Table Of Contents

nixCraft: Linux Tips, Hacks, Tutorials, And Ideas In Blog Format
http://www.cyberciti.biz/

Top 20 OpenSSH Server Best Security Practices

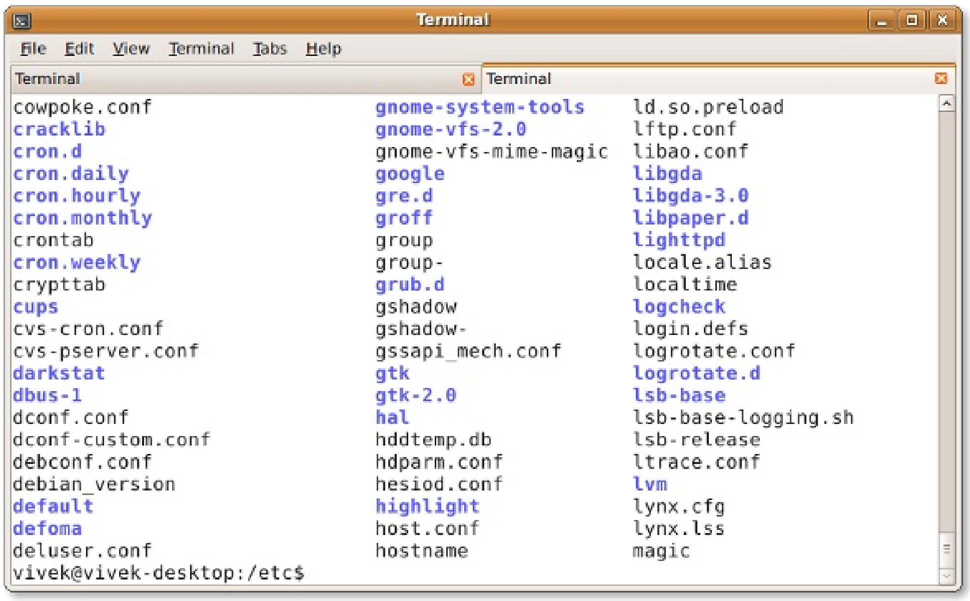
Posted By Vivek Gite | No Comments

[1]

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 [2] about OpenSSH zero day [3] 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



```
cowpoke.conf           gnome-system-tools    ld.so.preload
cracklib               gnome-vfs-2.0         lftp.conf
cron.d                 gnome-vfs-mime-magic  libao.conf
cron.daily             google                libgda
cron.hourly            gre.d                 libgda-3.0
cron.monthly           groff                 libpaper.d
crontab                group                 lighttpd
cron.weekly            group-                locale.alias
crypttab               grub.d                localtime
cups                   gshadow               logcheck
cvs-cron.conf          gshadow-              login.defs
cvs-pserver.conf       gssapi_mech.conf      logrotate.conf
darkstat               gtk                   logrotate.d
dbus-1                 gtk-2.0               lsb-base
dconf.conf             hal                   lsb-base-logging.sh
dconf-custom.conf      hddtemp.db            lsb-release
debconf.conf           hdparm.conf           ltrace.conf
debian_version         hesiod.conf           lvm
default                highlight             lynx.cfg
defoma                 host.conf             lynx.lss
deluser.conf           hostname              magic
vivek@vivek-desktop:/etc$
```

[4]

SSH Session in Action

#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 [5] 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
```

#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:

```
DenyUsers saroj anjali foo
```

You can also configure Linux PAM [6] allows or deny login via the sshd server. You can allow list of group name [7] 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:

```
ClientAliveInterval 300
ClientAliveCountMax 0
```

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 [8] 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
```

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:

```
HostbasedAuthentication no
```

#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:

```
PermitRootLogin no
```

However, bob made excellent [9] point:

> Saying "don't login as root" is horseshit. 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 decreast the chance you got spoofed as to your telnet host target, You'd get your password spoofed but not root's pw. Gimme a fucking break. this is 2005 - We have ssh, used properly it's secure. used improperly none of this 1989 bullshit 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:

```
Banner /etc/issue
```

Sample /etc/issue file:

```
----------------------------------------------------------------------------------------------
You are accessing a XYZ Government (XYZG) Information System (IS) that is provided for authorized use only.
By using this IS (which includes any device attached to this IS), you consent to the following conditions:

+ The XYZG routinely intercepts and monitors communications on this IS for purposes including, but not limited to,
penetration testing, COMSEC monitoring, network operations and defense, personnel misconduct (PM),
law enforcement (LE), and counterintelligence (CI) investigations.

+ At any time, the XYZG may inspect and seize data stored on this IS.

+ Communications using, or data stored on, this IS are not private, are subject to routine monitoring,
interception, and search, and may be disclosed or used for any XYZG authorized purpose.

+ This IS includes security measures (e.g., authentication and access controls) to protect XYZG interests--not
for your personal benefit or privacy.

+ Notwithstanding the above, using this IS does not constitute consent to PM, LE or CI investigative searching
or monitoring of the content of privileged communications, or work product, related to personal representation
or services by attorneys, psychotherapists, or clergy, and their assistants. Such communications and work
product are private and confidential. See User Agreement for details.
----------------------------------------------------------------------------------------------
```

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:

```
-A RH-Firewall-1-INPUT -s 192.168.1.0/24 -m state --state NEW -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -s 202.54.1.5/29 -m state --state NEW -p tcp --dport 22 -j ACCEPT
```

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

```
 -A RH-Firewall-1-INPUT -s ipv6network::/ipv6mask -m tcp -p tcp --dport 22 -j ACCEPT
```

Replace ipv6network::/ipv6mask with actual IPv6 ranges.

*BSD PF Firewall Configuration

If you are using PF firewall update /etc/pf.conf [10] as follows:

```
pass in on $ext_if inet proto tcp from {192.168.1.0/24, 202.54.1.5/29} to $ssh_server_ip port ssh flags S/SA synproxy state
```

#9: Change SSH Port and Limit IP Binding

By default SSH listen to all available interfaces and IP address on the system. Limit ssh port binding and change ssh port (by default brute forcing scripts only try to connects to port # 22). To bind to 192.168.1.5 and 202.54.1.5 IPs and to port 300, add or correct the following line:

```
Port 300
ListenAddress 192.168.1.5
ListenAddress 202.54.1.5
```

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 [11] attack and use john the ripper tool [12] to find out existing weak passwords. Here is a sample random password generator (put in your ~/.bashrc):

```
genpasswd() {
 local l=$1
      [ "$l" == "" ] && l=20
      tr -dc A-Za-z0-9_ < /dev/urandom | head -c ${l} | xargs
}
```

Run it:
genpasswd 16
Output:

uw8CnDVMwC6vOKgW

#11: Use Public Key Based Authentication

Use public/private key pair with password protection for the private key. See how to use RSA [13] and DSA key [14] 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 [15].

#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 [16]. 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 [17] 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 :

```
sshd : 192.168.1.2 172.16.23.12
```

See this FAQ about setting and using TCP wrappers [18] under Linux / Mac OS X and UNIX like operating systems.

#15: Disable Empty Passwords

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

```
PermitEmptyPasswords no
```

#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 prevents brute force attacks against SSH, use the following softwares:

- DenyHosts [19] 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.
- Fail2ban [20] is a similar program that prevents brute force attacks against SSH.
- security/sshguard-pf [21] protect hosts from brute force attacks against ssh and other services using pf.
- security/sshguard-ipfw [21] protect hosts from brute force attacks against ssh and other services using ipfw.
- security/sshguard-ipfilter [21] protect hosts from brute force attacks against ssh and other services using ipfilter.
- security/sshblock [22] block abusive SSH login attempts.
- security/sshit [23] checks for SSH/FTP bruteforce and blocks given IPs.
- BlockHosts [24] Automatic blocking of abusive IP hosts.
- Blacklist [25] Get rid of those bruteforce attempts.
- Brute Force Detection [26] 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 [27] 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:

```
#!/bin/bash
inet_if=eth1
ssh_port=22
$IPT -I INPUT -p tcp --dport ${ssh_port} -i ${inet_if} -m state --state NEW -m recent  --set
$IPT -I INPUT -p tcp --dport ${ssh_port} -i ${inet_if} -m state --state NEW -m recent  --update --seconds 60 --hitcount 5 -j DROP
```

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

```
$IPT -A INPUT  -i ${inet_if} -p tcp --dport ${ssh_port} -m state --state NEW -m limit --limit 3/min --limit-burst 3 -j ACCEPT
$IPT -A INPUT  -i ${inet_if} -p tcp --dport ${ssh_port} -m state --state ESTABLISHED -j ACCEPT
$IPT -A OUTPUT -o ${inet_if} -p tcp --sport ${ssh_port} -m state --state ESTABLISHED -j ACCEPT
# another one line example
# $IPT -A INPUT -i ${inet_if} -m state --state NEW,ESTABLISHED,RELATED -p tcp --dport 22 -m limit --limit 5/minute --limit-burst 5-j ACCEPT
```

See iptables man page for more details.

*BSD PF Example

The following will limits 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.

```
sshd_server_ip="202.54.1.5"
table <abusive_ips> persist
block in quick from <abusive_ips>
pass in on $ext_if proto tcp to $sshd_server_ip port ssh flags S/SA keep state (max-src-conn 20, max-src-conn-rate 15/5, overload <abusive_ips> flush)
```

#18: Use Port Knocking

Port knocking [28] 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:

```
$IPT -N stage1
$IPT -A stage1 -m recent --remove --name knock
$IPT -A stage1 -p tcp --dport 3456 -m recent --set --name knock2

$IPT -N stage2
$IPT -A stage2 -m recent --remove --name knock2
$IPT -A stage2 -p tcp --dport 2345 -m recent --set --name heaven

$IPT -N door
$IPT -A door -m recent --rcheck --seconds 5 --name knock2 -j stage2
$IPT -A door -m recent --rcheck --seconds 5 --name knock -j stage1
$IPT -A door -p tcp --dport 1234 -m recent --set --name knock

$IPT -A INPUT -m --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A INPUT -p tcp --dport 22 -m recent --rcheck --seconds 5 --name heaven -j ACCEPT
$IPT -A INPUT -p tcp --syn -j doo
```

- fwknop [29] is an implementation that combines port knocking and passive OS fingerprinting.
- Multiple-port knocking [30] Netfilter/IPtables only implementation.

#19: Use Log Analyzer

Read your logs using logwatch or logcheck [31]. These tools make your log reading life easier. It will go through your logs for a given period of time and make a report in the areas that you wish with the detail that you wish. Make sure LogLevel is set to INFO or DEBUG in sshd_config:

```
LogLevel INFO
```

#20: Patch OpenSSH and Operating Systems

It is recommended that you use tools such as yum [32], apt-get [33], freebsd-update [34] and others to keep systems up to date with the latest security patches.

Other Options

To hide openssh version, you need to update source code and compile openssh again. Make sure following options are enabled in sshd_config:

```
#  Turn on privilege separation
UsePrivilegeSeparation yes
```

```
# Prevent the use of insecure home directory and key file permissions
StrictModes yes
# Turn on  reverse name checking
VerifyReverseMapping yes
# Do you need port forwarding?
AllowTcpForwarding no
X11Forwarding no
#  Specifies whether password authentication is allowed.  The default is yes.
PasswordAuthentication no
```

Verify your sshd_config file before [35] restarting / reloading changes:

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

Tighter SSH security with two-factor [36] or three-factor (or more) [37] authentication.

References:

1. The official OpenSSH [38] project.
2. Forum thread: Failed SSH login attempts [39] and how to avoid brute ssh attacks
3. man pages sshd_config, ssh_config, tcpd, yum, and apt-get.

If you have a technique or handy software not mentioned here, please share in the comments below to help your fellow readers keep their openssh based server secure.

Download PDF version [40] (193K).

Article printed from nixCraft: **http://www.cyberciti.biz/tips**

URL to article: **http://www.cyberciti.biz/tips/linux-unix-bsd-openssh-server-best-practices.html**

URLs in this post:

[1] Image: **http://www.cyberciti.biz/tips/linux-unix-bsd-openssh-server-best-practices.html/openssh_logo**
[2] rumors: **http://isc.sans.org/diary.html?storyid=6742**
[3] zero day: **http://www.h-online.com/security/OpenSSH-zero-day-exploit-rumours-not-confirmed--/news/113731**
[4] Image: **http://www.cyberciti.biz/tips/linux-unix-bsd-openssh-server-best-practices.html/ssh-session**
[5] restart iptables: **http://www.cyberciti.biz/faq/howto-rhel-linux-open-port-using-iptables/**
[6] configure Linux PAM: **http://www.cyberciti.biz/tips/linux-pam-configuration-that-allows-or-deny-login-via-the-sshd-server.html**
[7] list of group name: **http://www.cyberciti.biz/tips/openssh-deny-or-restrict-access-to-users-and-groups.html**
[8] how to automatically log BASH / TCSH / SSH users: **http://www.cyberciti.biz/faq/linux-unix-login-bash-shell-force-time-outs/**
[9] excellent: **http://archives.neohapsis.com/archives/openbsd/2005-03/2878.html**
[10] /etc/pf.conf: **http://bash.cyberciti.biz/firewall/pf-firewall-script/**
[11] passwords against a dictionary: **http://www.cyberciti.biz/tips/linux-check-passwords-against-a-dictionary-attack.html**
[12] john the ripper tool: **http://www.cyberciti.biz/faq/unix-linux-password-cracking-john-the-ripper/**
[13] RSA: **http://www.cyberciti.biz/tips/ssh-public-key-based-authentication-how-to.html**
[14] DSA key: **http://www.cyberciti.biz/faq/ssh-password-less-login-with-dsa-publickey-authentication/**
[15] keychain software: **http://www.cyberciti.biz/faq/ssh-passwordless-login-with-keychain-for-scripts/**
[16] special tools such as rssh: **http://www.cyberciti.biz/tips/rhel-centos-linux-install-configure-rssh-shell.html**
[17] this blog post: **http://www.debian-administration.org/articles/590**
[18] FAQ about setting and using TCP wrappers: **http://www.cyberciti.biz/faq/tcp-wrappers-hosts-allow-deny-tutorial/**
[19] DenyHosts: **http://www.cyberciti.biz/faq/block-ssh-attacks-with-denyhosts/**
[20] Fail2ban: **http://www.fail2ban.org**
[21] security/sshguard-pf: **http://sshguard.sourceforge.net/**
[22] security/sshblock: **http://www.bsdconsulting.no/tools/**
[23] security/sshit: **http://anp.ath.cx/sshit/**
[24] BlockHosts: **http://www.aczoom.com/cms/blockhosts/**
[25] Blacklist: **http://blinkeye.ch/dokuwiki/doku.php/projects/blacklist**
[26] Brute Force Detection: **http://www.rfxn.com/projects/brute-force-detection/**
[27] IPQ BDB filter: **https://savannah.nongnu.org/projects/ipqbdb/**

[28] Port knocking: **http://en.wikipedia.org/wiki/Port_knocking**
[29] fwknop: **http://www.cipherdyne.org/fwknop/**
[30] Multiple-port knocking: **http://www.debian-administration.org/articles/268**
[31] logcheck: **http://logcheck.org/**
[32] yum: **http://www.cyberciti.biz/faq/rhel-centos-fedora-linux-yum-command-howto/**
[33] apt-get: **http://www.cyberciti.biz/tips/linux-debian-package-management-cheat-sheet.html**
[34] freebsd-update: **http://www.cyberciti.biz/tips/howto-keep-freebsd-system-upto-date.html**
[35] sshd_config file before: **http://www.cyberciti.biz/tips/checking-openssh-sshd-configuration-syntax-errors.html**
[36] two-factor: **http://www.linuxjournal.com/article/8957**
[37] three-factor (or more): **https://calomel.org/openssh.html**
[38] official OpenSSH: **http://www.openssh.com/**
[39] Failed SSH login attempts: **http://nixcraft.com/networking-firewalls-security/726-failed-ssh-login-attempts-how-avoid-brute-ssh-attacks.html**
[40] PDF version: **http://www.cyberciti.biz/files/pdf/nixcraft.tips.5489.pdf?5489**

Click here to print.