CSCI 400 Security Lab
Topic: Firewalls

Prof. Dietrich

October 11, 2017

Group size: 4 — Setup needed: Windows/MacOS X, Unix hosts, DETER.

1 General description

In this lab, we are building a firewall, using several approaches under Linux, FreeBSD, and possibly OpenBSD. This is partially based on a lab by Peter Reiher.

2 File with samples and results

The files are in lab-fw.zip. The page contains pointers to configurations (basic) and examples for setting up simple firewalls. You will have to develop more complex rule sets for the tasks.

3 Setting up a firewall

In this part of the lab, we will be setting up rule sets for hosts and networks. The purpose of this exercise is to introduce you to network access control schemes and the "principle of least privilege" through the use of iptables firewalls.

After this exercise, you will:

- understand the basics of stateful firewalls
- be able to apply that knowledge to configure a basic firewall in Linux using iptables.

3.1 Requirement

- You use the instructions provided with this lab. You should use DETER, as we will continue using the setup from the IDS lab.
- Your output should demonstrate that you have done the experiments, such as screenshots, Unix typescripts, and the firewall logs showing that packets were allowed across the perimeter, or that they were blocked.

4 Tasks

Half your team has been assigned to a Skunk Works Security Team within FrobozzCo, now working separately from the rest of the company. Your mission is to operate separately from the rest, to avoid previous catastrophes.

The other half of your team is working on replacing the previous network administrator "Just Accept" aka "accept all traffic." He liked to make things work, but it was when he changed the FrobozzCo firewall policy to `-j ACCEPT` and the internal LAN got exposed to some black hat zero-days that he signed his own pink slip. You must create an answer key for the hiring exam, just as you did for the security administrator and permissions.

4.1 Team 1: Adding a firewall to the previous network design

Reusing the previous experiment/lab, place a separate firewall host in front of your production servers running the web server on port 80, the IMAP server on port 143, and the NFS server. It is assumed that those hosts have ssh servers running on port 22 as well. No further restrictions should be placed on the server program configurations: all work with rules should be performed on the firewall you are creating.

4.1.1 iptables

1. In an Ubuntu (version 12.04, 14.04 or later preferred) environment, make sure `iptables` is installed. Call this host `firewall1`.
2. Install a permissive firewall setup. Scan your network from outside the firewall using `nmap`.
3. Establish rules to enact the following restrictions on the outside interface:
   - passively ignore any traffic inbound to the interface that says it's coming from the server itself (obvious spoof attempt).
   - allow new inbound TCP connections to the ssh, web, and NFS servers on their standard ports. The NFS server should only accept connections from a host named `client`. What other rules are necessary to make NFS work?
   - allow inbound and outbound ICMP ping requests and replies.
   - allow new outbound TCP connections to ssh and web servers.
   - passively ignore all other traffic. (Do not allow it or respond to it in any way.)
4. Again, scan your network from outside the firewall using `nmap`.

4.1.2 pf

1. In a FreeBSD environment, configure and enable `pf`. Note that on a multi-interface host, you must enable IP packet forwarding at the kernel level using `sysctl`. Call this host `firewall2`.
2. Repeat the experiment as in the previous section on iptables.

4.2 Team 2: Creating the hiring exam

4.2.1 FrobozzCo Firewall Test

This test is a part of the application process for the administrator position at FrobozzCo. You will be presented with a number of tasks and questions; the tasks must be executed on the server node of the DETER experiment we have created for this purpose (you can use the `client` node for testing). There are no short answer questions for this lab. See below for further instructions.

Setup

1. Log into DETER.
2. Create an instance of this exercise by following the instructions here, using

   `/share/education/PermissionsFirewalls_UCLA/permissions.ns`

   as your NS File.
3. In the "Idle-Swap" field, enter "1". This tells DETER to swap your experiment out if it is idle for more than one hour.
4. In the "Max. Duration" field, enter "6". This tells DETER to swap the experiment out after six hours.
5. Swap in your new lab.
6. After the experiment has finished swapping in, log in to the node via ssh.

As mentioned in DETER Instructions, changes to DETER nodes are lost when the nodes are swapped out. This means that you must manually save your work between working on nodes in order to keep it. However, this exercise includes experimental scripts to help you save and restore your work.

Saving your Work After completing the firewall-related exercises on the `server` node, cd into the `/root` directory and execute the script `/root/submit.sh`; like this:

```
$ ./submit.sh
```

...it will make a tarball of all the relevant files, including the firewall.sh script you have updated (it will also copy files for the Access Control/Permissions lab, but you do not need to bother with them; you should ignore the errors). You must copy or move the created tarball into your group directory, otherwise it will be lost upon swapout.

Restoring Your Work The experimental script `/root/restore.sh` on the server node, takes as input the path to a tarball created by submit.sh described above and restores the files to their proper locations on the system. This includes all the files you are asked to create or change in the other lab, as well as the `firewall.sh` script for this lab.

WARNING: These scripts do not back up all arbitrary changes you may have made to the node (e.g., changing a peripheral configuration file), and it does not "merge" system files with submission files – it only restores submission files copied by `ubmit.sh`. You shouldn't need to change anything else, but see `submit.sh` and `restore.sh` to see exactly what those scripts do, and do not delete any of your submission tarballs so that you can go back to an earlier version if need be.

To use the `restore.sh`, copy a tarball created by `submit.sh` into the `/root/` directory and execute this command:

```
$ ./restore.sh username-permissions-123123123.tar.gz
```

You will be asked if you want to automatically overwrite files, and if you want to selectively restore some files and not others. The options are self-explanatory.

Finally, if you don't trust the scripts, you can always make your own backups into your group directory and restore them by hand if you prefer.

NOTE: You do not need to run the submit and restore scripts with sudo. However, if you use sudo to run `submit.sh`, your tarball will be named after the root user. This is OK – just run `sudo ./restore.sh root-permissions-2134234243.tar.gz`.

Firewall Configuration The test server has a totally permissive firewall installed – it accepts all inbound and outbound traffic from all ports, protocols, addresses, interfaces, and states. This is basically like having no firewall at all.

Your task is to configure the firewall according to the principle of "least privilege". This means that it should be maximally restrictive while still permissive enough to allow a strictly defined set of tasks. While some of these rules can also be configured in the server software (this strategy is called *defense in depth*), we want you to implement the rules in `iptables` only – do not reconfigure the underlying software.

The firewall has been copied into the directory /root/firewall/ along with a script called extingui.sh to "put out the fire" and clear all the rules in case you make a mistake. The firewall is not enabled by default – to enforce the rules, execute:

```
/root/firewall/firewall.sh
```

... as the root user or using sudo:

```
sudo /root/firewall/firewall.sh
```

This will load the rules and start enforcing them. To make sure that you are removing all `iptables` rules, you should run `extingui.sh` in between every invocation of `firewall.sh` or rules might "stick around" which can be very confusing if you are trying to debug the system. This can be done like this as `root` (or with `sudo` as above):

```
/root/firewall/extingui.sh
```

If you make the server inaccessible with broken rules, don't worry – you can reboot the node in the DETER console, and since the firewall is not enabled by default, you can log in in order to fix it. Your files will still be on the experimental node as long as you don't swap out the experiment. (Of course, you can permanently save your files in your home directory.)

Finally, only your final product is evaluated – not the number of times you have to reboot the server. You should expect to lock yourself out a few times.

Your experimental nodes have at least two networks. The first is a control network between your node and `users.deterlab.net`. This is the network you use to connect to your nodes from users. Your nodes also have an ”experimental” network that connects all the machines in a given experiment. For this project, your experimental network connects server and client

The firewall only needs to limit the experimental network interface (the interface with the 10.1.x.x network) and should not ever limit the control network (192.x.x.x as of this writing) or you may cut yourself off from the node. The experimental interface is one of eth0-ethN and you can determine which using the command ifconfig and looking for the 10.x.x.x interface. Be warned that the specific interface used may change with a reboot or different experimental node.

See below for instructions on using environment variables to define the experimental interface in your `firewall.sh` script.

Here's what the firewall needs to do:

1. passively ignore any traffic inbound to the interface that says it's coming from the server itself (obvious spoof attempt). The server uses the localhost loopback device lo for internal traffic, so it should never see incoming traffic from its own IP on the experimental network interface. (See test case 11, below)

2. Allow all established traffic on the experimental network interface. Established or related traffic is traffic that is part of previously accepted new connections.

3. Accept new connections on the experimental network (10.1.x.x) of the types listed below:

   (a) Inbound TCP connections to the OpenSSH, Apache, and MySQL servers on their standard ports. (Test cases 1, 3, 5)

   (b) The MySQL server should only accept connections from the client host.

   (c) Inbound UDP connections to the server ports 10000 to 10005 from the host client. (Test case 8)

   (d) Inbound ICMP ping requests. (Test case 6)

   (e) Outbound ICMP ping replies. (Test case 7)

   (f) Outbound TCP connections to any OpenSSH, SMTP, and Apache (on standard ports). (Test cases 2, 4)

   (g) Outbound UDP connections to the ports 10006 to 10010 on host client from the server. (Test case 9)

4. passively ignore all other traffic. (Do not allow it or respond to it in any way.) (Test case 10)

There are many online resources and tutorials for `iptables` configuration – feel free to use them. Be aware, however, not all tutorials emphasize the principle of least privilege and may give you overly permissive advice! In order to properly configure the firewall, you must consider the basic ways the firewall can differentiate traffic and allow only the specific types you require to properly function.

Please read the helpful advice in the lab file for configuring and testing your firewall.

4.3 Meeting of the teams

After completing the two team tasks above, have the two teams meet. Discuss and summarize the successes and challenges you encountered in both teams and include that in your report.

5 Deliverables

1. A report describing all your findings above.
2. A zip file containing:
   - Any rule sets developed and logs (text only) recorded.
   - The file created by the submit.sh script.
3. Submit by the class on October 18, 2017.

6 Grading

Points will be subtracted if any of the pieces of the deliverables are missing or incomplete.