

CS166 Project 2: Sentinel

Due on Tuesday, March 17, 2009 at 11:59 pm EST

Project Out: March 2, 2009

Silly Premise

The virus wars of 1997 hurt you, financially, socially and in other ways you can only reveal to your therapist. Many went to prison, and once out, the big ones went to work for security companies. The small ones, never arrested, went on to work for the same people they were once attempting to exploit. It was the fish in the middle, people like you, people uncharming enough to not be the media's pets, but at the same time intelligent enough to not be able to escape prison. Prison made you more focused, you swore you'd never get involved in this business again, make your living writing little code, after all you could write better code than anyone else.

Consulting, securing networks, was going well for you until a new piece of code landed on your desk, a code which clearly indicated a major bug in one of the most popular operating systems out there. Someone needed to patch this up, and you did not need to get involved in this, but there was a reason those events were called a war, the big bosses, the charismatic fools would never believe your claims, and the code you have so far makes exploitation of this worm simple enough. You could work to secure your own companies, work on releasing a worm, and telling the world the solution, making those who left you for dead suffer, but still...going to prison would really suck. But then again who would send someone who helped secure thousands of computer networks, informed the world of an obvious bug, and provided the computing world with a solution to their own worm to prison? It was obviously other people's mistake. Besides, the jury would agree, the big fish screwed up, it was nearly completely their fault. Now if only you kept good records of your communications and the like, you'd probably never go to prison, perhaps, moving to a state with lax laws might be good, but then again a consultant really doesn't need to work out of anywhere ...

Introduction

In this assignment, you will develop a worm that exploits a variety of fabricated security holes in Linux machines.

We will provide code and instructions to allow you to exploit the security holes. The primary emphasis of this project will be on the other parts of worm writing: probing for vulnerable machines, propagating and spreading, and so forth.

Virtual Machines

You should work on this assignment using special VMware virtual machines that run on host machines of the Internet lab. Specifically, you are provided two virtual machines to work with, both Linux systems. They are:

- The development virtual machine where you will be doing all of your development work. It's got everything installed that you need to get the assignment done. We've installed the Java JDK and JRE, Eclipse, and all the Java docs you'll need. When you load up the virtual machine, you'll see the eclipse shortcut and all the Java docs shortcuts. If you run eclipse it'll automatically be set to the right workspace and everything. We've done everything possible to make it as easy as possible for you to work in the virtual machine. If you'd like to add anything, please let a TA know so we can help you with it. Note that there is no stencil code provided for this assignment.
- The "exploitable" virtual machine basically doesn't have anything on it. We've stripped it down as much as possible to save space on the file system. It's mostly for testing on. It does, however, have all standard UNIX utilities, including network clients like ftp and ssh, and runs sshd, telnetd, and the Apache webserver. It also has vim and emacs, so you can edit files on the machine.

We would like to have set up VMware on the Internet lab machines so that the virtual machines are constrained to an isolated virtual network (a *virtual LAN*). However, we only have host-only networking available.

Do not modify the VMware settings on the Internet lab machines!

If you modify the VMware settings, you may allow the worm to spread in the wild. This will cause undue annoyance to other users and may get you in trouble with the law.

The development virtual machine is located at:

```
/course/cs166/asgn/sentinel/vms/sentinel-dev.
```

The sample target virtual machine is located at:

```
/course/cs166/asgn/sentinel/vms/sentinel-target.
```

Do not modify these virtual machines directly, you need to copy them to your individual cs166 project directory. These are available as `/u/<login>.project`, so for instance if your user name were `pmeier` your project directory would be `/u/pmeier.project`. Please copy the virtual machine before modifying any code, not doing so is a violation of the collaboration policy.

You can log into the development VM and the target VM with the username `root` and the password `i<3166`. (To log into the target machine, you must first press "enter" at the LILO boot prompt.)

Trying to work in the VMs without having them copied over will not save any of your work. Remember to use the shared folders from the previous assignment!

The Worm

The overall goal of the assignment is to create a worm that can propagate by transferring itself to other vulnerable computers. That's the general goal and it's really up to you how you implement this. However, we're giving you instructions and information on how to gain use the exploits to gain root access.

What needs to be done in order to get your fully working worm is:

1. Deliver an innocuous payload to the victim machine. This can be anything you want, provided it is benign. E.g., you may change the message of the day located in `/etc/motd`, but you should not remove or corrupt system files.
2. Set the program to start at boot time.
3. Connect to a target machine (another VM you are running). You can find another machine's IP address by running `ifconfig`.
4. Probe the target machine for any of the various security holes described below. You can use any method you like (such as trying to exploit the hole) to see if the hole exists, but you must make sure that your worm can exploit any machine that has any one of the three security holes.
5. Use the appropriate exploit to gain root access to the target machine.
6. Transfer the worm files over to the target machine. Here, you can choose any method for transferring the files over.
7. Start the worm at the target machine. Tell the target machine to execute the worm program so it can infect other machines. (Make sure that you hardcode the IPs of your target VMs, so you don't inadvertently attack another student!)
8. Repeat steps 3–8

The Exploits

Each target machine will have one of the following security holes (the sample machine has all of them):

- The `login(1)` program may contain (perhaps due to a malicious compiler?) a back door. Entering the username “backdoorman” followed by any non-empty password in a telnet connection (or locally) will log you into the root account. A telnet class is provided for you in the development VM.
- A webpage located at `/hamlet.html` may contain an application that allows you to see how many lines in Hamlet have a given word in them. However, this application is written very insecurely, and the webserver is misconfigured. The application invokes a cgi script at `/cgi-bin/hamlet`, which takes a GET query with the variable “word”. Everything in “word” after a semicolon will be executed as a separate command! The application has write permission for the file `/etc/passwd`; thus, to gain root access, you can add an entry to this file with uid of 0 and an empty password: executing the command “`echo mallory::0:0:::/bin/sh >> /etc/passwd ;`” will add a user named mallory, with no password, with root permissions. Try querying the application with both valid and malicious queries to try and find the HTTP request you should send in order to exploit the machine.
- RFC 862 specifies a standard “echo” service that runs on port 7, accepts all connections, and sends all characters received back to the source. This service, which may be running on a machine, has a buffer overflow: a 1024-character buffer is used to hold the characters, and thus larger packets can overflow the buffer and overwrite the return address.

As it turns out, the return address is located 1036 bytes after the beginning of the buffer. Shellcode is provided for you in the development VM. There is some minor variation in where

the stack is located in the victim process, so we have also provided you with start and end locations for where the stack will most probably be, in the support code. Be warned though, exploits have very demanding specifications, some students last year had trouble with the placement of noops. Never assume that something *should* work, when working in the gray like you are, things sometimes just don't work for no apparent reason.

This shellcode will open a remote shell with root permissions at TCP port 20000. You can execute commands by sending them to this shell followed by `\n\0` (a newline and a zero character).

It is highly recommended that you read the following websites:

<http://mixter.void.ru/exploit.html>

<http://insecure.org/stf/smashstack.html>

Extra Credit

Since this an open ended assignment, there's a lot of extra credit opportunity. Come see a TA on hours if you think that your idea might be worthy of extra credit.

Some ideas that might work are: Finding your own vulnerability, for instance trying to brute force passwords.

Handing in

We will be having an electronic handin script that will be available in the VMs on the Virtual Network. More information on how to do this will be posted to the website and the handout will be changed to reflect this.

You will be required have a README that documents everything you did and if there are any bugs in the project. You should also comment your code so that we can read it easily.

Grading

We will be having interactive grading for this assignment (And most likely for all future assignments). We will be grading you based not just on functionality but also on the design, efficiency, commenting, and documentation.

Internet Lab Rules

For this project, all your coding must be carried out in virtual machines running in the Internet Lab. You should at no time remove code from the virtual machines or attempt to run it outside these virtual machines. The TAs will take a dim view of attempts to run or take code out of the Virtual Machines.

The Internet Lab has around 16 individual machines, and it should be obvious that this class has significantly more students than machines. Keeping this in mind, you might want to plan your schedule so that none or very little of your work is compressed into the final moments before this project is due. In the event that the Internet Lab is full, you can also ssh into one of the Internet Lab computers from another computer, and open the VM through the ssh session. However, this

will not run as quickly as working on the computer itself. Our use of the Internet lab is contingent upon everyone respecting the computers in there, not carrying drinks in, and not abusing access to this lab. Doing any of those things would lead to 166 being thrown out of the Internet Lab, and a total existence failure for most of the course. This would not bode well for you, and you should hence not engage in such actions.

Also, remember you are provided with virtual machines in the Internet lab to develop your assignment, and you should be coding within those. Do not attempt to move code out of the virtual machine.

NOTE: This information may change this semester, and if it does, this will be noted on the website, and an email will be sent out on the listserv.

Reading Material

- Lecture slides on operating systems, access control, and malicious code, available on the course website.
- Excerpts from the book *The Art of Computer Virus Research and Defense* by Peter Ször, Addison-Wesley (2005), ISBN 0321304543, available at

<http://www.certifiedsecuritypro.com/content/view/137/56/>

- Slides of the presentation *Fighting Computer Virus Attacks* given by Peter Ször at the USENIX Security 2004 conference, available at

<http://www.usenix.org/events/sec04/tech/slides/szor.zip>

- RFC 862, describing the Echo protocol

<http://www.faqs.org/rfcs/rfc862.html>

- RFC 2616, describing the Hypertext Transfer Protocol.

<http://www.faqs.org/rfcs/rfc2616.html>

- RFC 854, describing the Telnet Protocol

<http://www.faqs.org/rfcs/rfc854.html>