

Top 20 OpenSSH Server Best Security Practices ssh linux aix

Article Number: 672 | Rating: Unrated | Last Updated: Fri, May 15, 2020 11:36 AM

OpenSSH is the implementation of the SSH protocol. OpenSSH is recommended for remote login, making backups, remote file transfer via scp or sftp, and much more. SSH is perfect to keep confidentiality and integrity for data exchanged between two networks and systems. However, the main advantage is server authentication, through the use of public key cryptography. From time to time there are **rumors** about OpenSSH **zero day** exploit. Here are a few things you need to tweak in order to improve OpenSSH server security.

Default Config Files and SSH Port

- `/etc/ssh/sshd_config` – OpenSSH server configuration file.
- `/etc/ssh/ssh_config` – OpenSSH client configuration file.
- `~/.ssh/` – Users ssh configuration directory.
- `~/.ssh/authorized_keys` or `~/.ssh/authorized_keys` – Lists the public keys (RSA or DSA) that can be used to log into the user's account
- `/etc/nologin` – If this file exists, sshd refuses to let anyone except root log in.
- `/etc/hosts.allow` and `/etc/hosts.deny` : Access controls lists that should be enforced by tcp-wrappers are defined here.
- **SSH default port** : TCP 22

#1: Disable OpenSSH Server

Workstations and laptop can work without OpenSSH server. If you need not to

provide the remote login and file transfer capabilities of SSH, disable and remove the SSHD server. CentOS / RHEL / Fedora Linux user can disable and remove openssh-server with yum command:

```
# chkconfig sshd off
```

```
# yum erase openssh-server
```

Debian / Ubuntu Linux user can disable and remove the same with apt-get command:

```
# apt-get remove openssh-server
```

You may need to update your iptables script to remove ssh exception rule. Under CentOS / RHEL / Fedora edit the files /etc/sysconfig/iptables and /etc/sysconfig/ip6tables. Once done restart iptables service:

```
# service iptables restart
```

```
# service ip6tables restart
```

#2: Only Use SSH Protocol 2

SSH protocol version 1 (SSH-1) has man-in-the-middle attacks problems and security vulnerabilities. SSH-1 is obsolete and should be avoided at all cost. Open sshd_config file and make sure the following line exists:

```
Protocol 2
```

```
Protocol 2
```

```
Protocol 2
```

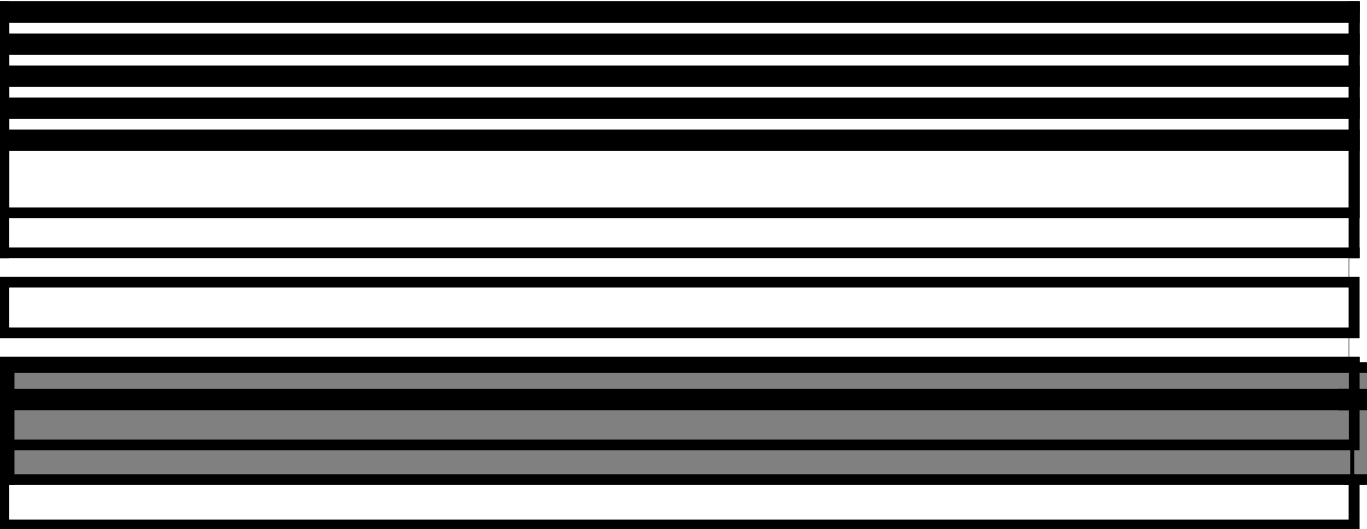
#3: Limit Users' SSH Access

By default all systems user can login via SSH using their password or public key. Sometime you create UNIX / Linux user account for ftp or email purpose. However, those user can login to system using ssh. They will have full access to system tools including compilers and scripting languages such as Perl, Python which can open network ports and do many other fancy things. One of my client has really outdated php script and an attacker was able to create a new account on the system via a php script. However, attacker failed to get into box via ssh because it wasn't in AllowUsers.

Only allow root, vivek and jerry user to use the system via SSH, add the following to sshd_config:

```
AllowUsers root vivek jerry
```

Alternatively, you can allow all users to login via SSH but deny only a few users, with the following line:



You can also configure Linux PAM allows or deny login via the sshd server. You can allow list of group name to access or deny access to the ssh.

#4: Configure Idle Log Out Timeout Interval

User can login to server via ssh and you can set an idel timeout interval to avoid unattended ssh session. Open sshd_config and make sure following values are configured:



You are setting an idle timeout interval in seconds (300 secs = 5 minutes). After this interval has passed, the idle user will be automatically kicked out (read as logged out). See how to automatically log BASH / TCSH / SSH users out after a period of inactivity for more details.

#5: Disable .rhosts Files

Don't read the user's ~/.rhosts and ~/.shosts files. Update sshd_config with the following settings:

```
IgnoreRhosts yes
IgnoreShosts yes
IgnoreUserKnownHosts yes
```

SSH can emulate the behavior of the obsolete rsh command, just disable insecure access via RSH.

#6: Disable Host-Based Authentication

To disable host-based authentication, update `sshd_config` with the following option:

```
[REDACTED]
```

#7: Disable root Login via SSH

There is no need to login as root via ssh over a network. Normal users can use `su` or `sudo` (recommended) to gain root level access. This also make sure you get full auditing information about who ran privileged commands on the system via `sudo`. To disable root login via SSH, update `sshd_config` with the following line:

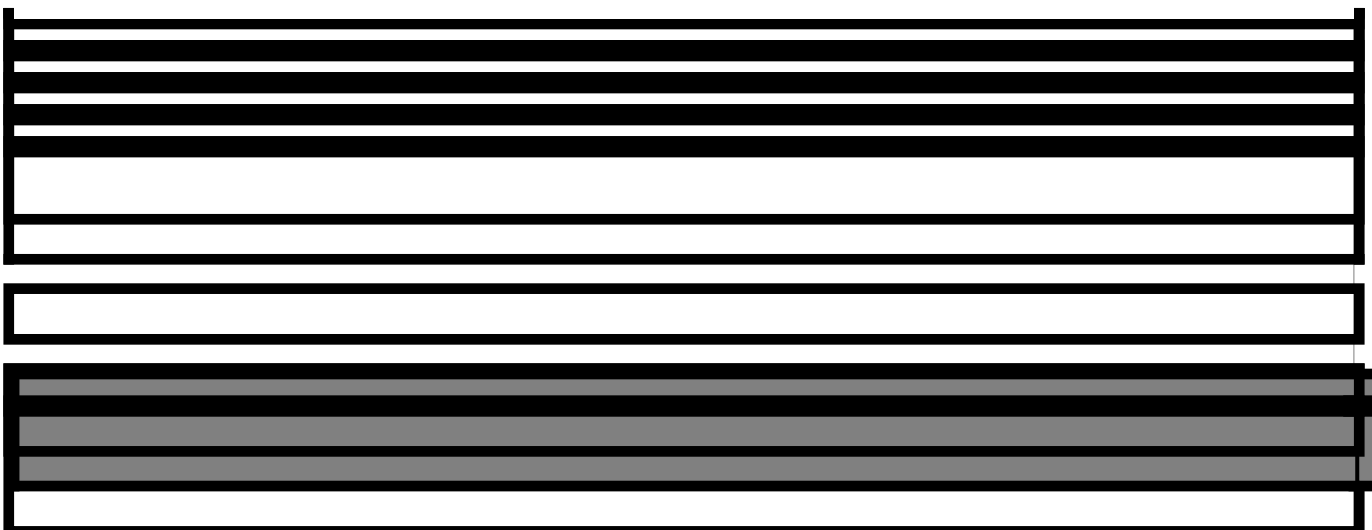
```
[REDACTED]
```

However, bob made excellent point:

*Saying “don’t login as root” is h*****t. It stems from the days when people sniffed the first packets of sessions so logging in as yourself and su-ing decreased the chance an attacker would see the root pw, and decrease the chance you got spoofed as to your telnet host target, You’d get your password spoofed but not root’s pw. Gimme a break. this is 2005 – We have ssh, used properly it’s secure. used improperly none of this 1989 will make a damn bit of difference. -Bob*

#8: Enable a Warning Banner

Set a warning banner by updating sshd_config with the following line:



Sample /etc/issue file:

[Redacted content]

[Redacted content]

[Redacted content]

Above is standard sample, consult your legal team for exact user agreement and legal notice details.

#8: Firewall SSH Port # 22

You need to firewall ssh port # 22 by updating iptables or pf firewall configurations. Usually, OpenSSH server must only accept connections from your LAN or other remote WAN sites only.

Netfilter (Iptables) Configuration

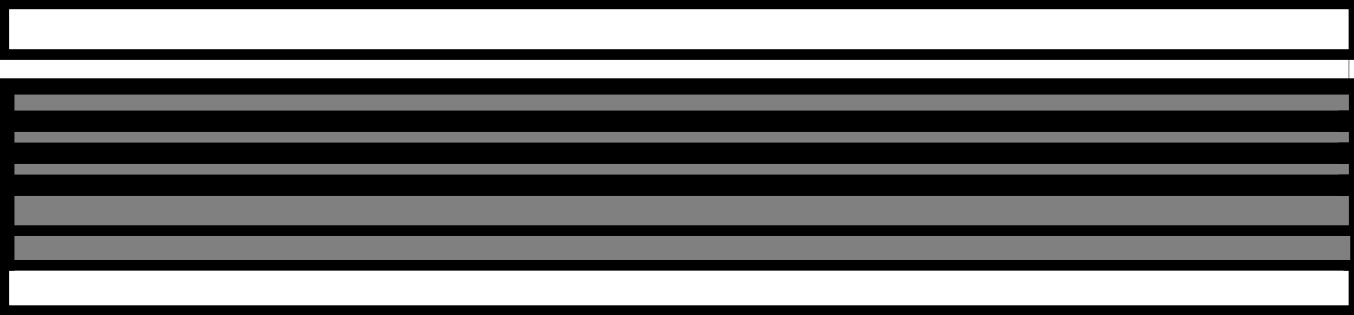
Update /etc/sysconfig/iptables (Redhat and friends specific file) to accept connection only from 192.168.1.0/24 and 202.54.1.5/29, enter:

```
iptables -F
iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT
iptables -A INPUT -s 202.54.1.5/29 -j ACCEPT
iptables -A INPUT -j REJECT --reject-with icmp-host-prohibited
iptables -F
```

If you've dual stacked sshd with IPv6, edit /etc/sysconfig/ip6tables (Redhat and friends specific file), enter:

```
ip6tables -F
ip6tables -A INPUT -s ipv6network::/ipv6mask -j ACCEPT
ip6tables -A INPUT -j REJECT --reject-with icmp6-host-prohibited
ip6tables -F
```

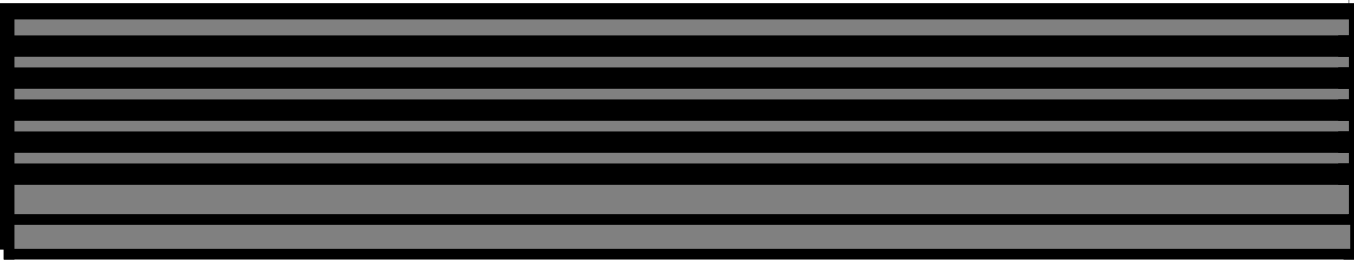
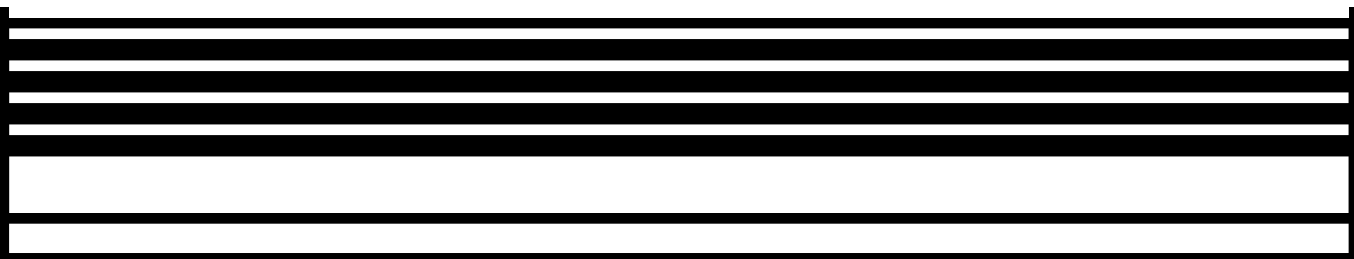
Replace ipv6network::/ipv6mask with actual IPv6 ranges.



A better approach to use proactive approaches scripts such as fail2ban or denyhosts (see below).

#10: Use Strong SSH Passwords and Passphrase

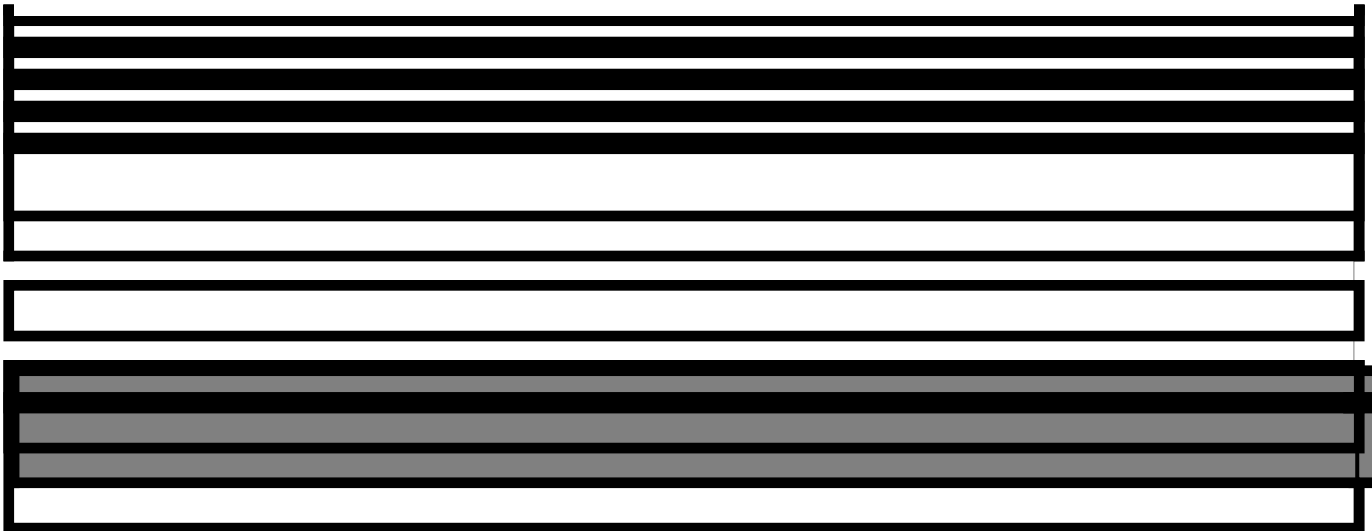
It cannot be stressed enough how important it is to use strong user passwords and passphrase for your keys. Brute force attack works because you use dictionary based passwords. You can force users to avoid passwords against a dictionary attack and use john the ripper tool to find out existing weak passwords. Here is a sample random password generator (put in your ~/.bashrc):



Run it:

```
genpasswd 16
```

Output:



#11: Use Public Key Based Authentication

Use public/private key pair with password protection for the private key. See how to use RSA and DSA key based authentication. Never ever use passphrase free key (passphrase key less) login.

#12: Use Keychain Based Authentication

keychain is a special bash script designed to make key-based authentication incredibly convenient and flexible. It offers various security benefits over passphrase-free keys. See how to setup and use keychain software.

#13: Chroot SSHD (Lock Down Users To Their Home Directories)

By default users are allowed to browse the server directories such as /etc/, /bin and so on. You can protect ssh, using os based chroot or use special tools such as rssh. With the release of OpenSSH 4.8p1 or 4.9p1, you no longer have to rely on third-party hacks such as rssh or complicated chroot(1) setups to lock users to their home directories. See this blog post about new ChrootDirectory directive to lock down users to their home directories.

#14: Use TCP Wrappers

TCP Wrapper is a host-based Networking ACL system, used to filter network access to Internet. OpenSSH does supports TCP wrappers. Just update your /etc/hosts.allow file as follows to allow SSH only from 192.168.1.2 172.16.23.12 :

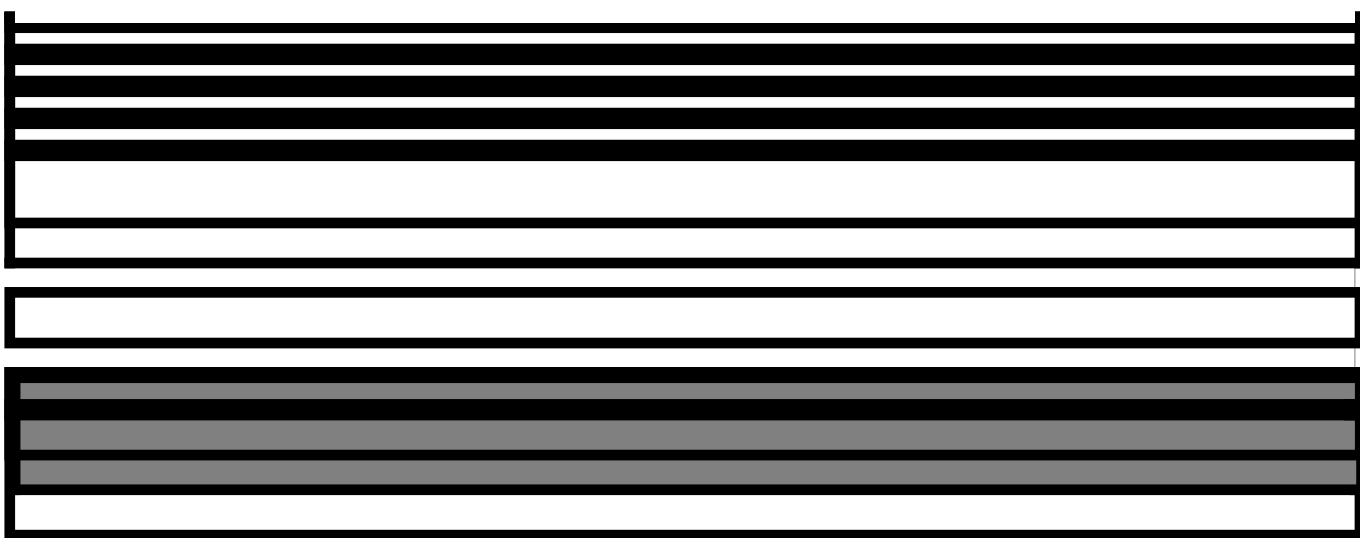
```
192.168.1.2: sshd
172.16.23.12: sshd
```

```
sshd: 192.168.1.2: sshd
sshd: 172.16.23.12: sshd
```

```
sshd: 192.168.1.2: sshd
sshd: 172.16.23.12: sshd
```

#15: Disable Empty Passwords

You need to explicitly disallow remote login from accounts with empty passwords, update `sshd_config` with the following line:



#16: Thwart SSH Crackers (Brute Force Attack)

Brute force is a method of defeating a cryptographic scheme by trying a large number of possibilities using a single or distributed computer network. To prevent brute force attacks against SSH, use the following softwares:

- DenyHosts is a Python based security tool for SSH servers. It is intended to prevent brute force attacks on SSH servers by monitoring invalid login

- attempts in the authentication log and blocking the originating IP addresses.
- Explains how to setup DenyHosts under RHEL / Fedora and CentOS Linux.
 - Fail2ban is a similar program that prevents brute force attacks against SSH.
 - security/sshguard-pf protect hosts from brute force attacks against ssh and other services using pf.
 - security/sshguard-ipfw protect hosts from brute force attacks against ssh and other services using ipfw.
 - security/sshguard-ipfilter protect hosts from brute force attacks against ssh and other services using ipfilter.
 - security/sshblock block abusive SSH login attempts.
 - security/sshit checks for SSH/FTP bruteforce and blocks given IPs.
 - BlockHosts Automatic blocking of abusive IP hosts.
 - Blacklist Get rid of those bruteforce attempts.
 - Brute Force Detection A modular shell script for parsing application logs and checking for authentication failures. It does this using a rules system where application specific options are stored including regular expressions for each unique auth format.
 - IPQ BDB filter May be considered as a fail2ban lite.

#17: Rate-limit Incoming Port # 22 Connections

Both netfilter and pf provides rate-limit option to perform simple throttling on incoming connections on port # 22.

Iptables Example

The following example will drop incoming connections which make more than 5 connection attempts upon port 22 within 60 seconds:

```
[REDACTED]
```

```
[REDACTED]
```

```
[REDACTED]
```

Call above script from your iptables scripts. Another config option:

```
[REDACTED]
```

```
[REDACTED]
```

```
[REDACTED]
```

See iptables man page for more details.

*BSD PF Example

The following will limit the maximum number of connections per source to 20 and rate limit the number of connections to 15 in a 5 second span. If anyone breaks our rules add them to our abusive_ips table and block them for making any further connections. Finally, flush keyword kills all states created by the matching rule which originate from the host which exceeds these limits.

```
[REDACTED]
```

```
[REDACTED]
```

```
[REDACTED]
```

#18: Use Port Knocking

Port knocking is a method of externally opening ports on a firewall by generating a connection attempt on a set of prespecified closed ports. Once a correct sequence of connection attempts is received, the firewall rules are dynamically modified to allow the host which sent the connection attempts to connect over specific port(s).

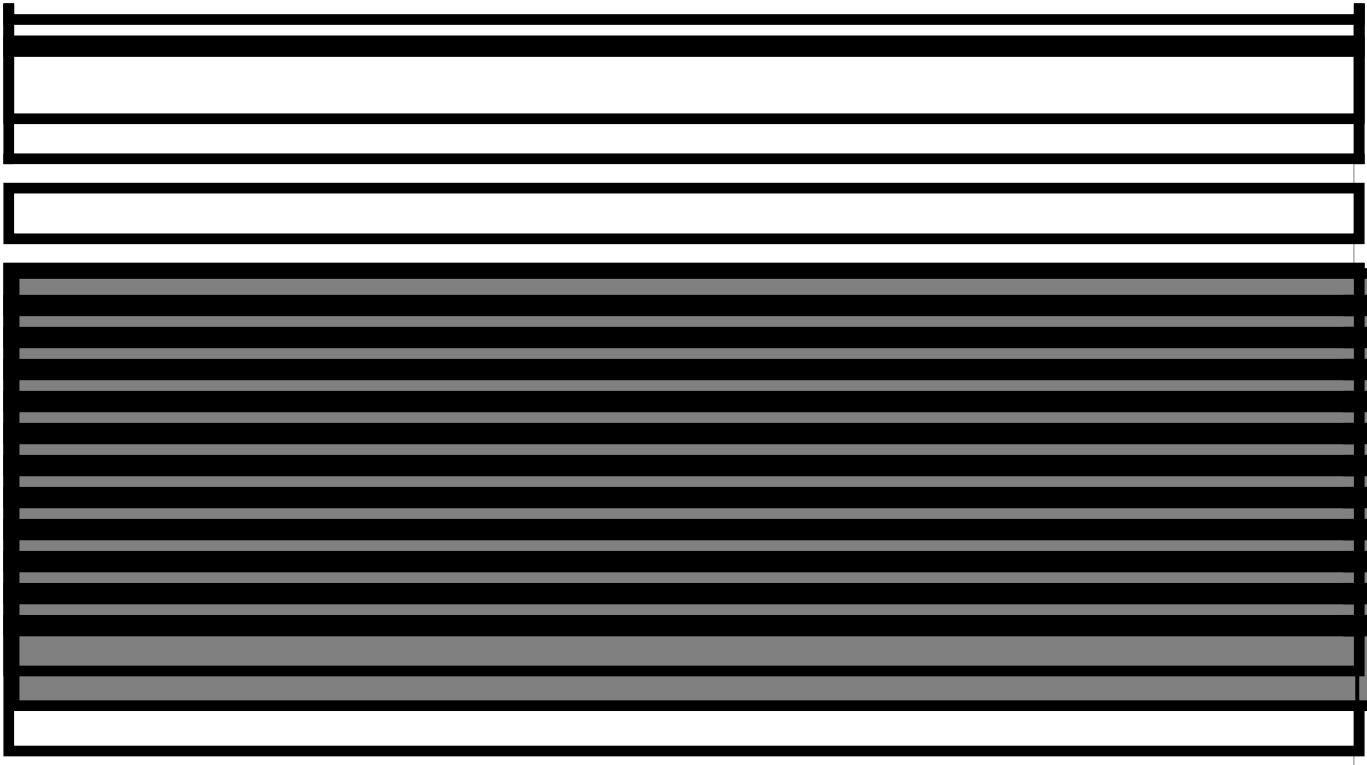
A sample port Knocking example for ssh using iptables:

```
[REDACTED]
```

- fwknop is an implementation that combines port knocking and passive OS fingerprinting.
- Multiple-port knocking Netfilter/IPtables only implementation.

#19: Use Log Analyzer

Read your logs using logwatch or logcheck. These tools make your log reading life



Verify your sshd_config file before restarting / reloading changes:

```
# /usr/sbin/sshd -t
```

Posted - Fri, May 15, 2020 11:36 AM. This article has been viewed 7025 times.

Online URL: <http://kb.ictbanking.net/article.php?id=672>