ssh/sshd_config

```
Port 22222                # ポート番号 22222/TCP で sshd を稼働させる
```

■確認方法

コマンド（/bin/netstat）を使って、設定したポートで稼動していることを確認します。

Host: sshsrv

```
[root@sshsrv ~]# /bin/netstat -pantu ⏎
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address       State       PID/Program name
tcp        0      0 0.0.0.0:22222       0.0.0.0:*             LISTEN      904/sshd
tcp        0      0 127.0.0.1:25        0.0.0.0:*             LISTEN      980/master
tcp        0      0 :::22222            :::*                  LISTEN      1042/sshd
tcp        0      0 ::1:25              :::*                  LISTEN      980/master
udp        0      0 0.0.0.0:68          0.0.0.0:*                         793/dhclient
```

【コラム VI:ポート番号を指定してログインする】

SSH サービスの待ち受けポートを 22/tcp 以外にしている場合には -p オプションでポート番号を指定してログインします。ポート番号 22222/tcp の場合には、次のようになります。

```
[hanako@sshcc ~]$ /usr/bin/ssh -p 22222 hanakosrv@sshsrv ⏎
```

オプションの詳細については、第 6 章：付録 B　SSH 関連コマンドを参照してください。

4.2.5. known_hosts ファイルをハッシュ化する【推奨】

SSH サービスに関わる事案の中には、別の SSH サーバへの不正アクセスに、SSH サーバ上の known_hosts ファイルに格納されているホスト名や IP アドレスが悪用されてしまったという事案も報告されています。特に、SSH サーバをゲートウェイ（ログインホスト）として構成している場合には、ユーザのホームディレクトリにある.ssh/known_hosts ファイルのハッシュ化を実施してください。

known_hosts ファイルには、「ホスト名 キー種別 ホスト公開鍵」が格納されています。ssh、scp などの SSH コマンドは、SSH サービスに接続する度に、SSH サーバから受信したホスト公開鍵と known_hosts に記録されているホスト公開鍵とを照合することで、既知のサーバか否かを確認しています。

■設定方法

ssh の設定ファイル（/etc/ssh/ssh_config）で設定します。

File: /etc/ssh/ssh_config

```
HashKnownHosts yes        # ホスト名や IP アドレスをハッシュ化する
```

■確認方法

ユーザのホームディレクトリにある.ssh/known_hosts ファイルの先頭列が、ホスト名や IP アドレスとして可読な形式ではないことを確認します。ハッシュ化済のエントリの先頭には、"|1|"が記載されています。known_hosts ファイルがハッシュ化されていない場合には、コマンド（/usr/bin/ssh-keygen）を使って、ハッシュ化してください。その際、known_hosts.old ファイルの削除を忘れずに実施してください。

Host: sshsrv

■既存の known_hosts ファイルをハッシュ化する

```
[hanakosrv@sshsrv .ssh]$ ssh-keygen -H ⏎
/home/hanakosrv/.ssh/known_hosts updated.
Original contents retained as /home/hanakosrv/.ssh/known_hosts.old
WARNING: /home/hanakosrv/.ssh/known_hosts.old contains unhashed entries
Delete this file to ensure privacy of hostnames
```

■known_hosts.old ファイル（ハッシュ化未実施）

```
[hanakosrv@sshsrv .ssh]$ cat known_hosts.old ⏎
192.168.1.200 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA7gBfuo5STGGNxQbNHBEIQQvSnwShmmSRGdWDjP9mGgLbZVYuO
J6ny6cVGpVvE2aNuVRnc+knT65eHF1aQZOt+tIGvlV5c+8ObwNo7MB9yHl2m5AWu5WQ8UcI3ZmXNDapOw5UXKy/viBp9BqPLFN
lXLFpMNHkeWubCotc/dZzuVmsZgZ1BG9fJfNdvYjNsbS6WAdpp9qXzDrl+6nWlLKuUazPvA5r7KZqvOZSm5rUFGqepFiblOaf4
Psjmc8y1qVO4wqUm5sNMTBm2XofxTepnVAx5xrehWVPCd17lJcEbrBWCs/vYmCDun9vHrtEJDGpT8mn4PiYZKKYrsNSOBE/5w==
```

■known_hosts ファイル（ハッシュ化済）

```
[hanakosrv@sshsrv .ssh]$ cat known_hosts ⏎
|1| dJaVQ/LE/aKuOy13+Z1NrPNj8Qc=|D6zc9n3ZiXUcWyVN88rLLWw5hgA= ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA7g
Bfuo5STGGNxQbNHBEIQQvSnwShmmSRGdWDjP9mGgLbZVYuOJ6ny6cVGpVvE2aNuVRnc+knT65eHF1aQZOt+tIGvlV5c+8ObwNo7
MB9yHl2m5AWu5WQ8UcI3ZmXNDapOw5UXKy/viBp9BqPLFNlXLFpMNHkeWubCotc/dZzuVmsZgZ1BG9fJfNdvYjNsbS6WAdpp9qX
zDrl+6nWlLKuUazPvA5r7KZqvOZSm5rUFGqepFiblOaf4Psjmc8y1qVO4wqUm5sNMTBm2XofxTepnVAx5xrehWVPCd17lJcEbrB
WCs/vYmCDun9vHrtEJDGpT8mn4PiYZKKYrsNSOBE/5w==

[hanakosrv@sshsrv .ssh]$ rm known_hosts.old ⏎
[hanakosrv@sshsrv .ssh]$
```

4.2.6. ログインを特定のユーザだけに許可する【オプション】

OpenSSH のデフォルト設定では、OS にログインできるすべてユーザが SSH サービスを利用してリモートからログインできる状態になっています。管理者権限を持っている root ユーザが一番狙われるユーザです。SSH サービスに限らず、リモートから root ユーザでは直接ログインできない設定とし、root 権限での操作は、一般ユーザでログインした後に、su や sudo コマンドを利用してください。また、商用製品でよく利用されているアカウント（例：oracle）も、踏み台として悪用されるので、一般ユーザについても SSH サービスを利用してリモートからログインできるユーザを絞り込んでください。

■設定方法

sshd の設定ファイル(/etc/ssh/sshd_config)で設定した後、設定を反映します(/sbin/service sshd restart)。

File: /etc/ssh/sshd_config

```
PermitRootLogin no        # root でのログインを許可しない
AllowUsers hanakosrv      # hanakosrv のみのログインを許可する
```

■確認方法

root ユーザでログイン（[hanako@sshcc ~]$ /usr/bin/ssh root@sshsrv）を試みた後、ログファイル（/var/log/secure）に、root ユーザからのアクセス失敗が記録されていることを確認します。

Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/tail -f /var/log/secure ⏎
Aug  3 00:13:15 sshsrv sshd[1387]: User root from sshcc not allowed because not listed in AllowUsers
Aug  3 00:13:15 sshsrv sshd[1388]: input_userauth_request: invalid user root
Aug  3 00:13:19 sshsrv unix_chkpwd[1389]: password check failed for user (root)
Aug  3 00:13:19 sshsrv sshd[1387]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0
tty=ssh ruser= rhost=sshcc  user=root
Aug  3 00:13:21 sshsrv sshd[1387]: Failed password for invalid user root from 192.168.1.1 port 54412 ssh2
```

一般ユーザでログイン（[hanako@sshcc ~]$ /usr/bin/ssh hanakosrv@sshsrv）（[taro@sshcc ~]$ /usr/bin/ssh taro@sshsrv）を試みた後、ログファイル（/var/log/secure）で、許可したユーザのみがログインできていることを確認します。

Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/tail -f /var/log/secure ⏎

# hanakosrv はログイン可
Aug  3 00:16:52 sshsrv sshd[1390]: Accepted password for hanakosrv from 192.168.1.1 port 52550 ssh2
Aug  3 00:16:53 sshsrv sshd[1390]: pam_unix(sshd:session): session opened for user hanakosrv by (uid=0)

# taro はログイン不可
Aug  3 00:20:07 sshsrv sshd[1431]: User taro from sshcc not allowed because not listed in AllowUsers
Aug  3 00:20:07 sshsrv sshd[1432]: input_userauth_request: invalid user taro
```

4.2.7. IPv6 を使わない場合には、無効化する【オプション】

OpenSSH のデフォルト設定では、IPv6 及び IPv4 の両方で SSH サービスが稼動します。IPv6 で SSH サービス提供する場合には、IPv6 に対応したファイアウォールや IDS の導入や設定が必要です。IPv6 で SSH サービスを提供する必要がない場合には、sshd の IPv6 を明示的に無効化してください。

■設定方法

sshd の設定ファイル(/etc/ssh/sshd_config)で設定した後、設定を反映します(/sbin/service sshd restart)。

File: /etc/ssh/sshd_config

```
AddressFamily inet        # any から inet に変更
#ListenAddress ::         # 明示的にコメントアウト
```

■確認方法

コマンド（/bin/netstat）を使って、IPv6 が無効化されていることを確認します。

Host: sshsrv

```
[root@sshsrv ~]# /bin/netstat -pantu ⏎
Active Internet connections (servers and established)

# AddressFamily any の場合
Proto Recv-Q Send-Q Local Address       Foreign Address       State       PID/Program name
tcp        0      0 0.0.0.0:22          0.0.0.0:*             LISTEN      1081/sshd
tcp        0      0 127.0.0.1:25        0.0.0.0:*             LISTEN      981/master
tcp        0      0 :::22               :::*                  LISTEN      1081/sshd    #IPv6
tcp        0      0 ::1:25              :::*                  LISTEN      981/master   #IPv6

# AddressFamily inet の場合
Proto Recv-Q Send-Q Local Address       Foreign Address       State       PID/Program name
tcp        0      0 0.0.0.0:22          0.0.0.0:*             LISTEN      1176/sshd
tcp        0      0 127.0.0.1:25        0.0.0.0:*             LISTEN      981/master
tcp        0      0 ::1:25              :::*                  LISTEN      981/master   #IPv6
```

4.2.8. 弱い暗号方式やメッセージ認証コードを利用しない【オプション】

セキュリティ機構の危殆化への対応として、弱い暗号方式やメッセージ認証コードを使用しないことが推奨されています[b]。SSH のプロトコルでは、候補の中から暗号方式、メッセージ認証コードで使用できるものを選択します。このため、候補の中から弱い暗号方式やメッセージ認証コードを予め削除しておいてください。

■設定方法

sshd の設定ファイル(/etc/ssh/sshd_config)で設定した後、設定を反映します(/sbin/service sshd restart)。なお、設定は、次の手順で進めます。

(1)「**■確認方法**」を用いて、使用できる暗号方式、メッセージ認証コードを調査します。
(2) 使用できる暗号方式（Ciphers）から、CBC（Cipher Block Chaining）モードを省いて sshd の設定ファイル（/etc/ssh/sshd_config）に設定します。
＜sshd の設定ファイル（/etc/ssh/sshd_config）から省く暗号方式（Ciphers）＞
3des-cbc
aes128-cbc
aes192-cbc
aes256-cbc
blowfish-cbc
cast128-cbc
(3) 使用できるメッセージ認証コード（MACs）から MD5 と 96bit を省いて sshd の設定ファイル（/etc/ssh/sshd_config）に設定します。
＜sshd の設定ファイル（/etc/ssh/sshd_config）から省くメッセージ認証コード（MACs）＞
hmac-md5
hmac-md5-96
hmac-md5-96-etm@openssh.com
hmac-md5-etm@openssh.com
hmac-sha1-96
hmac-sha1-96-etm@openssh.com

File: /etc/ssh/sshd_config

```
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,arcfour
MACs
hmac-sha1,umac-64@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-ripemd160,hmac-ripemd160@openssh.com
```

■確認方法

デバックモードでログイン（[root@sshsrv ~]# /usr/bin/ssh -vv localhost）を試みた後、表示されているデバックログを確認します。

Host: sshsrv

```
[root@sshsrv ~]# /usr/bin/ssh -vv localhost ⏎
OpenSSH_5.3p1, OpenSSL 1.0.1e-fips 11 Feb 2013
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug2: ssh_connect: needpriv 0
debug1: Connecting to localhost [::1] port 22.
debug1: connect to address ::1 port 22: Connection refused
```

b) CPNI-957037
SSH 通信において一部データが漏えいする可能性
http://jvn.jp/vu/CPNI-957037/

```
debug1: Connecting to localhost [127.0.0.1] port 22.
debug1: Connection established.
debug1: permanently_set_uid: 0/0
debug1: identity file /root/.ssh/identity type -1
debug1: identity file /root/.ssh/identity-cert type -1
debug1: identity file /root/.ssh/id_rsa type -1
debug1: identity file /root/.ssh/id_rsa-cert type -1
debug1: identity file /root/.ssh/id_dsa type -1
debug1: identity file /root/.ssh/id_dsa-cert type -1
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.3
debug1: match: OpenSSH_5.3 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.3
debug2: fd 3 setting O_NONBLOCK
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug2: kex_parse_kexinit: diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,
diffie-hellman-group14-sha1,diffie-hellman-group1-sha1
debug2: kex_parse_kexinit: ssh-rsa-cert-v01@openssh.com,ssh-dss-cert-v01@openssh.com,
ssh-rsa-cert-v00@openssh.com,ssh-dss-cert-v00@openssh.com,ssh-rsa,ssh-dss
                                        #SSH クライアントが利用できる暗号方式（Ciphers）
debug2: kex_parse_kexinit: aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,
3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se
debug2: kex_parse_kexinit: aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,
3des-cbc,blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se
                                        #SSH クライアントが利用できるメッセージ認証コード（MACs）
debug2: kex_parse_kexinit: hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-sha2-256,hmac-sha2-512,
hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96
debug2: kex_parse_kexinit: hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-sha2-256,hmac-sha2-512,
hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96
debug2: kex_parse_kexinit: none,zlib@openssh.com,zlib
debug2: kex_parse_kexinit: none,zlib@openssh.com,zlib
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit: first_kex_follows 0
debug2: kex_parse_kexinit: reserved 0
debug2: kex_parse_kexinit: diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,
diffie-hellman-group14-sha1,diffie-hellman-group1-sha1
debug2: kex_parse_kexinit: ssh-rsa,ssh-dss
                                        #SSH サーバが利用できる暗号方式（Ciphers）
debug2: kex_parse_kexinit: aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,arcfour
debug2: kex_parse_kexinit: aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,arcfour
                                        #SSH サーバが利用できるメッセージ認証コード（MACs）
debug2: kex_parse_kexinit: hmac-sha1,umac-64@openssh.com,hmac-sha2-256,hmac-sha2-512,
hmac-ripemd160,hmac-ripemd160@openssh.com
debug2: kex_parse_kexinit: hmac-sha1,umac-64@openssh.com,hmac-sha2-256,hmac-sha2-512,
hmac-ripemd160,hmac-ripemd160@openssh.com
debug2: kex_parse_kexinit: none,zlib@openssh.com
debug2: kex_parse_kexinit: none,zlib@openssh.com
debug2: kex_parse_kexinit:
debug2: kex_parse_kexinit:
```

```
debug2: kex_parse_kexinit: first_kex_follows 0
debug2: kex_parse_kexinit: reserved 0
debug2: mac_setup: found hmac-sha1
debug1: kex: server->client aes128-ctr hmac-sha1 none    #使用する暗号方式とメッセージ認証コード
debug2: mac_setup: found hmac-sha1
debug1: kex: client->server aes128-ctr hmac-sha1 none    #使用する暗号方式とメッセージ認証コード
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<2048<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug2: dh_gen_key: priv key bits set: 158/320
debug2: bits set: 998/2048
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug2: no key of type 0 for host localhost
debug2: no key of type 2 for host localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 7b:90:8c:a5:a6:97:5c:ae:17:ec:16:f4:da:52:2f:a3.
Are you sure you want to continue connecting (yes/no)? no
Host key verification failed.
[root@sshsrv ~]#
```

4.3 鍵管理

公開鍵認証（ユーザ秘密鍵ファイル＋パスフレーズ）を用いた SSH サービスの提供にあたっては、利用者に次の事項を徹底させてください。

4.3.1. ユーザ秘密鍵ファイルにパスフレーズをつけること【推奨】

公開鍵認証に必要な鍵を生成する際に、パスフレーズに入力が求められます。必ず、パスフレーズを設定してください。また、「ユーザ秘密鍵ファイルにパスフレーズをつけること」に加えて、下記にも留意してください。

- ユーザ秘密鍵ファイルとパスフレーズは、どちらも他人に渡してはいけない。
- パスフレーズはなるべく長く複雑にし、他人に推測されないようにしなければならない。
- ユーザ秘密鍵ファイルを電子メール等でそのまま送付してはいけない。

Host: sshcc

■公開鍵認証に必要な鍵を生成する

```
[hanako@sshcc .ssh]$ /usr/bin/ssh-keygen -C "hanakosrv with passphrase" ⏎
enerating public/private rsa key pair.
Enter file in which to save the key (/home/hanakogw/.ssh/id_rsa): id_rsa_srv ⏎
Enter passphrase (empty for no passphrase): ＜パスフレーズ＞ ⏎  ←必ずパスフレーズを設定してください。
Enter same passphrase again:  ⏎
Your identification has been saved in id_rsa_srv.
Your public key has been saved in id_rsa_srv.pub.
The key fingerprint is:
cf:82:94:60:93:67:7c:75:18:61:e3:be:78:06:9a:6f hanakosrv with passphrase
The key's randomart image is:
+--[ RSA 2048]----+
|          *+.    |
|     o   +.o     |
|    = + . .      |
|   . = o .       |
|      o S .      |
|     . + = .     |
|      + o *      |
|       .E+       |
|       ..        |
+-----------------+

[hanako@sshcc .ssh]$ ls -l id_rsa* ⏎
-rw-------. 1 hanako users 1743 Aug  2 14:14 id_rsa_srv
-rw-r--r--. 1 hanako users  404 Aug  2 14:14 id_rsa_srv.pub
```

■ユーザ秘密鍵ファイルのパスフレーズを変更する

```
[hanako@sshcc .ssh]$ /usr/bin/ssh-keygen -p -f id_rsa_srv ⏎
Enter old passphrase:  ⏎
Key has comment 'id_rsa_srv'
Enter new passphrase (empty for no passphrase):＜パスフレーズ＞ ⏎  ←必ずパスフレーズを設定してください。
Enter same passphrase again:  ⏎
Your identification has been saved with the new passphrase.
```

4.3.2. SSH サーバ上にユーザ秘密鍵ファイルを置かない【推奨】

SSH サービスに関わる事案の中には、SSH サーバ上にユーザ秘密鍵ファイルを格納しておいたために、別の SSH サーバへの不正アクセスに、ユーザ秘密鍵ファイル（特に、パスフレーズが設定されていないユーザ秘密鍵ファイル）が悪用されてしまったという事案も報告されています。

多段や複数の SSH サーバ接続が必要な場合には、ProxyCommand、ssh-agent の活用を検討してください（5 章：付録を参照のこと）。

【コラム VII:ユーザ公開鍵ファイル、ユーザ秘密鍵ファイル】

SSH サービスの公開鍵認証では、ユーザ公開鍵ファイル（id_rsa.pub）、ユーザ秘密鍵ファイル（id_rsa）を使用します。ユーザ秘密鍵ファイルには、PRIVATE KEY という文字列が記載されていますので、パスフレーズで保護し、決して、他人に渡してはいけません。

ユーザ公開鍵ファイル

- デフォルトのファイル名は id_rsa.pub
- SSH サーバ上のユーザのホームディレクトリにある.ssh/authorized_keys ファイルに登録するデータ
- 電子メール等で送付しても問題なし

[hanako@sshcc .ssh]$ cat id_rsa.pub ⏎

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAsokzzBWUuRVASS7Vx9Ydt61DikihxBszuIAe
egQGoOin95zpWeImQU5pAGE3EARrTxLarEitUQEcVqER9P7vU/PBoYmd6yllGcsq
oTORvMHCrySx8BKSlF7R3kzUDrlHr5UivW6XcOJXjWRowJlcUnoca1aDhtEWwvxp
sQYr2H9dSvvrRF6XU6/q6tOx6xDrNOK+FFOkJLQiWWoi3YdkwLEzO/8tD2JQUOrQ
efCLTBMJlXTsM7rU2xQzSybIps8arVSRDZgWht2lMCW3SNLUwzo1fXQOgBTLr5TS
3jkaXa4fiMaOmeAF+ozXNwWOZXfu66uP8pEeUsU+hyHbUqFNdQ== hanako with
passphrase
```

ユーザ秘密鍵ファイル

- デフォルトのファイル名は id_rsa
- SSH クライアントに保持し、パスフレーズで保護する。決して、他人に渡してはいけない。
- 電子メール等でそのまま送付してはいけない。

[hanako@sshcc .ssh]$ cat id_rsa ⏎

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,8625FC7ADDA5827A

zqx/KAXMF+HU+txz2S7jK7hByEzPeA1ybWdq5QYb4UDOBPdAJ7ot7ynWxC6G1UOf
[illegible]
6g27I/LMJ/USaZ6isXEvDwZMvhA8sPKd6wKJUW2oA+IglAS+49VIpw==
-----END RSA PRIVATE KEY-----
```

また、パスフレーズの設定されていないユーザ秘密鍵ファイルには特徴がありますので、簡単に見分けられてしまいます。

パスフレーズが設定されていない
ユーザ秘密鍵ファイル

- 暗号化に必要な情報が記載されていない。
- 先頭が MII あるいは MIIBu で始まっている。

[hanako@sshxx .ssh]$ cat id_rsa ⏎

```
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEAyXqu5sE99SiSOXLsjuYBWjo+wFmPhdgyOArt+KV4NcVzqRjO
[illegible]
POyCJ2cgfsx/M5djLlCh7FDeuLhBm4TraBBtjj8c4HjRSFmpag==
-----END RSA PRIVATE KEY-----
```

パスフレーズが設定されている
ユーザ秘密鍵ファイル

- 暗号化に必要な情報が記載されている。

[hanako@sshcc .ssh]$ cat id_rsa ⏎

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,8625FC7ADDA5827A

zqx/KAXMF+HU+txz2S7jK7hByEzPeA1ybWdq5QYb4UDOBPdAJ7ot7ynWxC6G1UOf
[illegible]
6g27I/LMJ/USaZ6isXEvDwZMvhA8sPKd6wKJUW2oA+IglAS+49VIpw==
-----END RSA PRIVATE KEY-----
```

5　付録 A　利用形態別ガイド

ユーザ秘密鍵ファイルを安全に取り扱うために、ここでは、代表的な 3 つの利用形態について紹介します。いずれの場合も、ユーザ秘密鍵ファイルは、クライアント PC に保持します。

- 利用方法 A
  SSH クライアント PC（sshxx）から、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にログインする。
- 利用方法 B
  SSH クライアント PC（sshxx）から、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にファイルを転送する。
- 利用方法 C
  SSH クライアント PC（sshxx）から、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にログインした後、接続先ホスト（sshsrv）からデータサーバ（sshdb）にファイルを転送する。

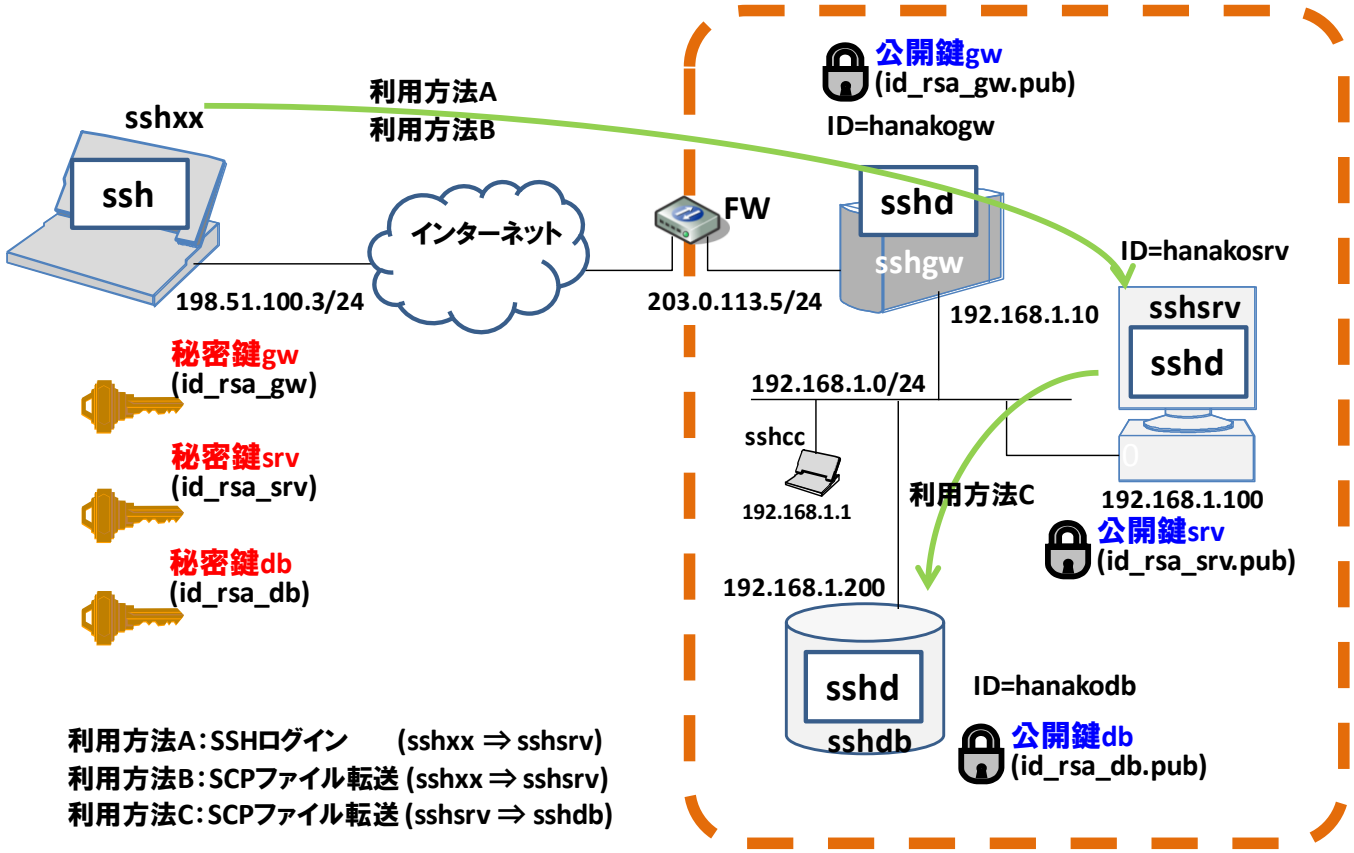


図 2：利用形態別

5.1 クライアントとサーバの設定例

第 4 章の設定を元に、利用形態別（図 2）のネットワーク構成における SSH クライアント（sshcc、sshxx）と SSH サーバ（sshgw、sshsrv、sshdb）の設定例を示します。

5.1.1. SSH クライアントの設定

（1）ssh の設定ファイル（/etc/ssh/ssh_config）

表 3：クライアント PC（sshcc、sshxx）

File: /etc/ssh/ssh_config

```
# $NCA: ssh_config,v1.0 2014/12/27 sshcc, sshxx $

HashKnownHosts yes                      #4.2.5【推奨】
                                        #4.2.8【オプション】
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,arcfour
MACs
hmac-sha1,umac-64@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-ripemd160,hmac-ripemd160@openssh.com

Host *
  GSSAPIAuthentication yes
  ForwardX11Trusted yes
  SendEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
  SendEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
  SendEnv LC_IDENTIFICATION LC_ALL LANGUAGE
  SendEnv XMODIFIERS
```

5.1.2. SSH サーバの設定

（1）iptables の設定ファイル（/etc/sysconfig/iptables）

表 4：ログインホスト（sshgw）

File: /etc/sysconfig/iptables

```
# $NCA: iptables,v1.0 2014/12/27 sshgw $

-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp -s 198.51.100.0/24 --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A OUTPUT -p tcp -d 192.168.1.100 --dport 22 -j ACCEPT
-A OUTPUT -j DROP

# インバウンド                          #4.1.2【推奨】
#   クライアント PC（sshxx）のネットワーク（198.51.100.0/24）からの 22/tcp アクセス：許可
#   上記以外：拒否
# アウトバウンド                        #4.1.4【オプション】
#   接続先ホスト（192.168.1.100）への 22/tcp アクセス：許可
#   上記以外：拒否
```

表 5：接続先ホスト（sshsrv）

File: /etc/sysconfig/iptables

```
# $NCA: iptables,v1.0 2014/12/27 sshsrv $

-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp -s 192.168.1.0/24 --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A OUTPUT -p tcp -d 192.168.1.200 --dport 22 -j ACCEPT
-A OUTPUT -j DROP

# インバウンド                          #4.1.2【推奨】
#   同一ネットワーク（192/168.1.0/24）からの 22/tcp アクセス：許可
#   上記以外：拒否
# アウトバウンド                        #4.1.4【オプション】
#   データサーバ（192.168.1.200）への 22/tcp アクセス：許可
#   上記以外：拒否
```

表 6：データサーバ（sshdb）

File: /etc/sysconfig/iptables

```
# $NCA: iptables,v1.0 2014/12/27 sshdb $

-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp -s 192.168.1.0/24 --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A OUTPUT -j DROP

# インバウンド                          #4.1.2【推奨】
#   同一ネットワーク（192/168.1.0/24）からの 22/tcp アクセス：許可
#   上記以外：拒否
# アウトバウンド                        #4.1.4【オプション】
#   すべて：拒否
```

（2）ssh の設定ファイル（/etc/ssh/ssh_config）

表 7：ログインホスト（sshgw）、接続先ホスト（sshsrv）

File: /etc/ssh/ssh_config

```
# $NCA: ssh_config,v1.0 2014/12/27 sshgw $

HashKnownHosts yes                      #4.2.5【推奨】
                                        #4.2.8【オプション】
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,arcfour
MACs
hmac-sha1,umac-64@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-ripemd160,hmac-ripemd160@openssh.com

Host *
  GSSAPIAuthentication yes
  ForwardX11Trusted yes
  SendEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
  SendEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
  SendEnv LC_IDENTIFICATION LC_ALL LANGUAGE
  SendEnv XMODIFIERS
```

（3）sshd の設定ファイル（/etc/ssh/sshd_config）

表 8：ログインホスト（sshgw）、接続先ホスト（sshsrv）、データサーバ（sshdb）

File: /etc/ssh/sshd_config

```
# $NCA: sshd_config,v1.0 2014/12/27 sshgw, sshsrv, sshdb $

#Port 22                                #・・・コメントアウト（先頭に＃があること）を確認
Port 22222                              #4.2.4【推奨】・・・Port 22222 追記

#AddressFamily any                      #・・・コメントアウトを確認
AddressFamily inet                      #4.2.7【オプション】・・・AddressFamily inet 追記
#ListenAddress ::                       #4.2.7【オプション】
```

```
Protocol 2                              #4.2.1【推奨】

#PermitRootLogin yes                    #・・・コメントアウトを確認
PermitRootLogin no                      #4.2.6【オプション】・・・PermitRootLogin no 追記を推奨
                                        #4.2.6【オプション】
AllowUsers hanakogw hanakosrv hanakodb

#RSAAuthentication yes                  #・・・コメントアウトを確認
RSAAuthentication no                    #4.2.2【推奨】・・・RSAAuthentication no 追記
PubkeyAuthentication yes                #4.2.2【推奨】

RhostsRSAAuthentication no              #4.2.2【推奨】・・・RhostsRSAAuthentication no 追記
HostbasedAuthentication no              #4.2.2【推奨】・・・HostbasedAuthentication no 追記

#PasswordAuthentication yes             #・・・コメントアウトを確認
PasswordAuthentication no               #4.2.2【推奨】・・・PasswordAuthentication no 追記
PermitEmptyPasswords no                 #4.2.2【推奨】・・・PermitEmptyPasswords no 追記

#ChallengeResponseAuthentication yes    #・・・コメントアウトを確認
ChallengeResponseAuthentication no      #4.2.2【推奨】・・・ChallengeResponseAuthentication no 追記

GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
UsePAM yes

AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS

#AllowTcpForwarding yes                 #・・・コメントアウトを確認
AllowTcpForwarding no                   #4.2.3【推奨】・・・AllowTcpForwarding no 追記
X11Forwarding no                        #4.2.3【推奨】・・・X11Forwarding no 追記
#X11Forwarding yes                      #・・・コメントアウトを確認
MaxStartups 10:30:100                   #4.1.3【推奨】・・・MaxStartups 10:30:100 追記

Subsystem       sftp    /usr/libexec/openssh/sftp-server

                                        #4.2.8【オプション】・・・Ciphers、MACs 追記
Ciphers aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,arcfour
MACs
hmac-sha1,umac-64@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-ripemd160,hmac-ripemd160@openssh.com
```

5.2　利用方法 A

SSH クライアント PC（sshxx）から、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にログインする方法です。「利用方法 A」の方法でログインホスト（sshgw）を経由させると、公開鍵認証の場合、ログインホスト（sshgw）上にユーザ秘密鍵ファイル（id_rsa_srv）を格納しなければなりません。この状況は、ユーザ秘密鍵ファイル（id_rsa_srv）を、他人に渡していることに近いと言えます。また、ログインホスト（sshgw）に不正アクセスによる侵害が発生した場合、ユーザ秘密鍵ファイルを窃取されてしまう可能性があります。

ここでは、ログインの方法として、ProxyCommand と ssh-agent を使った利用形態を紹介します。

5.2.1.　ProxyCommand

コマンド（/usr/bin/ssh）のオプション ProxyCommand を利用して、接続先ホストのユーザ秘密鍵ファイル（id_rsa_srv）を SSH クライアント PC（sshxx）上に保持したまま、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にログインする方法について説明します。

■前提条件

ログインホスト（sshgw）に nc（netcat）がインストールされていること。

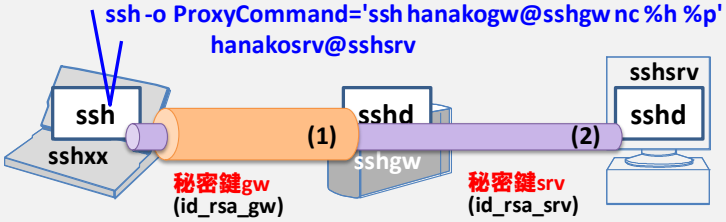
■操作

SSH クライアント PC（sshxx）での操作は、次の通りです。

Host: sshxx

■ユーザ hanakogw でユーザ秘密鍵(id_rsa_gw)を使って sshgw にログインした後、
ユーザ hanakosrv でユーザ秘密鍵(id_rsa_srv)を使って sshsrv にログインする

```
[hanako@sshxx ~]$ /usr/bin/ssh -i ~/.ssh/id_rsa_srv -o ProxyCommand='ssh hanakogw@sshgw -i ~/.ssh/id_rsa_gw nc %h %p' hanakosrv@sshsrv ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_gw':<パスフレーズ> ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_srv':<パスフレーズ> ⏎
Last login: Sun Aug  3 02:34:44 2014 from sshgw
[hanakosrv@sshsrv ~]$
```



「-o ProxyCommand='ssh hanakogw@sshgw nc %h %p'」が、ログインホスト（sshgw）を指定するオプションで、最初にログインホスト（sshgw）のパスフレーズもしくはパスワード入力が求められ、次に接続先ホスト（sshsrv）のパスフレーズもしくはパスワード入力が求められます。

この場合、ユーザ秘密鍵ファイル（id_rsa_srv）を用いた接続先ホスト（sshsrv）の公開鍵認証は、ログインホスト（sshgw）経由となります。これにより、ログインホスト（sshgw）上にユーザ秘密鍵ファイル（id_rsa_srv）を保持することなく、接続先ホスト（sshsrv）にログインできます。

■事前設定による操作性向上

ログインホスト（sshgw）を指定するオプションをユーザ毎の ssh 設定ファイル（~/.ssh/config）で設定することにより、操作性を改善できます。

File: ~/.ssh/config

```
Host sshsrv                                 # 接続先ホストの設定
    ProxyCommand    ssh hanakogw@sshgw nc %h %p
    HostName        sshsrv
    IdentityFile    ~/.ssh/id_rsa_srv       # 接続先ホストのユーザ秘密鍵ファイル

Host sshgw                                  # ログインホストの設定
    HostName        sshgw
    IdentityFile    ~/.ssh/id_rsa_gw        # ログインホストのユーザ秘密鍵ファイル
```

Host: sshxx

■ユーザ hanakogw でユーザ秘密鍵(id_rsa_gw)を使って sshgw にログインした後、
ユーザ hanakosrv でユーザ秘密鍵(id_rsa_srv)を使って sshsrv にログインする

```
[hanako@sshxx ~]$ /usr/bin/ssh hanakosrv@sshsrv ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_gw':＜パスフレーズ＞ ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_srv':＜パスフレーズ＞ ⏎
Last login: Sun Aug  3 02:34:44 2014 from sshgw
[hanakosrv@sshsrv ~]$
```

【コラム VIII:ユーザの気づきにより不正アクセスを発見】

公開鍵認証およびパスワード認証でログインした際、デフォルトで Last login（最終ログイン日時）が表示されます。

Host: sshxx

```
[hanako@sshxx ~]$ /usr/bin/ssh hanakosrv@sshsrv ⏎
hanakosrv@sshsrv's password:＜パスワード＞
Last login: Sun Aug  3 01:16:53 2014 from sshxx
[hanakosrv@sshsrv ~]$
```

ある日突然、利用者から「見覚えのないログインがある」という連絡が入り、不正アクセス事案が発生していたことが明らかになることがあります。「見覚えのないログインがある」発見のトリガの一つが、Last login の確認です。Last login の表示の時刻と発信元を確認するだけでも、不正アクセス事案の早期発見につながります。Last login の履歴参照コマンド（/usr/bin/last）とあわせて、利用者の皆さんに Last login の確認を呼びかけてみてください。

Host: sshxx

```
[hanakosrv@sshsrv ~]$ /usr/bin/last ⏎
hanakosrv   pts/0        sshxx           Sun Aug  3 11:39   still logged in
hanakosrv   pts/0        sshxx           Sun Aug  3 02:42 - 02:45  (00:02)
hanakosrv   pts/0        sshxx           Sun Aug  3 02:26 - 02:26  (00:00)
```

5.2.2. ssh-agent

SSH エージェントを利用して、接続先ホストのユーザ秘密鍵ファイル（id_rsa_srv）を SSH クライアント PC（sshxx）上に保持したまま、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にログインする方法について説明します。SSH エージェントは、SSH クライアント上でユーザ秘密鍵とパスフレーズを預かると共に、ユーザ秘密鍵に関する照会を、ユーザの代わりに応答する仕組みで、パスフレーズの入力の軽減などを図ることができます。

■SSH エージェントの起動

ユーザ秘密鍵（id_rsa_gw）（id_rsa_srv）とパスフレーズを SSH エージェントに預けます。

Host: sshxx

■SSH エージェントを起動する

```
[hanako@sshxx ~]$ eval `/usr/bin/ssh-agent` ⏎
Agent pid 32033
```

■SSH エージェントにユーザ秘密鍵(id_rsa_gw)(id_rsa_srv)を預ける

```
[hanako@sshxx ~]$ /usr/bin/ssh-add ~/.ssh/id_rsa_gw ⏎
Enter passphrase for /home/hanako/.ssh/id_rsa_gw:＜パスフレーズ＞ ⏎
Identity added: /home/hanako/.ssh/id_rsa_gw (/home/hanako/.ssh/id_rsa_gw)

[hanako@sshxx ~]$ /usr/bin/ssh-add ~/.ssh/id_rsa_srv ⏎
Enter passphrase for /home/hanako/.ssh/id_rsa_srv:＜パスフレーズ＞ ⏎
Identity added: /home/hanako/.ssh/id_rsa_srv (/home/hanako/.ssh/id_rsa_srv)
```

■SSH エージェントが保持しているユーザ秘密鍵を確認する

```
[hanako@sshxx ~]$ /usr/bin/ssh-add -l ⏎
2048 9b:c7:f4:1f:9a:bf:2c:f3:f7:f5:c4:84:c4:fe:a0:9d id_rsa_gw (RSA)
2048 e3:45:f6:14:bb:8f:0c:cb:bd:51:34:91:bb:1e:77:d8 id_rsa_srv (RSA)
```

■操作

SSH クライアント PC（sshxx）での操作は、次の通りです。

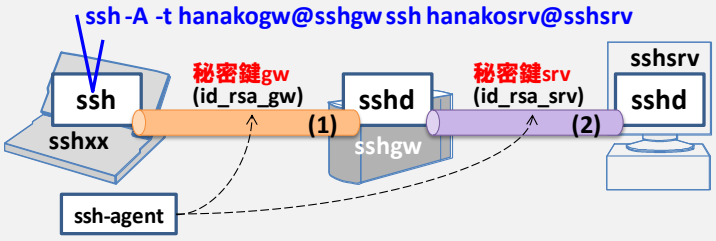
Host: sshxx

■認証エージェントフォワードを指定して、
ユーザ hanakogw のユーザ秘密鍵(id_rsa_gw)を使って sshgw にログインする
次に、ユーザ hanakosrv のユーザ秘密鍵(id_rsa_srv)を使って sshsrv にログインする

```
[hanako@sshxx ~]$ /usr/bin/ssh -A hanakogw@sshgw ⏎
Last login: Sun Aug  3 14:05:31 2014 from sshxx
[hanakogw@sshgw ~]$ /usr/bin/ssh hanakosrv@sshsrv ⏎
Last login: Sun Aug  3 14:06:31 2014 from sshgw
[hanakosrv@sshsrv ~]$
```

あるいは、

```
[hanako@sshxx ~]$ /usr/bin/ssh -A -t hanakogw@sshgw ssh hanakosrv@sshsrv ⏎
Last login: Sun Dec 28 19:57:36 2014 from sshgw
[hanakosrv@sshsrv ~]$
```



■SSH エージェントの終了

SSH エージェントを用いることで、パスフレーズの入力を省くことができるなどの操作性が向上します。その一方、パスフレーズで有効化したユーザ秘密鍵をメモリ上で保持するため、SSH クライアントのセキュリティ確保が重要となります。一連の操作が終わった時点で、SSH エージェントが保持しているユーザ秘密鍵を破棄し、SSH エージェントを終了しましょう。

Host: sshxx

■SSH エージェントが保持するユーザ秘密鍵を破棄する

```
[hanako@sshxx ~]$ /usr/bin/ssh-add -D ⏎
All identities removed.
[hanako@sshxx ~]$ /usr/bin/ssh-add -l ⏎
The agent has no identities.
```

■SSH エージェントを終了する

```
[hanako@sshxx ~]$ eval `ssh-agent -k` ⏎
Agent pid 32146 killed
```

5.3　利用方法 B

SSH クライアント PC（sshxx）から、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にファイルを転送する方法です。ここでは、ProxyCommand と ssh-agent を使った利用形態を紹介します。

5.3.1.　ProxyCommand

コマンド（/usr/bin/ssh）のオプション ProxyCommand を利用して、接続先ホストのユーザ秘密鍵ファイル（id_rsa_srv）を SSH クライアント PC（sshxx）上に保持したまま、ログインホスト（sshgw）を経由して、ファイルを転送する方法について説明します。

■前提条件

ログインホスト（sshgw）に nc（netcat）がインストールされていること。

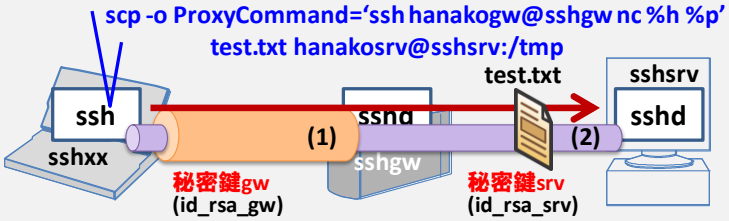
■操作

SSH クライアント PC（sshxx）での操作は、次の通りです。

Host: sshxx

■ユーザ秘密鍵(id_rsa_gw)(id_rsa_srv)を使って、ファイル test.txt を転送(sshxx＝＞sshsrv)する

```
[hanako@sshxx ~]$ /usr/bin/scp -i ~/.ssh/id_rsa_srv -o ProxyCommand='ssh hanakogw@sshgw -i ~/.ssh/id_rsa_gw nc %h %p' test.txt hanakosrv@sshsrv:/tmp ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_gw':＜パスフレーズ＞ ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_srv':＜パスフレーズ＞ ⏎
test.txt                                      100%    0       0.0KB/s   00:00
```



■ユーザ秘密鍵(id_rsa_gw)(id_rsa_srv)を使って、ファイル test.txt を転送(sshsrv＝＞sshxx)する

```
[hanako@sshxx ~]$ /usr/bin/scp -i ~/.ssh/id_rsa_srv -o ProxyCommand='ssh hanakogw@sshgw -i ~/.ssh/id_rsa_gw nc %h %p' hanakosrv@sshsrv:/tmp/test.txt test2.txt ⏎
test.txt                                      100%    0       0.0KB/s   00:00
```

ログインホスト（sshgw）を指定するオプションをユーザ毎の ssh 設定ファイル（~/.ssh/config）で設定した場合には、操作性を改善できます。設定方法は、5.2 節：利用方法 A ProxyCommand を参照してください。

Host: sshxx

■ファイル test.txt を転送(sshxx＝＞sshsrv)する

```
[hanako@sshxx ~]$ /usr/bin/scp test.txt hanakosrv@sshsrv:/tmp ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_gw':＜パスフレーズ＞ ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_srv':＜パスフレーズ＞ ⏎
test.txt                                      100%    0       0.0KB/s   00:00
```

5.3.2. ssh-agent

SSH エージェントを利用して、接続先ホストのユーザ秘密鍵ファイル（id_rsa_srv）を SSH クライアント PC（sshxx）上に保持したまま、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にファイルを転送する方法について説明します。

■SSH エージェントの起動

ユーザ秘密鍵（id_rsa_gw）（id_rsa_srv）とパスフレーズを SSH エージェントに預けます。
操作方法は、5.2.2 節：利用方法 A ssh-agent を参照してください。

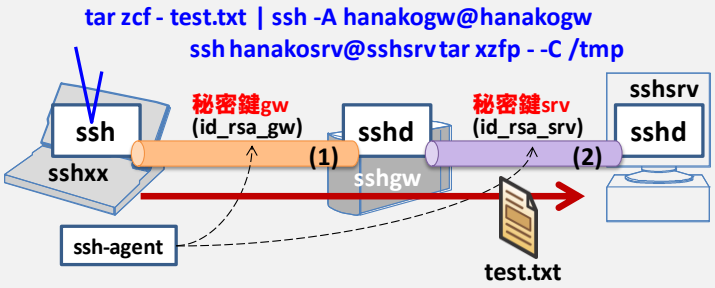
■操作

SSH クライアント PC（sshxx）での操作は、次の通りです。

Host: sshxx

■認証エージェントフォワードを指定して、
ユーザ秘密鍵(id_rsa_gw)(id_rsa_srv)を使って、ファイル test.txt を転送(sshxx＝＞sshsrv)する

```
[hanako@sshxx ~]$ /bin/tar zcf - test.txt | ssh -A hanakogw@sshgw ssh hanakosrv@sshsrv tar xzfp - -C /tmp ⏎
[hanako@sshxx ~]$
```



■ユーザ秘密鍵(id_rsa_gw)(id_rsa_srv)を使って、ファイル test.txt を転送(sshsrv＝＞sshxx)する

```
[hanako@sshxx ~]$ ssh -A hanakogw@sshgw ssh hanakosrv@sshsrv tar zcf - test.txt -C /tmp | /bin/tar xzfp - ⏎
[hanako@sshxx ~]$
```

■SSH エージェントの終了

一連の操作が終わった時点で、SSH エージェントが保持しているユーザ秘密鍵を破棄し、SSH エージェントを終了しましょう。操作方法は、5.2.2 節：利用方法 A ssh-agent を参照してください。

5.4　利用方法 C

SSH クライアント PC（sshxx）から、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にログインした後、接続先ホスト（sshsrv）からデータサーバ（sshdb）にファイルを転送する方法です。ここでは、ProxyCommand と ssh-agent を使った利用形態を紹介します。

5.4.1.　ProxyCommand

コマンド（/usr/bin/ssh）のオプション ProxyCommand と SSH エージェントとを利用して、データサーバのユーザ秘密鍵ファイル（id_rsa_db）を SSH クライアント PC（sshxx）上に保持したまま、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）からデータサーバ（sshdb）にファイルを転送する方法について説明します。

■前提条件

ログインホスト（sshgw）に nc（netcat）がインストールされていること。

■操作

SSH クライアント PC（sshxx）での操作は、次の通りです。

Host: sshxx

■SSH エージェントを起動する

```
[hanako@sshxx ~]$ eval `/usr/bin/ssh-agent` ⏎
Agent pid 8499
```

■SSH エージェントにユーザ秘密鍵(id_rsa_db)を預ける

```
[hanako@sshxx ~]$ /usr/bin/ssh-add ~/.ssh/id_rsa_db ⏎
Enter passphrase for /home/hanako/.ssh/id_rsa_db:＜パスフレーズ＞ ⏎
Identity added: /home/hanako/.ssh/id_rsa_db (/home/hanako/.ssh/id_rsa_db)
```

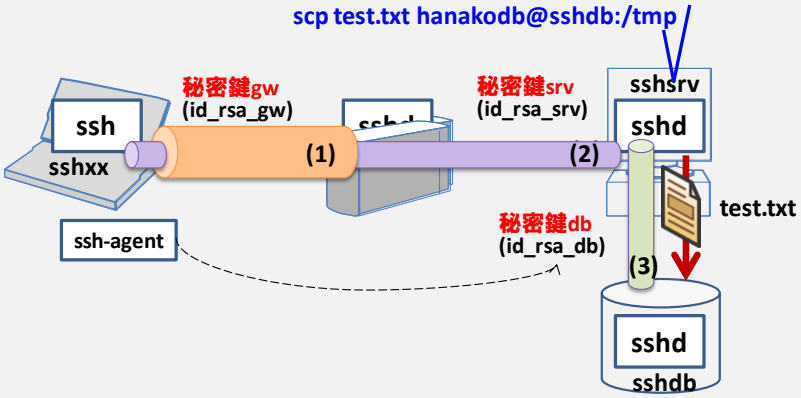
■ユーザ hanakogw でユーザ秘密鍵(id_rsa_gw)を使って sshgw にログインした後、
**　ユーザ hanakosrv でユーザ秘密鍵(id_rsa_srv)を使って sshsrv にログインする**

```
[hanako@sshxx ~]$ /usr/bin/ssh -A -i ~/.ssh/id_rsa_srv -o ProxyCommand='ssh hanakogw@sshgw -i ~/.ssh/id_rsa_gw nc %h %p' hanakosrv@sshsrv ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_gw':＜パスフレーズ＞ ⏎
Enter passphrase for key '/home/hanako/.ssh/id_rsa_srv':＜パスフレーズ＞ ⏎
Last login: Sun Dec 28 19:57:36 2014 from sshgw
[hanakosrv@sshsrv ~]$
```

■ファイル test.txt を転送（sshsrv＝＞sshdb）する

[hanakosrv@sshsrv ~]$ scp test.txt hanakodb@sshdb:/tmp ⏎

test.txt 100% 0 0.0KB/s 00:00



■ファイル test.txt を転送（sshdb＝＞sshsrv）する

[hanakosrv@sshsrv ~]$ scp hanakodb@sshdb:/tmp/test.txt test2.txt ⏎

test.txt 100% 0 0.0KB/s 00:00

■sshsrv、sshgw ログアウトした後、SSH エージェントを終了する

[hanako@sshxx ~]$ eval `ssh-agent -k` ⏎

Agent pid 8499 killed

5.4.2. ssh-agent

SSH エージェントを利用して、データサーバのユーザ秘密鍵ファイル（id_rsa_db）を SSH クライアント PC（sshxx）上に保持したまま、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にファイルを転送する方法について説明します。

■操作

SSH クライアント PC（sshxx）での操作は、次の通りです。

Host: sshxx

■SSH エージェントを起動する

```
[hanako@sshxx ~]$ eval `/usr/bin/ssh-agent` ⏎
Agent pid 8500
```

■SSH エージェントにユーザ秘密鍵(id_rsa_gw)(id_rsa_srv)(id_rsa_db)を預ける

```
[hanako@sshxx ~]$ /usr/bin/ssh-add ~/.ssh/id_rsa_gw ⏎
Enter passphrase for /home/hanako/.ssh/id_rsa_gw:＜パスフレーズ＞ ⏎
Identity added: /home/hanako/.ssh/id_rsa_gw (/home/hanako/.ssh/id_rsa_gw)

[hanako@sshxx ~]$ /usr/bin/ssh-add ~/.ssh/id_rsa_srv ⏎
Enter passphrase for /home/hanako/.ssh/id_rsa_srv:＜パスフレーズ＞ ⏎
Identity added: /home/hanako/.ssh/id_rsa_srv (/home/hanako/.ssh/id_rsa_srv)

[hanako@sshxx ~]$ /usr/bin/ssh-add ~/.ssh/id_rsa_db ⏎
Enter passphrase for /home/hanako/.ssh/id_rsa_db:＜パスフレーズ＞ ⏎
Identity added: /home/hanako/.ssh/id_rsa_db (/home/hanako/.ssh/id_rsa_db)
```

■認証エージェントフォワードを指定して、
ユーザ hanakogw のユーザ秘密鍵(id_rsa_gw)を使って sshgw にログインする
次に、ユーザ hanakosrv のユーザ秘密鍵(id_rsa_srv)を使って sshsrv にログインする

```
[hanako@sshxx ~]$ /usr/bin/ssh -A hanakogw@sshgw ⏎
Last login: Sun Dec 28 18:10:54 2014 from sshxx
[hanakogw@sshgw ~]$ /usr/bin/ssh -A hanakosrv@sshsrv ⏎
Last login: Sun Dec 28 19:24:04 2014 from sshgw
[hanakosrv@sshsrv ~]$
```

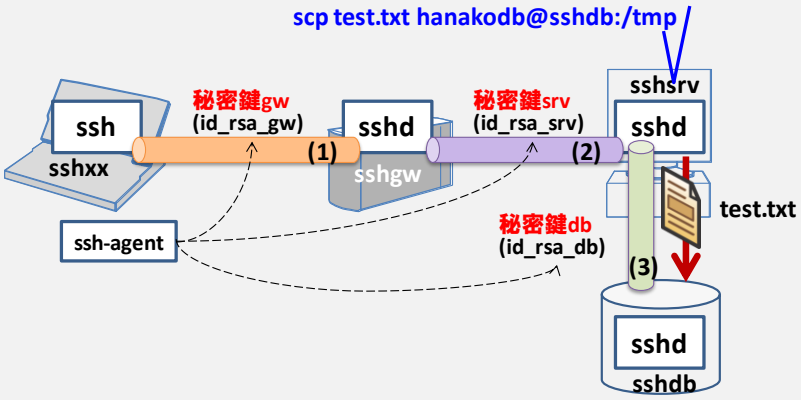
あるいは、

```
[hanako@sshxx ~]$ /usr/bin/ssh -A -t hanakogw@sshgw ssh -A hanakosrv@sshsrv ⏎
Last login: Sun Dec 28 19:57:36 2014 from sshgw
[hanakosrv@sshsrv ~]$
```

■ファイル test.txt を転送（sshsrv＝＞sshdb）する

[hanakosrv@sshsrv ~]$ scp test.txt hanakodb@sshdb:/tmp ⏎

test.txt 100% 0 0.0KB/s 00:00



■ファイル test.txt を転送（sshdb＝＞sshsrv）する

[hanakosrv@sshsrv ~]$ scp hanakodb@sshdb:/tmp/test.txt test2.txt ⏎

test.txt 100% 0 0.0KB/s 00:00

■sshsrv、sshgw ログアウトした後、SSH エージェントを終了する

[hanako@sshxx ~]$ eval `ssh-agent -k` ⏎

Agent pid 8500 killed

6　付録 B　SSH 関連コマンド

本章では、一般的な SSH 関連コマンドを紹介します。

- [ssh]

ssh コマンドは、リモートマシンにログインしたり、リモートマシン上でコマンドを実行するときに使います。

従来このような用途には rlogin、rsh、telnet といったコマンドが使われていましたが、これらのコマンドでは通信内容を暗号化していません。そのため、ネットワーク上に流れる通信内容は容易に読み取ることが可能です。ssh では、通信内容を暗号化することにより、傍受されてもその内容を容易には読み取られないようにします。

使用例

ssh host1	host1 にログインする。host1 でのユーザ名はローカル側のユーザ名をそのまま使う。
ssh user@host1	host1 にユーザ名 user としてログインする。
ssh -l user host1	上と同じだがユーザ名を -l オプションで指定する。
ssh -p 22222 user@host1	サーバの待ち受けポートを 22/tcp 以外にしている場合には -p オプションでポート番号を指定する。左記は 22222/tcp の例。
ssh -v user@host1	-v オプションをつけると、リモートマシン接続時の sshd とのやりとりを表示する。
ssh -l user host1 cmd	host1 上でユーザ名 user として cmd を実行する。

- [scp]

scp コマンドは二つのホスト間でファイルをコピーします。

scp コマンドの引数にファイル名を指定する際、リモートホスト側のファイル名には、ホスト名やユーザ名も合せて指定します。実際の通信には ssh を使って行われ、パスワード認証であればパスワードを、公開鍵認証であればユーザ秘密鍵を使うためのパスフレーズの入力が要求されます。

使用例

scp file1 host:file2	ローカルホストのカレントディレクトリにある file1 をリモートホスト host のホームディレクトリ上に file2 としてコピーする。file2 を省略した場合、ファイル名は file1 となる。
scp user@host:tmp/file1 file2	リモートホスト host のユーザ user のホームディレクトリにある tmp ディレクトリ配下の file1 をローカルホストのカレントディレクトリに file2 としてコピーする。

scp user@host:tmp/file1 file2	リモートホスト host のユーザ user のホームディレクトリにある tmp ディレクトリ配下の file1 をローカルホストのカレントディレクトリに file2 としてコピーする。
scp -p file1 host:	ファイルのタイムスタンプを保存したいときは cp コマンドと同じく -p オプションを指定する。
scp -p -P 22222 file1 user@host:tmp/file2	サーバの待ち受けポートが 22/tcp 以外の場合には -P オプションでポート番号を指定する。

- [sftp]

sftp は、ftp(1)コマンドと同じく対話的にファイル転送を行うためのプログラムです。

全ての操作は ssh で作られた暗号化通信路を通じて行なわれます。また、公開鍵認証やデータ圧縮など ssh で提供されている機能の多くをそのまま使用することができます。

使用例

sftp user@host	リモートホスト host にユーザ user としてログインし、ftp コマンドと同様にコマンド操作を行う。
sftp -o 'Port 22222' host	リモートホストが 22/tcp 以外のポートを使っている場合には -o オプションでポートを指定する。

- [ssh-keygen]

ssh-keygen は ssh コマンドで使用する鍵対の生成、管理、フォーマット変換を行います。

ssh-keygen では SSH プロトコルバージョン 1 で使う RSA 鍵と、SSH プロトコルバージョン 2 で使う RSA 鍵や DSA 鍵を生成することができます。生成する鍵のタイプは -t オプション（-t rsa / -t dsa）で指定します。引数なしで起動したときには SSH プロトコルバージョン 2 用の RSA 鍵を生成します。

使用例

ssh-keygen -f .ssh/id_rsa	新たに鍵対を生成し、ユーザ秘密鍵を.ssh/id_rsa に、ユーザ公開鍵を.ssh/id_rsa.pub に保存する。生成した鍵を保存する際、鍵を保護するためのパスフレーズを要求する。
ssh-keygen -p -f .ssh/id_rsa	ユーザ秘密鍵のパスフレーズを変更するときは -p オプションを使用する
ssh-keygen -l -f .ssh/id_rsa	-f オプションで指定したユーザ秘密鍵に対応するユーザ公開鍵のフィンガープリントを出力する。
ssh-keygen -e -f .ssh/id_rsa	OpenSSH の鍵ファイルから RFC4716 形式のユーザ公開鍵を出力する。
ssh-keygen -i -f .ssh/id_rsa	RFC4716 形式のユーザ公開鍵ファイルから OpenSSH 形式のユーザ公開鍵に変換する。putty などで鍵を生成するユーザ公開鍵は RFC4716 形式になっている。

- [ssh-agent]

  ssh-agent は公開鍵認証（RSA、DSA）で使うユーザ秘密鍵を保持します。

  通常、X ウィンドウセッションやログインセッションの最初に ssh-agent を起動し、他の全てのプログラムを ssh-agent プログラムに対するクライアントとして起動します。ssh-agent に関する環境変数を設定し、ssh-add コマンドでユーザ秘密鍵を ssh-agent に登録しておくことで、他のマシンに ssh ログインする際の公開鍵認証が ssh-agent を通じて行なわれます。

  使用例

eval `ssh-agent`	ssh-agent を起動し、必要な環境変数の設定を行う。
ssh-agent /bin/sh	ssh-agent を起動し、その子プロセスとしてシェルを起動する。

- [ssh-add]

  ssh-add は ssh-agent にユーザ秘密鍵を登録するために使います。

  引数なしで起動した場合には、ファイル~/.ssh/id_rsa、~/.ssh/id_rsa、~/.ssh/identity にあるユーザ秘密鍵が登録されます。コマンドライン引数で鍵ファイルを指定することもできます。パスフレーズが必要な場合には、ユーザにパスフレーズの入力を要求します。パスフレーズ入力はユーザのターミナルデバイスから読み込まれます。

  使用例

ssh-add -l	ssh-agent に登録されているユーザ秘密鍵に対応するユーザ公開鍵のフィンガープリントを出力する。
ssh-add .ssh/id_rsa	ssh-agent に指定したユーザ秘密鍵を登録する。
ssh-add -d .ssh/id_rsa	ssh-agent から指定したユーザ秘密鍵を削除する。

7　付録 C　クラウドでのセキュリティ設定

クラウドサービス上でサーバを構築する場合は、通常インターネット回線を通じて SSH サービスにアクセスすることになりますが、インターネット全体からのアクセスを許可するのではなく、運用で利用する IP アドレス以外からのアクセスを拒否することでセキュリティを高めることができます。
ここでは、クラウドサービスを想定して、発信元 IP アドレスを制限する方法を紹介します。

なお、クラウドサービスの場合はシステム間連携や管理ツール用に API が用意されていたり、システム設定を行う Web インターフェースが提供されている場合がありますので、それらのインターフェースについてもセキュアな設定にしておく必要があります。詳細については、各クラウドサービスのセキュリティページを参照してください。

- Amazon Web Services

AWS ではセキュリティグループを利用して発信元 IP アドレスを制限することができます。設定は、AWS マネジメントコンソールにログインし、EC2 ホーム＞NETWORK & SECURITY＞Security Groups へ移動して行います。詳細については以下のドキュメントを参考にしてください。EC2-classis と VPC で作成するセキュリティグループが異なる点に注意が必要です。この点も以下のドキュメントで説明されています。

> Amazon EC2 セキュリティグループ
http://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/using-network-security.html

なおアウトバウンドのトラフィックのルールを追加できるのは VPC インスタンスのみです。

AWS 以外のクラウドサービスにおいても同様にセキュリティ設定ができます。以下にいくつかのサービスにおける概略と詳細情報へのリンクを掲載します。

- さくら VPS

OS 上の iptables を利用して発信元 IP アドレスを制限します。詳細については以下のドキュメントを参考にしてください。

> iptables の設定方法
http://support.sakura.ad.jp/manual/vps/security/iptables.html

- IDC Frontier セルフクラウド

Web 上のファイアウォール設定で発信元 IP アドレスを制限します。

> セルフクラウド ご利用ガイド（ネットワーク編）
http://www.idcf.jp/cloud/self/manual/03.html

- GMO クラウド

Web 上のファイアウォール設定で発信元 IP アドレスを制限します。

> GMO クラウド Public クラウドコンソール
http://support.gmocloud.com/public/guide/portal_manual/fw.html

8　付録 D　Windows 環境での ssh-agent 利用

Windows 環境でも、SSH エージェント（ssh-agent）を利用できます。ここでは、Tera Term（SSH コマンド相当）[c]と Pagent（ssh-agent 相当）[d]とを組合せて、SSH クライアント PC（sshxx）から、ログインホスト（sshgw）を経由して、接続先ホスト（sshsrv）にログインする事例を紹介します。

■SSH エージェントの起動

pagent.exe を実行した後、タスクトレイ上のアイコンから Add key を選択して、ユーザ秘密鍵を追加します（図 3）。なお、OpenSSH の ssh-keygen でユーザ鍵を作成した場合には、pagent に追加する前に、ユーザ秘密鍵を PuTTY 形式に変換する必要があります[e]。

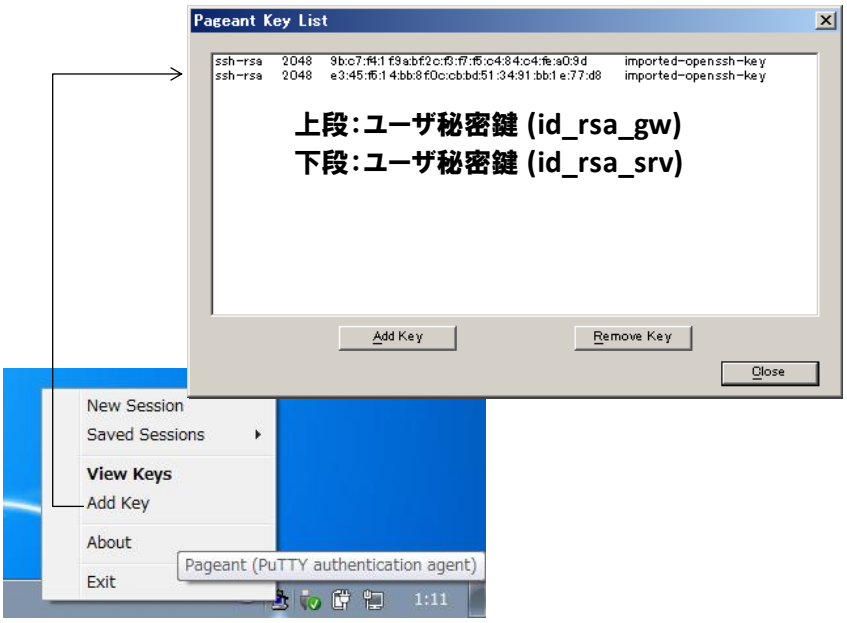


図 3：pagent.exe へのユーザ秘密鍵の追加

c) Tera Term
http://sourceforge.jp/projects/ttssh2/
d) PuTTY Download Page
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
e) PuTTYgen を使って、OpenSSH 形式のユーザ秘密鍵を PuTTY 形式に変換できます。

■操作

ttermpro.exe を実行し、ログインホスト（sshgw）を指定した後、「エージェント転送する」、「Pagent を使う」のチェックボックスをオンにして、ユーザ名（hanakogw）でアクセスします（図 4）。

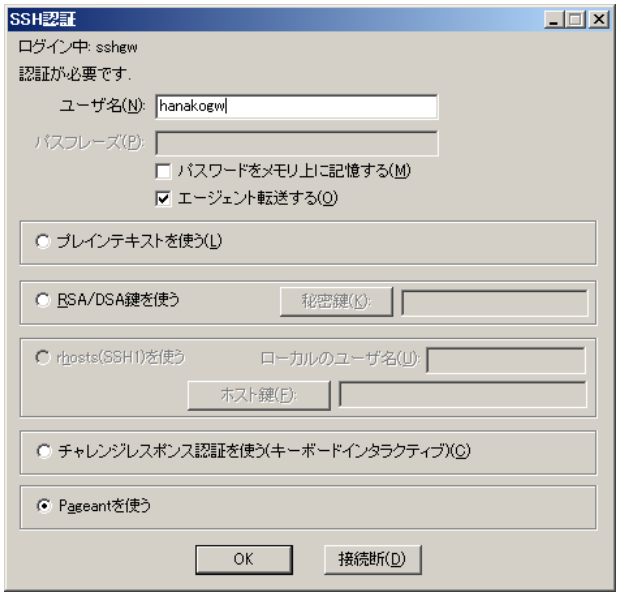


図 4：ttermpro.exe を用いたログインホストへのアクセス

ログインホストへのログインが終了した後、接続先サーバ（sshsrv）を指定して、ユーザ名（hanakosrv）でアクセスします（[hanakogw@sshgw ~]$ ssh hanakosrv@sshsrv）。「エージェント転送要求を受け入れますか？」というダイアログが表示されますので、「はい」を選択すれば接続が完了します（図 5）。

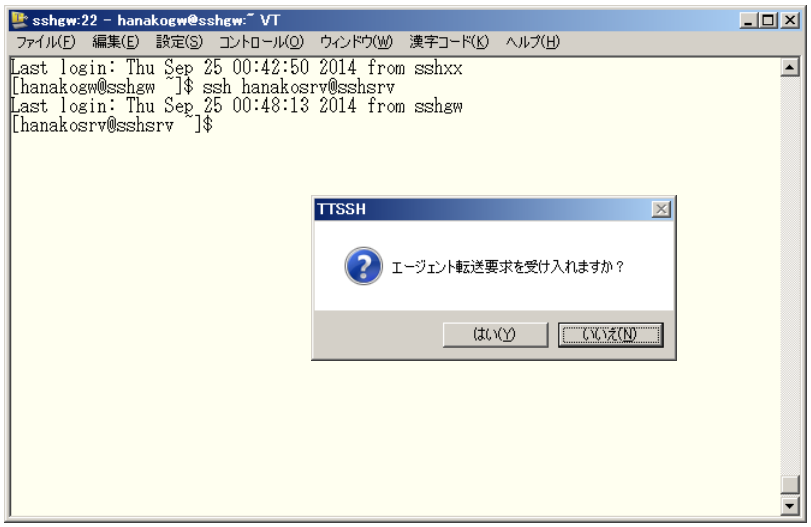


図 5：ttermpro.exe を用いたログインホスト経由での接続先ホストへのアクセス

■SSH エージェントの終了

タスクトレイ上のアイコンから Exit を指定して、SSH エージェントを終了します。

SSH サーバセキュリティ設定検討 WG
執筆および協力者一覧

茂岩祐樹	DeNA CERT	株式会社ディー・エヌ・エー
寺田真敏	HIRT	株式会社日立製作所
徳田敏文	IBM-CSIRT	日本アイ・ビー・エム株式会社
戸田洋三	JPCERT/CC	一般社団法人 JPCERT コーディネーションセンター
中野渡敬教	Fuji Xerox-CERT	富士ゼロックス株式会社
中村暢宏	YIRD	ヤフー株式会社
馬場亮一	KEK CSIRT	高エネルギー加速器研究機構
平岡洋介	JSOC	株式会社ラック
宮本久仁男	NTTDATA-CERT	株式会社 NTT データ
宮本靖	Met-CIRT	メットライフ生命保険株式会社
ラウリ コルツパルン	CDI-CIRT	株式会社サイバーディフェンス研究所