

Using tb-sshdfilter to Thwart Automated ssh Attacks

True Blade Systems, Inc.
<http://www.trueblade.com/>

Eric V. Smith
eric@trueblade.com

The Problem

- Many ssh break in attempts, trying to brute force passwords.
- Often using non-existent accounts.
- Occasionally using real accounts like: root, bin, ...
- Typical logwatch output:

sshd:

Authentication Failures:

```
root (200.48.211.2): 2835 Time(s)
root (serv1.mysobe.net): 578 Time(s)
root (vps.webowosso.net): 578 Time(s)
root (61-218-130-20.hinet-ip.hinet.net): 59 Time(s)
root (208-114-60-69.serverpronto.com): 25 Time(s)
root (danticgroup.com): 24 Time(s)
root (dadns.charlotte.america.net): 12 Time(s)
ftp (serv1.mysobe.net): 3 Time(s)
ftp (vps.webowosso.net): 3 Time(s)
root (210.123.133.51): 3 Time(s)
sshd (danticgroup.com): 3 Time(s)
adm (61-218-130-20.hinet-ip.hinet.net): 2 Time(s)
apache (danticgroup.com): 2 Time(s)
bin (danticgroup.com): 2 Time(s)
```

Solution Constraints

- We have to allow ssh into the boxes; there's only so much hardening of ssh that's possible.
- It's not clear which ports ISP's (including hotels, hot spots, etc.) are going to allow through. We know we have not yet seen a problem with port 22.
- It must be reliable. Remote access to machines that are hours away from us is dependent upon this working correctly all the time.

Possible Solutions (1)

- Use MaxAuthTries. Attackers will then only try once per connection; connections are pretty cheap.
- Rate limit ssh connections using netfilter.
- Use an alternate port. While useful, it's security through obscurity and there's the ISP port blocking problem.
- Force public key authentication. This is good, and I might require it, but I'd like to get access to my systems from machines that might not have the key (kiosks).
- Require stronger passwords.
- Port knocking. I've never liked this; it seems error prone and complex. Plus I need to have scripts that talk between machines. Again, the ISP port problem.

Possible Solutions (2)

- Dynamically add lines to `/etc/hosts_deny` to block hosts that are attacking us. Only works for those programs using tcp wrappers library. `sshd_sentry` does this.
- Dynamically add rules to netfilter/iptables to block traffic from hosts that are attacking us. We can either block just sshd traffic, or all traffic. This is our solution.

Options for finding addresses to block

- Monitor log files for bad attempts, such as DenyHosts does. Problems: Log files are huge, tailing has issues, window between attempt and when we scan the log file during which an attack can occur.
- Run sshd with -e (send to stdout, not to syslog), capture and parse the output. This is our solution. The idea is taken from the original sshdfilter.

How tb-sshdfilter works

- Starts up, reads command line parameters and config file.
- Clears netfilter rule – rule must already exist; can specify any criteria you want for blocked traffic.
`/sbin/iptables --table filter --append INPUT --jump SSHD`
- Daemonizes itself.
- Starts sshd with -e option.
- Looks at output of sshd, decides to block if:
 - Unknown userid: immediate block
 - Connect/disconnect: immediate block
 - bad password: block after 3 attempts
- Sends sshd output to syslog.
- Every few minutes checks to see if any rules have expired.

Command line options

usage: tb-sshdfilter [options]

options:

-h, --help show this help message and exit
-pPORT, --port=PORT port to listen on
-d, --daemon become a daemon
-n, --no-daemon do not become a daemon
-rREPURGE_TIME, --repurge-time=REPURGE_TIME
min seconds between purging old rules
-bMAXBLOCK_TIME, --maxblock-time=MAXBLOCK_TIME
seconds to block hostile IP addresses
-cMAXCHANCES, --maxchances=MAXCHANCES
number of invalid password guesses allowed
-sSSHD, --sshd=SSHD full path of sshd
-iIPTABLES, --iptables=IPTABLES
full path of iptables
-RRULE, --rule=RULE name of iptables rule for ssh connections
-CCONFIG, --config=CONFIG
name of config file to read regular
expressions from
-PPIDFILE, --pidfile=PIDFILE
name of file to contain pid

Startup

- Add a new init.d service called tb-sshdfilter, comes with an FC2 init.d script to do this.
- Uses `/etc/sysconfig/tb-sshdfilter` for command line and init.d options.
- Unless you're using the `--port` option, you want to disable the normal `sshd` service.

Possible problems with our solution

- If you typo the username, you're out of luck for a week (from the same host). This is mitigated by the fact that I never type a username. It's either configured in the client (SecureCRT, putty) or it uses the same username as the host you're on (ssh client). However, if you work on a system where your username doesn't match the username on the server, you might want to think about this.
- You can accidentally block all traffic between machines, not just ssh traffic. I did this last week, and it took me a couple of hours to figure out what was happening.

Why write tb-sshdfilter, when sshdfilter exists?

- I don't like how sshdfilter requires an installation script to modify the regular expressions used to parse sshd output. I put them in a separate config file.
- sshdfilter does not support listening on alternate ports.
- I think an entirely new init.d script is needed, so that this can be its own service, not sshdfilter's simple edit to sshd's init.d script. This, in combination with the prior point, allows testing (or actual usage) to have both regular sshd and tb-sshdfilter running at the same time.
- I wanted tb-sshdfilter to daemonize itself, instead of starting with "&" from a shell or script.
- I want to restructure it so the iptables logic is outside of the sshd output parser. I might want to have the rules go into a database, for example.
- The way sshdfilter lists rules is over-specified. It doesn't need to contain the tcp ports again, that could be specified where the rules is originally invoked.
- My python is better than my perl.

Improvements

- Have an option to re-assert the rules. If the underlying table gets blown away, there's no way to reload the rules that `tb-sshdfilter` previously added.
- Have a way to persist the rules so they live across reboots, `tb-sshdfilter` restarts, etc.
- Break into multiple files. In particular `daemonize()` and `Executer()` could go in libraries, but the way it is now makes for easy deployment.

References

- **tb-sshdfilter**
<http://trueblade.com/projects/tb-sshdfilter/tb-sshdfilter-1.0.tar.gz>
- **sshdfilter**
<http://www.csc.liv.ac.uk/~greg/sshdfilter/>
- **DenyHosts**
<http://denyhosts.sourceforge.net>
- **sshd_sentry**
http://linuxmafia.com/pub/linux/security/sshd_sentry/
- **Slashdot discussion of ssh attacks**
<http://it.slashdot.org/it/05/07/16/1615233.shtml?tid=172>