Securing OpenSSH Server [Part 1]

Posted by ahowells on Thu 10 Jan 2008 at 10:50

I'm sure if you're responsible for any server connected to the Internet with a "real" IP address, or if you port forward :22 to a box on your LAN, your logs will contain a lot of failed SSH attempts. Here we'll look at some simple solutions.

The logged entries will probably look something like the following:

```
sshd[22768]: Invalid user carol from 91.194.85.65
sshd[22770]: Invalid user carolr from 91.194.85.65
sshd[22772]: Invalid user sergi from 91.194.85.65
sshd[22774]: Invalid user quala from 91.194.85.65
```

Now these are an annoyance, but not much of a concern, as none of these usernames exist on my system - its just filling up my logs, nothing more.

There are other more alarming messages hiding in your logs too, like these:

```
sshd[17810]: Failed password for root from 91.194.85.65 port 44229 ssh2
sshd[17811]: Failed password for root from 91.194.85.65 port 38154 ssh2
sshd[17814]: pam_unix(ssh:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=91.194.85.65  user=r
sshd[17815]: pam_unix(ssh:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=91.194.85.65  user=r
```

Both of these logs come from different physical boxes, with IPs located in totally different /16's and yet it is the same IP address attacking both? Scary! Even more worrying is the fact the attacks are many hours apart, this guy has been hammering hundreds or thousands of servers all day...

What happens if they eventually get lucky? The "best case" scenario is they get an unprivileged account on your system, with which they can:

- Use you as a platform to launch further SSH attacks.
- Add your box to a "botnet" and launch DDoS attacks.
- Attempt to exploit a local vulnerability leading to privilege escalation.
- Send many thousands of spam messages through your server.
- Steal information, delete information.

If they end up with root@yourserver, things get more interesting - they can literally do anything they want with the box. You're screwed. Really.

Every single decision you ever make about these sort of issues will involve balancing usability and accessibility vs. security and safety. There are lots of options available to you, some more "annoying" to real users than others.

1. DenyHosts

   Simply put this is "`apt-get install denyhosts`" and the default configuration is pretty good. It'll send you alert e-mails if you want them and the level at which block compromised IPs with `/etc/hosts.deny` is tunable.

   Pros:

   - User has only a few attempts to get correct password.
   - Susceptability to brute force attack vector reduced lots.
   - Naughty IPs can be removed after a set time, which is helpful if you've blocked yourself accidentally.
   - Very simple to install on your system

   Cons:

   - Both fail2ban and DenyHosts had security vulnerabilities recently which could allow people to DoS your box by adding the whole world to `/etc/hosts.deny`

2. Firewall

   A simple firewall rule can totally drop all traffic to your SSH server if the traffic isn't coming from your IP.

   Pros:

   - Stops dead most attacks, you just aren't there!
   - Fairly trivial to setup. Few dependencies.

Cons:

- Impossible to maintain if your server is being accessed by lots of users from many changing IPs.
- You can't access your box *easily* from friends or whilst you're away on holiday from a cybercafe.
- If you bugger up when setting this up, you could break access to your server completely. :(

3. Strong Passwords

   This is often overlooked, but a real winner. Ensure that all accounts on your system have strong passwords using cracklib.

   - minimum of 8-10 characters advised
   - mixed case, require at least 1-2 capitals
   - numerics, require at least one number
   - throw in a symbol for good measure

   Generating these passwords is easy with pwgen, to make really strong passwords invoke like so: "`pwgen -syB 23`"

   Pros:

   - The simplest option
   - Even a ten or twelve character *strong* password has trillions of possibilities. Very hard to break.

   Cons:

   - none, except having to remember them ;)

4. Public Key Authentication

   Remembering all these strong passwords can be a chore. Often when you're dealing with 2-3+ boxes it's easier to just install the public part of a certificate in `~/.ssh/authorized_keys`.

   Then you may authenticate to the server with your private key.

   Simply put: drop your key in authorized_keys and try to login by doing "`ssh -i key.file user@your.hostname.here`". Once you have this working you can disable other authentication methods.

   Pros:

   - Fairly immune to bruteforce
   - Single sign-on` (same key!) on many servers without loss in security, you can use simpler (but still secure!)
   - Passwords for sudo on your server to get root

   Cons:

   - none really, once you've mastered the basics
   - keeping your key with you?

5. sshd_config

   Using options such as "AllowGroups" to limit the users which are considered valid for logins, regardless of the shell which may be set in /etc/passwd.

   Perhaps run the daemon on a non-standard port so that scans only looking at :22 don't detect you're even running SSH.

Most of the things suggested above are covered in manual pages and other articles on this site, I am planning to write up some more in-depth coverage as part of this series, including coverage of `sshd_config` "Best Practice", cracklib and PAM, authentication using public key, etc.

This is the first article I've written in a long time, hope you enjoyed it.

For reference you might want to take a look at our earlier collection of advice upon Keeping SSH access secure.

This article can be found online at the **Debian Administration** website at the following bookmarkable URL:

- http://www.debian-administration.org/articles/573