By Ryan

Published: 2008-01-16 16:51

SSH: Best PracticesIntroduction

Are you using SSH in the best waypossible? Have you configured it to be as limited and secure aspossible? The goal of this document is to kick in the new year with some best practices for SSH: why you should use them, how to set them up, and how to verify that they are in place.

All of the examples below assume that you are using EnGarde Secure Linux but any modern Linux distribution will do just fine since, as far as I know, everybody ships OpenSSH.

SSHv2 vs. SSHv1

There are numerous benefits to using the latest version of the SSHprotocol, version 2, over it's older counterpart, version 1 and I'm notgoing into a lot of details on those benefits here - if you'reinterested, see the URL in the reference below or Google around. Thatbeing said if you don't have an explicit reason to use the olderversion 1, you should always be using version 2.

To use SSHv2 by default but permit SSHv1, locate the "Protocol" line in your sshd_config file and change it to:

```
Protocol 2,1
```

When doing 2,1 please note that the protocol selection is left up tothe client. Most clients will default to v2 and "fall back" to v1,while legacy clients may continue to use v1. To force everybody to useSSHv2, change it to:

```
Protocol 2
```

When you make this change don't forget to generate the appropriate HostKey's as well! SSHv2 requires the following keys:

```
# HostKeys for protocol version 2
```

```
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
```

While SSHv1 requires:

```
# HostKey for protocol version 1
HostKey /etc/ssh/ssh_host_key
```

Once your changes are made, restart the SSH daemon:

```
# /etc/init.d/sshd restart
```

```
[ SUCCESSFUL ] Secure Shell Daemon
[ SUCCESSFUL ] Secure Shell Daemon
```

From another machine, try SSH'ing in. You can use the `-v option` tosee which protocol is being used, and the '-oProtocol=' option to forceone or the other - for example, "ssh -v -oProtocol=2 " would force protocol version 2.

Binding to a Specific Address or Non-Standard Port

If you're running SSH on an internal, firewalled, workstation thenyou can probably skip this section, but if you're running SSH on afirewall or on a machine with two network interfaces, this section isfor you.

Out of the box OpenSSH will bind to every available network address;while convenient and suitable for most installations, this is far fromoptimal. If your machine has two or more interfaces then the odds arethat one is "trusted" and the other is "untrusted." If this is thecase, and you don't need nor want SSH access coming in on the untrustedinterface, then you should configure OpenSSH to listen on a specificinterface.

To have OpenSSH only bind to your internal interface, 192.168.0.1 inthe example below, locate the following line in your sshd_config file:

```
ListenAddress 0.0.0.0
```

and change the 0.0.0.0 to 192.168.0.1:

```
ListenAddress 192.168.0.1
```

To verify that this change took, restart OpenSSH and look at netstat:

```
# /etc/init.d/sshd restart
```

```
[ SUCCESSFUL ] Secure Shell Daemon
[ SUCCESSFUL ] Secure Shell Daemon
```

```
# netstat -anp | grep sshd
```

```
tcp        0      0 192.168.0.1:22          0.0.0.0:*               LISTEN      7868/sshd
```

As you can see, the sshd daemon is now only listening on192.168.0.1. SSH requests coming in **any other interface** will be ignored.

Similarly, you may want to change the port that the SSH daemon bindsto. Sometimes there is a functional need for this (ie, your employerblocks outbound 22/tcp) but there is also security-through-obscurityvalue in this as well. While not providing any real security benefitagainst a determined attacker, moving the SSH daemon off of port 22protects you against automated attacks which assume that the daemon isrunning on port 22.

To have OpenSSH bind to a port other than port 22, 31337 in theexample below, locate the following line in your sshd_config file:

```
Port 22
```

and change the 22 to 31337:

```
Port 31337
```

To verify that this change took, restart OpenSSH and, again, look at netstat:

```
# netstat -anp | grep sshd
```

```
tcp        0      0 192.168.0.1:31337       0.0.0.0:*               LISTEN      330/sshd
```

Finally, to SSH into a host whose SSH daemon is listening on a non-standard port, use the -p option:

```
ssh -p 31337 user@192.168.0.1
```

Using TCP Wrappers

TCP Wrappers are used to limit access to TCP services on yourmachine. If you haven't heard of TCP Wrappers you've probably heard of/etc/hosts.allow and /etc/hosts.deny: these are the two configurationfiles for TCP Wrappers. In the context of SSH, TCP Wrappers allow youto decide what specific addresses or networks have access to the SSHservice.

To use TCP Wrappers with SSH you need to make sure that OpenSSH wasbuilt with the -with-tcp-wrappers. This is the case on any moderndistribution.

As I indicated earlier, TCP Wrappers are configured by editing the/etc/hosts.deny and /etc/hosts.allow files. Typically you tellhosts.deny to deny everything, then add entries to hosts.allow topermit specific hosts access to specific services.

An example:

```
#
# hosts.deny    This file describes the names of the hosts which are
#               *not* allowed to use the local INET services, as decided
#               by the '/usr/sbin/tcpd' server.
#
ALL: ALL
#
# hosts.allow   This file describes the names of the hosts which are
#               allowed to use the local INET services, as decided
#               by the '/usr/sbin/tcpd' server.
#
sshd: 207.46.236. 198.133.219.25
```

In the example above, access to SSH is limited to the network207.46.236.0/24 and the address 198.133.219.25. Requests to any otherservice from any other address are denied by the "ALL: ALL" inhosts.deny. If you try to SSH into a machine and TCP Wrappers deniesyour access, you'll see something like this:

```
ssh_exchange_identification: Connection closed by remote host
```

This simple configuration change significantly hardens yourinstallation since, with it in place, packets from hostile clients aredropped very early in the TCP session -- and before they can do any real damage to a potentially vulnerable daemon.

Public Key Authentication

The last item I will cover is public key authentication. One of thebest things you can do to tighten the security of your SSH installationis to disable password authentication and to use public keyauthentication instead. Password authentication is suboptimal for manyreasons, but mostly because people choose bad passwords and attackersroutinely try to brute-force passwords. If the systems administratorhas chosen a bad password and he's permitting root logins... game over.

Public key authentication is no silver bullet - similarly, peoplegenerate passphrase-less keys or leave ssh-agents running when theyshouldn't - but, in my opinion, it's a much better bet.

Just about every distribution ships with public key authentication enabled, but begin by making sure it is:

```
RSAAuthentication yes
PubkeyAuthentication yes
```

Both of these options default to "yes" and the "RSAAuthentication"option is for SSHv1 and the "PubkeyAuthentication" option is for SSHv2.If you plan on using this authentication method exclusively, whileyou're there, you may want to disable password authentication:

```
PasswordAuthentication no
```

Before you proceed, make sure you have a terminal open on yourtarget machine. Once you restart the SSH daemon you will no longer beable to log in without a key... which we haven't generated yet!

Once you're sure, restart the SSH daemon:

```
# /etc/init.d/sshd restart
```

```
[ SUCCESSFUL ] Secure Shell Daemon
[ SUCCESSFUL ] Secure Shell Daemon
```

Now, from your desktop, try to SSH in to your target machine:

```
$ ssh rwm@brainy
```

```
Permission denied (publickey,keyboard-interactive).
```

We're locked out! This is a **good** thing. The next step, on your desktop, is to generate a key:

```
$ ssh-keygen -t dsa -C "Ryan's SSHv2 DSA Key (Jan 2008)"
```

```
Generating public/private dsa key pair.
Enter file in which to save the key (/home/rwm/.ssh/id_dsa):
Enter passphrase (empty for no passphrase): **********
Enter same passphrase again: **********
Your identification has been saved in /home/rwm/.ssh/id_dsa.
Your public key has been saved in /home/rwm/.ssh/id_dsa.pub.
The key fingerprint is:
```

```
98:4d:50:ba:ee:8b:79:be:b3:36:75:8a:c2:4a:44:4b Ryan's SSHv2 DSA Key (Jan 2008)
```

A few notes on this:

- You can generate a DSA (-t dsa), RSA (-t rsa), or SSHv1 (-t rsa1) key. In the example above I'm using dsa.
- I like to put the date I generated the key in the comment (-C) field, that way I can change it out every so often.
- You're entering a passphrase, not a password. Use a long stringwith spaces and punctuation. The longer and more complicated the better!

The command you just ran generated two files - id_dsa, your privatekey and id_dsa.pub, your public key. It is critical that you keep yourprivate key private, but you can distribute your public key to anymachines you would like to access.

Now that you have generated your keys we need to get the public keyinto the ~/.ssh/authorized_keys file on the target machine. The bestway to do this is to copy-and-paste it - begin by concatenating thepublic key file:

```
$ cat .ssh/id_dsa.pub
```

```
ssh-dss AAAAB3NzaC1kc3MAAACBAL7p6bsg5kK4ES9BWLPCNABl20iQQB3R0ymaPMHK...
... ds= Ryan's SSHv2 DSA Key (Jan 2008)
```

This is a very long string. Make sure you copy **all of it** and thatyou do NOT copy the newline character at the end. In other words, copyfrom the "ssh" to the "2008)", but not past that.

The next step is to append this key to the end of the~/.ssh/authorized_keys file on your target machine. Remember thatterminal I told you to keep open a few steps ago? Type the followingcommand into it, pasting the key you've just copied into the area notedKEY:

```
echo "KEY"  >> ~/.ssh/authorized_keys
```

For example:

```
echo "ssh-dss AAAA5kS9BWLPCN...s= Ryan's SSHv2 DSA Key (Jan 2008)"  >> ~/.ssh/authorized_keys
```

Now, try to SSH in again. If you did this procedure correctly then instead of being denied access, you'll be prompted for your passphrase:

```
$ ssh rwm@brainy
```

```
Enter passphrase for key '/home/rwm/.ssh/id_dsa':
Last login: Thu Jan 10 14:37:14 2008 from papa.engardelinux.org
[rwm@brainy ~]$
```

Viola! You're now logged in using public key authentication instead of password authentication.

In Summary...

SSH is a wonderful tool and is every systems administrators secondbest friend (Perl, of course, being the first :). It allows you to readyour email from anywhere, provided you still use a terminal-based mailreader. It allows you to tunnel an xterm or X11 application from yourhome server to your desktop at work. It provides you a far superioralternative to FTP in SFTP and SCP.

SSH is great but just like any tool, it's only as good as you useit. I hope that you found value in some of my best practices and if youhave any of your own, leave them in the comments!

Before I go, here are some additional resources on SSH:

- **The OpenSSH Project**
- **SSH, The Secure Shell: The Definitive Guide**
- **Introduction to SSH Versions 1 and 2**
- **Knock, Knock, Knockin' on EnGarde's Door (with FWKNOP)**