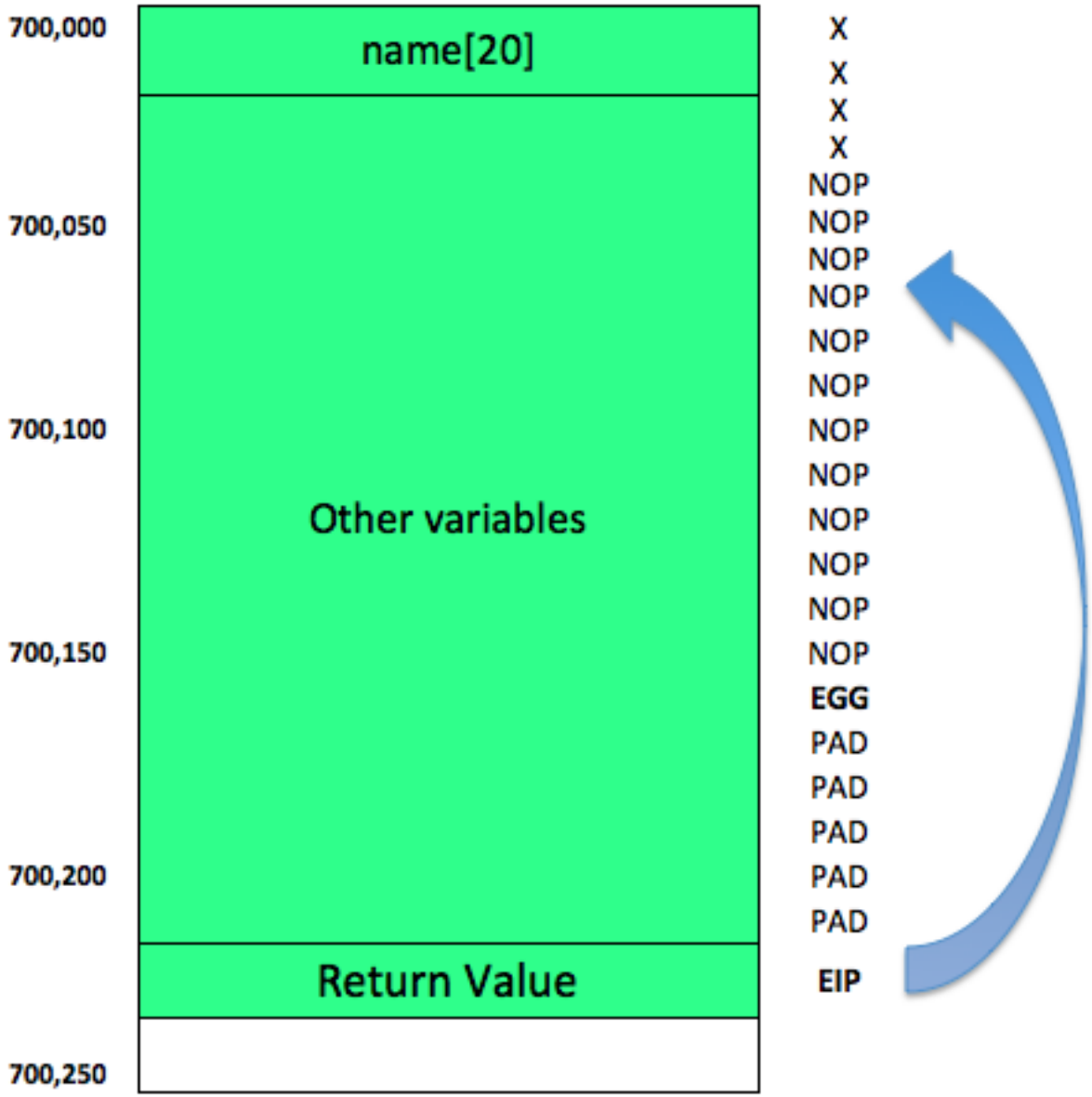
缓冲区溢出

- 覆盖栈中的函数返回地址
- 当函数返回(ret)时，EIP被设置成以上地址
- 该地址指向NOP Sled，并进一步滑向shellcode

Buffer Overflow Using Return Value

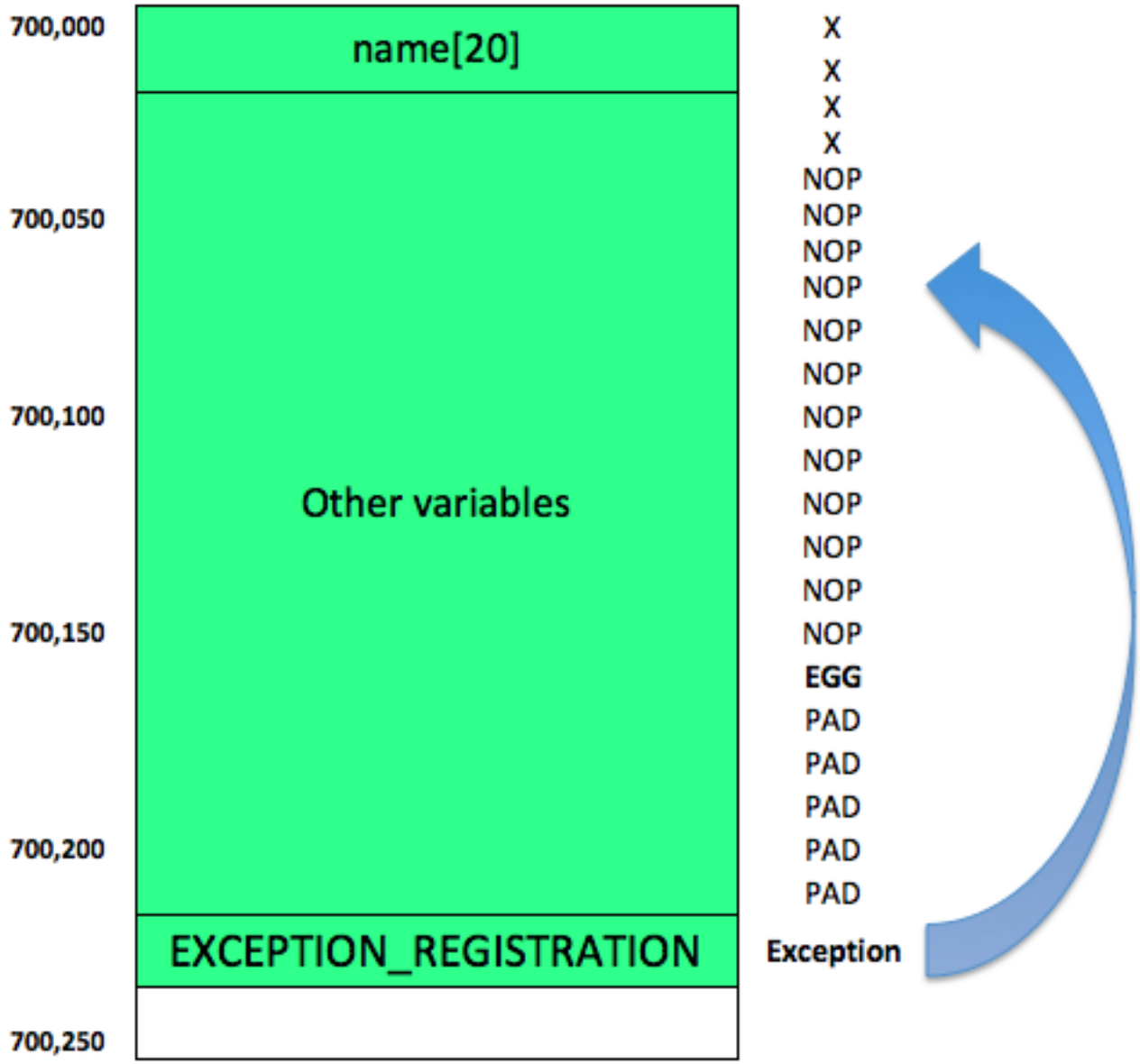


SEH

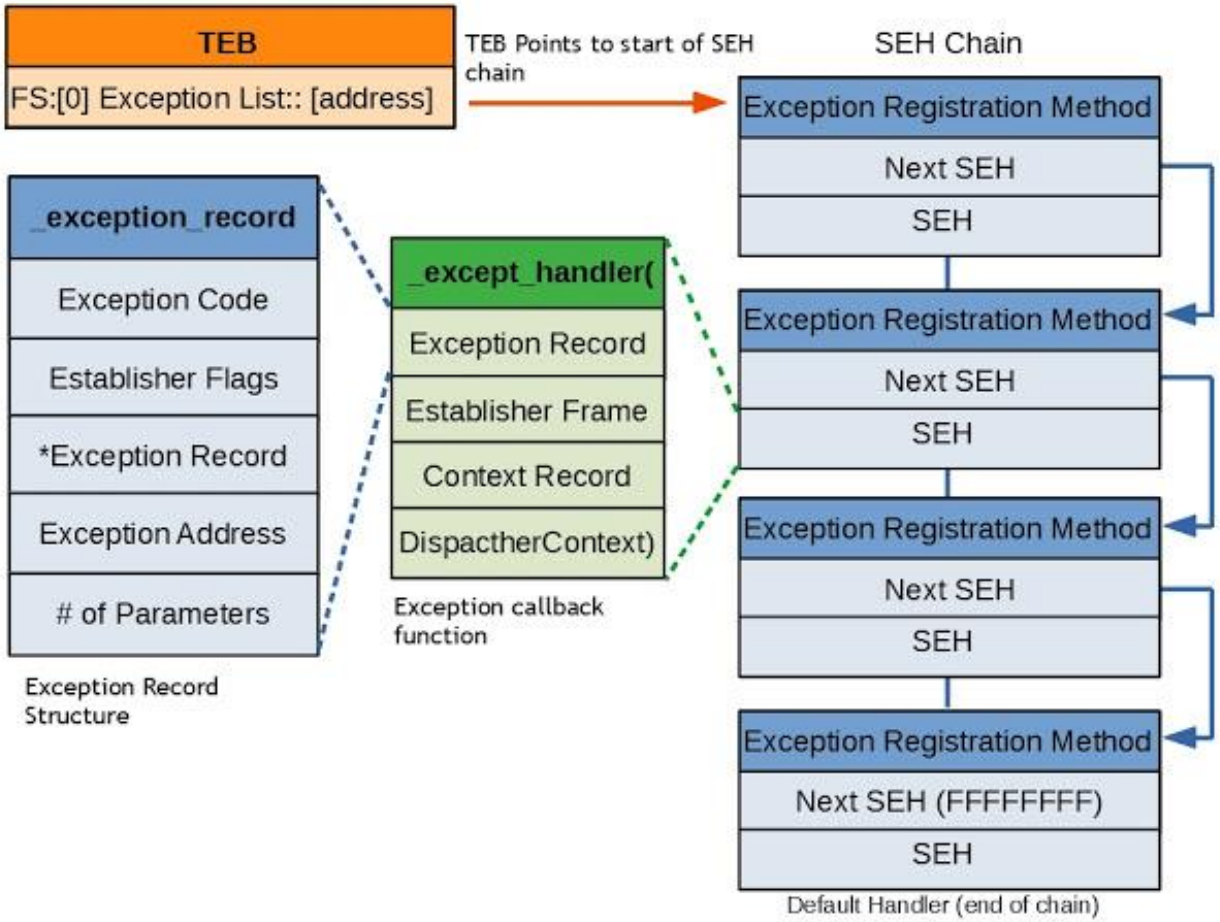
- SEH（Structured Exception Handling）是Windows提供的异常处理机制。

- 覆盖SEH
- 触发exception
- 修改 exception handler， 指向 NOP Sled，并进一步滑向shellcode

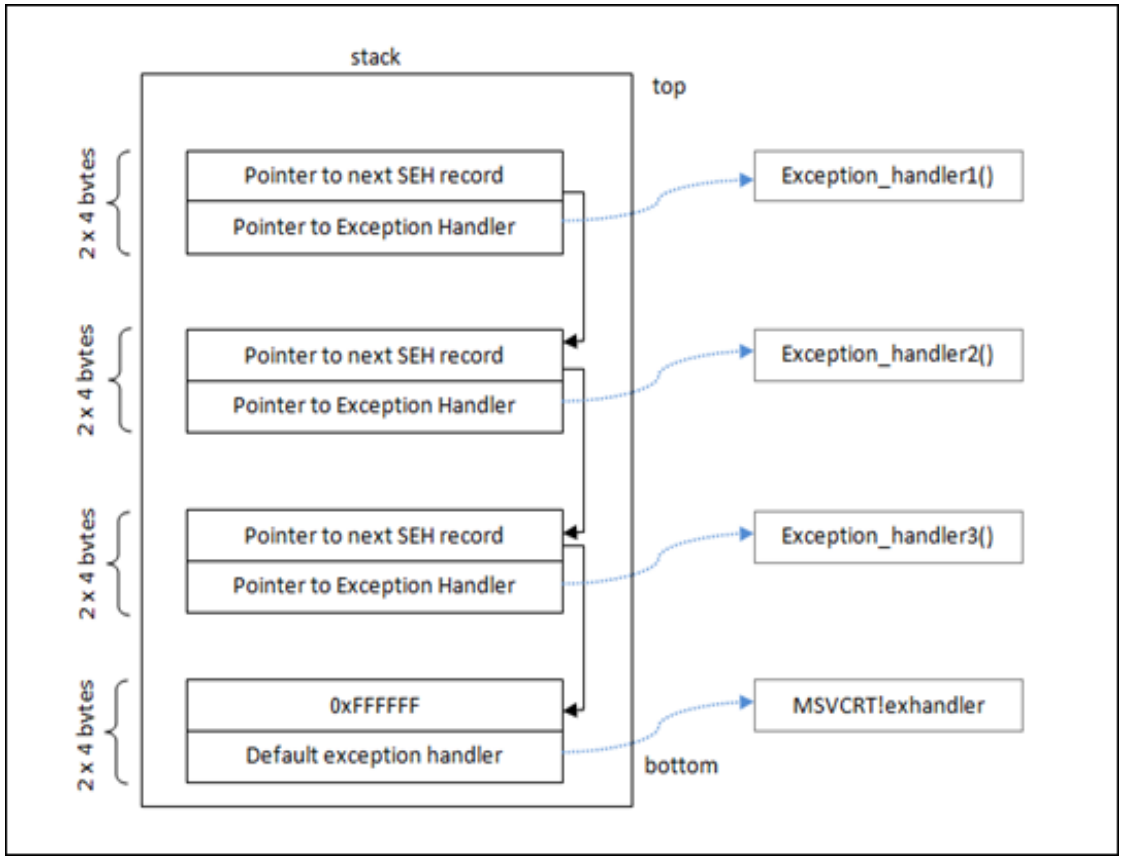
Buffer Overflow Using SEH



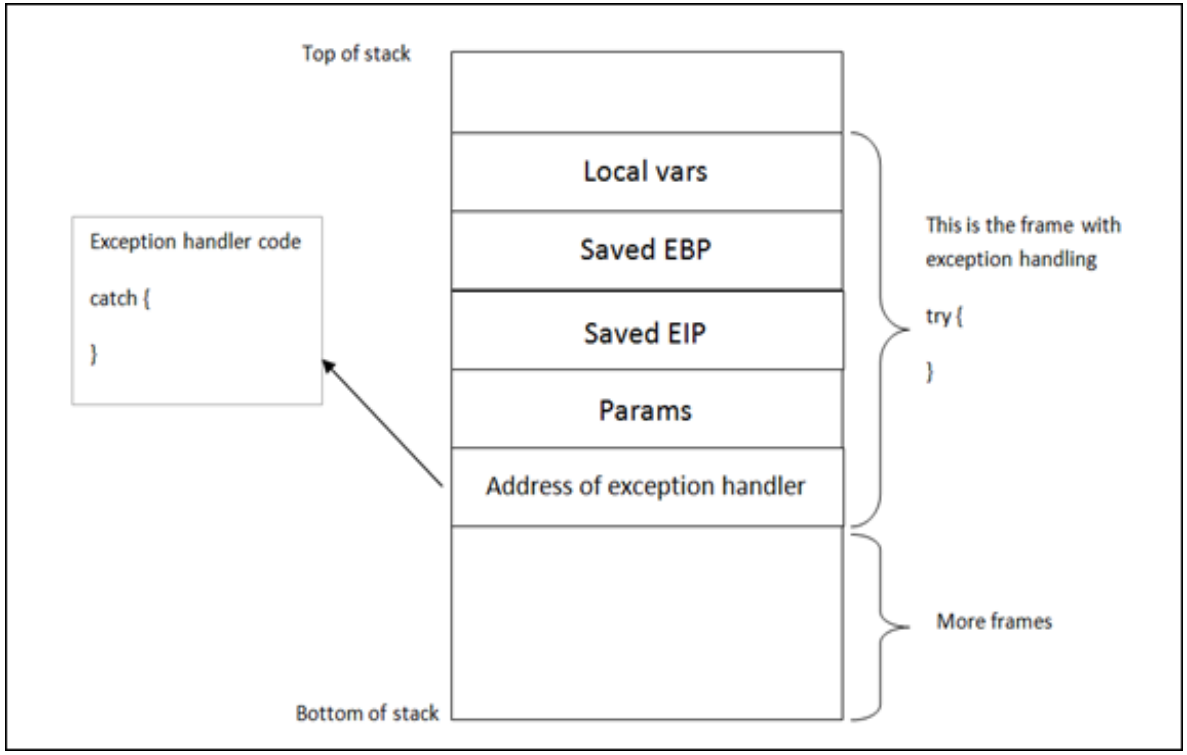
SEH中的异常处理


TEB
FS:[0] Exception List:: [address]
TEB Points to start of SEH chain
SEH Chain
Exception Registration Method
Next SEH
SEH
Exception Registration Method
Next SEH
SEH
Exception Registration Method
Next SEH
SEH
Exception Registration Method
Next SEH (FFFFFFFF)
SEH
Default Handler (end of chain)
_except_handler(
Exception Record
Establisher Frame
Context Record
DispactherContext)
Exception callback function
_exception_record
Exception Code
Establisher Flags
*Exception Record
Exception Address
# of Parameters
Exception Record Structure

SEH中的异常处理


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

Exception Handler结构

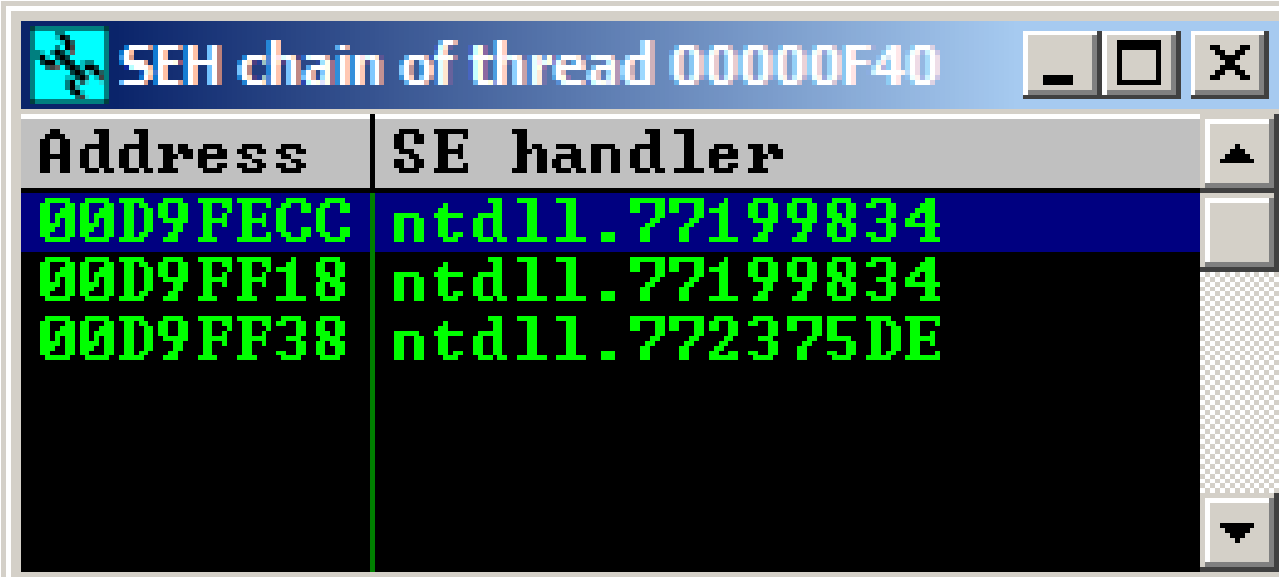


SEH Example Code

```
#include <stdio.h>
#include <windows.h>

DWORD MyExceptionHandler(void)
{
            printf("In exception handler....");
            ExitProcess(1);
            return 0;
}

int main()
{
             __try
             {
                  __asm
{
     // Cause an exception
                            xor eax,eax
                            call eax
}

            }
            __except(MyExceptionHandler())
            {
                   printf("oops...");
             }
             return 0;
}
```

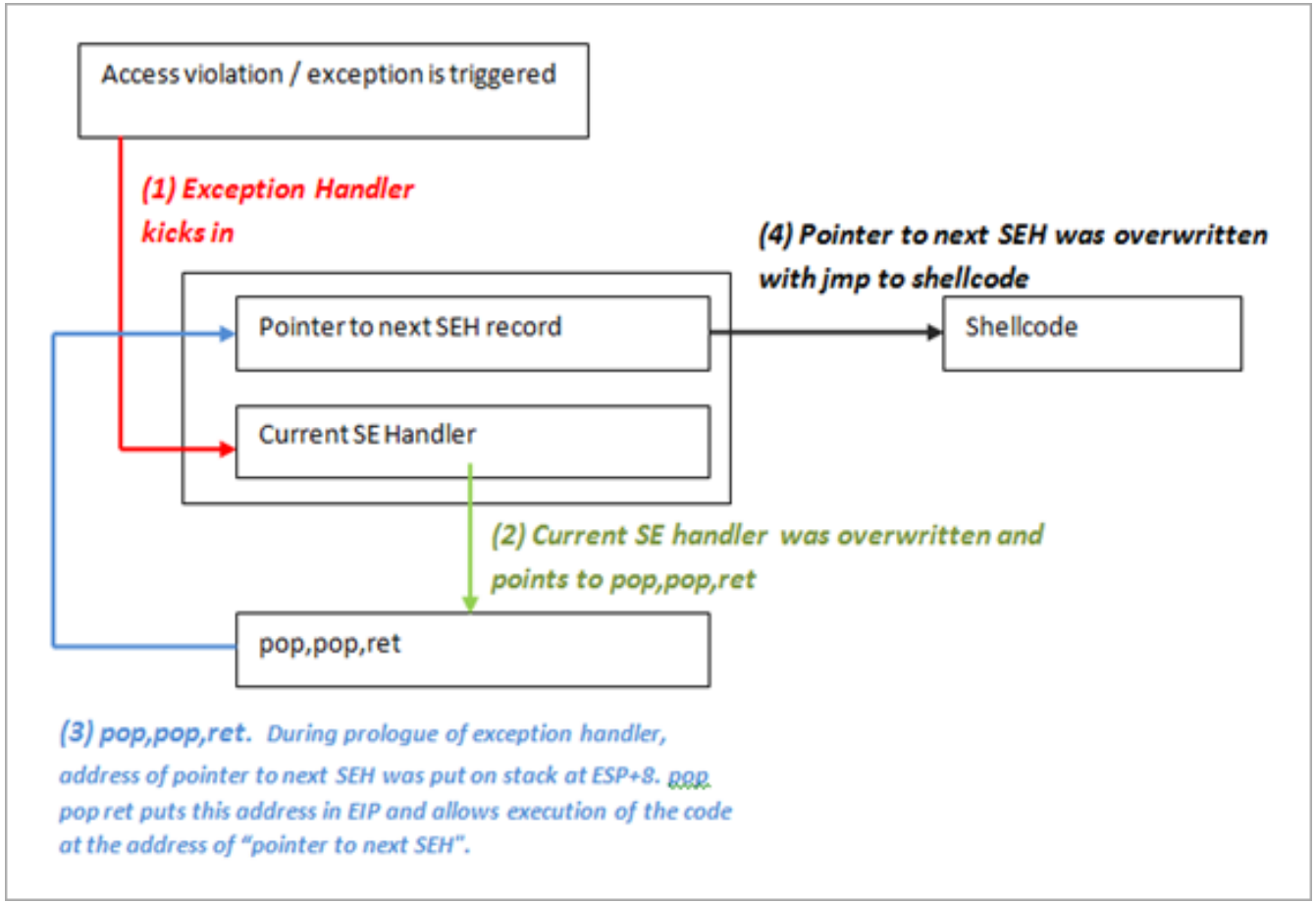
SEH Chain in Immunity

- View-->SEH Chain
- Notepad的 SEH Chain



Follow Address in Stack

```
00D9FECC  00D9FF18  ↑ ┘.  Pointer to next SEH record
00D9FED0  77199834  4ÿ↓w  SE handler
00D9FED4  003AD1CC  ╠╥:.
00D9FED8  00000000  ....
00D9FEDC  00D9FEE8  ¤▮┘.
00D9FEE0  76294911  ◄I►v  RETURN to kernel32.76294911
00D9FEE4  00000000  ....
00D9FEE8  00D9FF28  ( ┘.
00D9FEEC  771CE4B6  ╢Σ└w  RETURN to ntdll.771CE4B6
00D9FEF0  00000000  ....
00D9FEF4  77FC1BFC  ⁿ←ⁿw
00D9FEF8  00000000  ....
00D9FEFC  00000000  ....
00D9FF00  00000000  ....
00D9FF04  00000000  ....
00D9FF08  00000000  ....
00D9FF0C  00000000  ....
00D9FF10  00D9FEF4  ⌠▮┘.
00D9FF14  00000000  ....
00D9FF18  00D9FF38  8 ┘.  Pointer to next SEH record
00D9FF1C  77199834  4ÿ↓w  SE handler
00D9FF20  003ADB2C  ,╙:.
00D9FF24  00000000  ....
00D9FF28  00D9FF40  @ ┘.
00D9FF2C  771CE489  ëΣ└w  RETURN to ntdll.771CE489 from ntdll.771C
00D9FF30  7721D094  ö╝!w  ntdll.DbgUiRemoteBreakin
00D9FF34  00000000  ....
00D9FF38  FFFFFFFF      End of SEH chain
00D9FF3C  772375DE  ▌u#w  SE handler
00D9FF40  00000000  ....
00D9FF44  00000000  ....
00D9FF48  7721D094  ö╝!w  ntdll.DbgUiRemoteBreakin
00D9FF4C  00000000  ....
```


Access violation / exception is triggered
(1) Exception Handler kicks in
(4) Pointer to next SEH was overwritten with jmp to shellcode
Pointer to next SEH record
Shellcode
Current SE Handler
(2) Current SE handler was overwritten and points to pop,pop,ret
pop,pop,ret
(3) pop,pop,ret. During prologue of exception handler, address of pointer to next SEH was put on stack at ESP+8. pop pop ret puts this address in EIP and allows execution of the code at the address of "pointer to next SEH".

```
          Address       Content
.......................................................................................
             .             .
             .             .
             .             .
        00 00 60 40   XX XX XX XX   Address of Next SEH Record    The attacker
        00 00 60 44   XX XX XX XX   Address of SEH Handler        can overwrite
        00 00 60 48   XX XX XX XX                                 this part
        00 00 60 4C   XX XX XX XX                                 of the stack,
             .             .                                      but the Address
             .             .                                      of SEH Handler
             .             .                                      may not point here.
             .             .
             .             .
             .             .
.......................................................................................
             .             .
             .             .
             .             .
        10 20 30 40   Opcodes of a POP POP RET                    Another part of memory.
             .        instruction sequence
             .             .
             .             .
             .             .
.......................................................................................
```

溢出前

```
          Address       Content
.......................................................................................
             .             .
             .             .
             .             .
        00 00 60 40   EB 06 XX XX   Address of Next SEH Record    The attacker
        00 00 60 44   40 30 20 10   Address of SEH Handler        can overwrite
        00 00 60 48   Shellcode                                   this part
        00 00 60 4C   Shellcode                                   of the stack,
             .        Shellcode                                   but the Address
             .        Shellcode                                   of SEH Handler
             .        Shellcode                                   may not point here.
             .        Shellcode
             .             .
             .             .
.......................................................................................
             .             .
             .             .
             .             .
        10 20 30 40   Opcodes of a POP POP RET                    Another part of memory.
             .        instruction sequence
             .             .
             .             .
             .             .
.......................................................................................
```

溢出后

```
          Address       Content
.......................................................................................
             .             .
             .             .
             .             .
ESP ->   00 00 50 00   XX XX XX XX
         00 00 50 04   XX XX XX XX                                  The EXCEPTION_DISPOSITION
         00 00 50 08   40 60 00 00   Establisher Frame               Handler structure
         00 00 50 0C   XX XX XX XX                                  is set up on the stack
             .             .                                        after the exception is raised.
             .             .
             .             .
.......................................................................................
             .             .
             .             .
             .             .
         00 00 60 40   EB 06 XX XX   Address of Next SEH Record     The attacker
         00 00 60 44   40 30 20 10   Address of SEH Handler         can overwrite
         00 00 60 48   Shellcode                                    this part
         00 00 60 4C   Shellcode                                    of the stack,
             .         Shellcode                                    but the Address
             .         Shellcode                                    of SEH Handler
             .         Shellcode                                    may not point here.
             .         Shellcode
             .             .
             .             .
.......................................................................................
             .             .
             .             .
             .             .
EIP ->   10 20 30 40   Opcodes of a POP POP RET                     Another part of memory.
             .         instruction sequence
             .             .
             .             .
             .             .
.......................................................................................
```

异常发生后

在第一个POP执行时，ESP 指向00 00 50 00
在第一个POP执行后，ESP指向00 00 50 04
在第二个POP执行后，ESP 指向00 00 50 08

创建恶意输入

两种方式:

[一段任意的缓冲区填充 | NOP空指令滑行区 | Shellcode | 近跳转 | 短跳转 | PPR]

[一段任意的缓冲区填充 | 短跳转 | PPR | NOP空指令滑行区 | Shellcode]

JMP—Jump

Opcode	Instruction	Description
EB *cb*	JMP *rel8*	Jump short, relative, displacement relative to next instruction
E9 *cw*	JMP *rel16*	Jump near, relative, displacement relative to next instruction
E9 *cd*	JMP *rel32*	Jump near, relative, displacement relative to next instruction
FF /4	JMP *r/m16*	Jump near, absolute indirect, address given in *r/m16*
FF /4	JMP *r/m32*	Jump near, absolute indirect, address given in *r/m32*
EA *cd*	JMP *ptr16:16*	Jump far, absolute, address given in operand
EA *cp*	JMP *ptr16:32*	Jump far, absolute, address given in operand
FF /5	JMP *m16:16*	Jump far, absolute indirect, address given in *m16:16*
FF /5	JMP *m16:32*	Jump far, absolute indirect, address given in *m16:32*

https://web.archive.org/web/20090219101735/http://download.intel.com/design/PentiumII/manuals/24319102.PDF

创建恶意输入

Shellcode放在什么地方?

```
037BFF70  41414141 AAAA Pointer to next SEH record
037BFF74  42424242 BBBB SE handler
037BFF78  43434343 CCCC
037BFF7C  43434343 CCCC
037BFF80  43434343 CCCC
037BFF84  43434343 CCCC
037BFF88  43434343 CCCC
037BFF8C  43434343 CCCC
037BFF90  43434343 CCCC
037BFF94  43434343 CCCC
037BFF98  43434343 CCCC
037BFF9C  43434343 CCCC
037BFFA0  43434343 CCCC
037BFFA4  43434343 CCCC
037BFFA8  43434343 CCCC
037BFFAC  43434343 CCCC
037BFFB0  43434343 CCCC
037BFFB4  43434343 CCCC
037BFFB8  43434343 CCCC
037BFFBC  43434343 CCCC
037BFFC0  43434343 CCCC
037BFFC4  43434343 CCCC
037BFFC8  43434343 CCCC
037BFFCC  43434343 CCCC
037BFFD0  43434343 CCCC
037BFFD4  43434343 CCCC
037BFFD8  43434343 CCCC
037BFFDC  43434343 CCCC
037BFFE0  43434343 CCCC
037BFFE4  43434343 CCCC
037BFFE8  43434343 CCCC
037BFFEC  43434343 CCCC
037BFFF0  43434343 CCCC
037BFFF4  43434343 CCCC
037BFFF8  43434343 CCCC
037BFFFC  00434343 CCC. surgemai.00434343
```

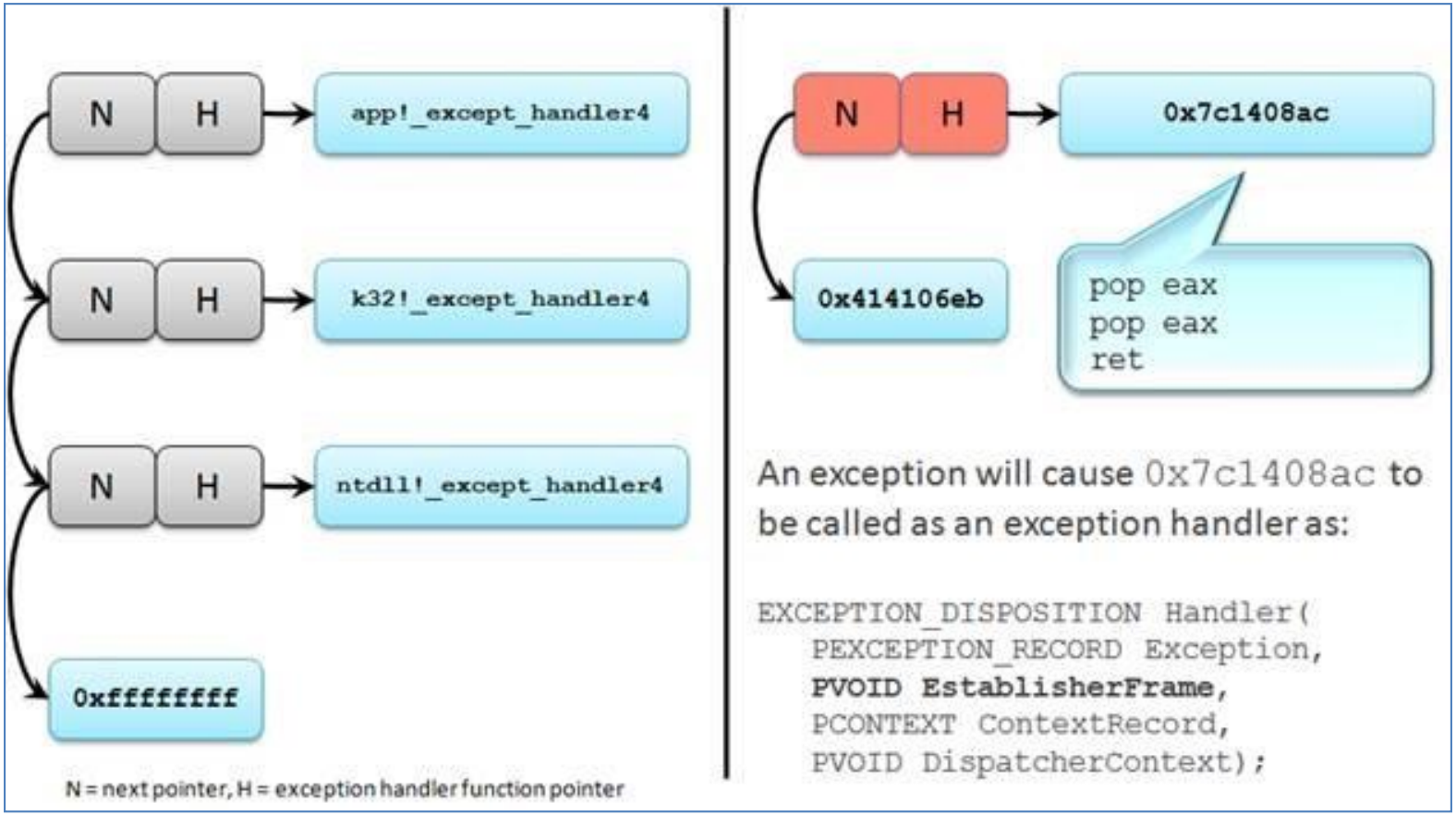
恶意输入

[一段任意的缓冲区填充 | NOP空指令滑行区 | Shellcode | 近跳转 | 短跳转 | PPR]

```
sjump = "\xEB\xF9\x90\x90"      # short jump
njump = "\xE9\xDD\xD7\xFF\xFF" # near jump
```

防御措施

- SEH 攻击


N
H
app!_except_handler4
N
H
k32!_except_handler4
N
H
ntdll!_except_handler4
0xffffffff
N = next pointer, H = exception handler function pointer
N
H
0x7c1408ac
0x414106eb
pop eax
pop eax
ret
An exception will cause 0x7c1408ac to be called as an exception handler as:
EXCEPTION_DISPOSITION Handler(
PEXCEPTION_RECORD Exception,
PVOID EstablisherFrame,
PCONTEXT ContextRecord,
PVOID DispatcherContext);

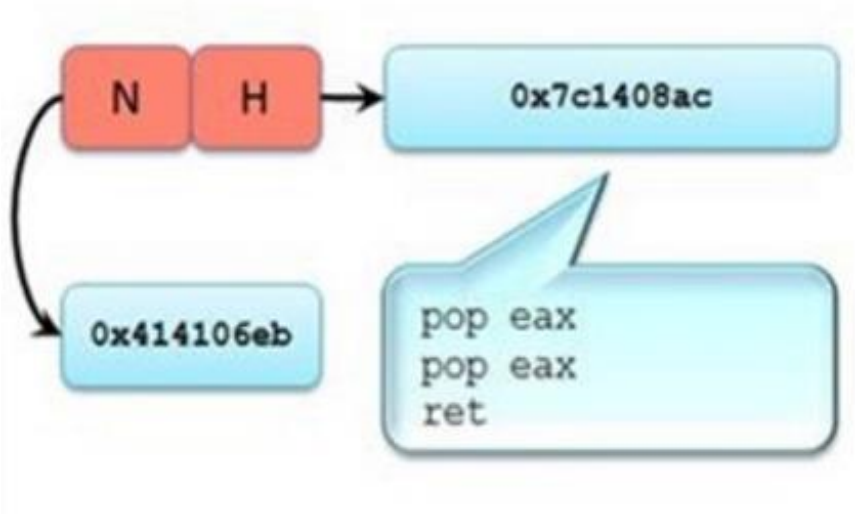
SAFESEH

- exception handler addresses白名单
- 通过compiler支持，需要重新编译代码

SEHOP

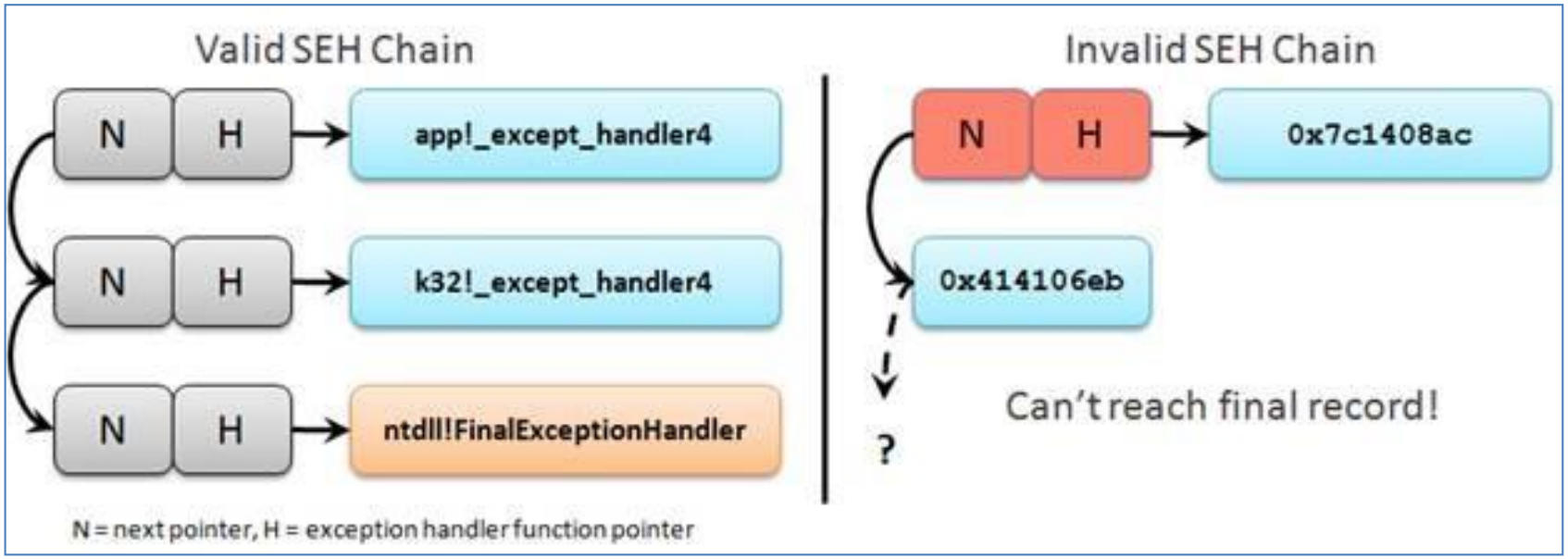
- Structured Exception Handling Overwrite Protection
- 验证exception handler 在使用前没有被修改
- 方法：
  - 添加一个额外的extra registration record
  - 遍历exception handler list，确保添加的record可到达

EXCEPTION_REGISTRATION_RECORD




```
typedef struct _EXCEPTION_REGISTRATION_RECORD
{
   struct _EXCEPTION_REGISTRATION_RECORD *Next;
   PEXCEPTION_ROUTINE                    Handler;
} EXCEPTION_REGISTRATION_RECORD, *PEXCEPTION_REGISTRATION_RECORD;
```

SEHOP


Valid SEH Chain
N
H
app!_except_handler4
N
H
k32!_except_handler4
N
H
ntdll!FinalExceptionHandler
N = next pointer, H = exception handler function pointer
Invalid SEH Chain
N
H
0x7c1408ac
0x414106eb
?
Can't reach final record!

实验任务

- 阅读“第14章”文件，实现对surgemail.exe (38k4-4ru)的渗透测试模块的编写（使用msf的方式）
  - 漏洞查找（Fuzz）
  - 漏洞利用
  - https://www.exploit-db.com/exploits/5259
- 实现对fs-wizard-setup.exe的渗透测试模块的编写（使用msf的方式）
  - https://www.exploit-db.com/exploits/47412

在线参考资料：https://www.offensive-security.com/metasploit-unleashed/writing-an-exploit/