Project 2: Firewall Design (Phase I)

CS 161 - Joseph/Tygar

November 12, 2006

1 Edits

If we need to make clarifications or corrections to this document after distributing it, we will post a new version to the class website and we will announce the new version on the class newsgroup. Please be patient, as this is a brand-new project that we're running for the first time.

11/12: Updated description of writeup required for Phase I submission. Added section describing grading policy.

11/8: Clarified that the time server and echo server are the same machine.

11/6: Clarified TCP vs. UDP for ports listed in functionality requirements. Changed wording from "Security requirements" to "Security goals".

2 Overview

2.1 Your Task

For this project, your group will play the role of a consulting company hired to design and test firewalls for corporate networks. The project will have two phases. In Phase I, you will be presented with a network and a set of requirements and asked to produce a firewall that protects the network while meeting the functional requirements. In Phase II, you will be given several competing implementations and your job will be to analyze and test them for weaknesses. This document describes Phase I. The specifications for Phase II will be posted at the end of Phase I.

2.2 Deadlines

- Phase I: Monday, November 13, 11:59 pm
- Phase II: Monday, November 20, 11:59 pm

2.3 Getting started

As of the first posting of this document, we are still ironing out some problems with DETER, though the project infrastructure should be up and fully functional within the next week. However, there is plenty to do in the meantime. To begin with, we suggest three things (which you should do first even if DETER were ready to go right away):

1. Read the specifications in this document and write some pseudocode rules like the ones we've seen in class.
2. Read the `iptables` tutorial at

   `http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html`.

   (More information in Section 5.1 below.) Then write a first-draft `iptables` version of your rulesets.
3. Skim the DETER tutorial at

   `http://www.isi.deterlab.net/tutorial/docwrapper.php3?docname=tutorial.html`.

You don't need to know most of the information in this tutorial, but you should start getting a feel for what DETER lets you do and what it's like to work with it.

3 DETER

DETER is a large-scale testbed on which complex network experiments can be run. Each project group will have an account on DETER that will give you access to the network to run your firewall and test it out. The home page for DETER is `http://www.isi.deterlab.net/`, and you can browse through the documentation there. More information will be posted to the website within the next week, as we get the bugs worked out.

4 Phase I Specifications

4.1 Network topology

The network you must defend appears in Figure 1. There are two firewalls separating the network into four distinct segments:

1. The *internet* is everything outside of this organization's network. It is completely untrusted.
2. There is a *DMZ*, where all publicly visible servers reside.
3. The internal *LAN* is where employees have their workstations. Some workstations are *admin* machines with additional access privileges (see Section 4.2).
4. There is also a *core net*, which holds the company's sensitive data and servers.

As shown in the figure, firewall FW1 links the DMZ to the internet, and firewall FW2 links the DMZ, the LAN, and the core net.

In practice, there will be only one internet machine ('remote'), one normal employee LAN machine ('user'), and one admin LAN machine ('admin').

4.2 Required functionality

There are several functionality requirements that your firewall must preserve:

HTTP service. The company needs its web server to serve content to the general public (the internet).

- The internet needs HTTP access to the web server (TCP port 80).
- The web server needs to be able to talk to the database in the core net on TCP port 3306.

Login service. Employees need to be able to log into their machines from home or when traveling. There is a login gateway in the DMZ that should be a single point of access.

- The internet needs SSH access to the login gateway (TCP port 22).
- The login gateway needs SSH access to machines on the LAN.

Time and echo services. The company provides a time/echo server that replies to queries by giving the time of day or repeating the message received.

- The internet and LAN need access to the time/echo server on TCP port 2000 (time) and TCP port 2001 (echo).

Employee workstations. Workstations on the LAN must have full access to the internet.

Administration. Admin machines on the LAN need ssh access to all DMZ and core net machines. Furthermore, ping (ICMP echo requests and responses) should be allowed to all machines from anywhere, as it is a useful diagnostic tool to see if a machine is up and reachable.

Signature server. Employees can submit documents to be signed by the company's private keys.

- LAN machines need access to the signing server on TCP port 1729.

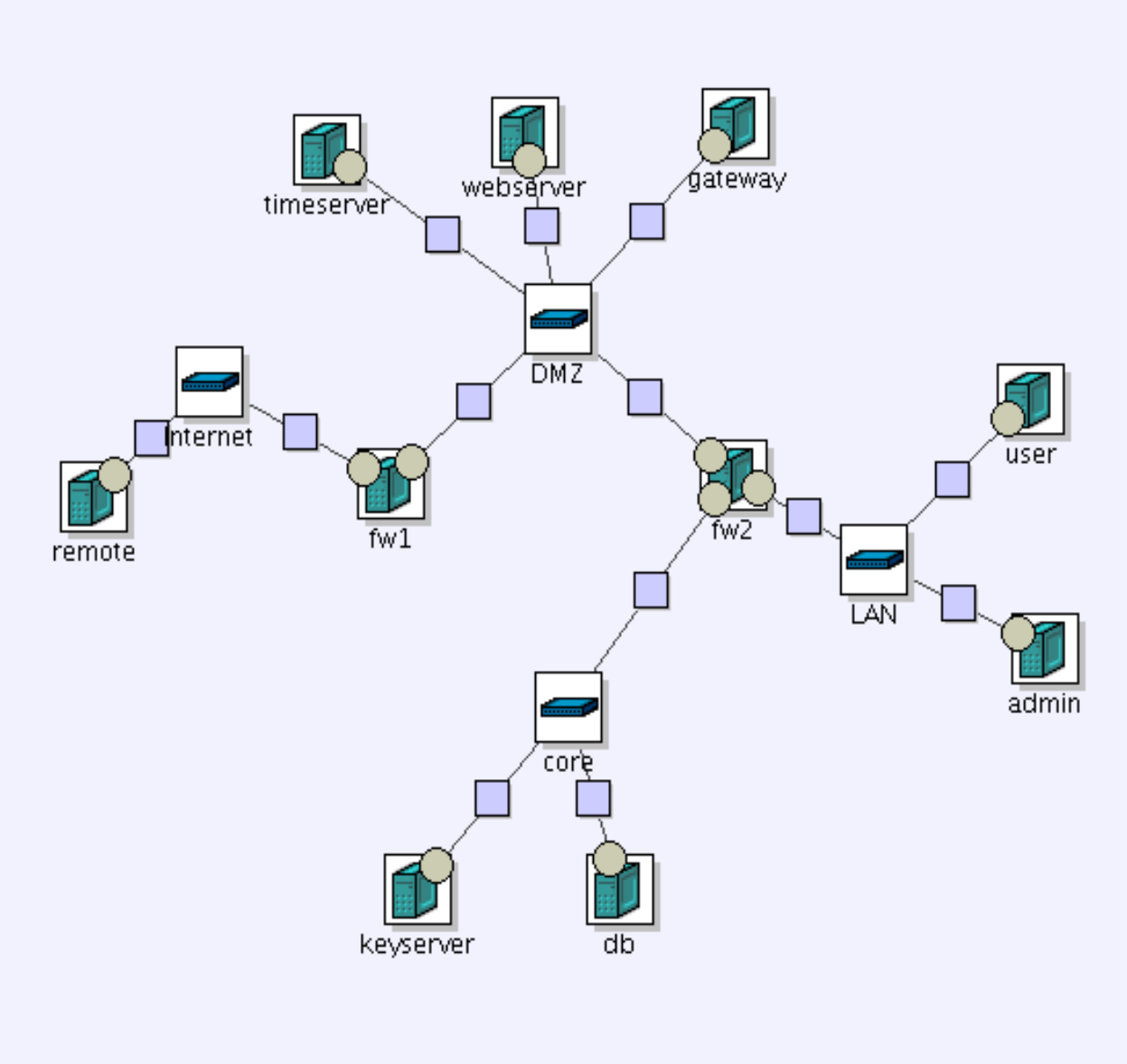

Figure 1: Graphical representation of the network topology. The node “remote” is outside the corporate network on the internet; all other nodes are within the corporate network.

4.3 Security goals

The security goals for this network are described here at a high level, and it is your job to interpret them into a concrete security policy. Assume that an attacker has full control over one machine on the internet and can send arbitrarily crafted network traffic at your network.

The security goals are very high-level for this project and focus on protecting the company's resources. Here are the main resources that need protection:

Key confidentiality/integrity. The signing server holds the company's private keys, and a compromise of these keys would be very bad. If the keys could be accessed directly by the attacker that would be devastating; if the attacker could have an arbitrary message signed then that is very bad too.

Database confidentiality/integrity. The database server holds data that is secret and very valuable to the company. Any access to that data, either reading or writing, would be a severe breach of security.

Workstation confidentiality. Each employee may have confidential information on his or her workstation. Access to any data on LAN workstations is a breach of security.

Web integrity/availability. The company website is its public face and therefore one of its key assets. Defacing the website or causing it to be unavailable is bad.

Login availability. Employees need remote access to do their work, so preventing them from logging in is undesirable.

Time integrity/availability. The time server lets employees synchronize their clocks, so having it unavailable is unfortunate and providing an incorrect time is very bad.

Many of these security goals can be enforced by your firewall, though not all, and some only partially. You are to write your firewall to preserve as many of these goals as possible, and you should also think about whether there are other desirable security goals not mentioned here. This is a high-level description, so you may need to interpret them a bit when formulating your packet-level policies. The network topology is designed to help enforce some separation and access constraints, and your firewall should take advantage of that.

5 Software and Tools

5.1 Firewall software

For this project you will write an `iptables` firewall to run on a linux system. `iptables` is close enough to the firewall syntax you learned in class that you should be able to pick it up relatively easily. Lots of information about `iptables` can be found here:

```
http://www.netfilter.org/documentation/index.html
```

In particular, you should start with the "Packet Filtering HOWTO". (Note that the HOWTO was written for the linux kernel 2.4 but your firewall will actually be running on linux kernel 2.6; however, there should not be significant differences.)

Here are some tips on getting started with `iptables`.

The paradigm for `iptables` is that you will call it once for each rule, adding that rule to the existing set. You can type each command at the command line or you can write a shell script to call it for each rule in your policy; the latter is recommended!

The first thing you should probably do is flush the tables, so that you can start fresh with a new ruleset:

```
iptables -F
```

You can do this at any time to start over; you should do this between calls to your firewall script (or just put it at the start of the script).

`iptables` has three different firewall rulesets, called chains. These chains are INPUT, OUTPUT, and FORWARD.

The INPUT chain filters traffic destined for the firewall machine itself. The OUTPUT chain filters traffic generated by the firewall machine. The FORWARD chain filters traffic routed through the firewall.

You will mainly be interested in the FORWARD chain. Note that even though the forwarding process involve traffic entering the firewall and then leaving, the INPUT and OUTPUT chains are never invoked during the forwarding process.

You might want to put some rules in the INPUT and OUTPUT chains to protect the firewall machine itself from attacks, but be careful not to block your own administrative access to the firewall!

The second thing you might want to do is turn on default deny. Default accept is the initial policy of iptables. The command to turn on default deny for the FORWARD chain is:

```
iptables -P FORWARD DROP
```

Now you can start adding rules. The command to add rules has the format:

```
iptables -A <chain>  .........  -j <target>
```

Where `<chain>` is INPUT, OUTPUT, or FORWARD, and `<target>` is ACCEPT or DROP. In place of `.......` you can specify rules that determine which packets get through.

For example, to allow pings to pass through, you can use

```
iptables -A FORWARD -p ICMP -j ACCEPT
```

The -p option specifies a protocol. Protocols include TCP, UDP, and ICMP.

You will need to specify interfaces in your rules. the -i flag and -o flag specify input and output interfaces. For example,

```
iptables -A FORWARD -p ICMP -i eth0 -o eth1 -j ACCEPT
```

Will only allow pings to pass from eth0 to eth1, but not the other way around. Of course, you also want to filter on source and destination IP addresses. You use the -s and -d flags, respectively, for example:

```
iptables -A FORWARD -p ICMP -s remote -d admin -j ACCEPT
```

This allows pings from the 'remote' machine to the 'admin' machine.

Note that in your firewalls, you will often need to use -s, -d, -i, and -o all in a single rule, to prevent IP spoofing attacks.

Also, you will need to filter on port numbers in your firewalls. To filter on destination port 80, for example, use flag `--destination-port 80`. The `--source-port` flag works in a similar way.

5.2 Testing tools

We are providing you with three network testing/analyzing tools to use both in testing your own firewall in Phase I and attacking other firewalls in Phase II. These three (plus normal utilities on your PATH, such as `ping` and `traceroute`) should be sufficient; if there is another tool you'd like to use for additional functionality, please check with your TA.

NMAP

NMAP tells you what ports are accessible. It tests whether a port that should be blocked is really blocked, and a port that should be open is indeed open. And it can do much, much more. Documentation for NMAP can be found at `http://insecure.org/nmap/docs.html`, or see its detailed man page.

HPING

HPING lets you craft packets and send them across the network. This is useful, for example, to verify that an outsider from the internet who is spoofing an internal IP cannot get access to the protected network. Documentation for HPING can be found at `http://wiki.hping.org/94`.

HPING has many capabilities; try to think like a hacker and come up with clever packets that might sneak through the firewall.

HPING in its most basic form will simply ping a remote host. For example,

```
hping3 admin
```

will ping the admin machine.

More interestingly, hping makes it easy to spoof the source IP address. For example,

```
hping3 -a keyserver admin
```

Will transmit packets to admin that have keyserver's source IP. Note that this works even from the remote machine. Your firewall design needs to consider this sort of spoofing.

NGREP

NGREP watches network traffic and displays the packets matching a rule you specify. This can be used in conjunction with HPING: HPING sends a packet and NGREP shows you the packet and any replies that go over the network. Documentation for NGREP can be found at `http://ngrep.sourceforge.net/usage.html`.

You will use nmap, ngrep, and hping on the 'remote' machine. You may also find it necessary to sniff traffic on the sensitive internal machines. To do this, you should use tcpdump.

One thing to be careful about is interfaces. Each DETER machine has a connection to the outside world, which is what you use when logging in from outside DETER. You do not want to sniff traffic on this network or send your packets over it. When using the tools, you can specify an interface. First, determine which interface you want, by running ifconfig. ifconfig gives you a list of interfaces and the IP address each interface links to.

IPs of the form `192.168.*.*` are to the control network, do not use these. IPs of the form `10.1.*.*` are for the internal network, DO use these.

Here is how to make the tools use the appropriate interface. Consider using interface eth0.

Call ngrep like this:

```
ngrep -d eth0 -e
```

On the BSD machines, use tcpdump instead, like this:

```
tcpdump -i eth0 -e
```

Note that the -e flag causes ngrep to display empty packets. empty packets are still interesting, so we want to see them. If you forget to include -e, empty packets will be displayed only as a hash mark, #, rather than a full header decode.

Specifics about where to find these tools will be posted with your DETER instructions.

6 Submitting

6.1 Writeup

For Phase I, you should primarily focus on writing your firewall scripts. You will have two scripts, one for each firewall in the network. You should also compose a short writeup that discusses the following points:

- How straightforward was writing these rulesets? What was the trickiest part?
- After completing these scripts, are you confident that you got them right? If you are unsure, what are you unsure about and why?
- What security goals were you not able to address to satisfaction with your firewall? Can you suggest other methods for dealing with any remaining security concerns?
- Did you have to make trade-offs between functionality and security? Were there any judgment calls you had to make when interpreting the high-level security goals? If so, please explain your reasoning.

Please keep your writeup under 2000 words. A writeup much shorter than that is fine; as long as nothing important is missing and everything is clear, graders strongly prefer to read short writeups.

6.2 Submission instructions

Here is how to submit for Phase I. This must be done on the instructional machines, and only one person from each group should run 'submit'.

Place your firewall rulesets (all scripts) and your writeup into a single tarball called phase1.tgz. Put phase1.tgz in a directory of its own, cd into that directory, and type

```
submit phase1
```

It should say "Submission complete. You should be hearing from us". If it doesn't say this, assume something is broken. If you can't get it to work, let your TA know as soon as possible. If it is 5 minutes before the deadline and it still isn't working, you may email your submission to your TA as a last resort.

6.3 Grading

Phase I and Phase II will each compose approximately 50% of the grade for project 2.

Here is a rough outline of how we will assign grades for Phase I (though we reserve the right to adjust this a bit if necessary):

- 30%: Complete functionality preserved.
- 50%: Secure against TA testing.
- 5%: Secure against peer testing.
- 15%: Writeup

For the TA testing, we will not grade harshly if you did not think of something creative that we tried, but we do expect your firewalls to be generally robust against network attacks.