

| base | SafeSEH | ASLR | NXCompat | OS Dll | Version, Modulename & Path |
|------|---------|-------|----------|--------|--|
| true | True | False | False | True | 6.3.9600.16384 [NLAppl.dll] (C:\Windows\system32\NLAppl.dll) |
| true | 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) |
| true | True | False | False | True | 7.0.9600.16384 [MSUCRT.dll] (C:\Windows\SYSTEM32\MSUCRT.dll) |
| true | True | False | False | True | 6.3.9600.16384 [CRYPTBASE.dll] (C:\Windows\SYSTEM32\CRYPTBASE.dll) |
| true | True | False | False | True | 6.3.9600.16384 [oledlg.dll] (C:\Windows\SYSTEM32\oledlg.dll) |
| true | True | False | False | True | 6.3.9600.16384 [dwmapi.dll] (C:\Windows\system32\dwmapi.dll) |
| true | True | False | False | True | 6.3.9600.17031 [ord.dll] (C:\Windows\SYSTEM32\ord.dll) |
| true | True | False | False | True | 6.3.9600.16384 [sechost.dll] (C:\Windows\SYSTEM32\sechost.dll) |
| true | True | False | False | True | 6.3.9600.16384 [USERENV.dll] (C:\Windows\SYSTEM32\USERENV.dll) |
| true | True | False | False | True | 6.3.9600.16384 [combase.dll] (C:\Windows\SYSTEM32\combase.dll) |
| true | False | False | False | False | -1.0- [LIBEAY32.dll] (C:\EFS Software2\Easy Chat Server\LIBEAY32.dll) |
| true | True | False | False | True | 11.00.9600.16384 [iertutil.dll] (C:\Windows\SYSTEM32\iertutil.dll) |
| true | True | False | False | True | 11.00.9600.16384 [WININET.dll] (C:\Windows\SYSTEM32\WININET.dll) |
| true | True | False | False | True | 6.3.9600.16384 [fwpuclnt.dll] (C:\Windows\System32\fwpuclnt.dll) |
| true | True | False | False | True | 6.3.9600.17031 [KERNEL32.DLL] (C:\Windows\SYSTEM32\KERNEL32.DLL) |
| true | True | False | False | True | 6.3.9600.16384 [MINIS1.DLL] (C:\Windows\SYSTEM32\MINIS1.DLL) |
| true | True | False | False | True | 6.3.9600.16384 [MSOCK32.dll] (C:\Windows\SYSTEM32\MSOCK32.dll) |
| true | True | False | False | True | 6.3.9600.16409 [SspiCli.dll] (C:\Windows\SYSTEM32\SspiCli.dll) |
| true | True | False | False | True | 6.3.9600.16384 [rsaenh.dll] (C:\Windows\system32\rsaenh.dll) |
| true | True | False | False | True | 6.3.9600.16384 [ole32.dll] (C:\Windows\SYSTEM32\ole32.dll) |
| true | True | False | False | True | 6.3.9600.16384 [SHLWAPI.dll] (C:\Windows\SYSTEM32\SHLWAPI.dll) |
| true | True | False | False | True | 6.3.9600.16384 [CRYPTSP.dll] (C:\Windows\SYSTEM32\CRYPTSP.dll) |
| true | True | False | False | True | 6.3.9600.16384 [USER32.dll] (C:\Windows\SYSTEM32\USER32.dll) |
| true | True | False | False | True | 6.3.9600.16384 [condlg32.dll] (C:\Windows\SYSTEM32\condlg32.dll) |
| true | True | False | False | True | 6.3.9600.16384 [kernel.appcore.dll] (C:\Windows\SYSTEM32\kernel.appcore.dll) |
| true | True | False | False | True | 6.3.9600.16384 [IPHLAPI.DLL] (C:\Windows\SYSTEM32\IPHLAPI.DLL) |
| true | True | False | False | True | 6.3.9600.16384 [napinsp.dll] (C:\Windows\system32\napinsp.dll) |
| true | True | False | False | True | 6.3.9600.16384 [uxtheme.dll] (C:\Windows\system32\uxtheme.dll) |
| true | True | False | False | True | 6.3.9600.16506 [OLEAUT32.dll] (C:\Windows\SYSTEM32\OLEAUT32.dll) |
| true | True | False | False | True | 6.3.9600.16384 [profapi.dll] (C:\Windows\SYSTEM32\profapi.dll) |
| true | True | False | False | True | 6.3.9600.17031 [SHELL32.dll] (C:\Windows\SYSTEM32\SHELL32.dll) |
| true | True | False | False | True | 6.3.9600.16384 [RPCRT4.dll] (C:\Windows\SYSTEM32\RPCRT4.dll) |
| true | True | False | False | True | 6.3.9600.16384 [DNSAPI.dll] (C:\Windows\SYSTEM32\DNSAPI.dll) |
| true | True | False | False | True | 6.3.9600.17031 [IMM32.DLL] (C:\Windows\system32\IMM32.DLL) |
| true | True | False | False | True | 6.3.9600.16384 [winnr.dll] (C:\Windows\System32\winnr.dll) |
| true | True | False | False | True | 6.10 [CONCTL32.dll] (C:\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b6414eccf146_x-ww_6.9.9600.16384_x-ww_6595b6414eccf146_x-ww\CONCTL32.dll) |
| true | True | False | False | True | 6.3.9600.16384 [MSCTF.dll] (C:\Windows\SYSTEM32\MSCTF.dll) |
| true | True | False | False | True | 6.3.9600.16384 [OLEPRO32.DLL] (C:\Windows\SYSTEM32\OLEPRO32.DLL) |
| true | True | False | False | True | 6.3.9600.17031 [KERNELBASE.dll] (C:\Windows\SYSTEM32\KERNELBASE.dll) |
| true | True | False | False | True | 6.3.9600.16384 [SHCORE.DLL] (C:\Windows\SYSTEM32\SHCORE.DLL) |
| true | True | False | False | True | 6.3.9600.16384 [mswsock.dll] (C:\Windows\System32\mswsock.dll) |
| true | 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) |
| true | 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

[Linkedin/sh377c0d3](#)

Twitter: [@sh377c0d3](#)

[Github/sh377c0d3](#)



Disclaimer



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



MOST OF THE CONTENT IS SOLELY GATHERED
FROM PUBLICLY AVAILABLE RESOURCES THAT
INCLUDE IMAGES AS WELL AS CONTENTS.

Agenda

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





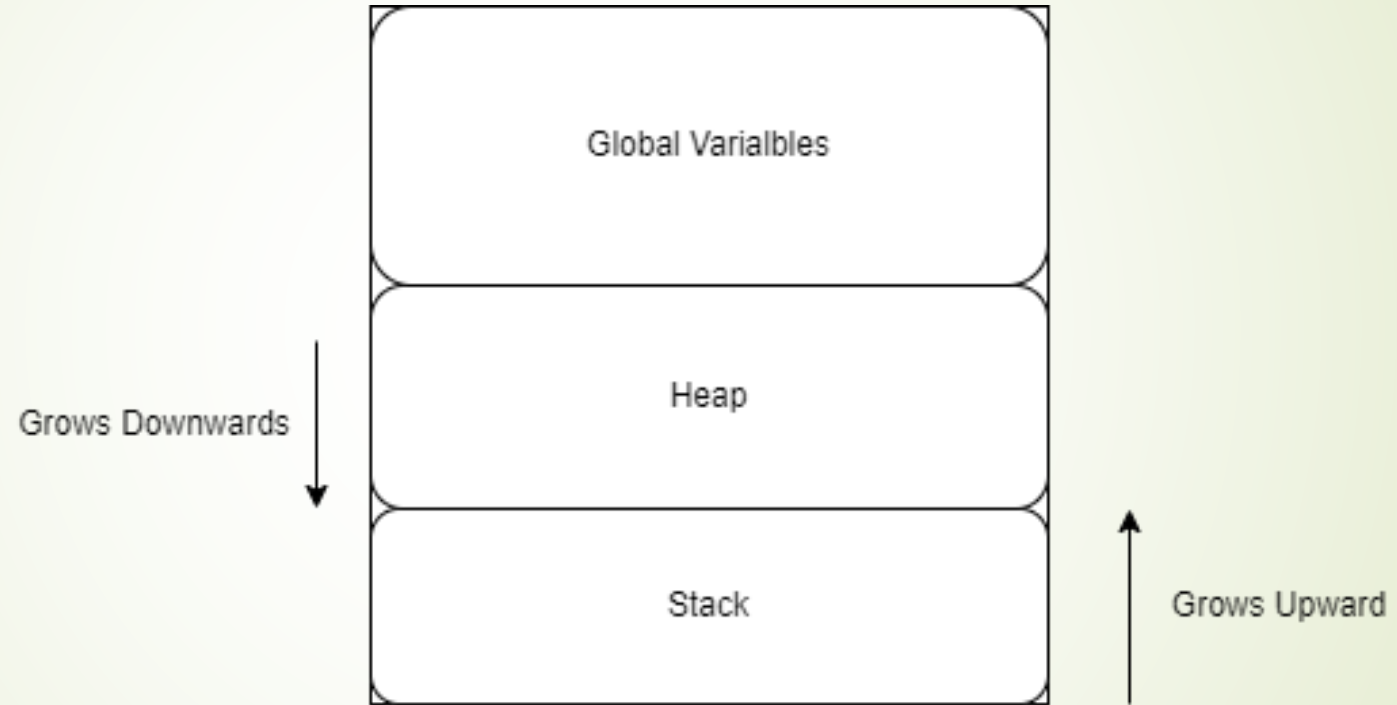
Basics

- 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?

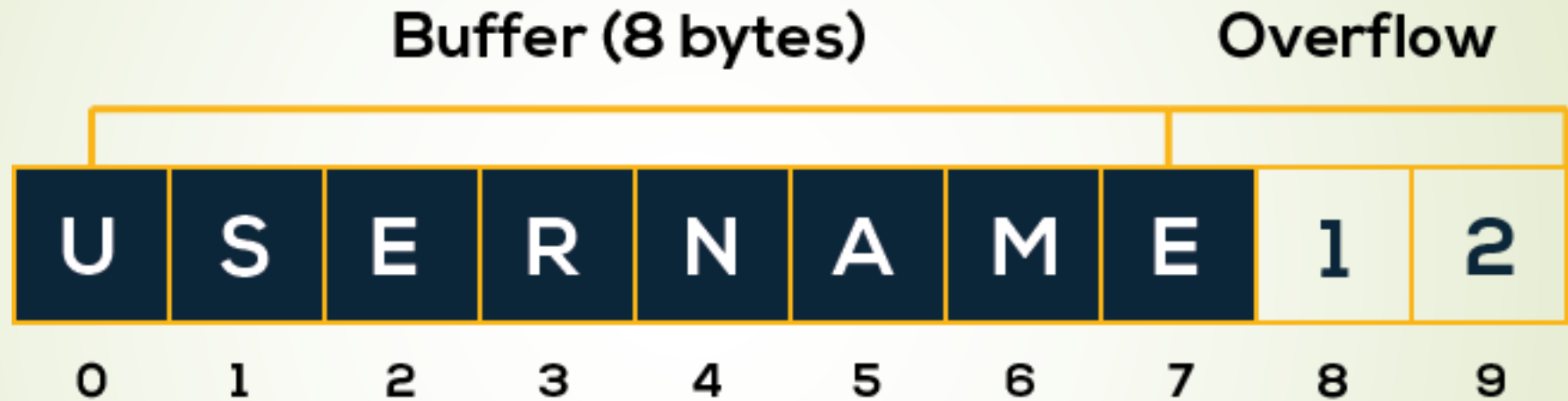
Basics (contd.) - Fuzzing

- 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("anonymous");  
s_string("\r\n");  
s_string("PASS ");  
s_string("anonymous");  
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

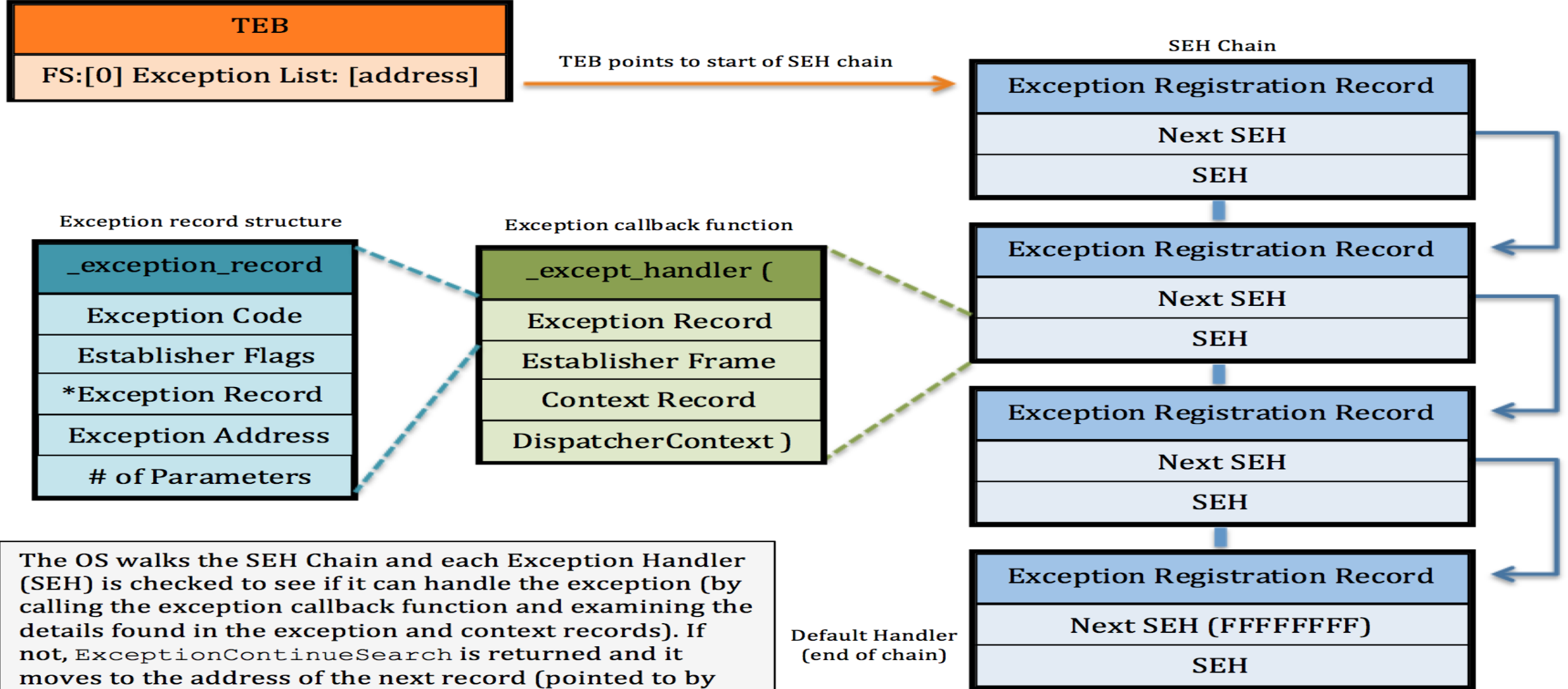
- A Windows feature that handles application's exceptions.
- Mechanism used by programmers
 - helps applications handle any unexpected conditions encountered during a program's runtime.
- Exception happens
 - Windows will pop-up a familiar dialog box to us
 - Which state that "Application Encountered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



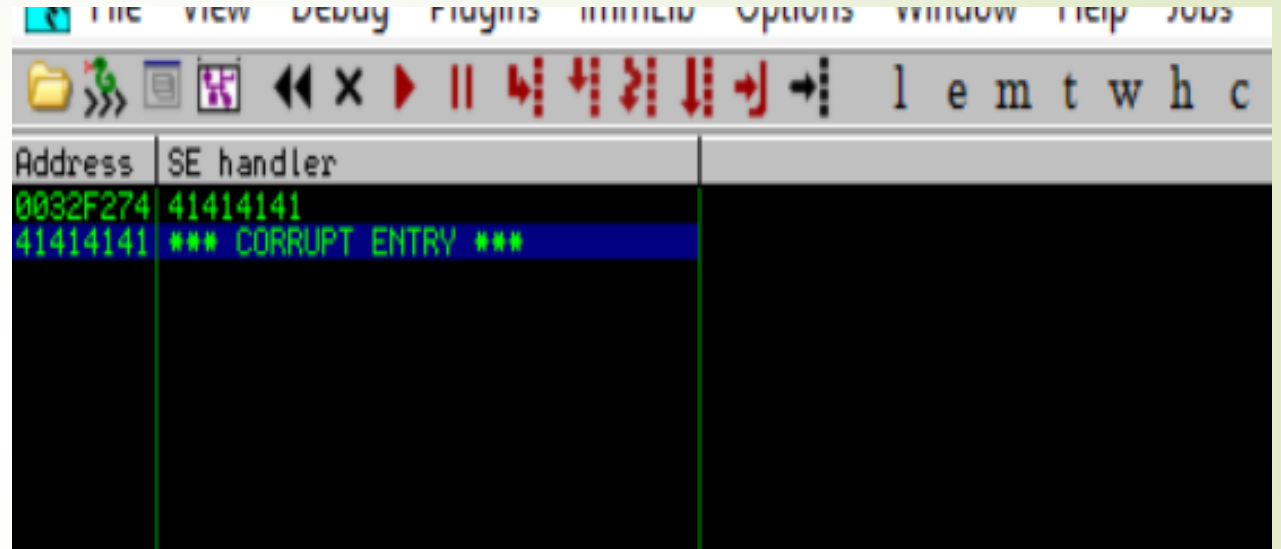
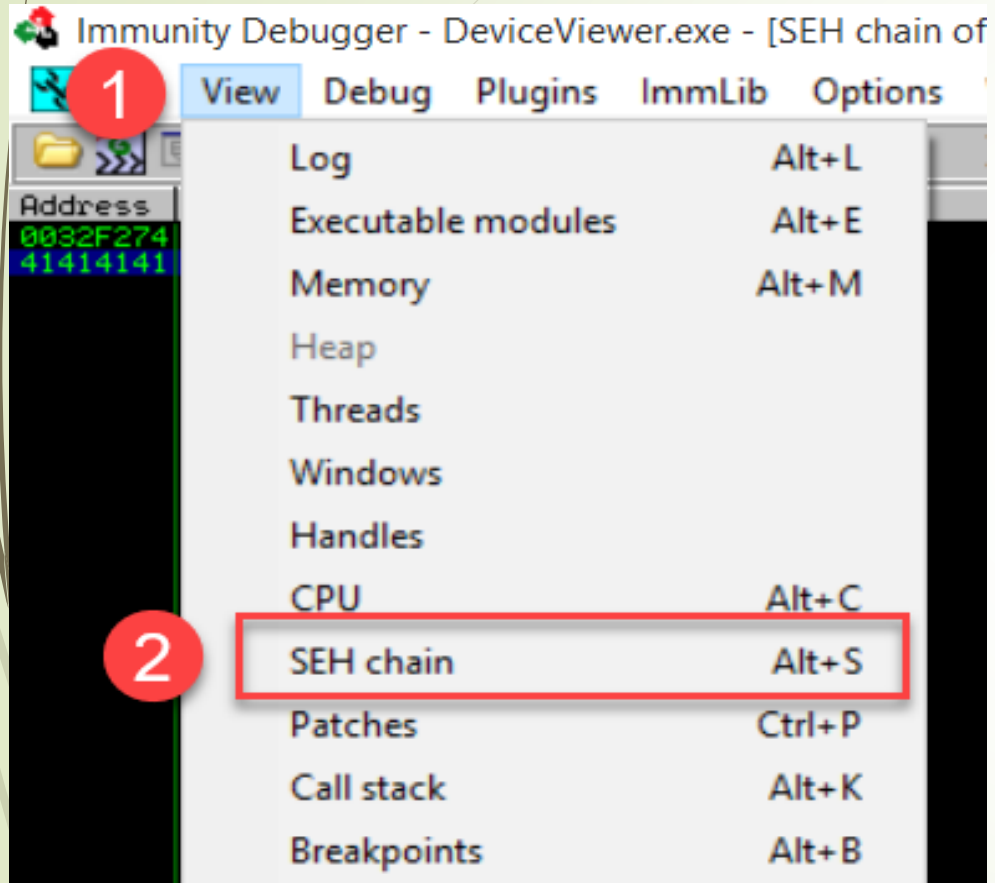
SEH (contd.)

- 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)



SEH (contd.)



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

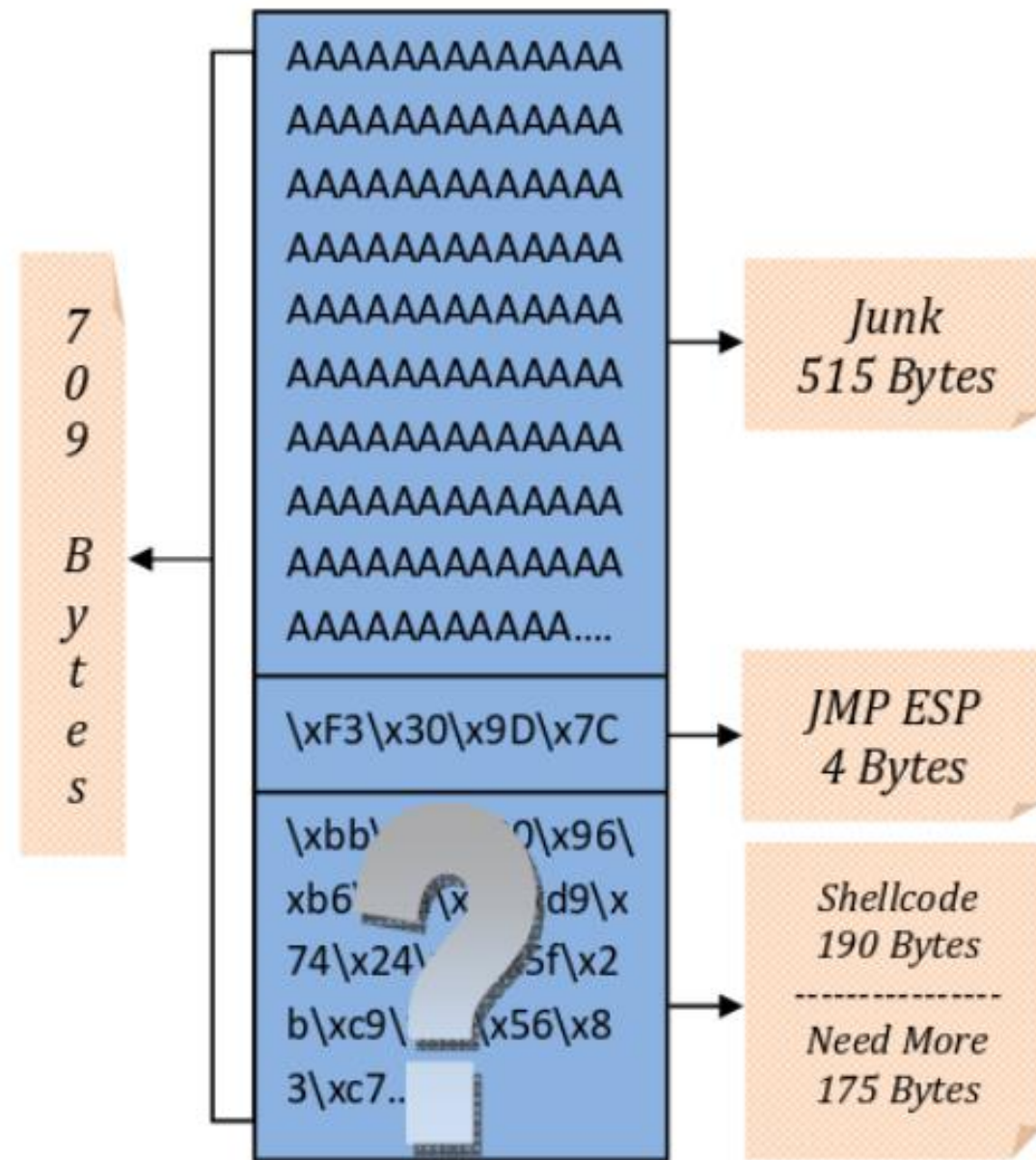
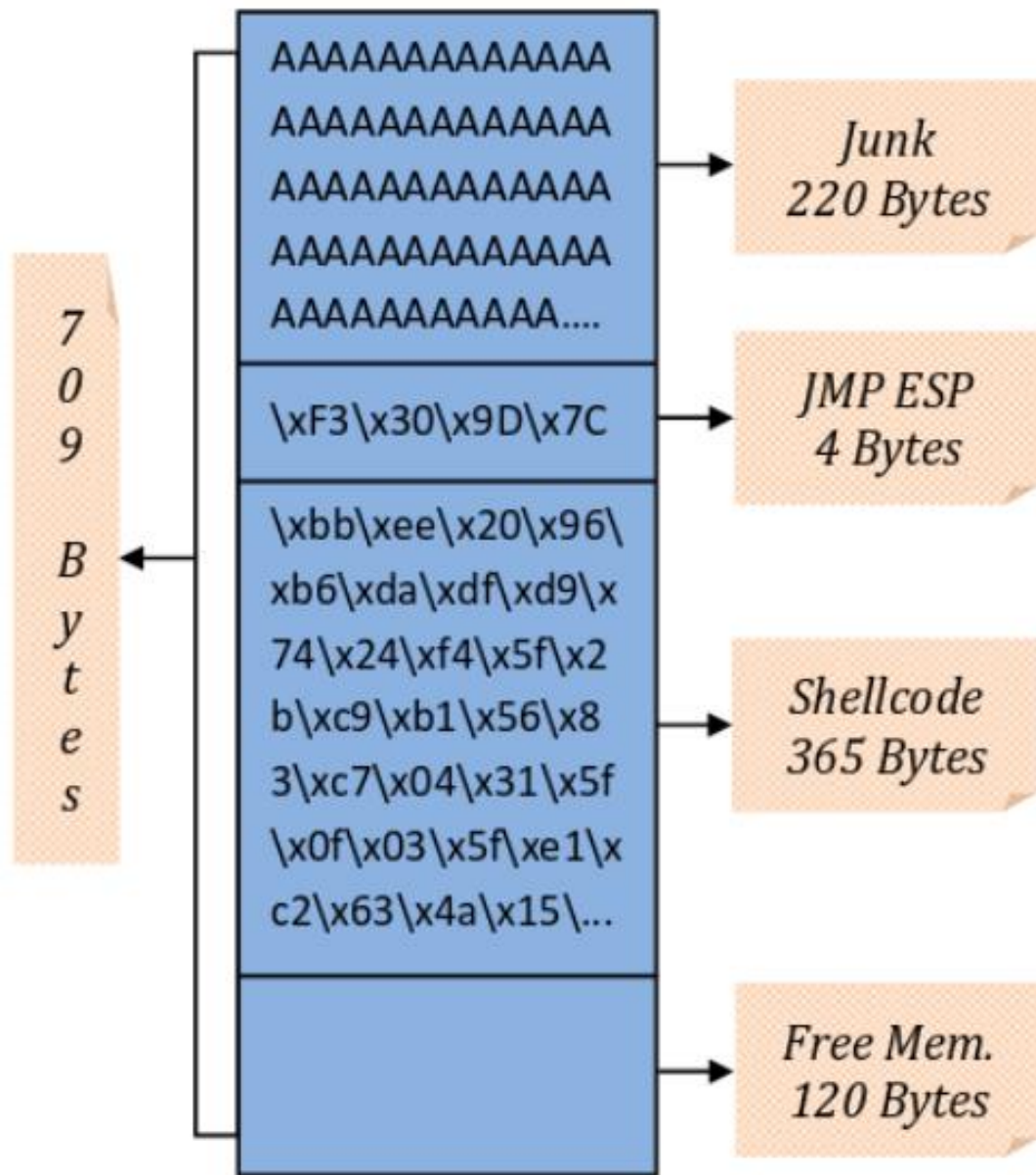


Egg Hunter

- 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!





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 century! 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
804843f: 83 e4 f0
8048442: ff 71 fc
8048445: 55
8048446: 89 e5
8048448: 51
8048449: 83 ec 04
804844c: 89 c8
804844e: 8b 40 04
8048451: 83 c0 04
8048454: 8b 00
8048456: 83 ec 0c
8048459: 50
804845a: e8 1c 00 00 00
804845f: 83 c4 10
8048462: 83 ec 0c
8048465: 68 20 85 04 08
804846a: e8 a1 fe ff ff
804846f: 83 c4 10
8048472: 90
8048473: 8b 4d fc
8048476: c9
8048477: 8d 61 fc
804847a: c3
```

```
lea    0x4(%esp),%ecx
and     $0xffffffff0,%esp
pushl   -0x4(%ecx)
push    %ebp
mov     %esp,%ebp
push    %ecx
sub     $0x4,%esp
mov     %ecx,%eax
mov     0x4(%eax),%eax
add     $0x4,%eax
mov     (%eax),%eax
sub     $0xc,%esp
push    %eax
call    804847b <copier>
add     $0x10,%esp
sub     $0xc,%esp
push    $0x8048520
call    8048310 <puts@plt>
add     $0x10,%esp
nop
mov     -0x4(%ebp),%ecx
leave
lea     -0x4(%ecx),%esp
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

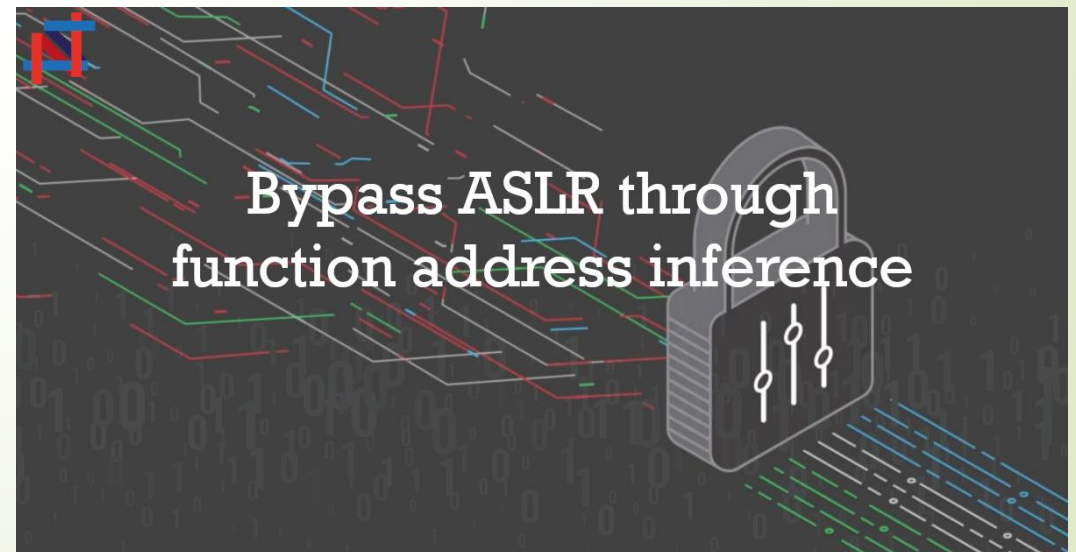
Exploit Development Protection (contd.)

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 ASLR to overwrite return point so that we can reach to our shellcode
- Nop - Sled



Exploit Research

DEP Bypass with ROP Chains

OPEN SECURITY
WWW.OPENSECURITY.IN
AJIN ABRAHAM
@AJINABRAHAM

Defeating Protection (contd.) - DEP

- 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, 0x000055555555220 in askUser ()

(gdb) x/16x \$rbp-32

| | | |
|------------------|---------------------|--------------------|
| 0x7fffffffef010: | 0x0000000000000000 | 0x0000000000000000 |
| 0x7fffffffef020: | 0x0000555555552d0 | 0x75c55e80bc05af00 |
| 0x7fffffffef030: | 0x00007fffffffef040 | 0x0000555555551db |
| 0x7fffffffef040: | 0x0000555555552d0 | 0x00007ffff7df3cb2 |
| 0x7fffffffef050: | 0x00007fffffffef138 | 0x00000000f7df3ad3 |
| 0x7fffffffef060: | 0x0000555555551c9 | 0x0000000000000000 |
| 0x7fffffffef070: | 0x0000000000000000 | 0x939f0000536ccb |
| 0x7fffffffef080: | 0x0000555555550e0 | 0x0000000000000000 |

(gdb) bt

#0 0x000055555555220 in askUser ()

#1 0x0000555555551db 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

