

Lecture 24: Password Protected Systems and the Annoying Dictionary Attack

Lecture Notes on “Computer and Network Security”

by Avi Kak (kak@purdue.edu)

May 5, 2009

©2009 Avinash Kak, Purdue University

Goals:

- The Dictionary Attack
- Thwarting the Dictionary Attack With Log Scanning
- Thwarting the Dictionary Attack With `iptables` Firewall Rules

24.1: The Dictionary Attack

- Scanning blocks of IP addresses for the vulnerabilities at the open ports is in many cases the starting point for breaking into a network.
- If you are not behind a firewall, it is easy to see such ongoing scans. All you have to do is to look at the access or the authorization logs of the services offered by a host in your network. **You will notice that the machines in your network are being constantly scanned for open ports and possible vulnerabilities at those ports.**
- In this lecture I will focus on how people try to break into port 22 that is used for the SSH service. This is a critical service since its use goes way beyond just remote login for terminal sessions. It is also used for secure pickup of email from a mail-drop machine and a variety of other applications.
- The most commonly used ploy to break into port 22 is to mount what is referred as a **dictionary attack** on the port. In a dictionary attack, the bad guys try a large number of common

names as possible account names on the target machine and, should they succeed in stumbling into a name for which there is actually an account on the target machine, they then proceed to try a large number of commonly used passwords for that account.

- If you are logged into a Ubuntu machine, you can see these attempts on an ongoing basis by running the following command line in a separate window

```
tail -f /var/log/auth.log
```

On RedHat systems, the same information is in the `/var/log/secure` file.

- I will now show just a **two minute segment** of this log produced on April 10, 2009 on the host `moonshine.ecn.purdue.edu`. To make it easier to see the user names being tried by the attacker, I have entered a line before each attempt highlighting the name. Note that the third line shown in each record is truncated because of obvious reasons. Nonetheless, you can see all of the relevant information is what is displayed. This scan was mounted from the IP address 61.163.228.117. If you enter this IP address in the query window of `http://www.ip2location.com/`, you will see that the attacker is logged into a network that belongs to the The Postal Information Technology Office in the city of Henan in China.

Account name tried: staff

Apr 10 13:59:59 moonshine sshd[32057]: Invalid user staff from 61.163.228.117
Apr 10 13:59:59 moonshine sshd[32057]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 13:59:59 moonshine sshd[32057]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.163.228.117
Apr 10 14:00:01 moonshine sshd[32057]: Failed password for invalid user staff from 61.163.228.117 port 40805 ssh2

Account name tried: sales

Apr 10 14:00:08 moonshine sshd[32059]: Invalid user sales from 61.163.228.117
Apr 10 14:00:08 moonshine sshd[32059]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:00:08 moonshine sshd[32059]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.163.228.117
Apr 10 14:00:10 moonshine sshd[32059]: Failed password for invalid user sales from 61.163.228.117 port 41066 ssh2

Account name tried: recruit

Apr 10 14:00:17 moonshine sshd[32061]: Invalid user recruit from 61.163.228.117
Apr 10 14:00:17 moonshine sshd[32061]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:00:17 moonshine sshd[32061]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.163.228.117
Apr 10 14:00:19 moonshine sshd[32061]: Failed password for invalid user recruit from 61.163.228.117 port 41303 ssh2

Account name tried: alias

Apr 10 14:00:26 moonshine sshd[32063]: Invalid user alias from 61.163.228.117
Apr 10 14:00:26 moonshine sshd[32063]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:00:26 moonshine sshd[32063]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.163.228.117
Apr 10 14:00:29 moonshine sshd[32063]: Failed password for invalid user alias from 61.163.228.117 port 41539 ssh2

Account name tried: office

Apr 10 14:00:36 moonshine sshd[32065]: Invalid user office from 61.163.228.117
Apr 10 14:00:36 moonshine sshd[32065]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:00:36 moonshine sshd[32065]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.163.228.117
Apr 10 14:00:38 moonshine sshd[32065]: Failed password for invalid user office from 61.163.228.117 port 41783 ssh2

Account name tried: samba

Apr 10 14:00:46 moonshine sshd[32067]: Invalid user samba from 61.163.228.117
Apr 10 14:00:46 moonshine sshd[32067]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:00:46 moonshine sshd[32067]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.163.228.117
Apr 10 14:00:47 moonshine sshd[32067]: Failed password for invalid user samba from 61.163.228.117 port 42027 ssh2

Account name tried: tomcat

Apr 10 14:00:55 moonshine sshd[32069]: Invalid user tomcat from 61.163.228.117
Apr 10 14:00:55 moonshine sshd[32069]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:00:55 moonshine sshd[32069]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.163.228.117
Apr 10 14:00:57 moonshine sshd[32069]: Failed password for invalid user tomcat from 61.163.228.117 port 42247 ssh2

Account name tried: webadmin

Apr 10 14:01:05 moonshine sshd[32071]: Invalid user webadmin from 61.163.228.117
Apr 10 14:01:05 moonshine sshd[32071]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:01:05 moonshine sshd[32071]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=61.163.228.117
Apr 10 14:01:07 moonshine sshd[32071]: Failed password for invalid user webadmin from 61.163.228.117 port 42488 ssh2

Account name tried: spam

```
Apr 10 14:01:14 moonshine sshd[32073]: Invalid user spam from 61.163.228.117
Apr 10 14:01:14 moonshine sshd[32073]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:01:14 moonshine sshd[32073]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 14:01:16 moonshine sshd[32073]: Failed password for invalid user spam from 61.163.228.117 port 42693 ssh2
```

Account name tried: virus

```
Apr 10 14:01:23 moonshine sshd[32075]: Invalid user virus from 61.163.228.117
Apr 10 14:01:23 moonshine sshd[32075]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:01:23 moonshine sshd[32075]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 14:01:25 moonshine sshd[32075]: Failed password for invalid user virus from 61.163.228.117 port 42917 ssh2
```

Account name tried: cyrus

```
Apr 10 14:01:32 moonshine sshd[32077]: Invalid user cyrus from 61.163.228.117
Apr 10 14:01:32 moonshine sshd[32077]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:01:32 moonshine sshd[32077]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 14:01:35 moonshine sshd[32077]: Failed password for invalid user cyrus from 61.163.228.117 port 43144 ssh2
```

Account name tried: oracle

```
Apr 10 14:01:42 moonshine sshd[32079]: Invalid user oracle from 61.163.228.117
Apr 10 14:01:42 moonshine sshd[32079]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:01:42 moonshine sshd[32079]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 14:01:45 moonshine sshd[32079]: Failed password for invalid user oracle from 61.163.228.117 port 43384 ssh2
```

Account name tried: mechael

```
Apr 10 14:01:52 moonshine sshd[32081]: Invalid user michael from 61.163.228.117
Apr 10 14:01:52 moonshine sshd[32081]: pam_unix(sshd:auth): check pass; user unknown
Apr 10 14:01:52 moonshine sshd[32081]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 14:01:54 moonshine sshd[32081]: Failed password for invalid user michael from 61.163.228.117 port 43634 ssh2
....
....
....
```

- In mounting a dictionary attack, the bad guys focus particularly on account names that a target machine could be expect to have with high probability. These include:

```
root
webmaster
webadmin
linux
```

admin
ftp
mysql
oracle
guest
postgres
test
sales
staff
user

and several others

- All of the log entries I showed earlier were for accounts that do not exist on `moonshine.ecn.purdue.edu`. What I show next is a concerted attempt to break into the machine through the `root` account that does exist on the machine. This attack is from the IP address 202.99.32.53. As before, if you enter this IP address in the query window of `http://www.ip2location.com/`, you will see that the attacker is logged into a network that belongs to the CNCGroup Beijing Province Network in Beijing, China. Note that is just a **three minute segment** of the log file.

```
Apr 10 16:23:20 moonshine sshd[32301]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:23:22 moonshine sshd[32301]: Failed password for root from 202.99.32.53 port 42273 ssh2

Apr 10 16:23:29 moonshine sshd[32303]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:23:32 moonshine sshd[32303]: Failed password for root from 202.99.32.53 port 42499 ssh2

Apr 10 16:23:39 moonshine sshd[32305]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
```

Apr 10 16:23:41 moonshine sshd[32305]: Failed password for root from 202.99.32.53 port 42732 ssh2

Apr 10 16:23:48 moonshine sshd[32307]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:23:50 moonshine sshd[32307]: Failed password for root from 202.99.32.53 port 42976 ssh2

Apr 10 16:23:58 moonshine sshd[32309]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:23:59 moonshine sshd[32309]: Failed password for root from 202.99.32.53 port 43208 ssh2

Apr 10 16:24:06 moonshine sshd[32311]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:24:08 moonshine sshd[32311]: Failed password for root from 202.99.32.53 port 43439 ssh2

Apr 10 16:24:15 moonshine sshd[32313]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:24:17 moonshine sshd[32313]: Failed password for root from 202.99.32.53 port 43659 ssh2

Apr 10 16:24:24 moonshine sshd[32315]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:24:26 moonshine sshd[32315]: Failed password for root from 202.99.32.53 port 43901 ssh2

Apr 10 16:24:33 moonshine sshd[32317]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:24:35 moonshine sshd[32317]: Failed password for root from 202.99.32.53 port 44128 ssh2

Apr 10 16:24:42 moonshine sshd[32319]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:24:44 moonshine sshd[32319]: Failed password for root from 202.99.32.53 port 44352 ssh2

Apr 10 16:24:51 moonshine sshd[32321]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:24:53 moonshine sshd[32321]: Failed password for root from 202.99.32.53 port 44577 ssh2

Apr 10 16:25:00 moonshine sshd[32323]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:25:01 moonshine sshd[32323]: Failed password for root from 202.99.32.53 port 44803 ssh2

Apr 10 16:25:09 moonshine sshd[32325]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:25:11 moonshine sshd[32325]: Failed password for root from 202.99.32.53 port 45024 ssh2

Apr 10 16:25:18 moonshine sshd[32327]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:25:20 moonshine sshd[32327]: Failed password for root from 202.99.32.53 port 45269 ssh2

Apr 10 16:25:27 moonshine sshd[32329]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:25:29 moonshine sshd[32329]: Failed password for root from 202.99.32.53 port 45496 ssh2

Apr 10 16:25:36 moonshine sshd[32331]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:25:38 moonshine sshd[32331]: Failed password for root from 202.99.32.53 port 45725 ssh2

Apr 10 16:25:45 moonshine sshd[32333]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:25:47 moonshine sshd[32333]: Failed password for root from 202.99.32.53 port 45951 ssh2

Apr 10 16:25:54 moonshine sshd[32335]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:25:56 moonshine sshd[32335]: Failed password for root from 202.99.32.53 port 46186 ssh2

Apr 10 16:26:03 moonshine sshd[32337]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:26:05 moonshine sshd[32337]: Failed password for root from 202.99.32.53 port 46402 ssh2

Apr 10 16:26:12 moonshine sshd[32339]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:26:14 moonshine sshd[32339]: Failed password for root from 202.99.32.53 port 46637 ssh2

Apr 10 16:26:21 moonshine sshd[32341]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 16:26:23 moonshine sshd[32341]: Failed password for root from 202.99.32.53 port 46859 ssh2

....
....
....

- As long as we are on the subject of looking at the `/var/log/auth.log` log file, in the same file you will also see numerous break-in entries that look like those shown below. These entries contain the special entry **failed - POSSIBLE BREAK-IN ATTEMPT!**. Although such entries look alarming at first sight, they are no more sinister than the examples I showed earlier. What triggers this particular form of log entry is when the local `sshd` daemon cannot reconcile the domain name from where SSH connection request is coming from with the IP address contained in the connection request. Shown below is a small segment of such an attack on `moonshine.ecn.purdue.edu` from the IP address `78.153.210.68`. As before, if you enter this address in the query window of `http://www.ip2location.com/`, you will discover that the attacker is logged into the network that belongs to PEM VPS Hosting Servers in the city of Carlow, Ireland. The attack represents a concerted attempt to break into the `root` account by guessing the password. I have abbreviated the first line of each attempt as indicated by the sequence of dots in such lines. An actual first line of each attempt looks like the following:

```
Apr 10 21:42:45 moonshine sshd[787]: reverse mapping checking \  
  getaddrinfo for 210-68.colo.sta.blacknight.ie [78.153.210.68] \  
  failed - POSSIBLE BREAK-IN ATTEMPT!
```

Here is just a **two minute segment** of such an attack:

```
Apr 10 21:41:58 moonshine sshd[757]: reverse mapping checking ..... [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 10 21:41:58 moonshine sshd[757]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 10 21:41:59 moonshine sshd[757]: Failed password for root from 78.153.210.68 port 43828 ssh2
```

```
Apr 10 21:42:01 moonshine sshd[759]: reverse mapping checking ..... [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
```

Apr 10 21:42:01 moonshine sshd[759]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:02 moonshine sshd[759]: Failed password for root from 78.153.210.68 port 43948 ssh2

Apr 10 21:42:03 moonshine sshd[761]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:04 moonshine sshd[761]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:06 moonshine sshd[761]: Failed password for root from 78.153.210.68 port 44058 ssh2

Apr 10 21:42:08 moonshine sshd[763]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:08 moonshine sshd[763]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:09 moonshine sshd[763]: Failed password for root from 78.153.210.68 port 44210 ssh2

Apr 10 21:42:11 moonshine sshd[765]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:11 moonshine sshd[765]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:12 moonshine sshd[765]: Failed password for root from 78.153.210.68 port 44330 ssh2

Apr 10 21:42:14 moonshine sshd[767]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:14 moonshine sshd[767]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:16 moonshine sshd[767]: Failed password for root from 78.153.210.68 port 44440 ssh2

Apr 10 21:42:17 moonshine sshd[769]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:17 moonshine sshd[769]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:19 moonshine sshd[769]: Failed password for root from 78.153.210.68 port 44568 ssh2

Apr 10 21:42:20 moonshine sshd[771]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:20 moonshine sshd[771]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:22 moonshine sshd[771]: Failed password for root from 78.153.210.68 port 44698 ssh2

Apr 10 21:42:23 moonshine sshd[773]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:23 moonshine sshd[773]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:25 moonshine sshd[773]: Failed password for root from 78.153.210.68 port 44818 ssh2

Apr 10 21:42:27 moonshine sshd[775]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:27 moonshine sshd[775]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:29 moonshine sshd[775]: Failed password for root from 78.153.210.68 port 44928 ssh2

Apr 10 21:42:30 moonshine sshd[777]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:30 moonshine sshd[777]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:32 moonshine sshd[777]: Failed password for root from 78.153.210.68 port 45089 ssh2

Apr 10 21:42:33 moonshine sshd[779]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:33 moonshine sshd[779]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:34 moonshine sshd[779]: Failed password for root from 78.153.210.68 port 45186 ssh2

Apr 10 21:42:36 moonshine sshd[781]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:36 moonshine sshd[781]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:37 moonshine sshd[781]: Failed password for root from 78.153.210.68 port 45299 ssh2

Apr 10 21:42:38 moonshine sshd[783]: reverse mapping checking [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:38 moonshine sshd[783]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:40 moonshine sshd[783]: Failed password for root from 78.153.210.68 port 45405 ssh2

```
Apr 10 21:42:41 moonshine sshd[785]: reverse mapping checking ..... [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:41 moonshine sshd[785]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:43 moonshine sshd[785]: Failed password for root from 78.153.210.68 port 45521 ssh2

Apr 10 21:42:45 moonshine sshd[787]: reverse mapping checking ..... [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:45 moonshine sshd[787]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:47 moonshine sshd[787]: Failed password for root from 78.153.210.68 port 45663 ssh2

Apr 10 21:42:48 moonshine sshd[789]: reverse mapping checking ..... [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:48 moonshine sshd[789]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:49 moonshine sshd[789]: Failed password for root from 78.153.210.68 port 45778 ssh2

Apr 10 21:42:51 moonshine sshd[791]: reverse mapping checking ..... [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:51 moonshine sshd[791]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:53 moonshine sshd[791]: Failed password for root from 78.153.210.68 port 45882 ssh2

Apr 10 21:42:54 moonshine sshd[793]: reverse mapping checking ..... [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:54 moonshine sshd[793]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:55 moonshine sshd[793]: Failed password for root from 78.153.210.68 port 46011 ssh2

Apr 10 21:42:57 moonshine sshd[795]: reverse mapping checking ..... [78.153.210.68] failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 21:42:57 moonshine sshd[795]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
Apr 10 21:42:58 moonshine sshd[795]: Failed password for root from 78.153.210.68 port 46123 ssh2
....
....
....
```

24.2: The Password File Embedded in the Conficker Worm

- When an attacker who has mounted a dictionary attack does find an installed account on the victim machine, the next challenge for the attacker is to gain entry into the account by making guesses at the password for the account. For example, the last two segments of the `auth.log` file shown in the previous section are for two concerted attempts by two different attackers to guess the password for the `root` account on `moonshine.ecn.purdue.edu`.
- In the context of guessing the passwords, it is interesting to examine the guesses that are embedded in the binary for the Conficker worm that we discussed in Lecture 22. Here are the 240 guesses that were taken from `http://onecare.live.com/standard/en-us/virusenc/virusencinfo.htm?VirusName=Worm:Win32/Conficker.B`

```
123
1234
12345
123456
1234567
12345678
123456789
1234567890
123123
12321
123321
123abc
123qwe
123asd
```

1234abcd
1234qwer
1q2w3e
a1b2c3
admin
Admin
administrator
nimda
qwewq
qweewq
qwerty
qweasd
asdsa
asddsa
asdzxc
asdfgh
qweasdzxc
q1w2e3
qazwsx
qazwsxedc
zxcxz
zxccxz
zxcvb
zxcvbn
passwd
password
Password
login
Login
pass
mypass
mypassword
adminadmin
root
rootroot
test
testtest
temp
temptemp
foofoo
foobar
default
password1
password12
password123
admin1
admin12
admin123
pass1
pass12
pass123
root123
pw123
abc123
qwe123

test123
temp123
mypc123
home123
work123
boss123
love123
sample
example
internet
Internet
nopass
nopassword
nothing
ihavenopass
temporary
manager
business
oracle
lotus
database
backup
owner
computer
server
secret
super
share
superuser
supervisor
office
shadow
system
public
secure
security
desktop
changeme
codename
codeword
nobody
cluster
customer
exchange
explorer
campus
money
access
domain
letmein
letitbe
anything
unknown
monitor
windows

files
academia
account
student
freedom
forever
cookie
coffee
market
private
games
killer
controller
intranet
work
home
job
foo
web
file
sql
aaa
aaaa
aaaaa
qqq
qqqq
qqqqq
xxx
xxxx
xxxxx
zzz
zzzz
zzzzz
fuck
12
21
321
4321
54321
654321
7654321
87654321
987654321
0987654321
0
00
000
0000
00000
00000
000000
0000000
1
11
111

1111
11111
111111
1111111
11111111
2
22
222
2222
22222
222222
2222222
22222222
3
33
333
3333
33333
333333
3333333
33333333
4
44
444
4444
44444
444444
4444444
44444444
444444444
5
55
555
5555
55555
555555
5555555
55555555
555555555
6
66
666
6666
66666
666666
6666666
66666666
7
77
777
7777
77777
777777
7777777
77777777
8
88

888
8888
88888
888888
8888888
88888888
9
99
999
9999
99999
999999
9999999
99999999

24.3: Thwarting the Dictionary Attack With Log Scanning

- Before getting to the subject of log scanning for protecting a computer/network against a dictionary attack, I should say quickly that if, say, the computer you want to protect is at your home and you want to be able to SSH into it from work without allowing others to be able to do the same, just a couple of entries in the `/etc/hosts.allow` and the `/etc/hosts.deny` files would keep all intruders at bay.

```
/etc/hosts.allow    :    sshd: xxx.xxx.xxx.xxx
```

```
/etc/hosts.deny    :    ALL: ALL
```

where `xxx.xxx.xxx.xxx` is the IP address from where you wish to connect to your home machine. Since `/etc/hosts.allow` takes precedence over `/etc/hosts.deny`, the above two entries will ensure that only you will be allowed SSH access into the machine.

- Let's now consider a more general situation of detecting repeated break-in attempts and temporarily (or, sometimes, permanently) blacklisting IP addresses from where the attacks are emanating.

- DenyHosts by Phil Schwartz is one of the best tools out there that keeps an eye on the `sshd` server logs (in `/var/log/auth.log` on Ubuntu machines) and takes immediate protective action if it sees signs of repeated unsuccessful attempts to break into the machine through the SSH service.
- You may think there is a bit of irony in making decisions on the basis of past “unsuccessful” attempts to break into a machine. If an intruder successfully logged in as root, the first thing he/she would probably do would be to eliminate all signs of his/her intrusion. So your security decisions will be based more on the actions of clumsy thieves who are unsuccessful and not on the actions of those who may have caused you serious harm in the past.
- But since it is reasonable to assume that even a successful thief would need to make a few attempts before hitting the jackpot, it makes eminent sense to use a tool like DenyHosts.
- The basic operation of DenyHosts is that if it sees a repeated unsuccessful attempt from an IP address, it places that IP address in the `/etc/hosts.deny` file. Subsequently, no further SSH connection from the same IP address would be honored. As to how many attempts at breaking in should qualify for blacklisting an IP address can be set by you in the configuration file of

DenyHosts.

- Thanks to Phil Schwartz, it is extremely easy to configure Deny-Hosts. The start/stop script and the configuration files are in

```
/usr/share/denyhosts/
```

The main config file is

```
/usr/share/denyhosts/denyhosts.cfg
```

The `README.txt` in the main installation directory tells you about the configuration file and a file called `daemon_control` that is used to start/stop the tool.

- To start, stop, and restart DenyHosts (you must restart Deny-Hosts if you change the config file), execute the following as root:

```
/usr/share/denyhosts/daemon-control  start
                                       stop
                                       restart
```

- DenyHosts makes its log entries in the file

```
/var/log/denyhosts
```

- What makes DenyHosts particularly powerful is its synchronization feature that allows it to download the IP addresses that have been blacklisted elsewhere. In that sense, the tool has the ability to give you advance protection.
- DenyHosts can also silently restore access privileges of a blacklisted IP address after a certain period of time whose duration is set in the configuration file.
- The homepage for DenyHosts is

`http://denyhosts.sourceforge.net/`

For Ubuntu specific information on the tool, visit

`http://ubuntuforums.org/showthread.php?t=450853`

- To show DenyHosts in action, here is a **45 second** segment of the `auth.log` file after DenyHosts was fired up. This represents an illegal attempt to break into `moonshine.ecn.purdue.edu` from someone at 190.12.41.50. If you enter this IP address in the query window of `http://www.ip2location.com`, you will discover that the intruder is logged into a network owned by an outfit called PUNTONET in the country of Ecuador.

tried to connect as root:

```
Apr 25 16:29:03 moonshine sshd[31037]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:03 moonshine sshd[31037]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:04 moonshine sshd[31037]: Failed password for root from 190.12.41.50 port 54042 ssh2
```

tried to connect as apple:

```
Apr 25 16:29:08 moonshine sshd[31039]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:08 moonshine sshd[31039]: Invalid user apple from 190.12.41.50  
Apr 25 16:29:08 moonshine sshd[31039]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:10 moonshine sshd[31039]: Failed password for invalid user apple from 190.12.41.50 port 54102 ssh2
```

tried to connect as magazine:

```
Apr 25 16:29:13 moonshine sshd[31041]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:13 moonshine sshd[31041]: Invalid user magazine from 190.12.41.50  
Apr 25 16:29:13 moonshine sshd[31041]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:15 moonshine sshd[31041]: Failed password for invalid user magazine from 190.12.41.50 port 54163 ssh2
```

tried to connect as sophia:

```
Apr 25 16:29:18 moonshine sshd[31043]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:18 moonshine sshd[31043]: Invalid user sophia from 190.12.41.50  
Apr 25 16:29:18 moonshine sshd[31043]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:20 moonshine sshd[31043]: Failed password for invalid user sophia from 190.12.41.50 port 54227 ssh2
```

tried to connect as janet:

```
Apr 25 16:29:23 moonshine sshd[31045]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:23 moonshine sshd[31045]: Invalid user janet from 190.12.41.50  
Apr 25 16:29:23 moonshine sshd[31045]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:25 moonshine sshd[31045]: Failed password for invalid user janet from 190.12.41.50 port 54289 ssh2
```

tried to connect as taylor:

```
Apr 25 16:29:28 moonshine sshd[31047]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:28 moonshine sshd[31047]: Invalid user taylor from 190.12.41.50  
Apr 25 16:29:28 moonshine sshd[31047]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:30 moonshine sshd[31047]: Failed password for invalid user taylor from 190.12.41.50 port 54351 ssh2
```

tried to connect as vanessa:

```
Apr 25 16:29:33 moonshine sshd[31049]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:33 moonshine sshd[31049]: Invalid user vanessa from 190.12.41.50  
Apr 25 16:29:33 moonshine sshd[31049]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:34 moonshine sshd[31049]: Failed password for invalid user vanessa from 190.12.41.50 port 54406 ssh2
```

tried to connect as alyson:

```
Apr 25 16:29:38 moonshine sshd[31051]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:38 moonshine sshd[31051]: Invalid user alyson from 190.12.41.50  
Apr 25 16:29:38 moonshine sshd[31051]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:39 moonshine sshd[31051]: Failed password for invalid user alyson from 190.12.41.50 port 54467 ssh2
```

tried again to connect as root:

```
Apr 25 16:29:42 moonshine sshd[31053]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:42 moonshine sshd[31053]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:44 moonshine sshd[31053]: Failed password for root from 190.12.41.50 port 54509 ssh2
```

tried again to connect as research:

```
Apr 25 16:29:48 moonshine sshd[31055]: reverse mapping .... [190.12.41.50] failed - POSSIBLE BREAK-IN ATTEMPT!  
Apr 25 16:29:48 moonshine sshd[31055]: Invalid user research from 190.12.41.50  
Apr 25 16:29:48 moonshine sshd[31055]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=  
Apr 25 16:29:50 moonshine sshd[31055]: Failed password for invalid user research from 190.12.41.50 port 54581 ssh2
```

AND FINALLY CAUGHT BY DENYHOSTS:

```
Apr 25 16:29:50 moonshine sshd[31060]: refused connect from ::ffff:190.12.41.50 (::ffff:190.12.41.50)
```

- So the intruder made 10 attempts before getting trapped by DenyHosts. **How many attempts an intruder is allowed to try before any further connection requests are summarily refused depends on the choices you make in the file denyhosts.cfg in the /usr/share/denyhosts/ directory.** I had the following setting in the config file for the log file segment shown above:

```
DENY_THRESHOLD_INVALID = 5  
DENY_THRESHOLD_VALID   = 10
```

where the first number sets the limit on how many times an intruder can try to gain entry with account names that do NOT exist in the `/etc/passwd` file and the second sets a similar limit on trying to gain entry through account names that actually do exist.

- Obviously, what values you choose for the two parameters shown above and other similar parameters in the config file depends on how much latitude you want to give the legitimate users of your host with regarding to any accidental mis-entry of user names and passwords.
- What I show next is an attack by a cleverer intruder. **What this intruder is attempting is not your classic dictionary attack. The intruder appears to know that he/she will be allowed only a limited number of attempts (probably from a prior manual attempt to break in with a number of different login names from conceivably a different IP address). So the intruder is trying only the login names that form the various substrings in the domain name of “moonshine.ecn.purdue.edu”.** **Note that the intruder is making only 4 attempts for each login name, one less than it takes to get disbarred by the config settings shown previously.** To see the source of the attack, enter the IP address 66.135.39.212 in the query window of <http://www.ip2location.com> and you will notice that this address belongs to a company called Zartana based in Brazil. In its description at LinkedIn, this company claims to be able to deliver 2,000,000 email messages per hour.

login tried: ecn (Attempt 1 as ecn)

```
May 5 10:11:23 moonshine sshd[27483]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - PO
May 5 10:11:23 moonshine sshd[27483]: Invalid user ecn from 66.135.39.212
May 5 10:11:23 moonshine sshd[27483]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:23 moonshine sshd[27483]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
May 5 10:11:25 moonshine sshd[27483]: Failed password for invalid user ecn from 66.135.39.212 port 33901 ssh2
```

login tried: ecn (Attempt 2 as ecn)

```
May 5 10:11:25 moonshine sshd[27485]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - PO
May 5 10:11:25 moonshine sshd[27485]: Invalid user ecn from 66.135.39.212
May 5 10:11:25 moonshine sshd[27485]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:25 moonshine sshd[27485]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
May 5 10:11:28 moonshine sshd[27485]: Failed password for invalid user ecn from 66.135.39.212 port 34028 ssh2
```

login tried: ecn (Attempt 3 as ecn)

```
May 5 10:11:29 moonshine sshd[27487]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - PO
May 5 10:11:29 moonshine sshd[27487]: Invalid user ecn from 66.135.39.212
May 5 10:11:29 moonshine sshd[27487]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:29 moonshine sshd[27487]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
May 5 10:11:31 moonshine sshd[27487]: Failed password for invalid user ecn from 66.135.39.212 port 34163 ssh2
```

login tried: ecn (Attempt 4 as ecn)

```
May 5 10:11:32 moonshine sshd[27489]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - PO
May 5 10:11:32 moonshine sshd[27489]: Invalid user ecn from 66.135.39.212
May 5 10:11:32 moonshine sshd[27489]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:32 moonshine sshd[27489]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
May 5 10:11:34 moonshine sshd[27489]: Failed password for invalid user ecn from 66.135.39.212 port 34282 ssh2
```

login tried: moonshine (Attempt 1 as moonshine)

```
May 5 10:11:35 moonshine sshd[27491]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - PO
May 5 10:11:35 moonshine sshd[27491]: Invalid user moonshine from 66.135.39.212
May 5 10:11:35 moonshine sshd[27491]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:35 moonshine sshd[27491]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
May 5 10:11:37 moonshine sshd[27491]: Failed password for invalid user moonshine from 66.135.39.212 port 34384 ssh2
```

login tried: moonshine (Attempt 2 as moonshine)

```
May 5 10:11:37 moonshine sshd[27493]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - PO
May 5 10:11:37 moonshine sshd[27493]: Invalid user moonshine from 66.135.39.212
May 5 10:11:37 moonshine sshd[27493]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:37 moonshine sshd[27493]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
May 5 10:11:40 moonshine sshd[27493]: Failed password for invalid user moonshine from 66.135.39.212 port 34514 ssh2
```

login tried: moonshine (Attempt 3 as moonshine)

```
May 5 10:11:41 moonshine sshd[27495]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - PO
May 5 10:11:41 moonshine sshd[27495]: Invalid user moonshine from 66.135.39.212
May 5 10:11:41 moonshine sshd[27495]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:41 moonshine sshd[27495]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=
May 5 10:11:43 moonshine sshd[27495]: Failed password for invalid user moonshine from 66.135.39.212 port 34637 ssh2
```

login tried: moonshine (Attempt 4 as moonshine)

```
May 5 10:11:43 moonshine sshd[27497]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - P
May 5 10:11:43 moonshine sshd[27497]: Invalid user moonshine from 66.135.39.212
May 5 10:11:43 moonshine sshd[27497]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:43 moonshine sshd[27497]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhos
May 5 10:11:46 moonshine sshd[27497]: Failed password for invalid user moonshine from 66.135.39.212 port 34759 ssh2
```

login tried: purdue (Attempt 1 as purdue)

```
May 5 10:11:47 moonshine sshd[27499]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - P
May 5 10:11:47 moonshine sshd[27499]: Invalid user purdue from 66.135.39.212
May 5 10:11:47 moonshine sshd[27499]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:47 moonshine sshd[27499]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhos
May 5 10:11:49 moonshine sshd[27499]: Failed password for invalid user purdue from 66.135.39.212 port 34906 ssh2
```

login tried: purdue (Attempt 2 as purdue)

```
May 5 10:11:49 moonshine sshd[27501]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - P
May 5 10:11:49 moonshine sshd[27501]: Invalid user purdue from 66.135.39.212
May 5 10:11:49 moonshine sshd[27501]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:49 moonshine sshd[27501]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhos
May 5 10:11:52 moonshine sshd[27501]: Failed password for invalid user purdue from 66.135.39.212 port 35030 ssh2
```

login tried: purdue (Attempt 3 as purdue)

```
May 5 10:11:52 moonshine sshd[27503]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - P
May 5 10:11:52 moonshine sshd[27503]: Invalid user purdue from 66.135.39.212
May 5 10:11:52 moonshine sshd[27503]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:52 moonshine sshd[27503]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhos
May 5 10:11:54 moonshine sshd[27503]: Failed password for invalid user purdue from 66.135.39.212 port 35189 ssh2
```

login tried: purdue (Attempt 4 as purdue)

```
May 5 10:11:55 moonshine sshd[27505]: reverse mapping checking getaddrinfo for server2.tusom.org [66.135.39.212] failed - P
May 5 10:11:55 moonshine sshd[27505]: Invalid user purdue from 66.135.39.212
May 5 10:11:55 moonshine sshd[27505]: pam_unix(sshd:auth): check pass; user unknown
May 5 10:11:55 moonshine sshd[27505]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhos
May 5 10:11:58 moonshine sshd[27505]: Failed password for invalid user purdue from 66.135.39.212 port 35321 ssh2
```

FINALLY TRAPPED BY DENYHOSTS

24.4: Thwarting the Dictionary Attack With iptables Firewall Rules

- It is also possible to thwart dictionary attacks with appropriate packet-level firewall rules. Here is a rule set by Rainer Krienke:

```
#!/bin/sh

# These iptables rules limit SSH access from any single IP address
# to five in any 25 second period.

INTERFACE="eth0"
EXTERNAL_NET="192.198.1.0/24"
PERIOD_EXTERNAL="25"
TRIES_EXTERNAL="5"

# Limit sshd access to connection requests from EXTERNAL_NET:
iptables -A INPUT -p tcp --dport 22 -i $INTERFACE --source $EXTERNAL_NET -m state \
    --state NEW -m recent --name EXTERNAL --set

# Log scan attack
iptables -A INPUT -p tcp --dport 22 -i $INTERFACE --source $EXTERNAL_NET -m state \
    --state NEW -m recent --name EXTERNAL --update --seconds $PERIOD_EXTERNAL \
    --hitcount $TRIES_EXTERNAL -j LOG --log-level warning --log-tcp-options \
    --log-ip-options --log-prefix "SF2-DELAY-ssh-scan "

# Limit the number of ssh connection attempts from one IP:
iptables -A INPUT -p tcp --dport 22 -i $INTERFACE --source $EXTERNAL_NET -m state \
    --state NEW -m recent --name EXTERNAL --update --seconds $PERIOD_EXTERNAL \
    --hitcount $TRIES_EXTERNAL -j DROP
```

See Lecture 18 for the syntax of **iptables** rules and how to place them in a shell executable file.