Offensive Software Exploitation

Summer 2020

Ali Hadi

@binaryz0ne

Part #5

Exploit Mitigation – Part #1

Preventing JMP/CALL ESP …

Exploit Mitigation

Cited [2]

- Finding and fixing every vulnerability is impossible
- It is possible to make exploitation more difficult through:
  - Memory page protection
  - Run-time validation
  - Obfuscation and Randomization
- Making every vulnerability non-exploitable is impossible

Types of Mitigation

- Compile Time Techniques
  - Stack Guards
  - SEH
- Runtime Techniques
  - DEP
  - ASLR
- Combination of both such as Control Flow Guard (CFG)

Timeline of Mitigation

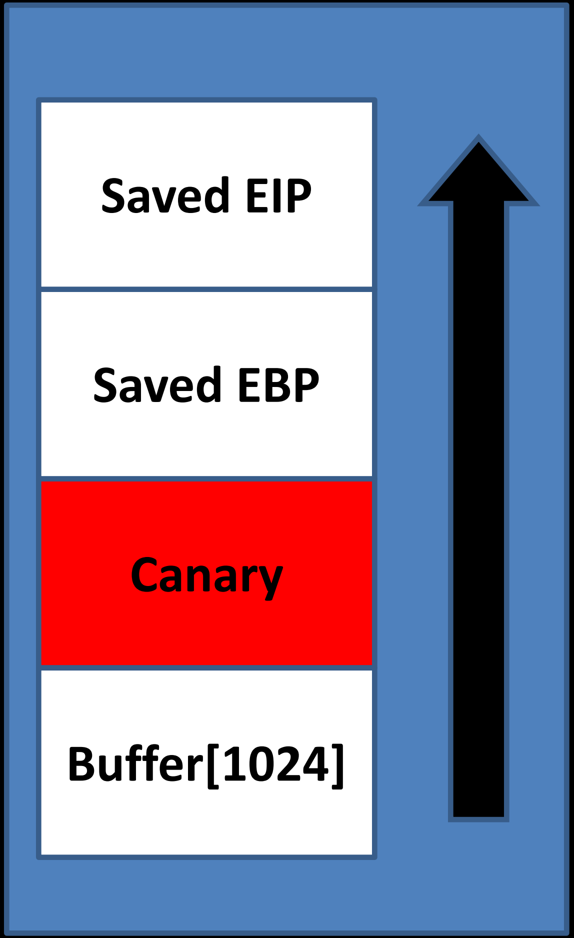
Cited [2]

- Windows 1.0 - Windows XP SP1
  - Corruption of stack and heap metadata is possible
- Windows Server 2003 RTM
  - Operating System is compiled with stack cookies
- Windows XP SP 2
  - Stack/heap cookies, SafeSEH, Software/Hardware DEP
- Windows Vista
  - Address Space Layout Randomization

Visual Studio /GS Flag

Cited [2]

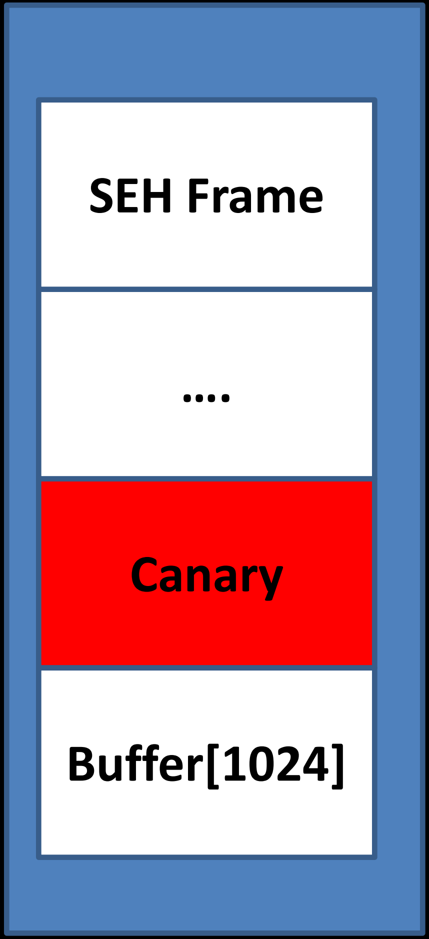
- Place a random "cookie" in the stack frame before frame pointer and return address
- Check cookie before using saved frame pointer and return address



Structured Exception Handling

Cited [2]

- Supports try, except blocks in C and C++ exceptions
- Nested SEH frames are stored on the stack
- Contain pointer to next frame and exception filter function pointer



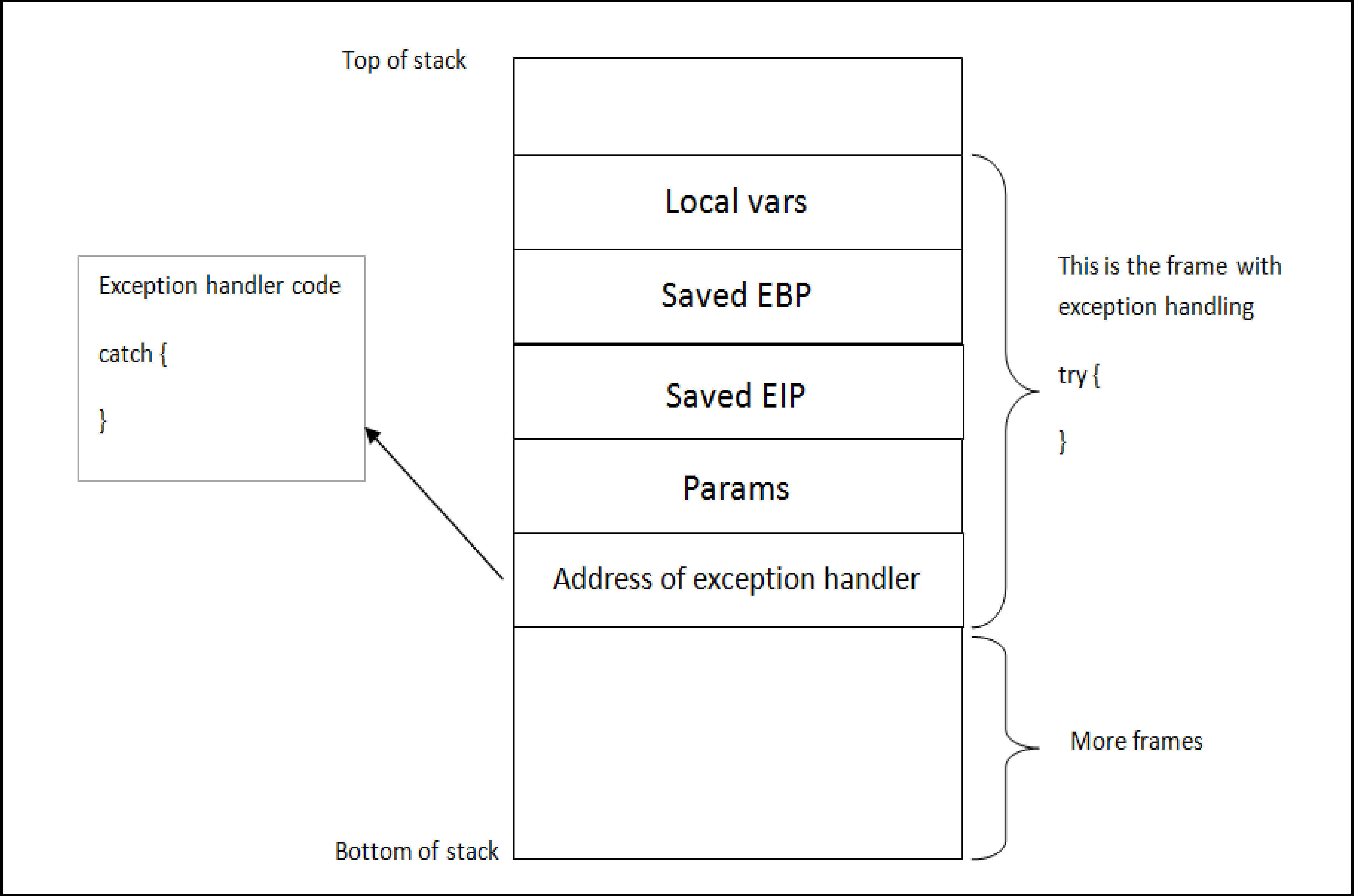


Figure originally from Peter “corelanc0d3r”, http://www.corelan.be/

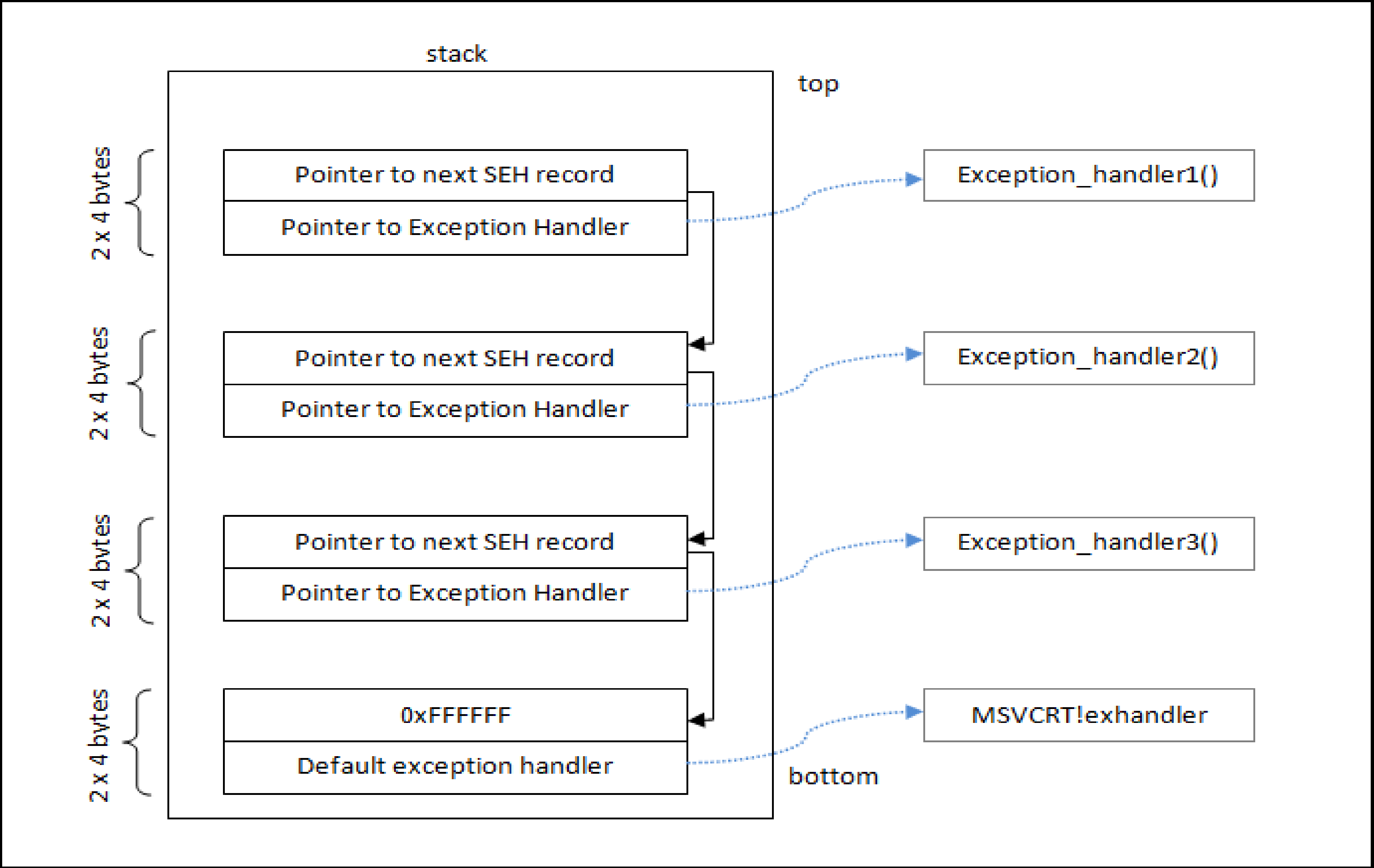

stack
top
2 x 4 bytes
Pointer to next SEH record
Pointer to Exception Handler
Exception_handler1()
2 x 4 bytes
Pointer to next SEH record
Pointer to Exception Handler
Exception_handler2()
2 x 4 bytes
Pointer to next SEH record
Pointer to Exception Handler
Exception_handler3()
2 x 4 bytes
0xFFFFFF
Default exception handler
MSVCRT!exhandler
bottom

Figure originally from Peter “corelanc0d3r”, http://www.corelan.be/

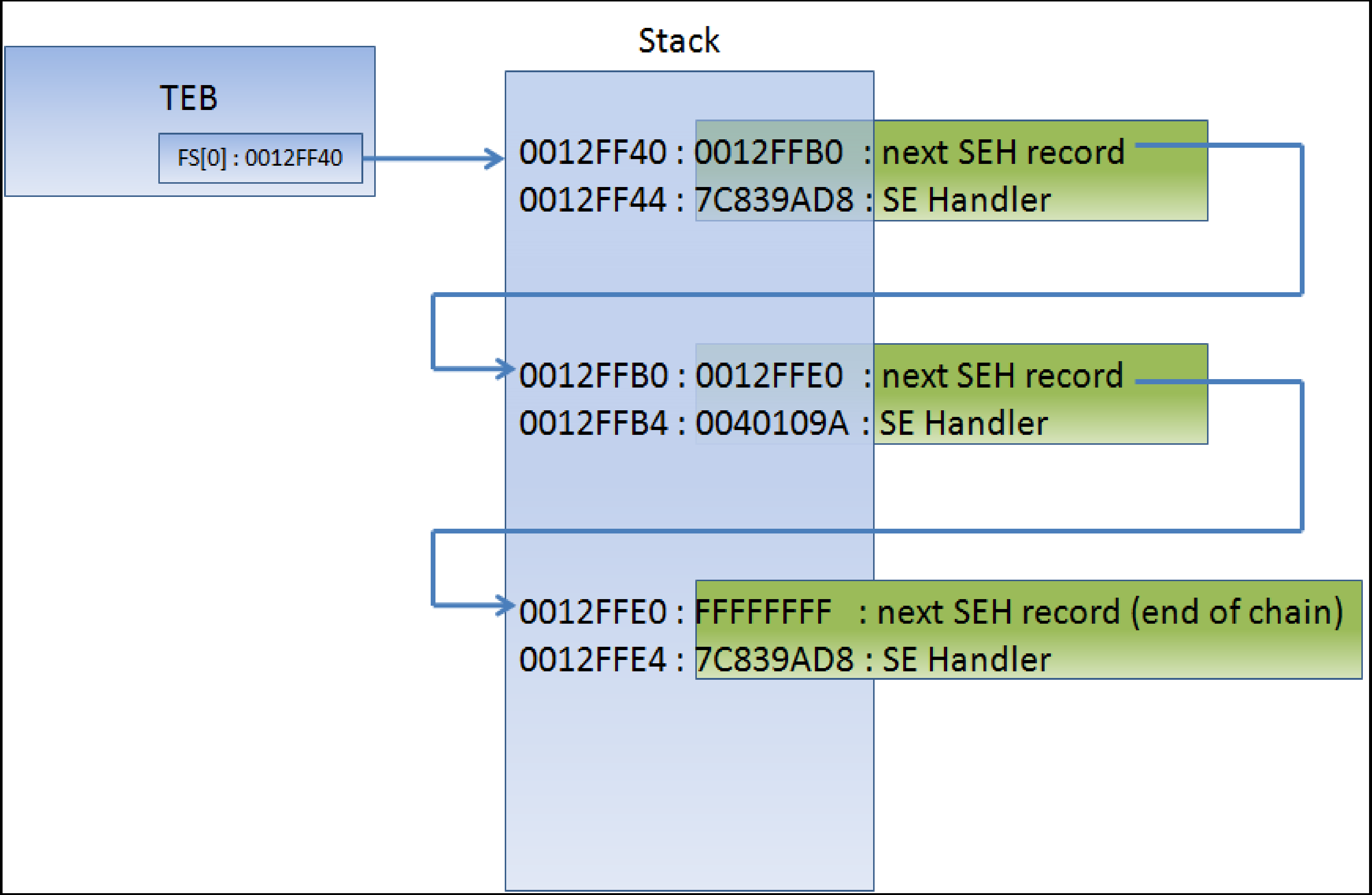

Stack
TEB
FS[0] : 0012FF40
0012FF40 : 0012FFB0 : next SEH record
0012FF44 : 7C839AD8 : SE Handler
0012FFB0 : 0012FFE0 : next SEH record
0012FFB4 : 0040109A : SE Handler
0012FFE0 : FFFFFFFF : next SEH record (end of chain)
0012FFE4 : 7C839AD8 : SE Handler

Figure originally from Peter “corelanc0d3r”, http://www.corelan.be/

Visual Studio /SafeSEH

Cited [2]

- Pre-registers all exception handlers in the DLL or EXE
- When an exception occurs, Windows will examine the pre-registered table and only call the handler if it exists in the table
- What if one DLL wasn't compiled w/ SafeSEH?
  - Windows will allow any address in that module as an SEH handler
  - This allows an attacker to still gain full control

References

[1] Peter "Corelanc0d3r", Exploit Writing (Jumping to Shellcode), https://www.corelan.be/index.php/2009/07/23/writing-buffer-overflow-exploits-a-quick-and-basic-tutorial-part-2/

[2] Memory Corruption 101, NYU Poly, Dino Dai Zovi

[3] Vulnserver, Stephen Bradshaw, http://grey-corner.blogspot.com/,

[4] Grayhat Hacking: The Ethical Hacker's Handbook, 3rd Edition

[5] The Shellcoders Handbook

[6] Exploit-DB: http://www.exploit-db.com/

[7] The Art of Exploitation, 2nd Edition

[8] Vulnerability Discovery, http://www.thegreycorner.com/2010/01/introduction-to-vulnerability-discovery.html

[9] SEH Based Overflow Exploit Tutorial, http://resources.infosecinstitute.com/seh-exploit/