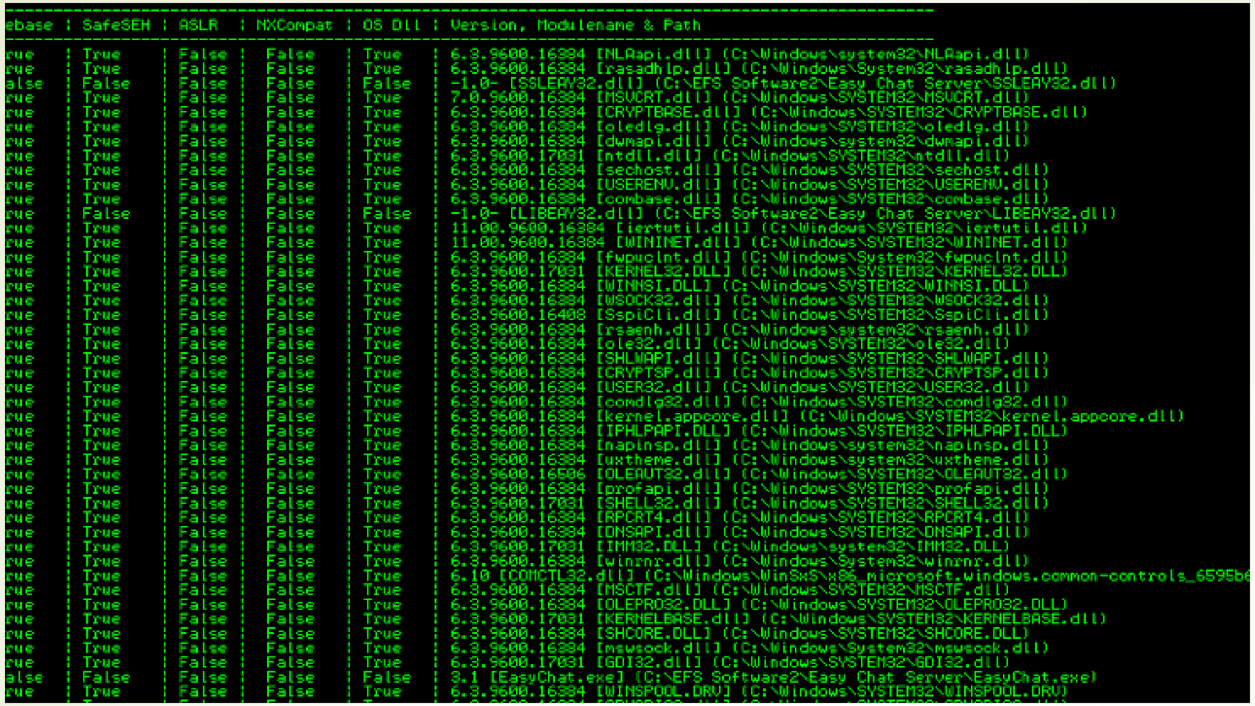


```
---------------------------------------------------------------------------------------------
ebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
---------------------------------------------------------------------------------------------
rue   | True    | False |  False   | True   | 6.3.9600.16384 [NLAapi.dll] (C:\Windows\system32\NLAapi.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [rasadhlp.dll] (C:\Windows\System32\rasadhlp.dll)
alse  | False   | False |  False   | False  | -1.0- [SSLEAY32.dll] (C:\EFS Software2\Easy Chat Server\SSLEAY32.dll)
rue   | True    | False |  False   | True   | 7.0.9600.16384 [MSVCRT.dll] (C:\Windows\SYSTEM32\MSVCRT.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [CRYPTBASE.dll] (C:\Windows\SYSTEM32\CRYPTBASE.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [oledlg.dll] (C:\Windows\SYSTEM32\oledlg.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [dwmapi.dll] (C:\Windows\system32\dwmapi.dll)
rue   | True    | False |  False   | True   | 6.3.9600.17031 [ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [sechost.dll] (C:\Windows\SYSTEM32\sechost.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [USERENV.dll] (C:\Windows\SYSTEM32\USERENV.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [combase.dll] (C:\Windows\SYSTEM32\combase.dll)
rue   | False   | False |  False   | False  | -1.0- [LIBEAY32.dll] (C:\EFS Software2\Easy Chat Server\LIBEAY32.dll)
rue   | True    | False |  False   | True   | 11.00.9600.16384 [iertutil.dll] (C:\Windows\SYSTEM32\iertutil.dll)
rue   | True    | False |  False   | True   | 11.00.9600.16384 [WININET.dll] (C:\Windows\SYSTEM32\WININET.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [fwpuclnt.dll] (C:\Windows\System32\fwpuclnt.dll)
rue   | True    | False |  False   | True   | 6.3.9600.17031 [KERNEL32.DLL] (C:\Windows\SYSTEM32\KERNEL32.DLL)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [WINNSI.DLL] (C:\Windows\SYSTEM32\WINNSI.DLL)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [WSOCK32.dll] (C:\Windows\SYSTEM32\WSOCK32.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16408 [SspiCli.dll] (C:\Windows\SYSTEM32\SspiCli.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [rsaenh.dll] (C:\Windows\system32\rsaenh.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [ole32.dll] (C:\Windows\SYSTEM32\ole32.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [SHLWAPI.dll] (C:\Windows\SYSTEM32\SHLWAPI.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [CRYPTSP.dll] (C:\Windows\SYSTEM32\CRYPTSP.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [USER32.dll] (C:\Windows\SYSTEM32\USER32.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [comdlg32.dll] (C:\Windows\SYSTEM32\comdlg32.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [kernel.appcore.dll] (C:\Windows\SYSTEM32\kernel.appcore.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [IPHLPAPI.DLL] (C:\Windows\SYSTEM32\IPHLPAPI.DLL)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [napinsp.dll] (C:\Windows\system32\napinsp.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [uxtheme.dll] (C:\Windows\system32\uxtheme.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16506 [OLEAUT32.dll] (C:\Windows\SYSTEM32\OLEAUT32.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [profapi.dll] (C:\Windows\SYSTEM32\profapi.dll)
rue   | True    | False |  False   | True   | 6.3.9600.17031 [SHELL32.dll] (C:\Windows\SYSTEM32\SHELL32.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [RPCRT4.dll] (C:\Windows\SYSTEM32\RPCRT4.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [DNSAPI.dll] (C:\Windows\SYSTEM32\DNSAPI.dll)
rue   | True    | False |  False   | True   | 6.3.9600.17031 [IMM32.DLL] (C:\Windows\system32\IMM32.DLL)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [winrnr.dll] (C:\Windows\System32\winrnr.dll)
rue   | True    | False |  False   | True   | 6.10 [COMCTL32.dll] (C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b6
rue   | True    | False |  False   | True   | 6.3.9600.16384 [MSCTF.dll] (C:\Windows\SYSTEM32\MSCTF.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [OLEPRO32.DLL] (C:\Windows\SYSTEM32\OLEPRO32.DLL)
rue   | True    | False |  False   | True   | 6.3.9600.17031 [KERNELBASE.dll] (C:\Windows\SYSTEM32\KERNELBASE.dll)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [SHCORE.DLL] (C:\Windows\SYSTEM32\SHCORE.DLL)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [mswsock.dll] (C:\Windows\System32\mswsock.dll)
rue   | True    | False |  False   | True   | 6.3.9600.17031 [GDI32.dll] (C:\Windows\SYSTEM32\GDI32.dll)
alse  | False   | False |  False   | False  | 3.1 [EasyChat.exe] (C:\EFS Software2\Easy Chat Server\EasyChat.exe)
rue   | True    | False |  False   | True   | 6.3.9600.16384 [WINSPOOL.DRV] (C:\Windows\SYSTEM32\WINSPOOL.DRV)
```

Exploit Development

The Art of Exploitation

About Me

MANISH SHARMA

Working at Gainsight

Linkedin/sh377c0d3

Twitter: @sh377c0d3

Github/sh377c0d3

Discalimer

THIS ENTIRE TALK IS MORE AT PERSONAL LEVEL AND DOESN'T CONTAIN OR RELATE TO ANY OF MY FORMER OR CURRENT PROFESSIONAL ASSOCIATIONS.

ALL CONTENT HAS BEEN SOURCED FROM PUBLICLY AVAILABLE SOURCES LIKE THE INTERNET. THE PRESENTER DOES NOT INFRINGE OR CLAIM RIGHTS OVER ANY CONTENT, IMAGES AND ANY OTHER WORK BEING PRESENTED HERE AND ALL RIGHTS BELONG TO RESPECTIVE OWNERS ONLY.

Agenda

- Basics
- Stack Smashing (with Demo)
- SEH
- Egg Hunter
- Shellcoding
- Protection against Exploit Development
- Defeating the Protection
- Q & A

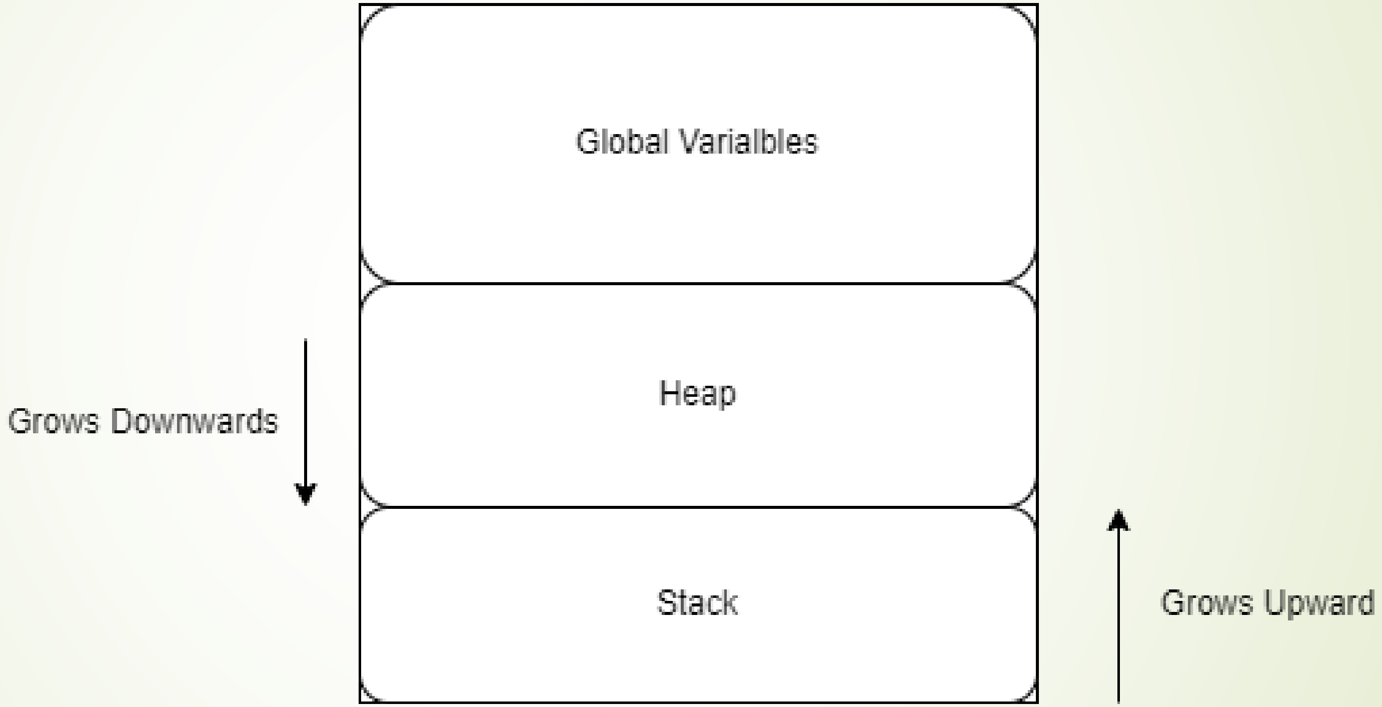
Basics

- Exploit
- Tools required for the process
    - Debuggers (Immunity Debugger, xdbg, ollydbg, gdb and more)
    - Fuzzers for fuzzing the software for crash
    - Metasploit-Framework
    - Mona.py
    - Programming
    - Reverse Engineering
    - Assembly
- What is Fuzzing?
- What is Buffer Overflow?

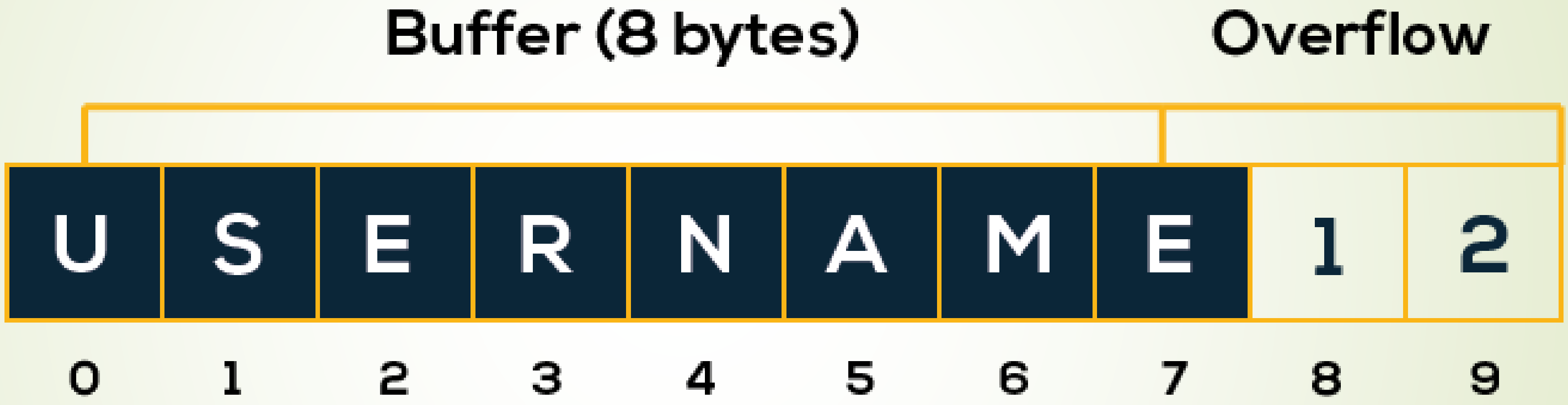
Basics (contd.) - Fuzzing

- Discovering faults in applications by providing unexpected input and monitoring for exceptions.
- **Types of fuzzers:**
  - Mutation-based
  - Generation-based
- **Fuzzing Targets:**
  - Environment variables and Arguments
  - Web application and server
  - File Format Network Protocol
  - Web browsers
  - In-memory

Basics (contd.) - Memory



Basics (contd.) – Buffer Overflow



Basics (contd.) – Steps for Exploit Dev.

1. Identify the Entry Point
2. Fuzz the application/software for a crash
3. Re-create the crash
4. Control the Execution
5. Hunt and eliminate bad characters
6. Generate shellcode for exploitation
7. Obtain a Shell

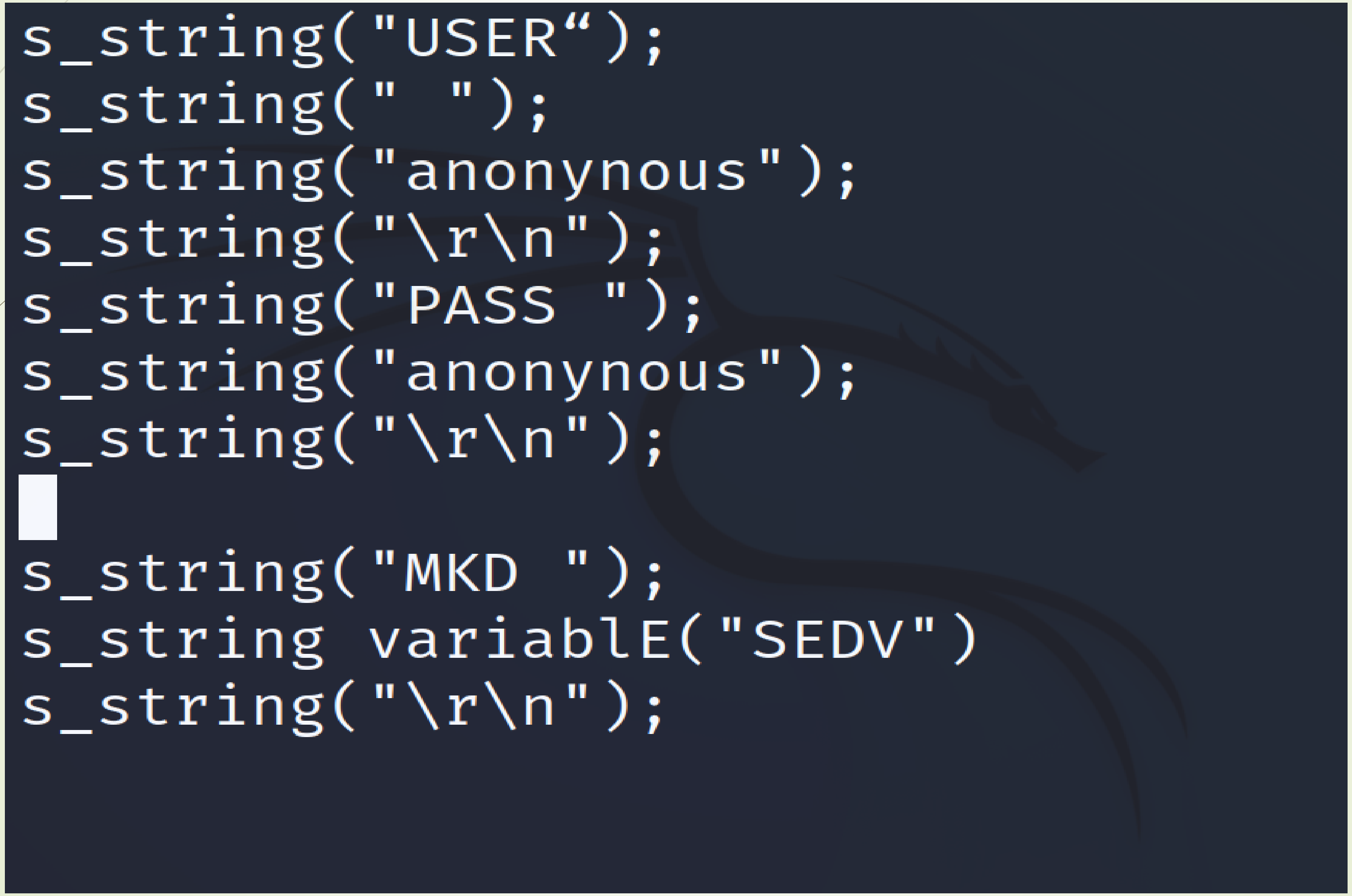
Stack Smashing

- Stack overflow, also called buffer overflow or stack-based buffer overflow
- It occurs due to a programmatic error.
- This may happen when the program Is Insecurely handling user-supplied data.
- The core of buffer overflow exploitation on Windows is the same as it is on Linux.

Stack Smashing (contd.)

- Vulnerable fields:
  - Form fields where text can be placed into
  - Command line arguments
  - Remote resources fetched by the application
  - Files parsed by an application

Spike Command : fuzz.spk



```
s_string("USER“);
s_string(" ");
s_string("anonynous");
s_string("\r\n");
s_string("PASS ");
s_string("anonynous");
s_string("\r\n");

s_string("MKD ");
s_string variablE("SEDV")
s_string("\r\n");
```

Before we start Exploitation : Linux

Compile:

```
gcc -fno-stack-protector -z execs-ack program.c -o program
```

Disable ASLR:

```
echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
```

Stack Smashing (contd.)

DEMO Time!

Windows and Linux Exploitation

SEH

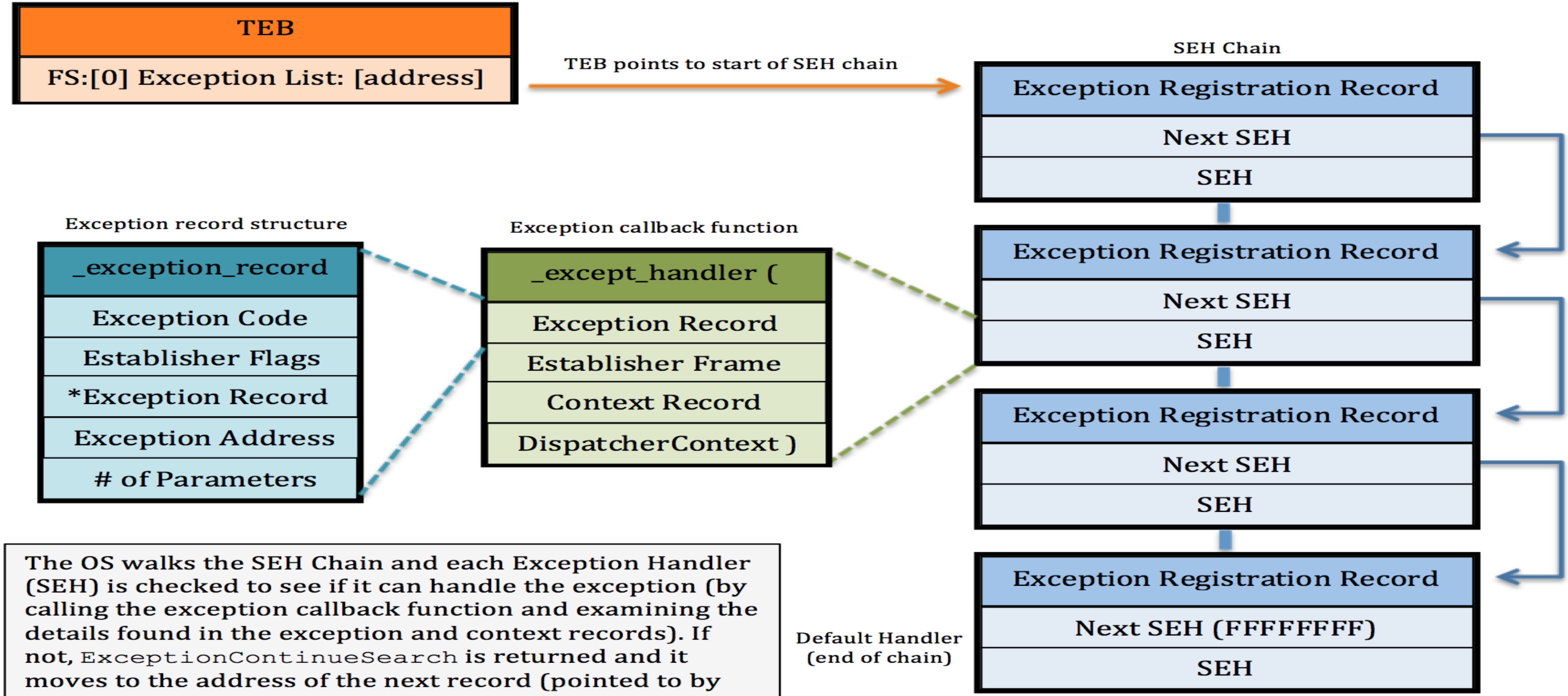
- A Windows feature that handles application’s exceptions.
- Mechanism used by programmers
  - helps applications handle any unexpected conditions encountered during a program’s runtime.
- Exception happens
  - Windows will pop-up a familiar dialog box to us
  - Which state that “Application Encountered an error” and then program will exit

SEH (contd.)

- SEH is LinkedList of exception handler
  - Contains pointer to the current exception handler record and the next exception handler.
- So, SEH are implemented in the form of a chain.
- Overwrite SEH with pointer to POP POP RETN instruction and overwrite nSEH with opcode to jump to attacker-controlled memory location.

Windows SEH Chain

(simplified)



TEB

FS:[0] Exception List: [address]

TEB points to start of SEH chain

SEH Chain

Exception Registration Record

Next SEH

SEH

Exception Registration Record

Next SEH

SEH

Exception Registration Record

Next SEH

SEH

Exception Registration Record

Next SEH (FFFFFFFF)

SEH

Default Handler
(end of chain)

Exception record structure

_exception_record

Exception Code

Establisher Flags

*Exception Record

Exception Address

# of Parameters

Exception callback function

_except_handler (

Exception Record

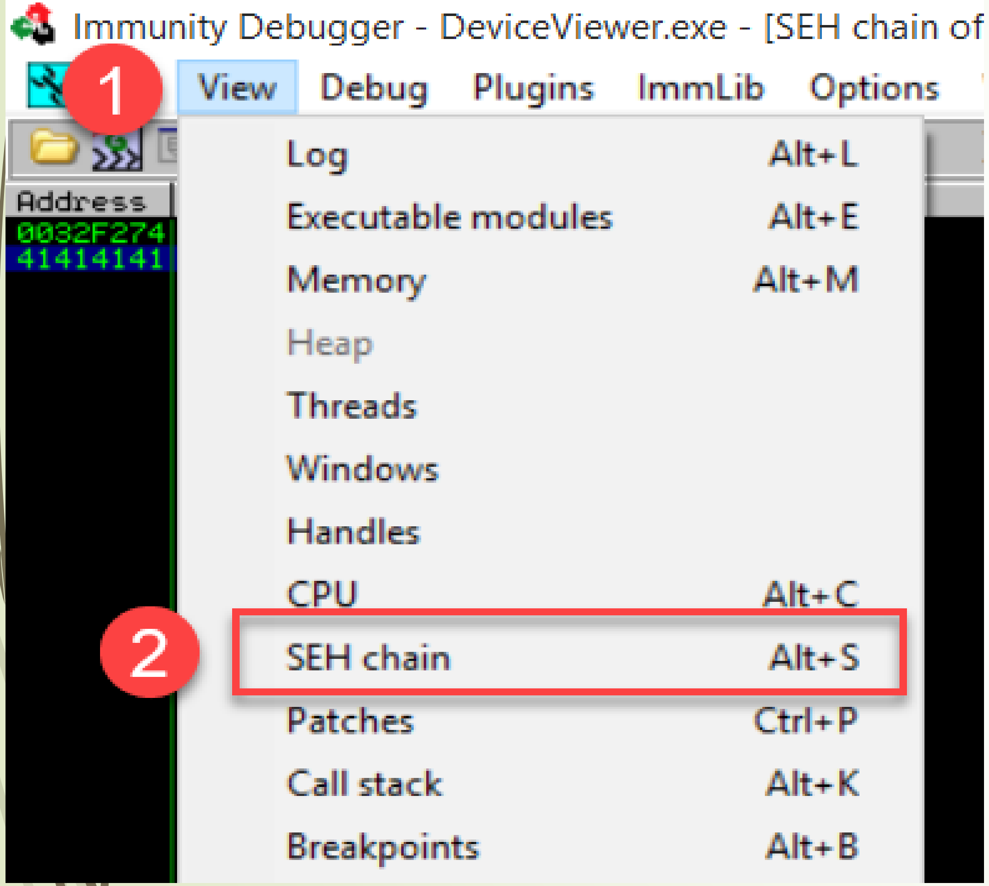
Establisher Frame

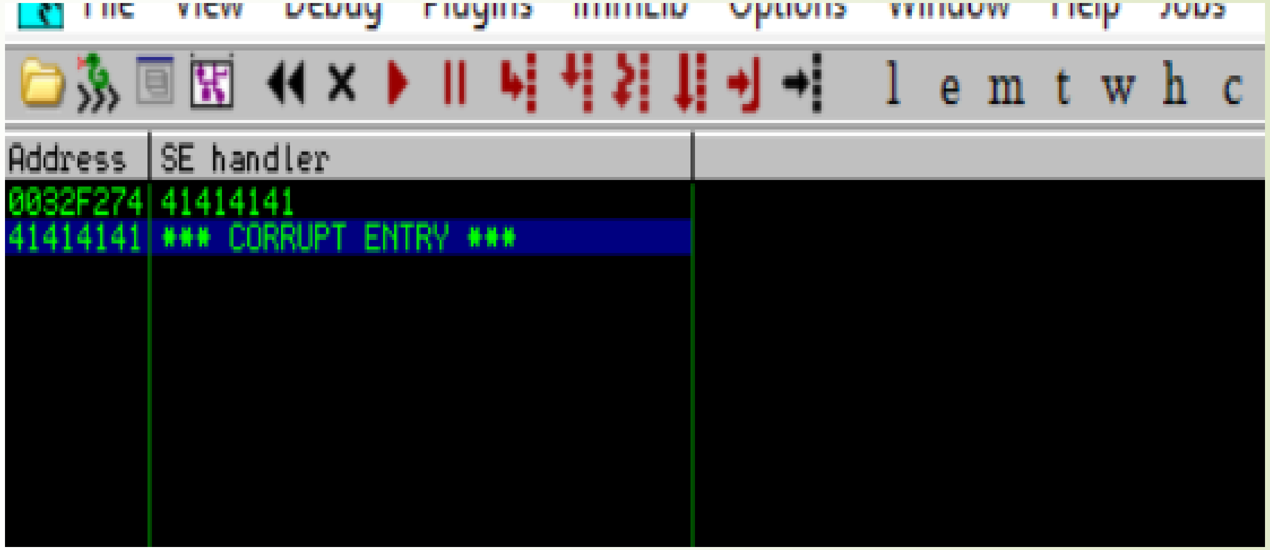
Context Record

DispatcherContext)

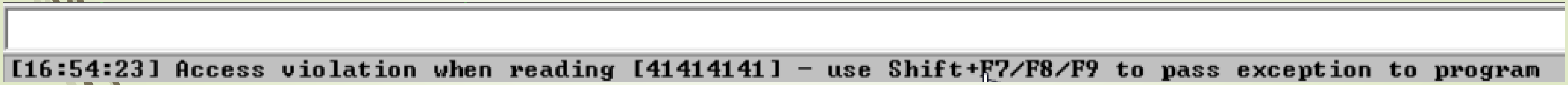
The OS walks the SEH Chain and each Exception Handler (SEH) is checked to see if it can handle the exception (by calling the exception callback function and examining the details found in the exception and context records). If not, `ExceptionContinueSearch` is returned and it moves to the address of the next record (pointed to by Next SEH) and continues down the chain until it finds a suitable exception handler or hits the last, default handler (FFFFFFFF)

SEH (contd.)





```
Address   SE handler
0032F274  41414141
41414141  *** CORRUPT ENTRY ***
```



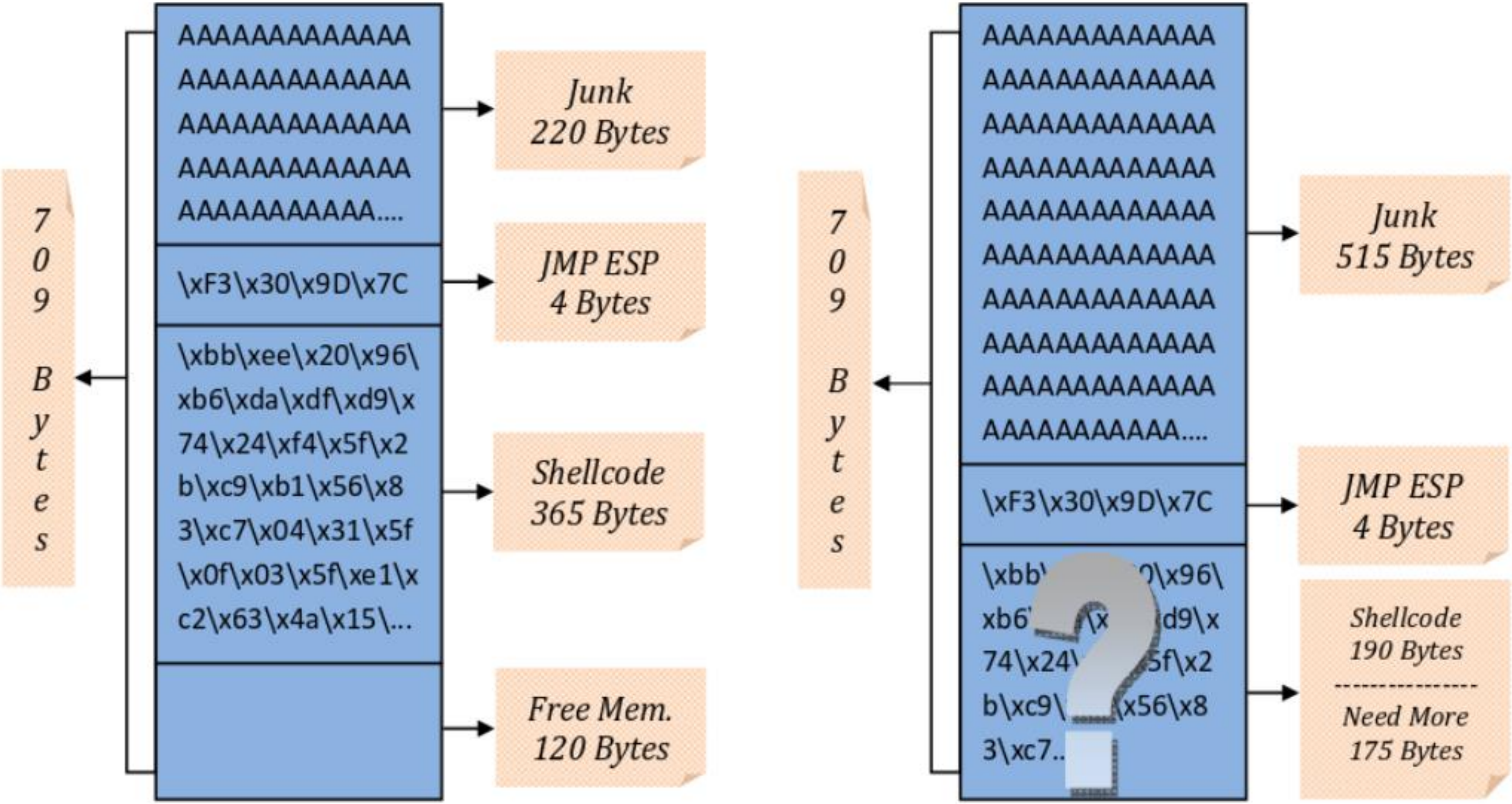
```
[16:54:23] Access violation when reading [41414141] - use Shift+F7/F8/F9 to pass exception to program
```

Egg Hunter

- Egg Hunter shellcode simply means small sized shellcode
- Writing shellcode to Exploit within a Limited space
- Shellcode won't fit in the available space
- Storing User input in the memory for long run than expected.
- Relays on system calls that have ability to traverse process memory

Egg Hunter (contd.)

- Egg Hunter can be generated in Immunity Debugger with the help of Mona.py
  - !mona egg –t r00t3r
- Simple Format of an Egg Hunter shell is:
  - EGGEGG + shellcode
  - Here EGGEGG is nothing, but tag or word repeated twice
- So, Step goes like this:
  1. Write a shellcode in the limited buffer to find EGGEGG
  2. Once the shellcode is executed, then it'll look for both occurrence of EGG
  3. Once EGGEGG is found it'll execute our desired exploit which is present after EGGEGG!



7 0 9 Bytes
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAA....
Junk
220 Bytes
\xF3\x30\x9D\x7C
JMP ESP
4 Bytes
\xbb\xee\x20\x96\
xb6\xda\xdf\xd9\x
74\x24\xf4\x5f\x2
b\xc9\xb1\x56\x8
3\xc7\x04\x31\x5f
\x0f\x03\x5f\xe1\x
c2\x63\x4a\x15\...
Shellcode
365 Bytes
Free Mem.
120 Bytes
7 0 9 Bytes
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAA....
Junk
515 Bytes
\xF3\x30\x9D\x7C
JMP ESP
4 Bytes
\xbb\ 0\x96\
xb6\ \x d9\x
74\x24\ 5f\x2
b\xc9\ x56\x8
3\xc7..
?
Shellcode
190 Bytes
Need More
175 Bytes

Shellcoding

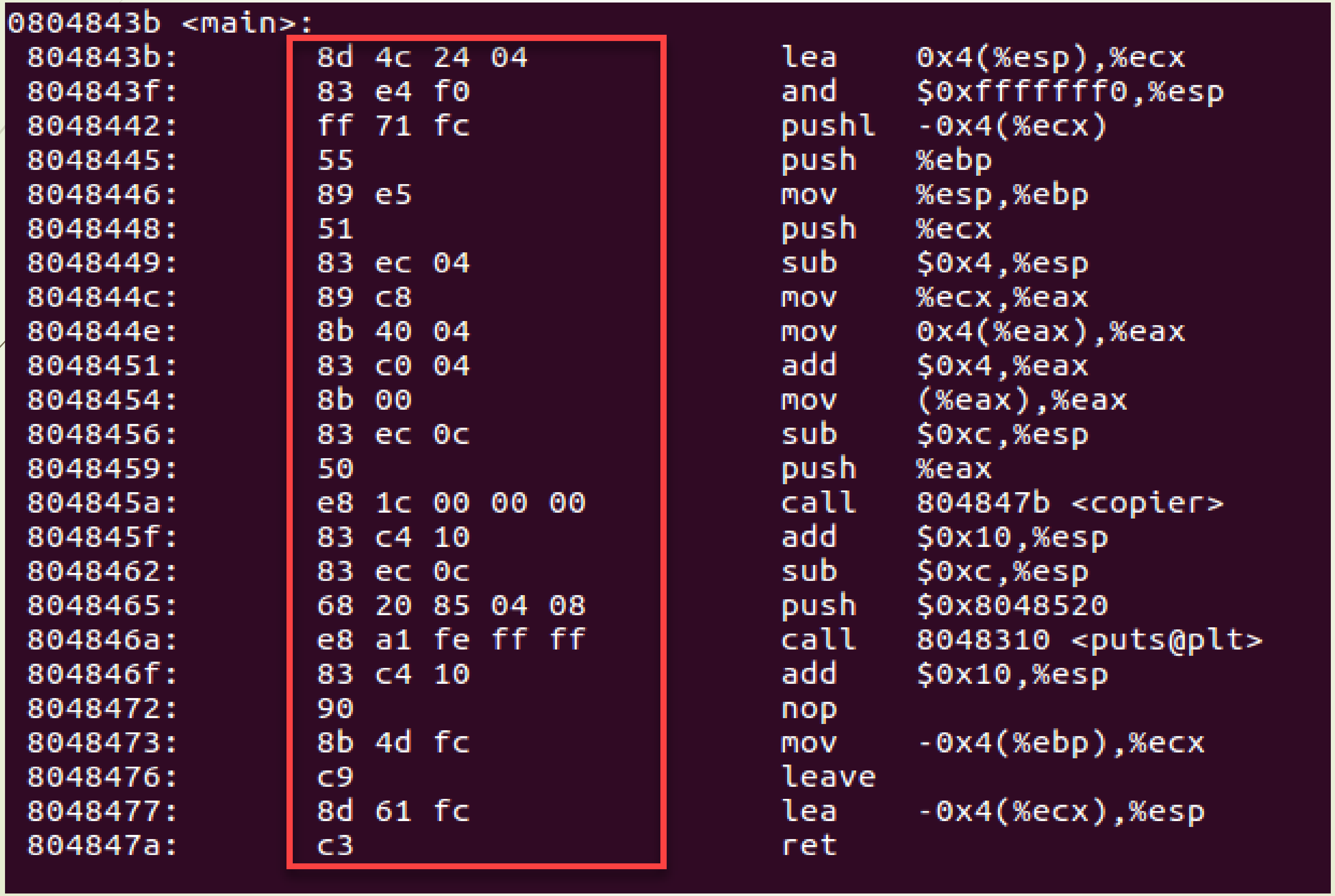
- What is shellcode?
  - Well, it's heart of every exploit.
- It's not even assembly
- It consists of raw processor opcodes
- These are raw bytes that are responsible for executing certain tasks.

Shellcoding (contd.)

- Do I have to write every time a new shellcode to exploit a software?
  - In 21st centaury! No, it's not required.
  - But it's good to know how to write it on your own.

- Then where we can find shellcode?
  - Metasploit-Framework will be our best friend
  - Also, don't forget "shellstorm"
  - Or just use an existing exploit

Shellcoding (contd.)



```
0804843b <main>:
 804843b:       8d 4c 24 04             lea    0x4(%esp),%ecx
 804843f:       83 e4 f0                and    $0xfffffff0,%esp
 8048442:       ff 71 fc                pushl  -0x4(%ecx)
 8048445:       55                      push   %ebp
 8048446:       89 e5                   mov    %esp,%ebp
 8048448:       51                      push   %ecx
 8048449:       83 ec 04                sub    $0x4,%esp
 804844c:       89 c8                   mov    %ecx,%eax
 804844e:       8b 40 04                mov    0x4(%eax),%eax
 8048451:       83 c0 04                add    $0x4,%eax
 8048454:       8b 00                   mov    (%eax),%eax
 8048456:       83 ec 0c                sub    $0xc,%esp
 8048459:       50                      push   %eax
 804845a:       e8 1c 00 00 00          call   804847b <copier>
 804845f:       83 c4 10                add    $0x10,%esp
 8048462:       83 ec 0c                sub    $0xc,%esp
 8048465:       68 20 85 04 08          push   $0x8048520
 804846a:       e8 a1 fe ff ff          call   8048310 <puts@plt>
 804846f:       83 c4 10                add    $0x10,%esp
 8048472:       90                      nop
 8048473:       8b 4d fc                mov    -0x4(%ebp),%ecx
 8048476:       c9                      leave
 8048477:       8d 61 fc                lea    -0x4(%ecx),%esp
 804847a:       c3                      ret
```

Shellcoding (contd.)

1

```
83 e4 f0 ff 71 fc 55 89 e5 51 83 ec 04 89 c8 8b 40 04 83 c0 04 8b 00 83 ec 0c 50
e8 1c 00 00 00 83 c4 10 83 ec 0c 68 20 85 04 08 e8 a1 fe ff ff 83 c4 10 90 8b 4d
fc c9 8d 61 fc c3
```

2

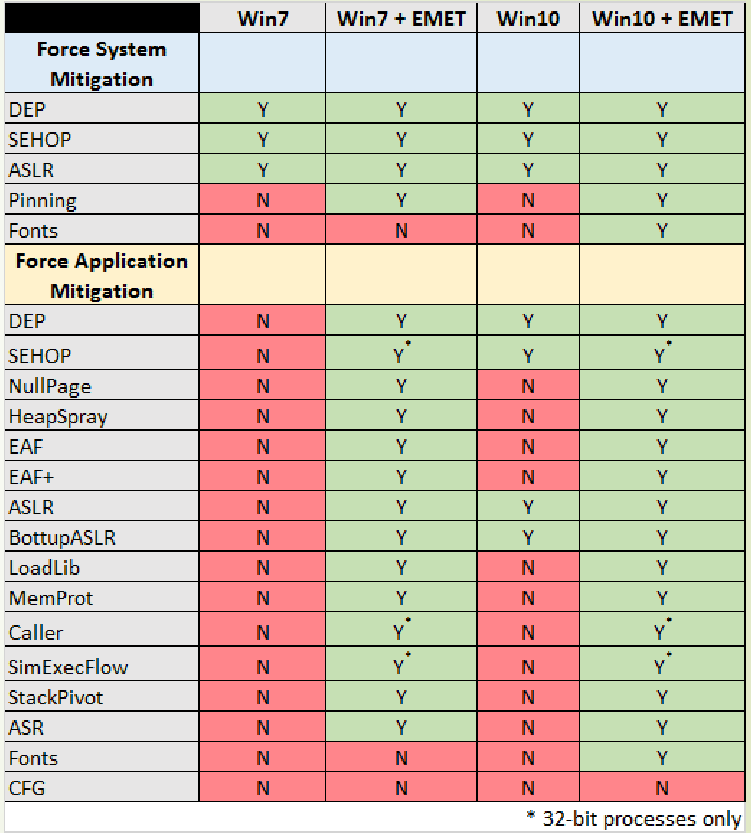
```
\x83\xe4\xf0\xff\x71\xfc\x55\x89\xe5\x51\x83\xec\x04\x89\xc8\x8b\x40\x
04\x83\xc0\x04\x8b\x00\x83\xec\x0c\x50\xe8\x1c\x00\x00\x00\x83\xc4\x
10\x83\xec\x0c\x68\x20\x85\x04\x08\xe8\xa1\xfe\xff\xff\x83\xc4\x10\x90
\x8b\x4d\xfc\xc9\x8d\x61\xfc\xc3
```

NOTE: Don’t forget to rearrange according to Big-endian or little-endian usage, also Bad characters too

Exploit Development Protection

- ASLR – Address Space Layer Randomization
- DEP – Data Execution Prevention
- Stack Canary : Just like coal mine canary
  - SafeSEH
- Tools:
  - EMET (Enhanced Mitigation Experience Toolkit)- DEP, ASLR, SEHOP and more

Note: In newer OS we cannot completely disable ASLR, DEP, SEHOP.



	Win7	Win7 + EMET	Win10	Win10 + EMET
Force System Mitigation				
DEP	Y	Y	Y	Y
SEHOP	Y	Y	Y	Y
ASLR	Y	Y	Y	Y
Pinning	N	Y	N	Y
Fonts	N	N	N	Y
Force Application Mitigation				
DEP	N	Y	Y	Y
SEHOP	N	Y*	Y	Y*
NullPage	N	Y	N	Y
HeapSpray	N	Y	N	Y
EAF	N	Y	N	Y
EAF+	N	Y	N	Y
ASLR	N	Y	Y	Y
BottupASLR	N	Y	Y	Y
LoadLib	N	Y	N	Y
MemProt	N	Y	N	Y
Caller	N	Y*	N	Y*
SimExecFlow	N	Y*	N	Y*
StackPivot	N	Y	N	Y
ASR	N	Y	N	Y
Fonts	N	N	N	Y
CFG	N	N	N	N

* 32-bit processes only

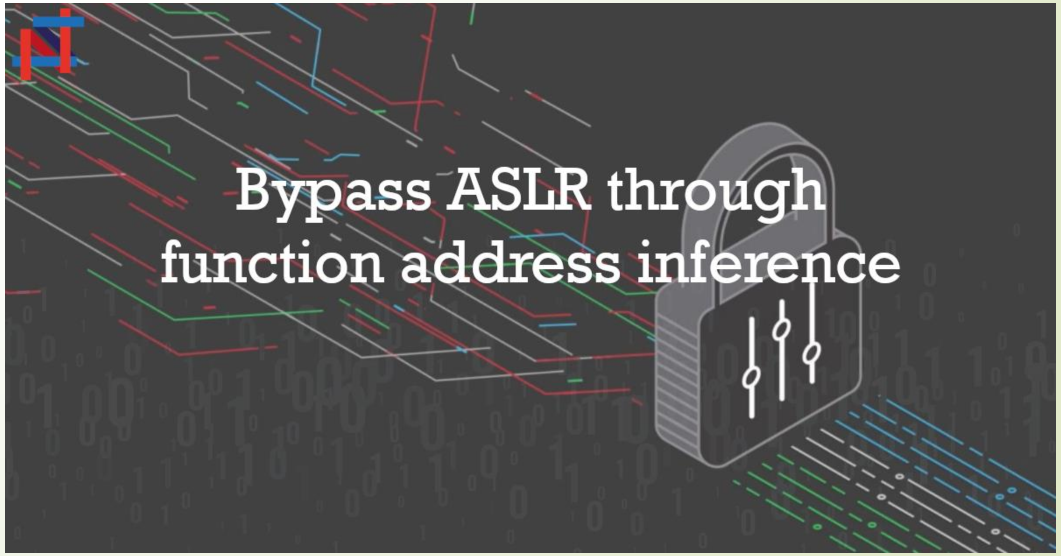
Exploit Development Protection (contd.)

Q: Well, what if we Implement DEF and ALSR together as a protection?

A: Yes, if both are implemented together then code execution is something which gets impossible to achieve in one shot!

Defeating Protection - ASLR

- Search and Use Non-randomized modules for JMP/CALL ESP
- Brute-force : Force ALSR to overwrite return point so that we can reach to our shellcode
- Nop - Sled



Defeating Protection (contd.) - DEP

Exploit Research

DEP Bypass with ROP Chains

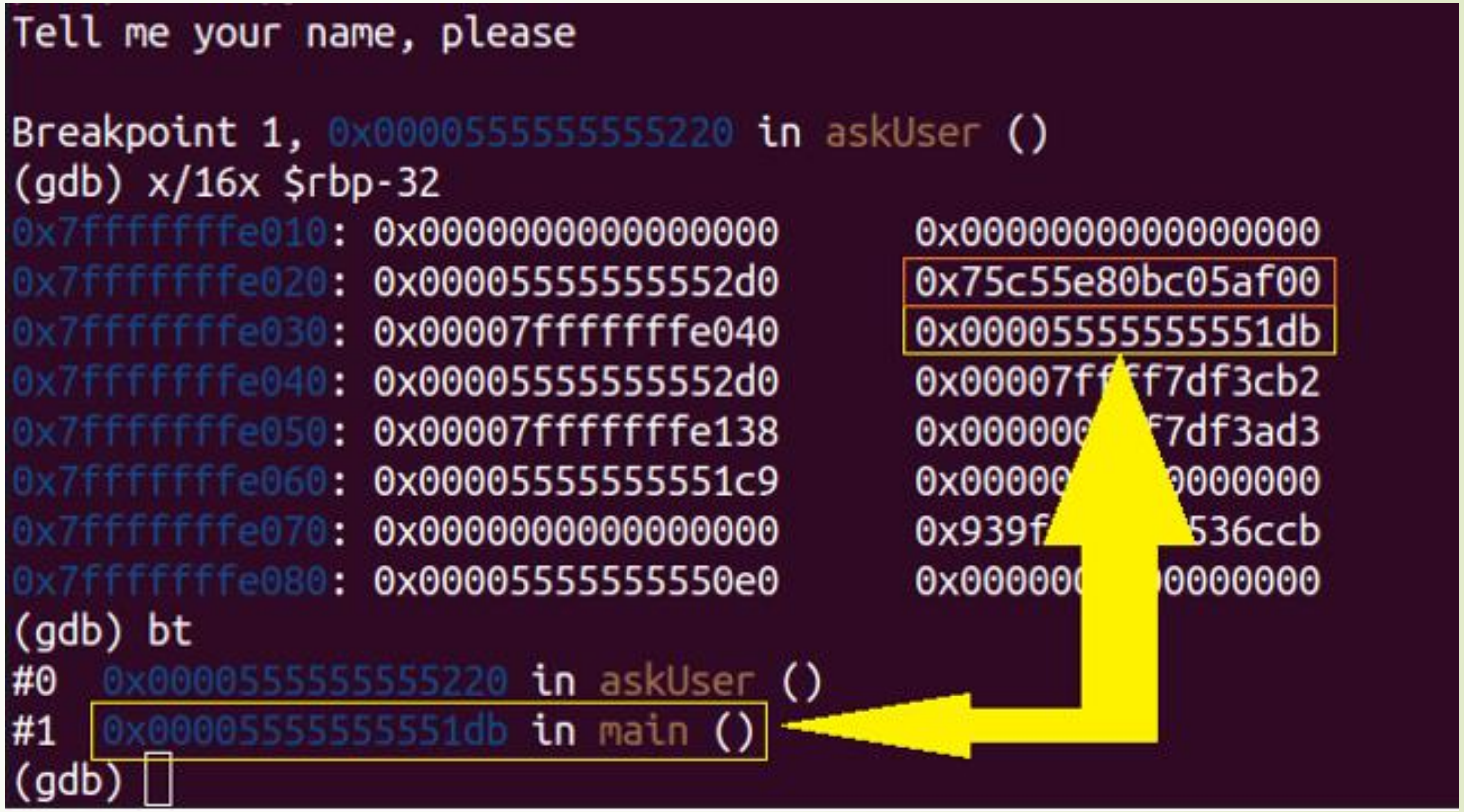
OPEN SECURITY
WWW.OPENSECURITY.IN
AJIN ABRAHAM
@AJINABRAHAM

- Return Oriented Programming (ROP)
  - Finding Multiple machine instructions in the program
  - Instructions are part of the stack, so no DEP involved
- Again, for ROP we have mona.py too

Defeating Protection (contd.) Stack Canary

- Hit and Trail : Try to find or guess the canary value

- David Litchfield
    - Defeating Stack Protection through SEH



```
Tell me your name, please

Breakpoint 1, 0x0000555555555220 in askUser ()
(gdb) x/16x $rbp-32
0x7fffffffe010: 0x0000000000000000      0x0000000000000000
0x7fffffffe020: 0x00005555555552d0      0x75c55e80bc05af00
0x7fffffffe030: 0x00007fffffffe040      0x00005555555551db
0x7fffffffe040: 0x00005555555552d0      0x00007ff  f7df3cb2
0x7fffffffe050: 0x00007fffffffe138      0x000000   f7df3ad3
0x7fffffffe060: 0x00005555555551c9      0x00000     0000000
0x7fffffffe070: 0x0000000000000000      0x939f       536ccb
0x7fffffffe080: 0x00005555555550e0      0x000000    0000000
(gdb) bt
#0  0x0000555555555220 in askUser ()
#1  0x00005555555551db in main ()
(gdb) 
```

Reminder !!!

Exploitation Demo Please!

Reference

- Basics - Link
- Buffer Overflow – Link
- Vulnserver Stack Smashing - Link
- Exploit Development - Link
- Exploit Protection - Link
- Egg Hunter - Link
- mona.py manual - Link
- SEH - Link
- ROP - Link
- David Litchfield Paper (bypass Stack-based Overflow) – Link

Thank You