Project 2: Penetration Testing (Phase II)

CS 161 - Joseph/Tygar

November 17, 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.

11/17: Clarified that scripts/code you write should be in tarball submitted.

11/16: Corrected name of scripts to `f1.sh` and `f2.sh`. Added more detail for journals. Clarified wordings.

Also changed specification to allow you to use your DETER accounts to gather any information you can find, not just sniffing packets.

2 Overview

2.1 Your task

In Phase II, you will have the chance to test the strength of other firewall implementations for the same network. You will play the role of the "tiger team" and try to break into the network to assess its security. Part of your job is to think creatively about how you might attack the network; part of your job is to be methodical and thorough in your evaluation.

The deadline for submitting Phase II is Monday, November 20, at 11:59 pm.

2.2 The implementations

Each group will be randomly assigned three implementations to test. Your firewall from Phase I will be distributed to other groups and you will be testing some of theirs. Furthermore, each group will receive a TA implementation as one of their three. You are expected to investigate each one to some degree, but you don't have to spend equal time on all three.

Do not discuss your firewalls with other groups to try to figure out whose firewalls you have and who has yours. You are of course free to do so after turning in Phase II, but while working on the project you should just think of them as three anonymous firewall implementations.

You will find your implementations in a tarball in `/groups/CS161/gXX/phase2`. Group 7, for example, will find a file called `g7.tgz` that will untar to create a directory called `g7` with three subdirectories under it, `g7-1`, `g7-2`, and `g7-3`. In each directory there are scripts called `f1.sh` and `f2.sh`. (Some implementations also have other scripts alongside those.) To run an implementation, just type `sudo f1.sh` on node fw1 and `sudo f2.sh` on node fw2 (you may have to do a `chmod +x f*.sh` first).

Some implementations may have minor functionality errors. But if you are having trouble running one at all, or it has major errors, please let your TA know so we can see whether it's due to changes we made.

2.3 Experiments

You will have to start a new experiment on DETER (not just swap in an old one) because there is a new NS file for Phase II. The network layout has not changed, but the startup scripts have changed and that requires a new NS file. You can find it at `/proj/CS161/ns/proj2phase2.ns`.

Please remember to swap out or terminate your experiments when you are not using them.

3 Attacking the Network

3.1 Security goals

The purpose of penetration testing is to discover whether there are vulnerabilities in a network setup that could lead to security goals being compromised. The security goals are given in the Phase I handout. There may be other security goals that are implied or that just haven't been mentioned; if you think there are other important security goals not listed that you can compromise, please explain them in your writeup.

3.2 Expectations

We do not know how difficult this part of the project will be. We expect that every group should be able to find at least a couple of vulnerabilities, but we don't expect any group to be able to compromise all of the security goals. Maybe every group will be able to find big holes in each firewall, or maybe everyone will struggle to find even one exploitable vulnerability. We'll be interested in hearing how it's going for you.

3.3 The rules

The Golden Rule: You may use any techniques that would be possible on the real internet, but you may not use any feature or artifact of DETER to aid your attacks.

For example, one of the trophies listed below is a message signed by the keyserver. Since you all have root access to the keyserver, you could ssh in and sign a message directly. Or even simpler, you could look for the key in the /proj shared filesystem and sign it that way. But of course this is not the intent; you must imagine you don't have such access and figure out a way to break in as if this were a real corporate network. In general, you'll have to provide enough details that we can reproduce your attack ourselves to see that it works.

For your attacks, you have the following resources:

- One computer on the internet. This is the machine 'remote' in the experiment network. You may use sudo on that machine; you have full control of it to launch your attacks.
- Any software already installed on that machine. This includes standard utilities such as ping, traceroute, and telnet, and also the tools nmap, hping3, and ngrep in /proj/CS161/toolBinaries.
- Any programs/scripts you write yourself to run on remote.
- You may look at and analyze the firewall scripts you are attacking; we are not assuming any attempts at "security through obscurity" here.
- You may also use your access to the network machines and to the shared filesystem to explore and gather any information you can find.

Please do not install additional software on any of the nodes unless you write it yourself for this project. If there is other software you wish to use, please let us know and we will consider adding it.

You should specifically avoid the following in executing your attacks:

- Taking advantage of the shared filesystem (except to run tools in /proj/CS161/toolBinaries and so on)
- Using the fact that some virtual nodes share a physical host
- Using your DETER account to access nodes other than remote
- Using the control network in any way

. . . and anything else that violates the Golden Rule.

We are making one significant exception to the Golden Rule: you may log in to any node on the network and access the shared filesystems to explore and gain information about the software/services running. Because of this, you will have a lot more information at your disposal than a real attacker is likely to have (though there are ways for a real attacker to gain such information). This includes sniffing network packets, reading CGI scripts, and so on. We are allowing this exception because it should make the process much easier for you and help you get more out of it, and because it would be impossible to enforce a prohibition on getting this kind of feedback.

It is important that you do not use your DETER access to execute an attack, however. We will be reproducing your attacks when grading, and we need to be able to do so without using any access the attacker doesn't have. So the idea is that you can use your DETER access to gain information that will help you find/create attacks, but the execution of those attacks must be done without relying on that access. (Imagine the attacker has some inside source for information but not granting access, or is just really lucky at guessing!)

4 Specification of Services

The behavior of each of the network's services is described here in more detail than previously specified.

HTTP service. The company has its website served by Apache running on webserver. Apache accepts HTTP connections on TCP port 80. If you connect to `http://webserver/cgi-bin/index.cgi` you should see a simple home page for the corporation that lets you enter a username and look up the privileges of a user ("user" or "admin"). We recommend using lynx to view the web page (e.g. `lynx http://...`).

Login service. An SSH server is running on gateway, and that is the point of entry for employees. For the purpose of this project, there are one admin and one normal user who will be accessing machines via SSH. The admin has login 'alice' and the normal user has login 'bob'. (You can look them up on the web interface.) Bob can authenticate with his password on gateway and user, and Alice can authenticate with her password on any BSD machine.

Time and echo services. The time service on timeserver's TCP port 2000 accepts a connection, waits for a line of text to be sent, then returns the time formatted for human readability.

The echo service on TCP port 2001 accepts a connection and then echoes back each line of text sent. Telnet works well for connecting to these services.

Signature server. The signature server accepts connections on TCP port 1729 and reads lines until EOF or a line containing only one period ('.'). When it reaches EOF or a period, it signs the data received and sends it back on the socket. We recommend using telnet to connect and send text to be signed. The signatures can be verified with the public key available at `http://webserver/corporate_public_key.asc`.

MySQL. A MySQL server is running on db and accepting connections on TCP port 3306. The web server contacts the database for the content of the webpage.

5 Attacks and Documentation

5.1 Task specifics

There are two sides to your task:

1. Do a reasonably thorough test of each firewall you are given to be sure it meets the required functionality and doesn't have any blatant security holes. This part will focus almost entirely on the firewalls and network traffic.

2. Be creative and think of other attacks to try that might compromise security goals through other means. This part will focus in part on the firewalls and network traffic, and in part on the applications running on the network.

For the first part, your documentation can be very terse if the functionality works and there are no major holes. If you do find missing functionality or clear security holes, please write them up and explain what you think the mistake is.

The second part is more interesting. We know that there are some vulnerabilities in some of the firewalls and in some of the applications, and you may be able to find and exploit some of those. It is also likely that there are vulnerabilities we don't know about that you might find; this is a complex system with several applications running on top of a nontrivial network topology, so there are bound to be holes somewhere.

When looking for vulnerabilities, use the security goals (listed in the handout for Phase I) to guide you. Can you compromise the signing key, either by actually acquiring the key or by getting a message signed by it? Can you access any data in the database that isn't explicitly available via the web page? Can you gain shell access to or view files on the user or admin machines? Can you deface the web page? Can you cause a denial of service on the webpage or the

login service (by exhausting a resource or by crashing the program in question, or by . . .)? Can you cause the time or echo servers to behave incorrectly or be unavailable? Be creative and see what you can do!

Don't set your expectations too high, however. This will be a difficult project and you probably won't find much that you can exploit. That's fine; keep a record of what you try and document your thought process and methodology.

5.2 Documentation tips

When looking for vulnerabilities to exploit, you should keep a log of everything you try. Your writeup should include not only details about your successes, but also information about what you tried that didn't work.

For example, let's say you think you might be able to cause a denial of service on the web server by sending it a whole bunch of requests all at once. If you use hping3 to bombard it with TCP connection requests but you find that you have no problem connecting with lynx, discuss why the attack didn't work and whether you think it was due to an error in your reasoning or a lack of resources or strong robustness on the part of apache or what. Data collected with ngrep or tcpdump may be useful here (remember, you're allowed to sniff on every machine even if you can't use your access for other purposes).

When you do find an attack that you think compromises one of the security goals, please be very explicit about how you did it; we should be able to reproduce your attack exactly after reading your description. Discuss what the vulnerability was and whether it could have been avoided given the project specifications.

5.3 Suggestions

We have a few specific suggestions of vulnerabilities to look for or paths to take in crafting your attacks.

- First and most obviously, look for misconfigurations in the firewalls. Can you access machines you shouldn't be able to on ports that shouldn't be open?
- alice and bob have password-based access to several machines. Can you compromise one or both of their passwords as the attacker?
- The website communicates with a MySQL database. Is there the possibility of a SQL injection attack? (See section 14.5.3 in the textbook or `http://www.securiteam.com/securityreviews/5DP0N1P76E.html`.)

You aren't guaranteed to find anything with these methods, but they are a good starting place.

5.4 Trophies

When you do find an attack that significantly compromises a security goal, capture some record of it that you can show off to demonstrate your achievement. For example, if you manage to deface the webpage, take a screenshot. If you get access to the keyserver, a signed message can be a trophy. Get creative. This aspect is mostly just for fun; grading will be based on your description (remember, we need to be able to reproduce your attack), but the trophies are hard evidence of your success!

6 Submitting

6.1 Writeup

At the top of your writeup, please be sure to list your names and your group number.

You will have two main sections in your writeup. First is your evaluation of the three firewalls based on functionality requirements and clear-cut security assessment (for example, do they let the remote node have direct access to anywhere it shouldn't have access?). This part should be short; no more than 500 words per firewall. Please refer to the firewalls by their numbers ("our third firewall", or "g7-2", etc.).

The second section describes your attempts to find creative and interesting attacks on the firewalls and the applications. For tests/attacks you tried that didn't work, or probing you did that revealed no weaknesses, give a brief description of what you did and why, as well as why you think it wasn't successful. For attacks that did succeed, you must give us enough information to reproduce the attack, and you must explain exactly what the attack accomplishes, that is, what security goal it violates and how. This part of the writeup should not exceed 3500 words.

On both parts, if you do not need to use the entire word limit, that's great! As long as you explain yourselves well and make it clear you did a complete job, a shorter writeup is always preferable to graders.

Your writeup should be in PDF, HTML, or plain text format.

6.2 Project journals

After the submission, each student should also send your TA a project journal like you did after the first project. This should include:

- A brief description of what you did on this project.
- A distribution of 120 points among your fellow team members. If you deviate much from an even distribution, please briefly explain why.

You have up to 24 hours after the deadline (plus any slip days you use) to send your journal to your TA by email. You are free do discuss your journals with your group if you like, but you also have the right to keep them private. We will not share your journals with anyone other than the TAs and professors.

6.3 Submission instructions

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

Put your writeup, any trophies you collect, and any scripts or code you've written that we'll need to reproduce your attacks into a tarball called phase2.tgz. Put phase2.tgz in a directory of its own, cd into that directory, and type

```
submit phase2
```

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.

6.4 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 II (though we reserve the right to adjust this a bit if necessary):

- 30%: Thorough evaluation of all three implementations (10% each).
- 40%: Descriptions of what you tried and your methodology, whether successful or not.
- 20%: Creative thinking in coming up with attacks to try and ways to test security, whether successful or not.
- 10%: Success in finding creative exploits that violate security goals (including showing trophies).

We should emphasize that the bulk of your grade here depends only on your methodology and your explanation; only 10% of the grade depends on how successful you were in your attacks, and we'll adjust grading on that 10% depending on how hard this turns out to be.