VULNERABILITIES AND EXPLOITS

Defense Against The Dark Arts

Brad Antoniewicz
Foundstone – McAfee Professional Services

WHO'S THIS GUY?!

Hi, I'm @brad_anton

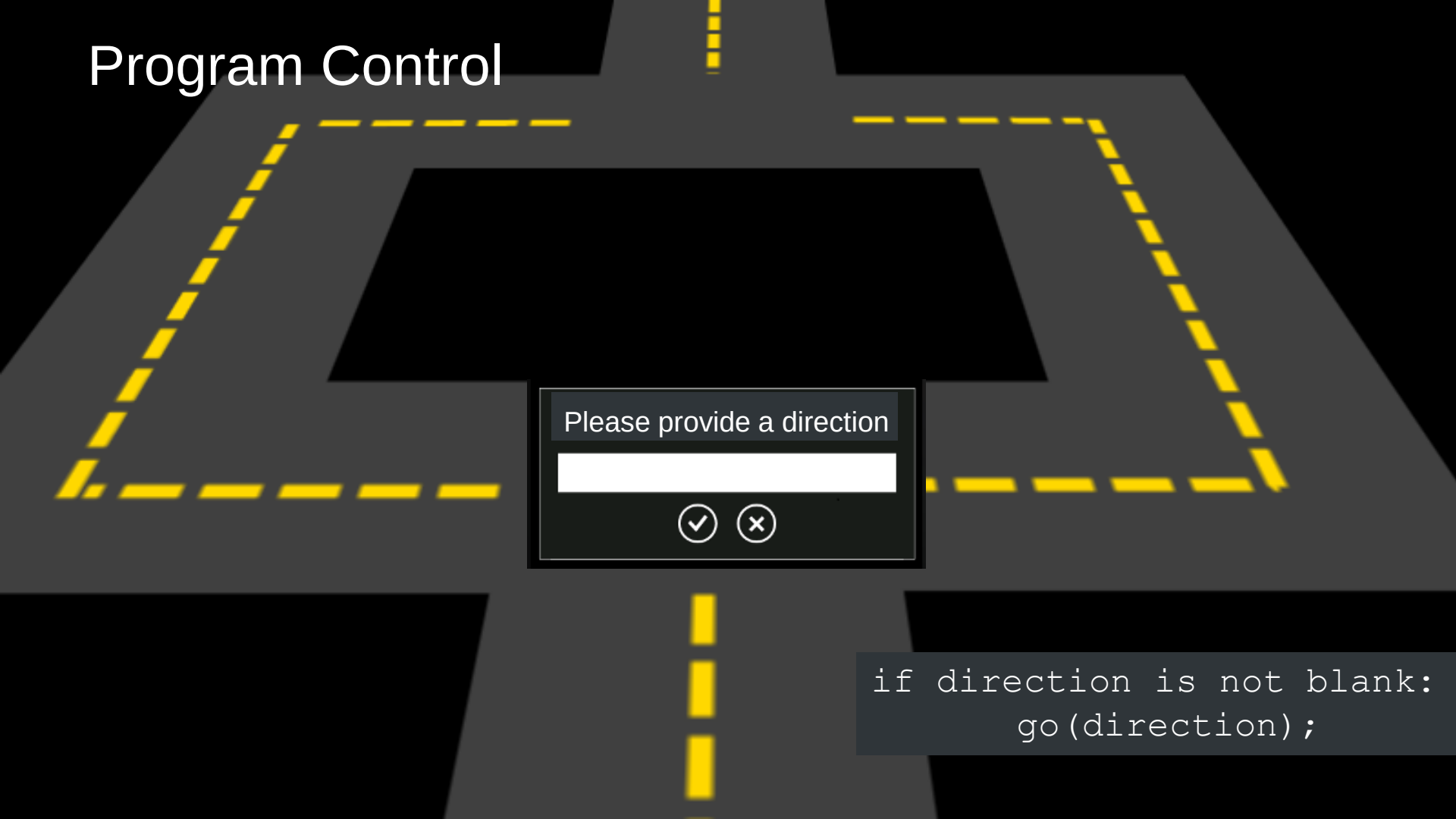
BLIZZARDPOCALYPSEMAGEDDON!?@!


Welcome to
Silent Hill
Oregon State

WHATS THIS ABOUT NOW?

- About “Hacking!
  - Real Lyfe
  - Trends
- WinDBG
  - Win-ternals via firehose
  - How to use
- Exploitz, son!
  - How the stack works
  - Exploit Stack-Based Vulnerabilitiy

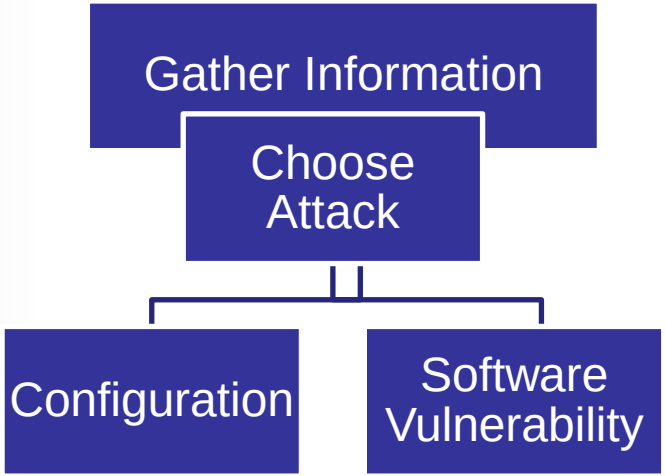
Program Control




```
if direction is not blank:
    go(direction);
```

Manipulating Software

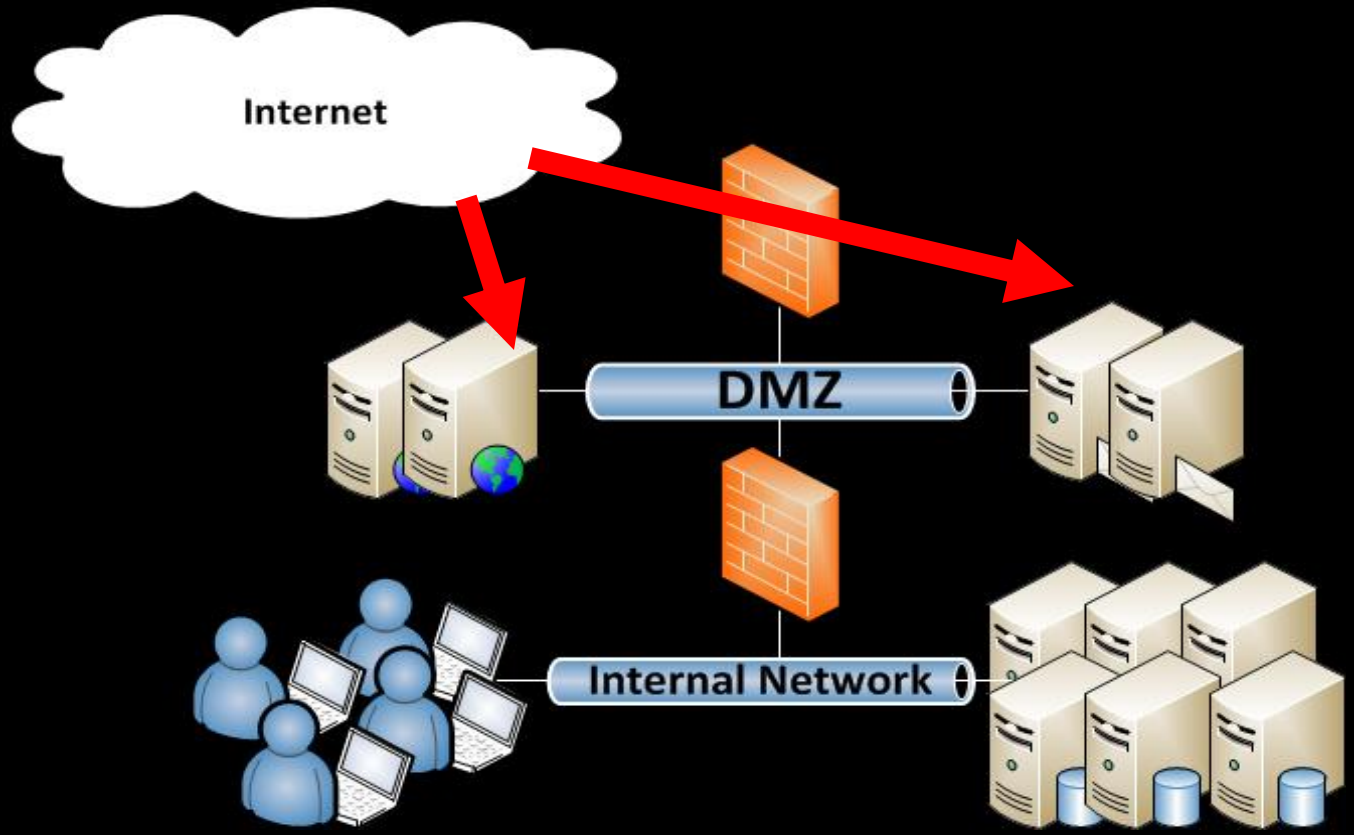
- Finding “bugs” which alter the behavior of the program
- Taking advantage of a misconfiguration or poor programming practice

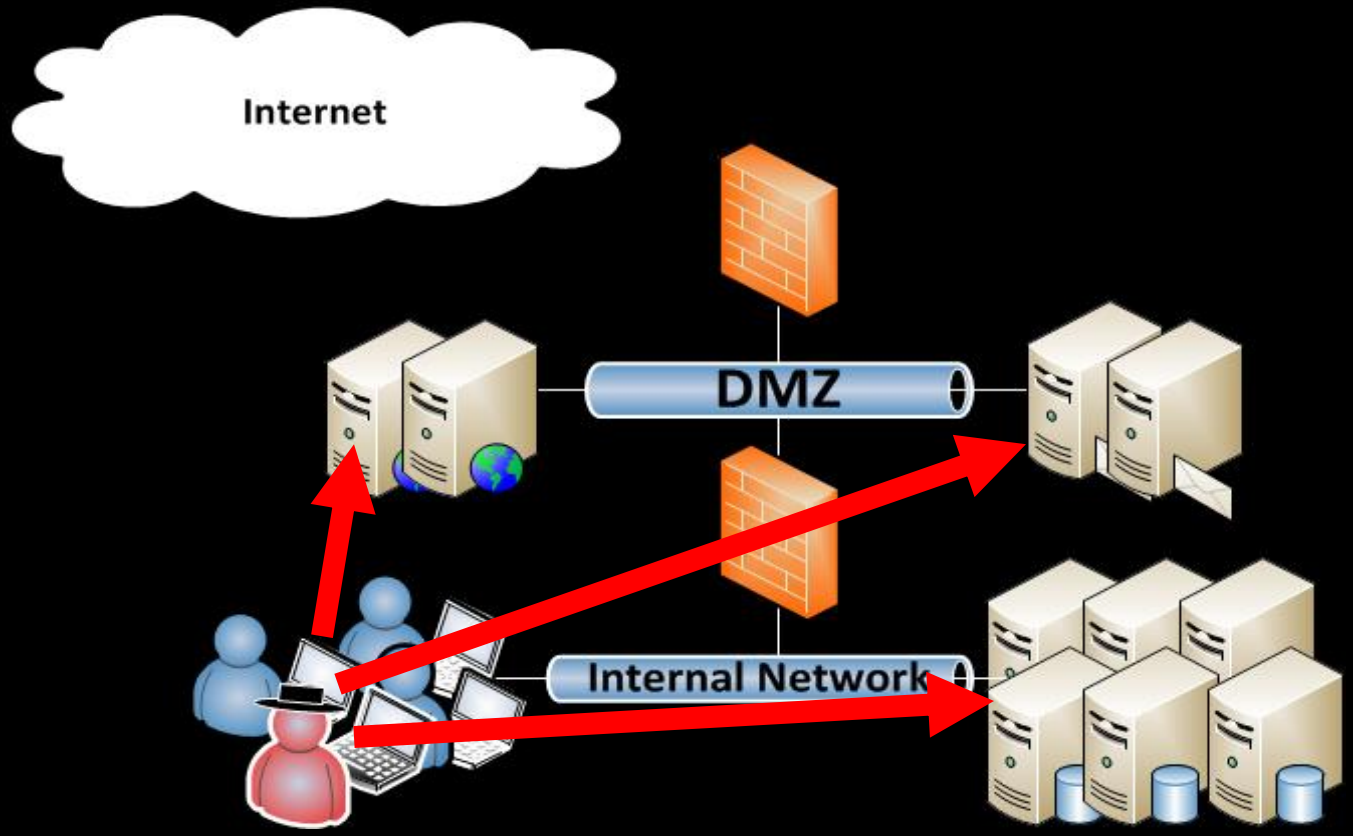


Hax? 4 wha?

Things change

Bug Bounty Programs


Internet
DMZ
Internal Network


Internet
DMZ
Internal Network

New zero-day vulnerability identified in all versions of IE

The flaw, which is being leveraged in "limited, targeted attacks," allows remote code execution, Microsoft warns.

by Steven Musil @stevenmusil


We'll give you the vuln, you exploit

WinDbg

+

Program-Freezer :)

WinDbg:6.12.0002.633 X86

File Edit View Debug Window Help

Command

Debuggee not connected

Ln 0, Col 0 | Sys 0:<None> | Proc 000:0 | Thrd 000:0 | ASM | OVR | CAPS | NUM

Attach to Process

1080 svchost.exe
1216 spoolsv.exe
1264 svchost.exe
1432 taskhost.exe
1508 dwm.exe
1536 explorer.exe
1572 vmtoolsd.exe
1976 vmtoolsd.exe
1100 dllhost.exe
1548 msdtc.exe
1152 SearchIndexer.exe
2424 svchost.exe
2456 sppsvc.exe
2488 svchost.exe
2544 wmpnetwk.exe
3220 WMIADAP.exe
3256 WmiPrvSE.exe
3736 iexplore.exe
1912 svchost.exe
1308 iexplore.exe

Sort
System order | By ID | By Executable

Process ID:
1308

Noninvasive

OK | Cancel | Help

File->Attach->2nd IE Process

Pid 1308 - WinDbg:6.2.9200.16384 X86

File Edit View Debug Window Help

Disassembly

Offset: @$scopeip Previous Next

```
77c940cd 8d0487          lea     eax,[edi+eax*4]
77c940d0 33ff            xor     edi,edi
77c940d2 47              inc     edi
77c940d3 83e11f          and     ecx,1Fh
77c940d6 d3e7            shl     edi,cl
77c940d8 f7d7            not     edi
77c940da 2138            and     dword ptr [eax],edi
77c940dc e9fce50200      jmp     ntdll!RtlpPopulateListIndex+0x
77c940e1 90              nop
77c940e2 90              nop
77c940e3 90              nop
ntdll!DbgBreakPointWithStatus:
77c940e4 8b442404        mov     eax,dword ptr [esp+4]
ntdll!RtlpBreakWithStatusInstruction:
77c940e8 cc              int     3
77c940e9 c20400          ret     4
ntdll!DbgUserBreakPoint:
77c940ec cc              int     3
77c940ed 90              nop
77c940ee c3              ret
77c940ef 90              nop
ntdll!DbgBreakPoint:
77c940f0 cc              int     3
77c940f1 c3              ret
77c940f2 90              nop
77c940f3 90              nop
77c940f4 90              nop
77c940f5 90              nop
77c940f6 90              nop
77c940f7 90              nop
ntdll!RtlInitString:
77c940f8 57              push    edi
77c940f9 8b7c240c        mov     edi,dword ptr [esp+0Ch]
77c940fd 8b542408        mov     edx,dword ptr [esp+8]
77c94101 c70200000000    mov     dword ptr [edx],0
77c94107 897a04          mov     dword ptr [edx+4],edi
77c9410a 0bff            or      edi,edi
77c9410c 741e            je      ntdll!RtlInitString+0x34 (77c9
77c9410e 83c9ff          or      ecx,0FFFFFFFFh
77c94111 33c0            xor     eax,eax
77c94113 f2ae            repne scas byte ptr es:[edi]
```

Command

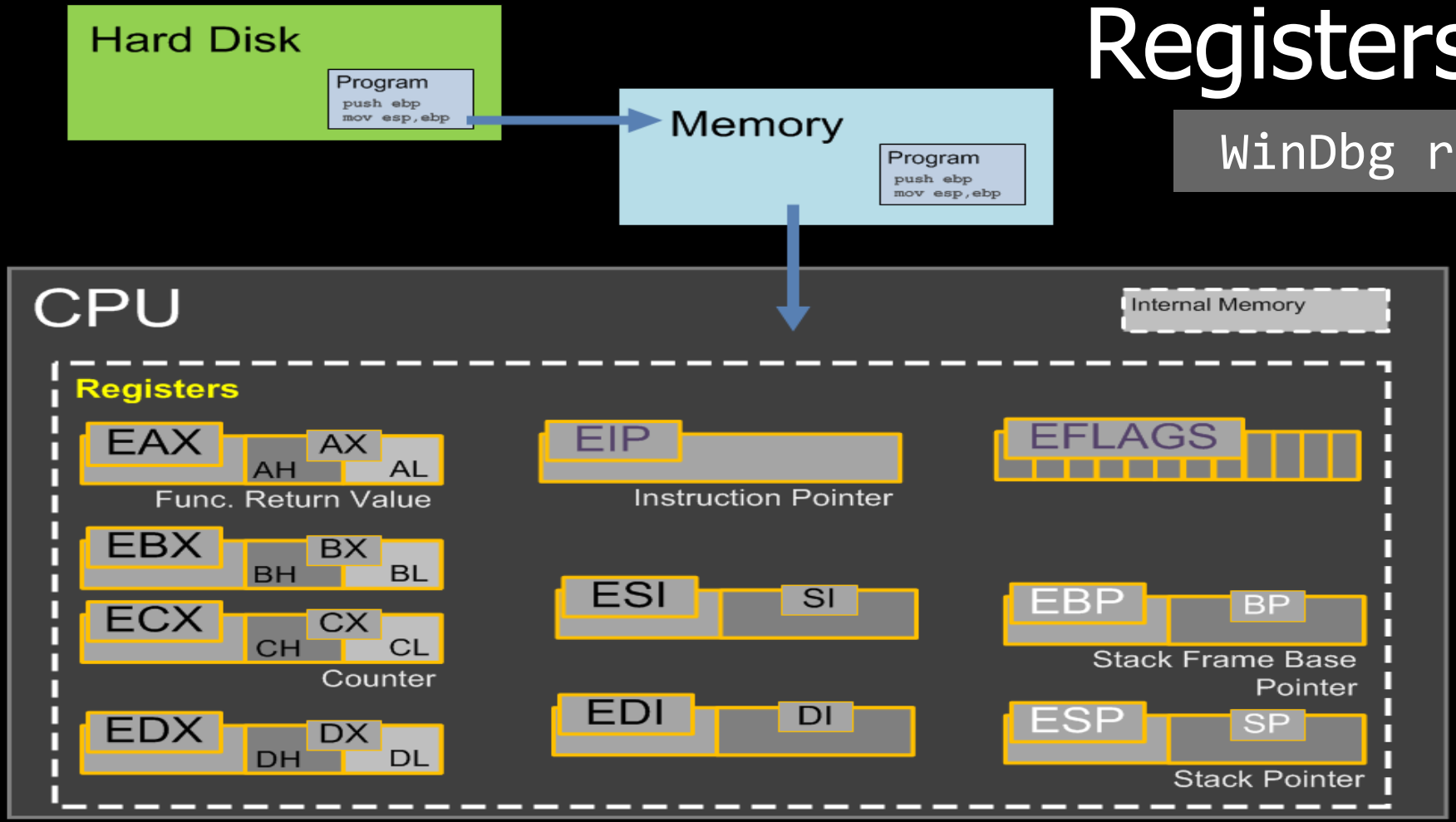
```
ModLoad: 76380000 763b5000   C:\Windows\system32\ws2_32.DLL
ModLoad: 76520000 76526000   C:\Windows\system32\NSI.dll
ModLoad: 756b0000 756f4000   C:\Windows\system32\dnsapi.DLL
ModLoad: 74460000 7447c000   C:\Windows\system32\iphlpapi.DLL
ModLoad: 74420000 74427000   C:\Windows\system32\WINNSI.DLL
ModLoad: 6e240000 6e26e000   C:\Windows\system32\MLANG.dll
ModLoad: 75cb0000 75cfc000   C:\Windows\system32\apphelp.dll
ModLoad: 68f00000 694b7000   C:\Windows\System32\mshtml.dll
ModLoad: 70c20000 70c4a000   C:\Windows\System32\msls31.dll
ModLoad: 752b0000 752b9000   C:\Windows\System32\VERSION.dll
ModLoad: 6a970000 6a9df000   C:\Windows\System32\ieapfltr.dll
ModLoad: 75b40000 75b48000   C:\Windows\System32\Secur32.dll
ModLoad: 75d10000 75d6f000   C:\Windows\system32\SXS.DLL
ModLoad: 73dc0000 73dcb000   C:\Windows\system32\msimtf.dll
ModLoad: 72020000 72052000   C:\Windows\system32\WINMM.dll
ModLoad: 74c20000 74c59000   C:\Windows\system32\MMDevAPI.DLL
ModLoad: 70400000 70430000   C:\Windows\system32\wdmaud.drv
ModLoad: 74360000 74364000   C:\Windows\system32\ksuser.dll
ModLoad: 74eb0000 74eb7000   C:\Windows\system32\AVRT.dll
ModLoad: 6dda0000 6ddab000   C:\Windows\system32\ImgUtil.dll
ModLoad: 6dd90000 6dd9e000   C:\Windows\System32\pngfilt.dll
ModLoad: 73d80000 73db6000   C:\Windows\system32\AUDIOSES.DLL
ModLoad: 735f0000 735f8000   C:\Windows\system32\msacm32.drv
ModLoad: 735d0000 735e4000   C:\Windows\system32\MSACM32.dll
ModLoad: 735c0000 735c7000   C:\Windows\system32\midimap.dll
ModLoad: 699c0000 69a72000   C:\Windows\System32\jscript.dll
ModLoad: 54430000 5443b000   C:\Windows\Downloaded Program Files\FSExp
ModLoad: 69580000 699b2000   C:\Windows\system32\mfc100u.dll
ModLoad: 694c0000 6957e000   C:\Windows\system32\MSVCR100.dll
ModLoad: 73530000 735b4000   C:\Windows\WinSxS\x86_microsoft.windows.c
ModLoad: 729b0000 729b5000   C:\Windows\system32\MSIMG32.dll
ModLoad: 73f50000 73f5d000   C:\Windows\system32\MFC100ENU.DLL
ModLoad: 68d50000 68e83000   C:\Windows\System32\msxml3.dll
(51c.260): Break instruction exception - code 80000003 (first chance)
eax=7ffad000 ebx=00000000 ecx=00000000 edx=77cff125 esi=00000000 edi=0
eip=77c940f0 esp=067eff14 ebp=067eff40 iopl=0         nv up ei pl zr n
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=0
ntdll!DbgBreakPoint:
77c940f0 cc              int     3
```

0:014>

Ln 0, Col 0 | Sys 0:<Local> | Proc 000:51c | Thrd 014:260 | ASM | OVR | CAPS | NUM

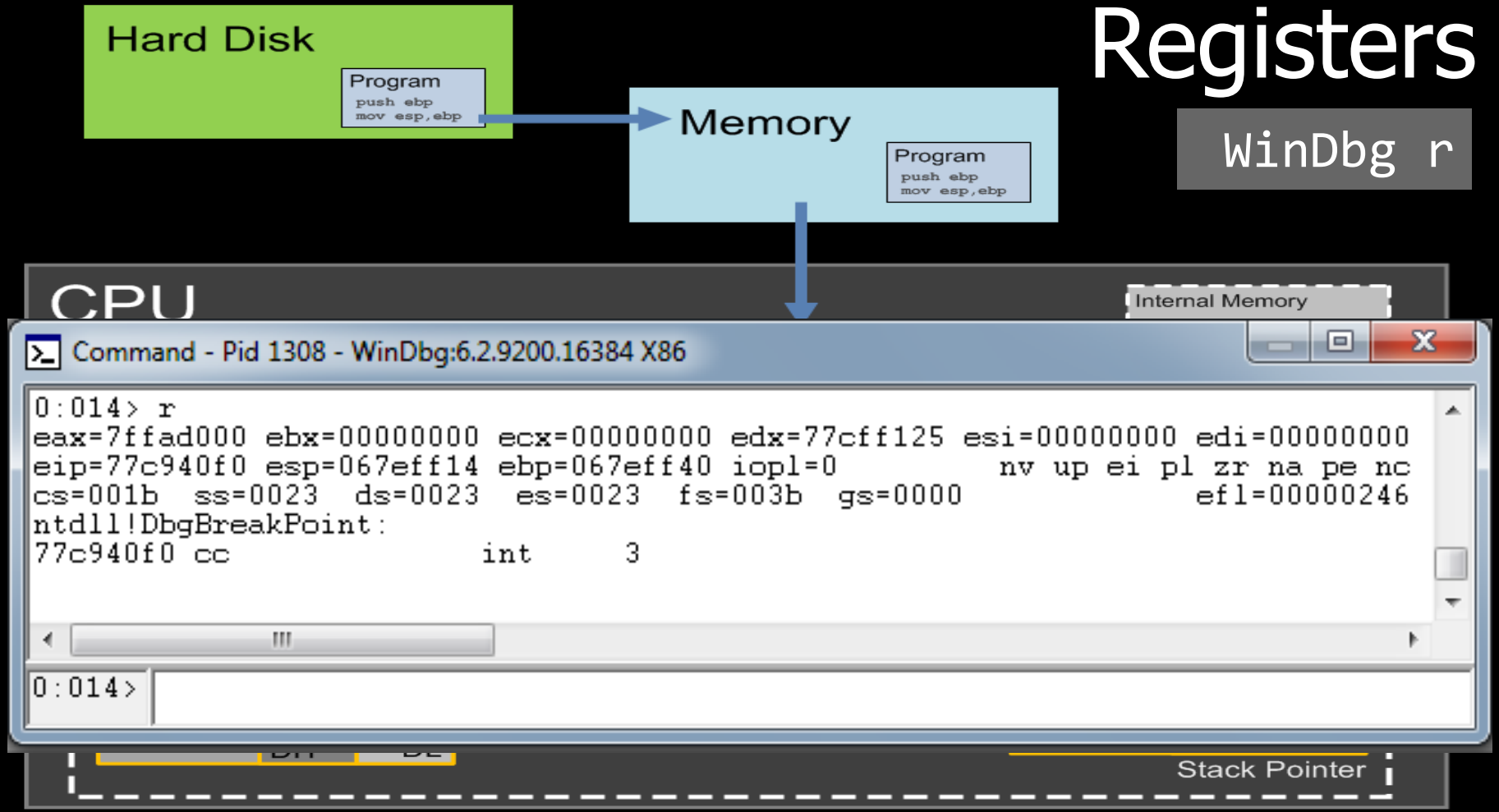
Registers

WinDbg r



Registers

WinDbg r



Program Memory

VMMap - Sysinternals: www.sysinternals.com

File Edit View Options Help

Process: notepad.exe
PID: 1396

Committed: 43,8[illegible] K

Private Bytes: 1,136 K

Working Set: 6,080 K

Type	Size	Committed	Private	Total WS	Private WS	Shareable...	Shared WS	Locked WS
Image	25,180 K	25,180 K	368 K	2,604 K	276 K	2,328 K	2,244 K	
Mapped File	12,708 K	12,708 K		528 K		528 K	516 K	
Shareable	17,788 K	3,912 K		2,260 K		2,260 K	2,252 K	
Heap	1,728 K	400 K	336 K	320 K	316 K	4 K	4 K	
Managed Heap								
Stack	256 K	76 K	76 K	12 K	12 K			
Private Data	596 K	28 K	28 K	28 K	24 K	4 K	4 K	
Page Table	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]			

Address	Type	Size	Com...	Private	Total ...	Priva...	Shar...	Sh...	Loc...	Blo...	Protection	Details
00010000	Heap (Shar...	64 K	64 K		4 K		4 K	4 K		1	Read/Write	Heap ID: 2 [COM
00020000	Mapped File	12 K	12 K	12 K	12 K		12 K			1	Copy on write	C:\Windows\Syst
00030000	Shareable	16 K	16 K		16 K		16 K	16 K		1	Read	
00040000	Shareable	8 K	8 K		8 K		8 K			1	Read	
00050000	Private Data	4 K	4 K	4 K	4 K	4 K				1	Read/Write	
00060000	Mapped File	412 K	412 K		192 K		192 K	192 K		1	Read	C:\Windows\Sys
000D0000	Shareable	800 K	24 K		20 K		20 K	20 K		4	Read	
001A0000	Private Data	4 K	4 K	4 K	4 K	4 K				1	Read/Write	
001B0000	Heap (Priva...	64 K	12 K	12 K	12 K	12 K				2	Read/Write	Heap ID: 3 [COM
001C0000	Private Data	4 K	4 K	4 K	4 K	4 K				1	Read/Write	
001D0000	[illegible]											

Timeline... Heap Allocations... Call Tree... Trace...

```
!teb (stack)
!peb (heap)
!address
```

Program Memory

VMMap - Sysinternals: www.sysinternals.com

File Edit View Options Help

Process: notepad.exe
PID: 1396

Committed: 43,8[illegible] K
Private Bytes: 1,136 K
Working Set: 6,080 K

Type	Private WS	Shareable...	Shared WS	Locked WS
Image	276 K	2,328 K	2,244 K	
Mapped		528 K	516 K	
Shareabl		2,260 K	2,252 K	
Heap	316 K	4 K	4 K	
Managed				
Stack	12 K			
Private D	24 K	4 K	4 K	
Page Tab				

Address	Protection	Details
000100	1 Read/Write	Heap ID: 2 [COM
000200	1 Copy on write	C:\Windows\Syst
000300	1 Read	
000400	1 Read	
000500	1 Read/Write	
000600	1 Read	C:\Windows\Syst
000D0	4 Read	
001A0	1 Read/Write	
001B0	2 Read/Write	Heap ID: 3 [COM
001C0	1 Read/Write	
001D0		

Command - Pid 1308 - WinDbg:6.2.9200.16384 X86

```
0:014> !teb
TEB at 7ffad000
    ExceptionList:        067eff30
    StackBase:            067f0000
    StackLimit:           067ec000
    SubSystemTib:         00000000
    FiberData:            00001e00
    ArbitraryUserPointer: 00000000
    Self:                 7ffad000
    EnvironmentPointer:   00000000
    ClientId:             0000051c . 00000260
    RpcHandle:            00000000
    Tls Storage:          00000000
    PEB Address:          7ffd9000
    LastErrorValue:       0
    LastStatusValue:      0
    Count Owned Locks:    0
    HardErrorMode:        0
0:014>
```

Timeline... Heap Allocations... Call Tree... Trace...

!teb (stack)
!peb (heap)
!address

Memory

Command - Pid 1308 - WinDbg:6.2.9200.16384 X86

```
0:014> !peb
PEB at 7ffd9000
    InheritedAddressSpace:    No
    ReadImageFileExecOptions: No
    BeingDebugged:            Yes
    ImageBaseAddress:         013a0000
    Ldr                       77d37880
    Ldr.Initialized:          Yes
    Ldr.InInitializationOrderModuleList: 002519f8 . 002f4a48
    Ldr.InLoadOrderModuleList:           00251968 . 002f4a38
    Ldr.InMemoryOrderModuleList:         00251970 . 002f4a40
            Base TimeStamp                     Module
         13a0000 4ce79912 Nov 20 01:46:58 2010 C:\Program Files\Internet Explorer\iexpl
        77c60000 4ce7b96e Nov 20 04:05:02 2010 C:\Windows\SYSTEM32\ntdll.dll
        68d50000 4ce7b8e9 Nov 20 04:02:49 2010 C:\Windows\System32\msxml3.dll
    SubSystemData:     00000000
    ProcessHeap:       00250000
    ProcessParameters: 00251108
    CurrentDirectory:  'C:\Users\Admin\Desktop\'
    WindowTitle:  'Microsoft.InternetExplorer.Default'
    ImageFile:    'C:\Program Files\Internet Explorer\iexplore.exe'
    CommandLine:  '"C:\Program Files\Internet Explorer\iexplore.exe" SCODEF:3736 CREDAT
    DllPath:      'C:\Program Files\Internet Explorer;C:\Windows\system32;C:\Windows\sy
    Environment:  002507f0
        =::=::\
        ALLUSERSPROFILE=C:\ProgramData
        APPDATA=C:\Users\Admin\AppData\Roaming
        CommonProgramFiles=C:\Program Files\Common Files
        COMPUTERNAME=WIN-EOLFS6K48D0
        ComSpec=C:\Windows\system32\cmd.exe
        FP_NO_HOST_CHECK=NO
        HOMEDRIVE=C:
0:014>
```

VMMap - Sysinternals: www.sysinternals.com

Shared WS	Locked WS
2,244 K	
516 K	
2,252 K	
4 K	
4 K	

Details
Heap ID: 2 [COM
C:\Windows\Syst
C:\Windows\Syst
Heap ID: 3 [COM

teb (stack)

!peb (heap)

!address

intel Security

Program Memory

VMMap - Sysinternals: www.sysinternals.com

File Edit View Options Help

Process: notepad.exe
PID: 1396

Committed:
Private Byt
Working Se

Command - Pid 1308 - WinDbg:6.2.9200.16384 X86

```
0:014> !address esp


Mapping file section regions...
Mapping module regions...
Mapping PEB regions...
Mapping TEB and stack regions...
Mapping heap regions...
Mapping page heap regions...
Mapping other regions...
Mapping stack trace database regions...
Mapping activation context regions...


Usage:                  Stack
Base Address:           067ec000
End Address:            067f0000
Region Size:            00004000
State:                  00001000    MEM_COMMIT
Protect:                00000004    PAGE_READWRITE
Type:                   00020000    MEM_PRIVATE
Allocation Base:        066f0000
Allocation Protect:     00000004    PAGE_READWRITE
More info:              ~14k

0:014>
```

Type	Private WS	Shareable...	Shared WS	Locked WS
Image	276 K	2,328 K	2,244 K	
Mapped		528 K	516 K	
Shareat		2,260 K	2,252 K	
Heap	316 K	4 K	4 K	
Manage				
Stack	12 K			
Private	24 K	4 K	4 K	
Page Ta				

Addres	har...	Sh...	Loc...	Blo...	Protection	Details
0001	4 K	4 K		1	Read/Write	Heap ID: 2 [COM
0002	12 K			1	Copy on write	C:\Windows\Syst
0003	16 K	16 K		1	Read	
0004	8 K			1	Read	
0005				1	Read/Write	
0006	192 K	192 K		1	Read	C:\Windows\Sys
000D	20 K	20 K		4	Read	
001A				1	Read/Write	
001B				2	Read/Write	Heap ID: 3 [COM
001C				1	Read/Write	
001D						

Timeline... Heap Allocations... Call Tree... Trace...

!teb (stack)
!peb (heap)
!address

Lab 1: Hello Mr. WinDbg

Viewing Memory: `dd, da, du`
Breakpoints: `bp <addr>`
Clear all: `bc *`
Stepping: `t, p`

Disassembly: `View->Disass.`
Conversion: `.formats`
Math: `?1+1`
Modules: `lm`

Extensions:

Process (inc heap): `!peb`
Thread (inc stack): `!teb`
What Addr?: `!address`

FLAW CLASSES AND VULNERABILITIES

(there are lots)

- Configuration
  - Weak Password
- Logic
  - Authorization Issues
- Storage
  - Inadequate Encryption
- Input Validation
  - Memory Corruption
  - Injection

Accessing memory in an invalid way which results in an **undefined behavior**

MEMORY CORRUPTION

Reading/Writing

Usually Stack or Heap

Originally unintended

Accessing memory in an invalid way which results in an **undefined behavior**

What we're looking to control

COMMON CATEGORIES*

- Lifetime Control ← Exploit Tomorrow
- Uninitialized Memory
- Array index calculations
- Buffer length calculations ← Exploit Today

*Just a few from http://cwe.mitre.org/

Taking advantage of a vulnerability

(Control the “undefined” behavior)

EXPLOIT? HUH?

Vulnerability Trigger	Invokes the software bug to obtain control of the program
Payload	Action to be performed when control is obtained

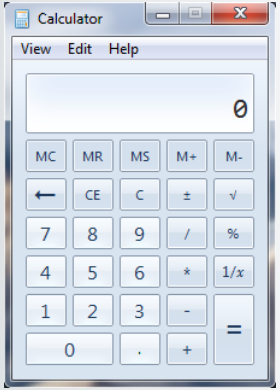
$-LOAD

Payload

"Shell" code – usually assembly code to execute a shell (e.g. `/bin/sh`)

$-LOAD

Payload



$-LOAD

Payload

↓

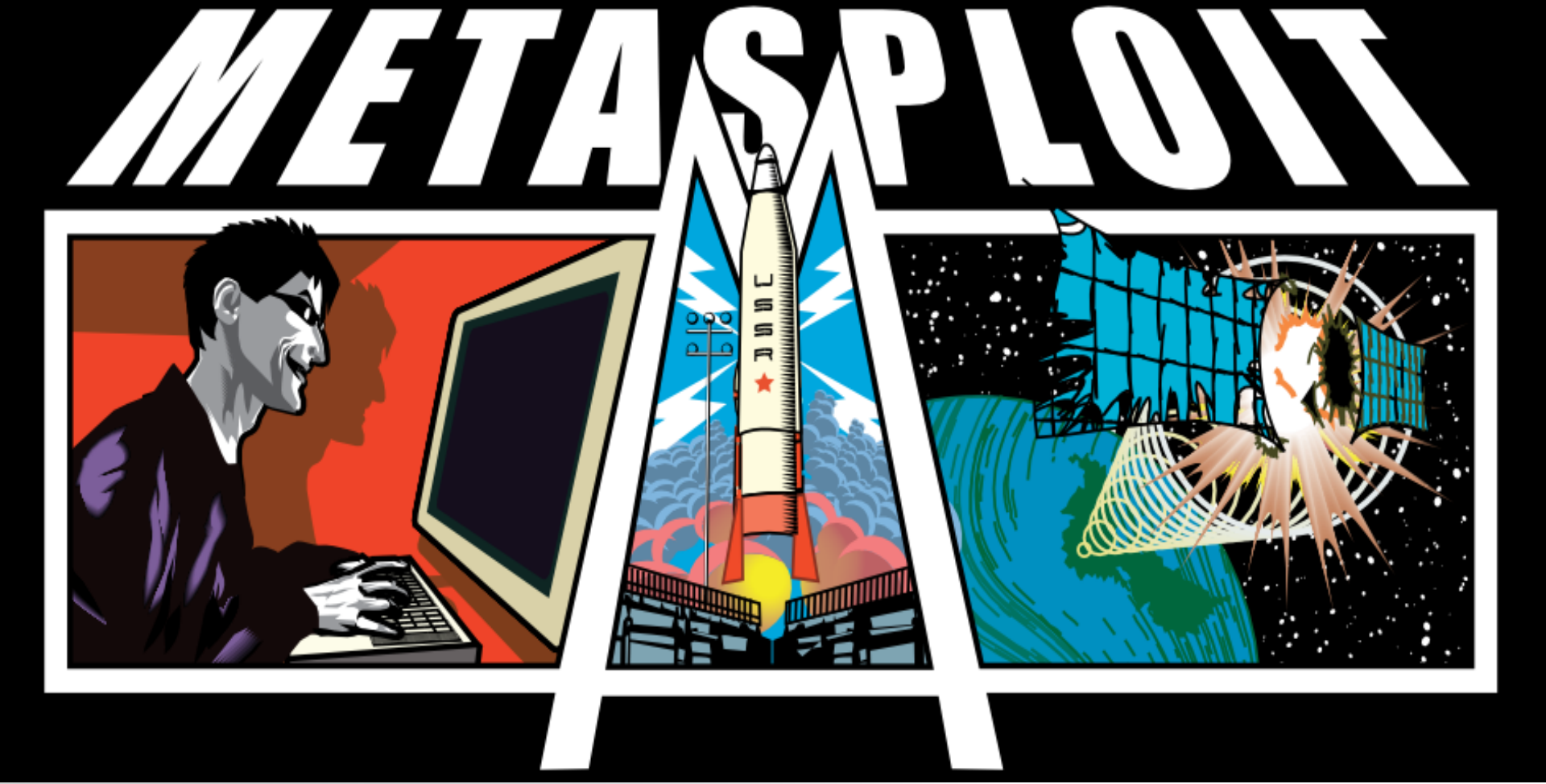
```
1?
```

$-LOAD

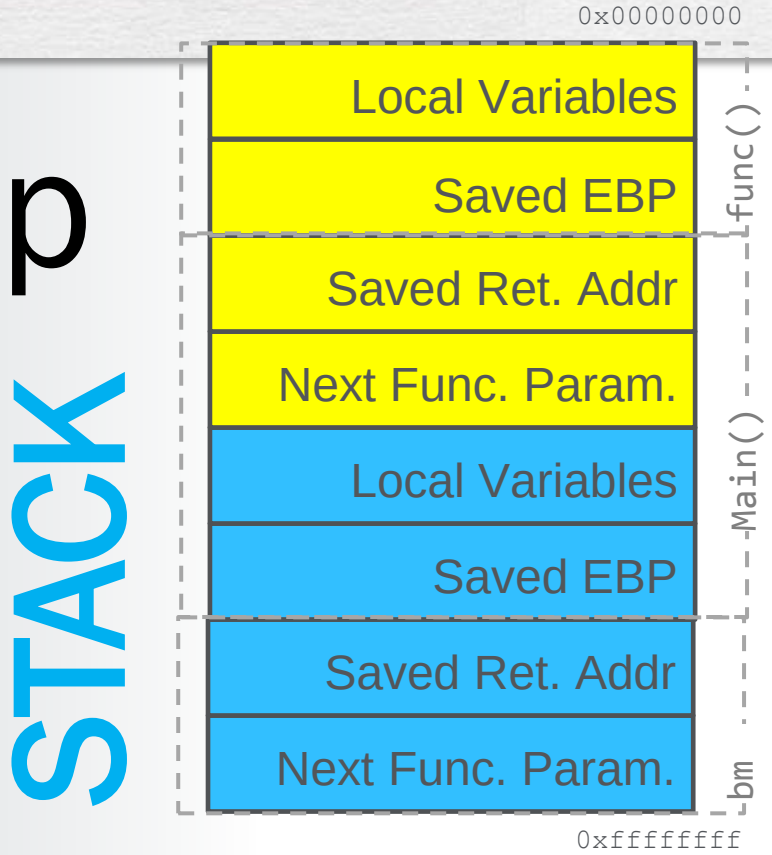
Payload

1?

```
bool isValid = 0;
char buf[255];

memcpy(buf, usrval, len);

...

if (checkPW(buf))
        isValid = 1;

...

return isValid;
```


METASPLOIT

Stack Recap


0x00000000
STACK
Local Variables
Saved EBP
Saved Ret. Addr
Next Func. Param.
Local Variables
Saved EBP
Saved Ret. Addr
Next Func. Param.
func()
Main()
0xffffffff

main()

1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

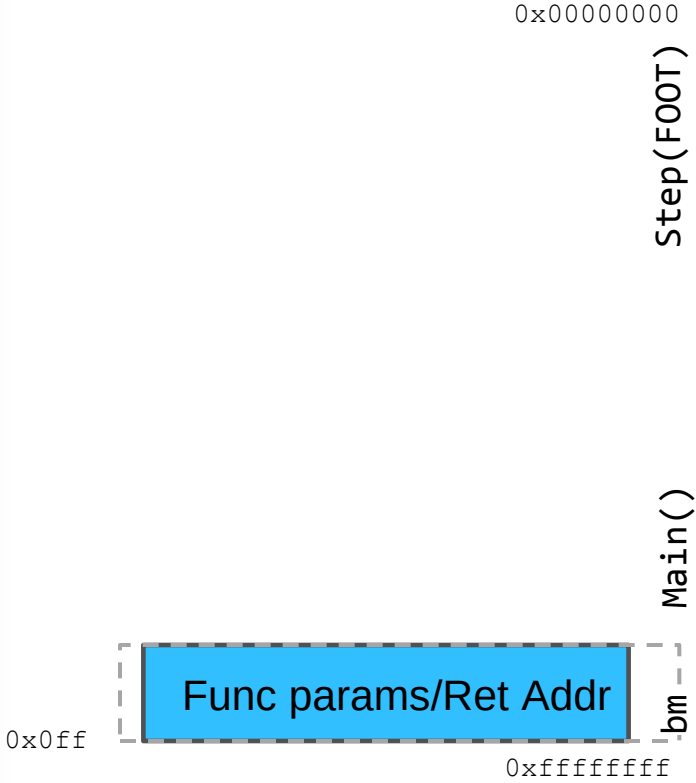
A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down

main()

1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



0x00000000

main()

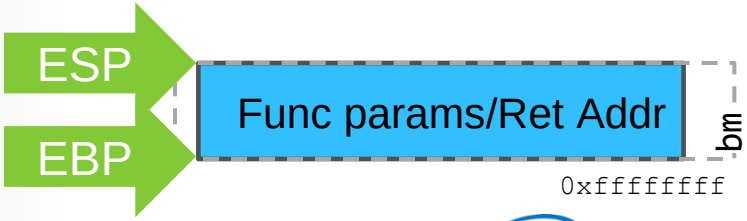
EIP

```
push ebp
mov ebp, esp
```

1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



main()

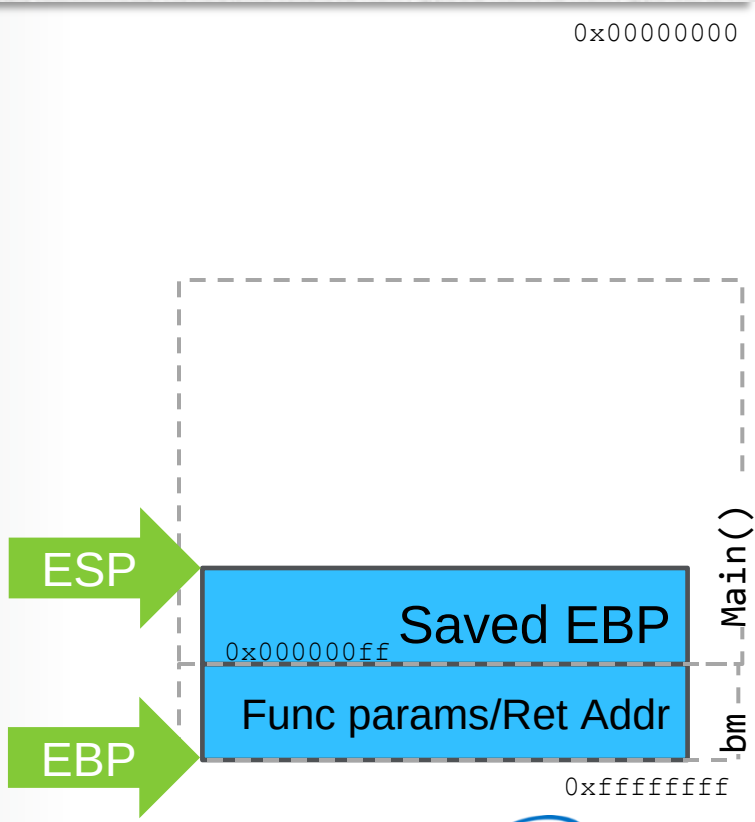
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

EIP

```
push ebp
mov ebp, esp
```

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



main()

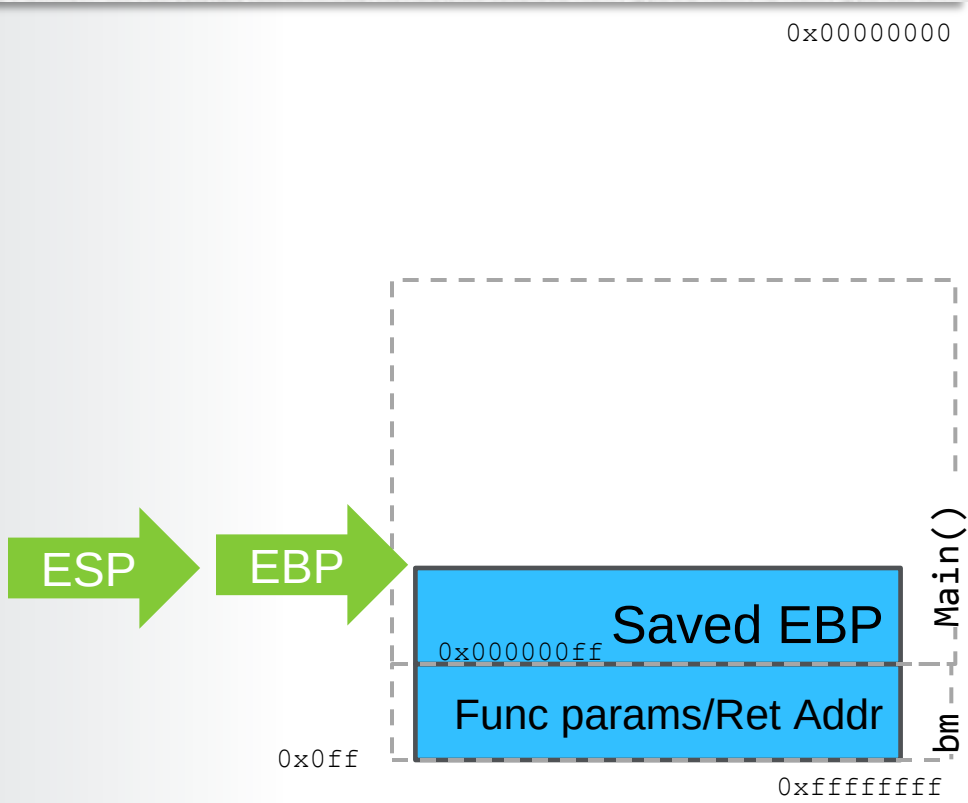
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

```
push ebp
mov ebp, esp
```

EIP → `sub esp,4h`

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down

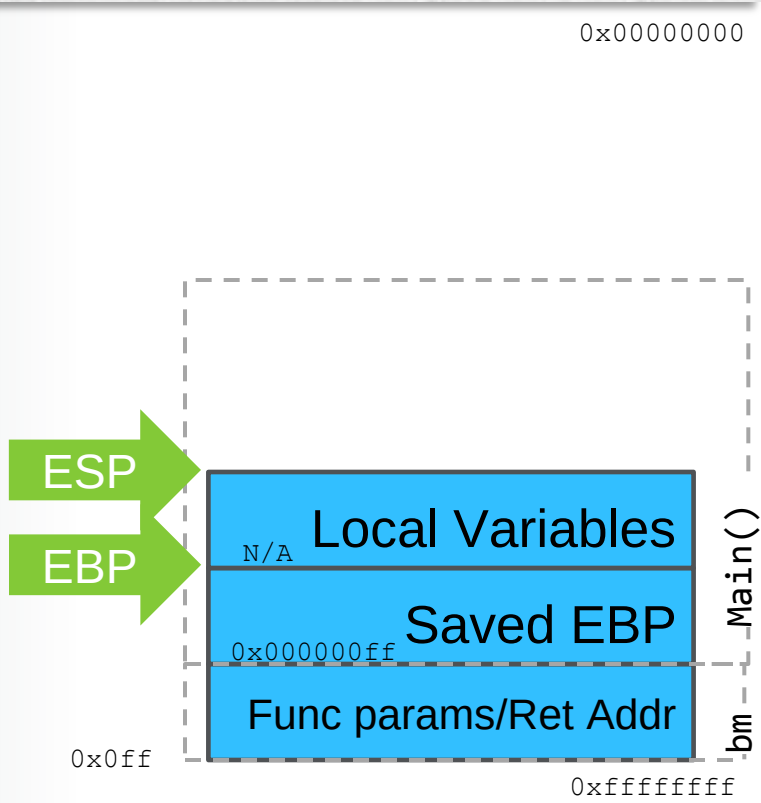


main()

EIP → Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



main()

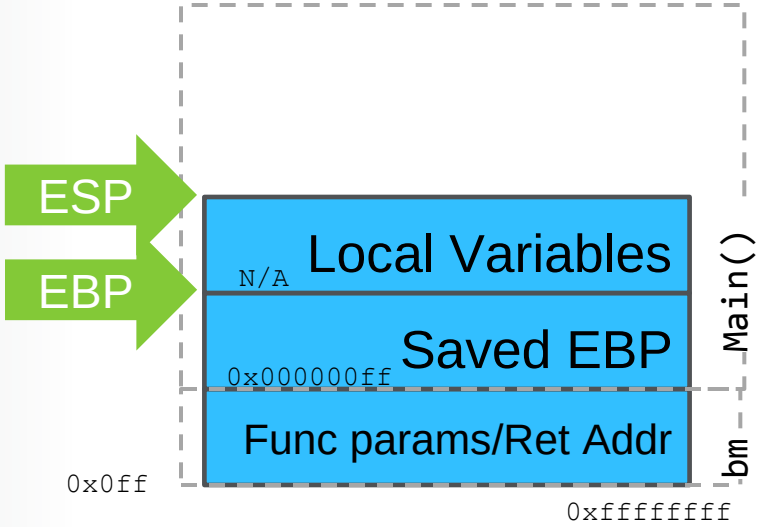
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

EIP

```
push 0x01001000
call Step()
```

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



main()

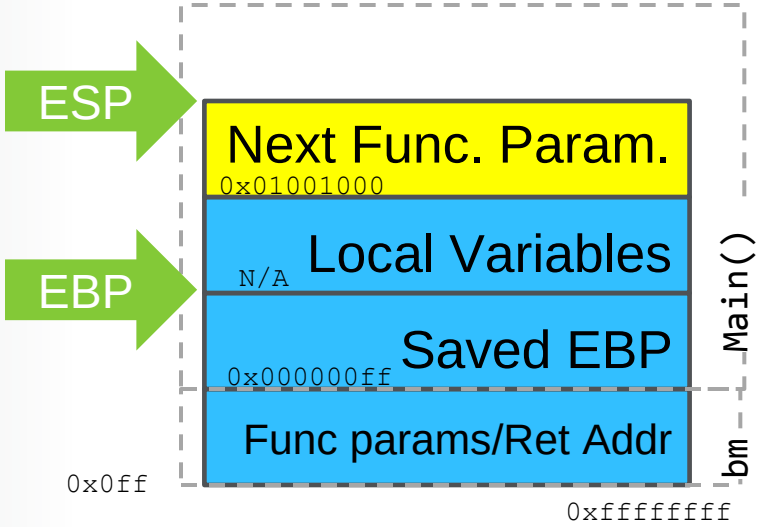
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

EIP

```
push 0x01001000
call Step()
```

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



main()

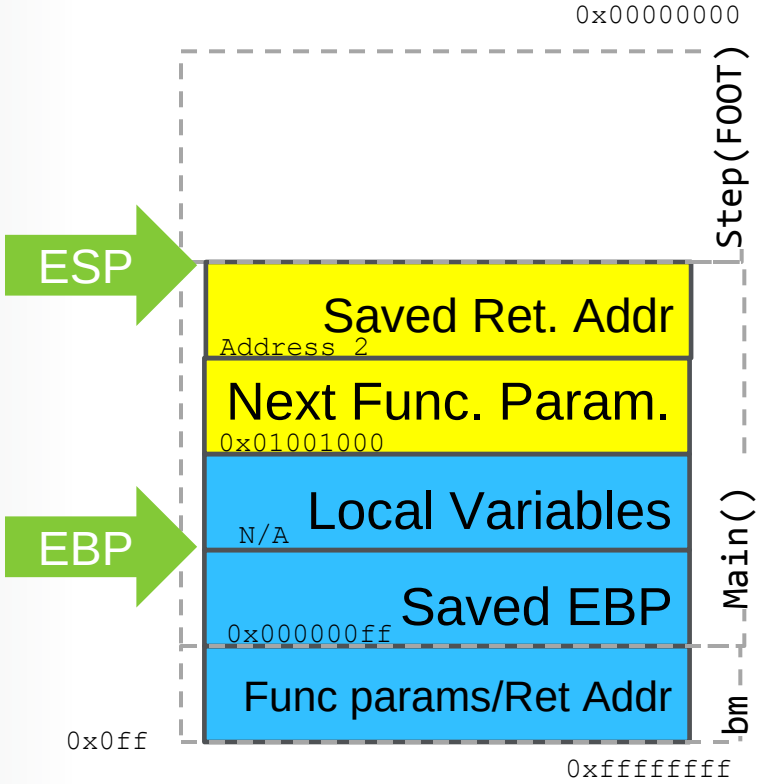
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

```
push 0x01001000
call Step()
```

Step(FOOT)

EIP

B. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



main()

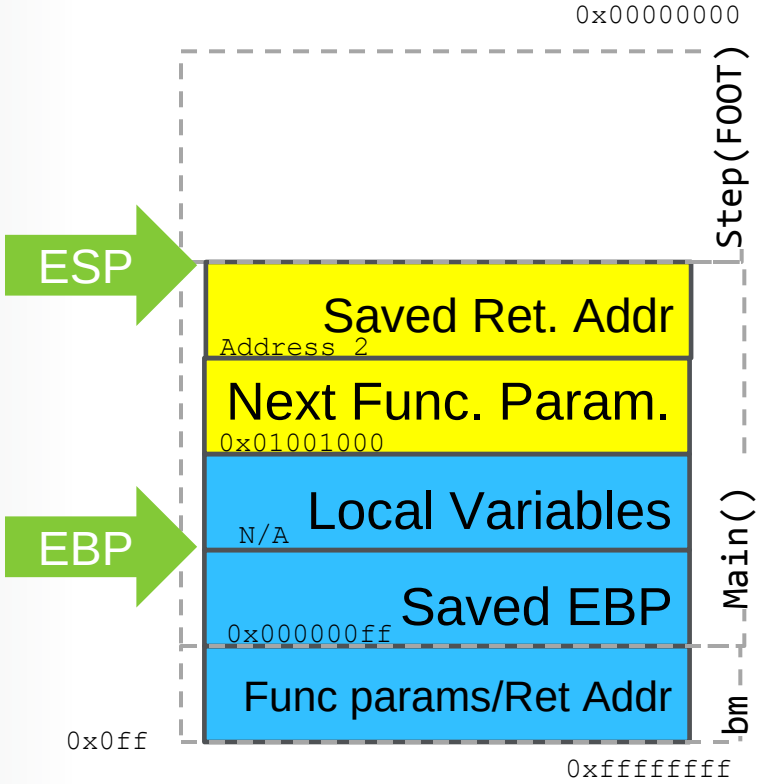
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

EIP

```
push ebp
mov ebp, esp
```

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



main()

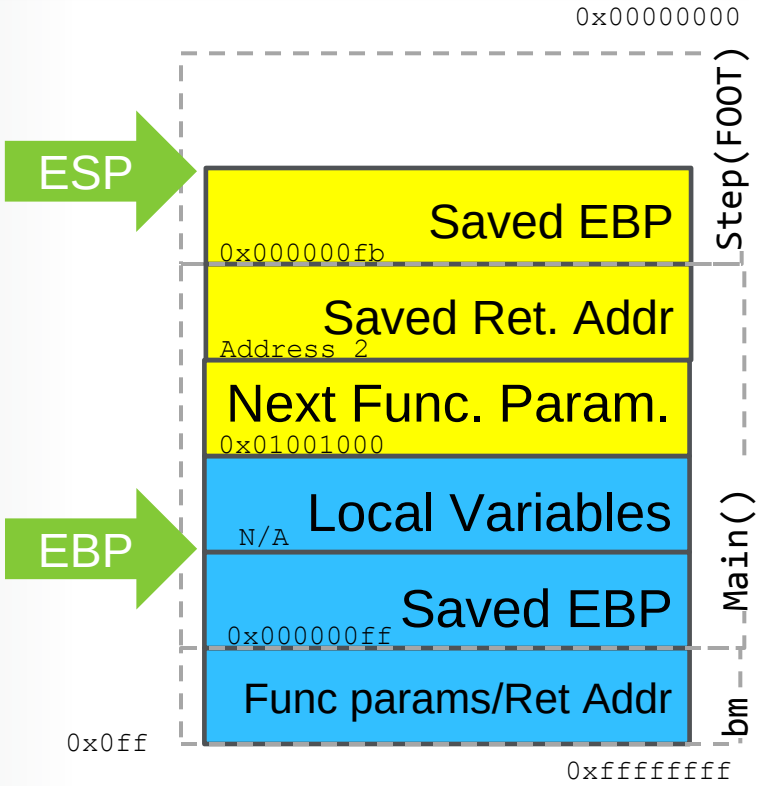
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

EIP

```
push ebp
mov ebp, esp
```

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



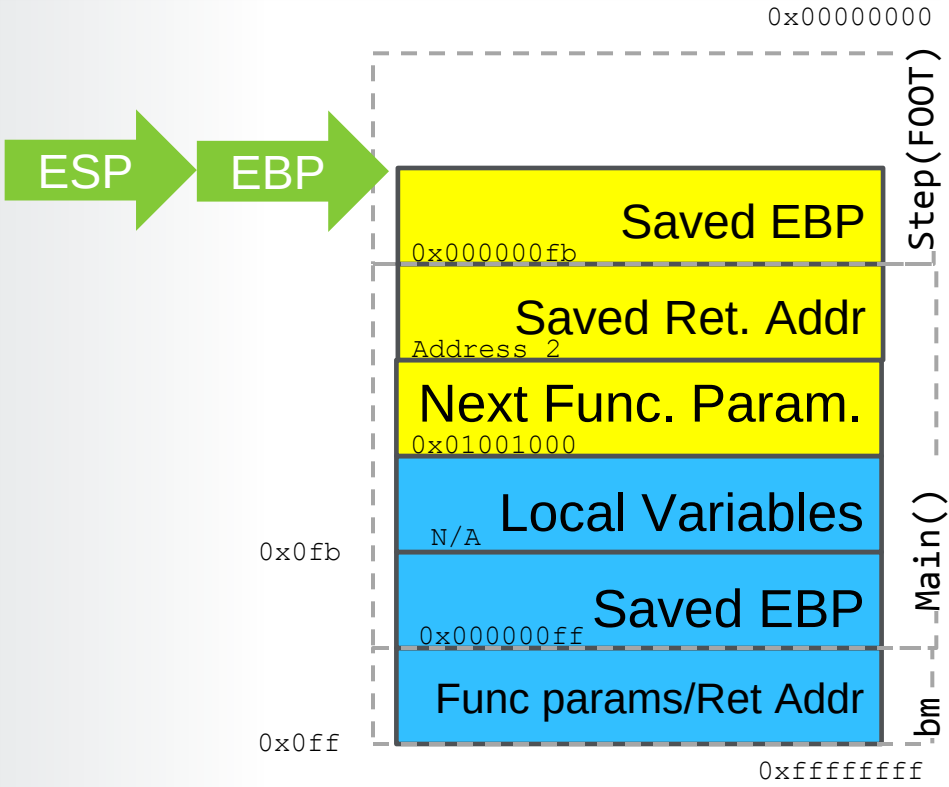
main()

1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

EIP

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



main()

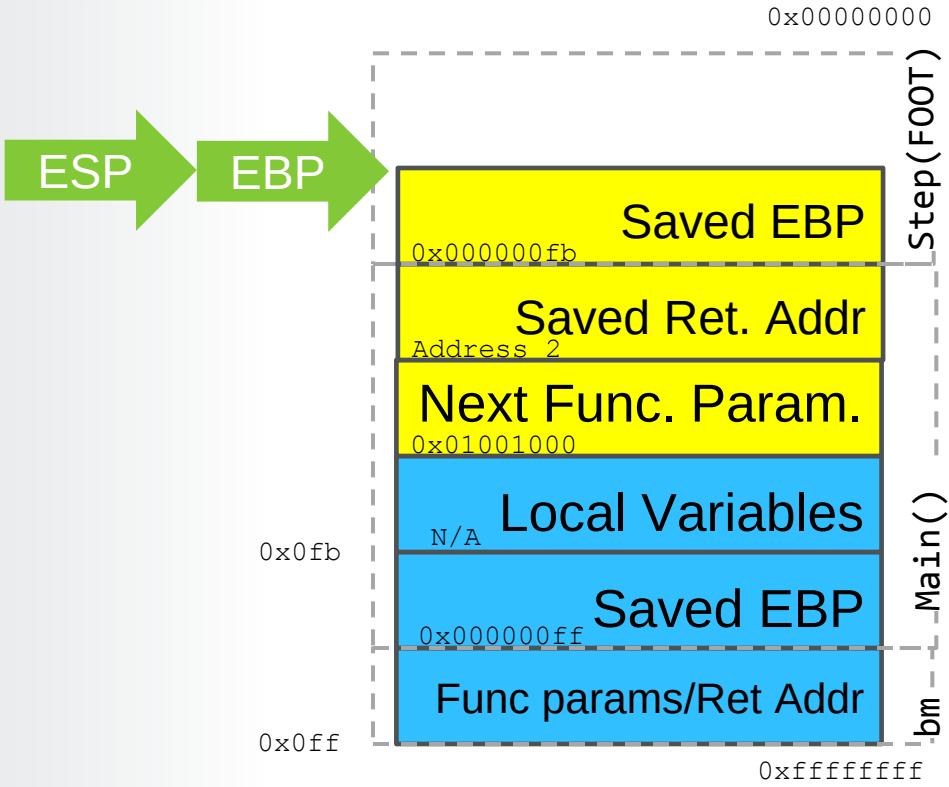
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

A. Lift FOOT foot

EIP → Move FOOT foot forward

C. Put FOOT foot down

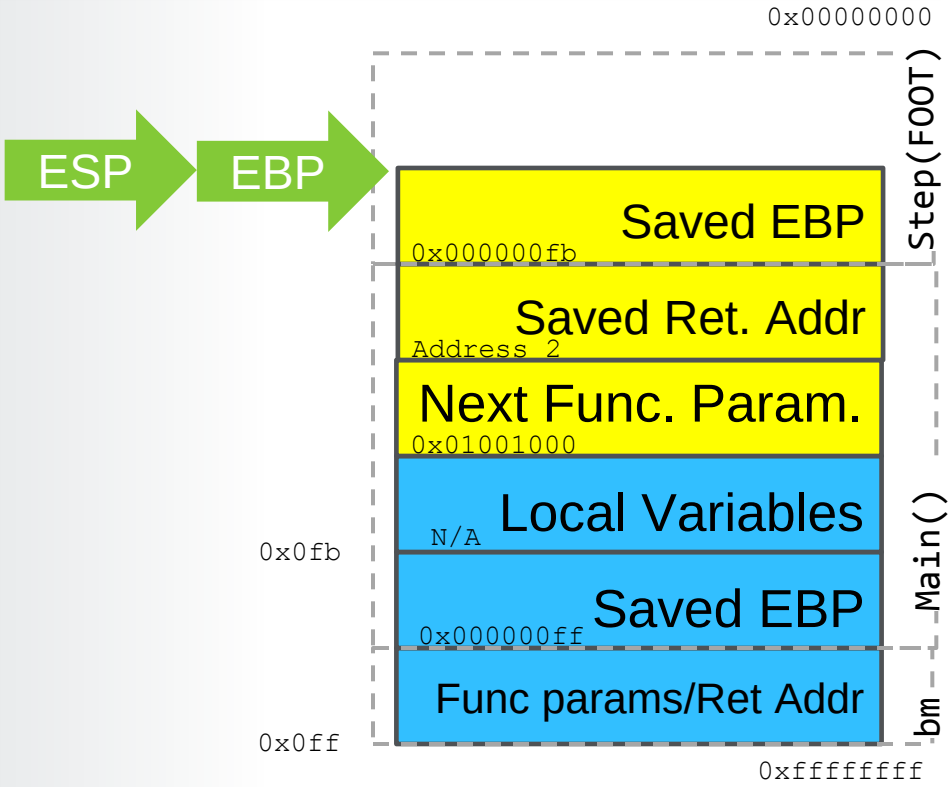


main()

1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
EIP Put FOOT foot down



main()

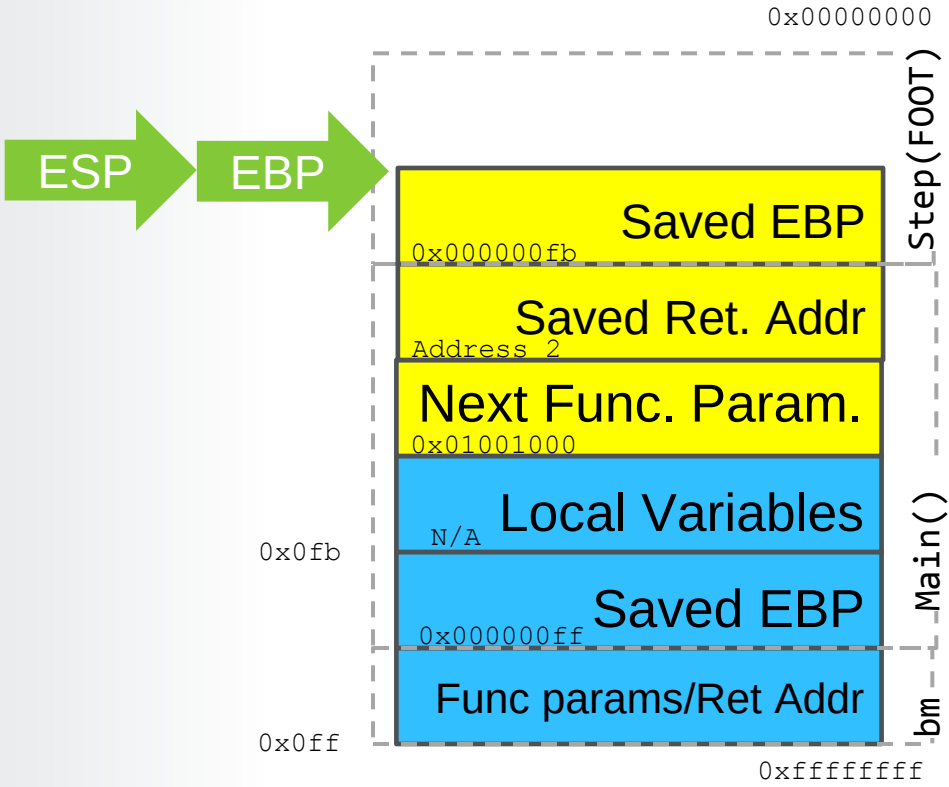
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down

EIP

```
pop ebp
ret 4
```



main()

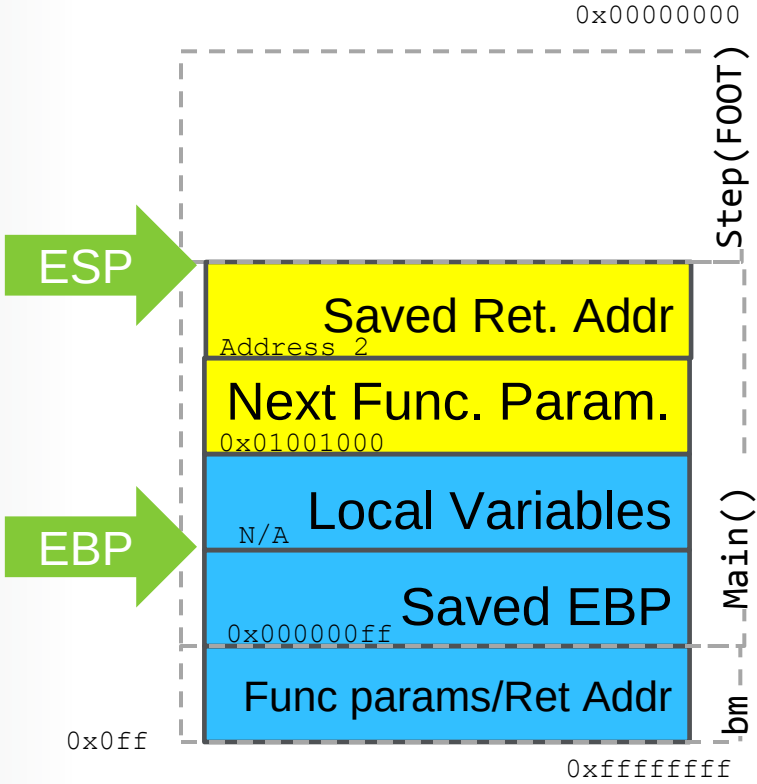
1. Step(Left);
2. Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

EIP

```
pop ebp
ret 4
```

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down



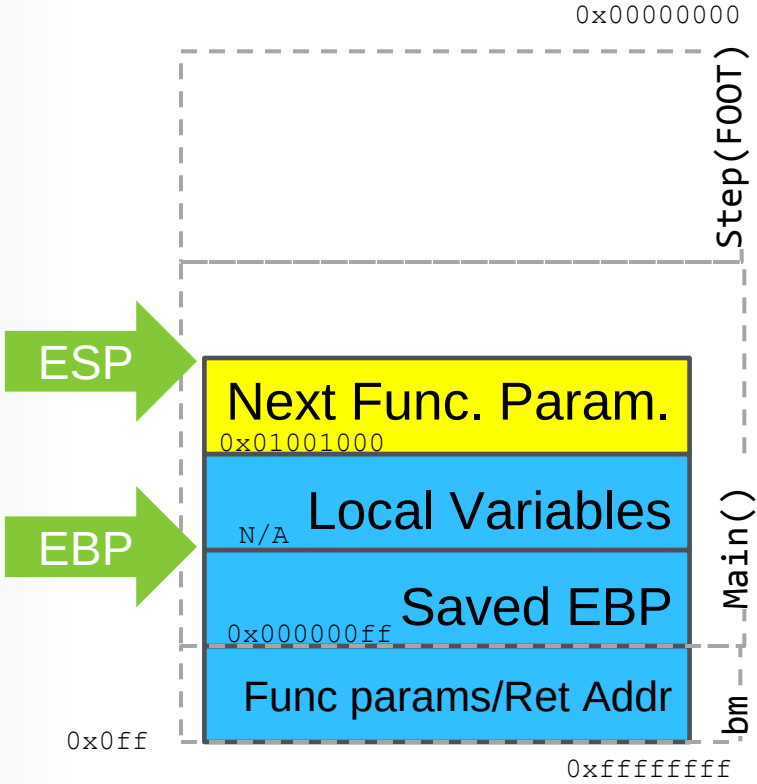
main()

1. Step(Left);
2. EIP → Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down

pop ebp
ret 4



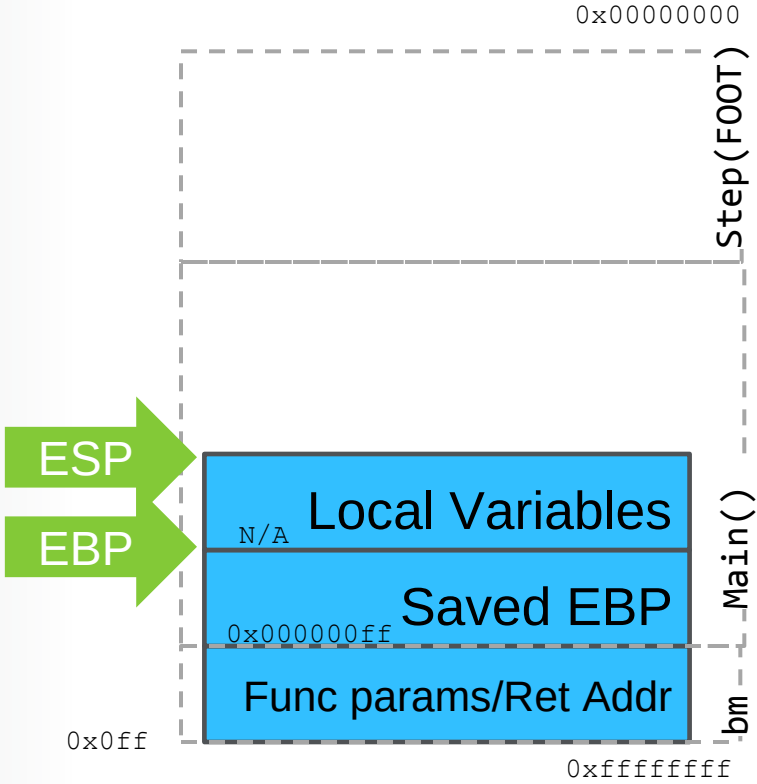
main()

1. Step(Left);
2. EIP → Step(Right);
3. Step(Left);
4. Step(Right);

Step(FOOT)

A. Lift FOOT foot
B. Move FOOT foot forward
C. Put FOOT foot down

```
pop ebp
ret 4
```



WINDBG KEEPS ON COMING!

Viewing Call Stack:

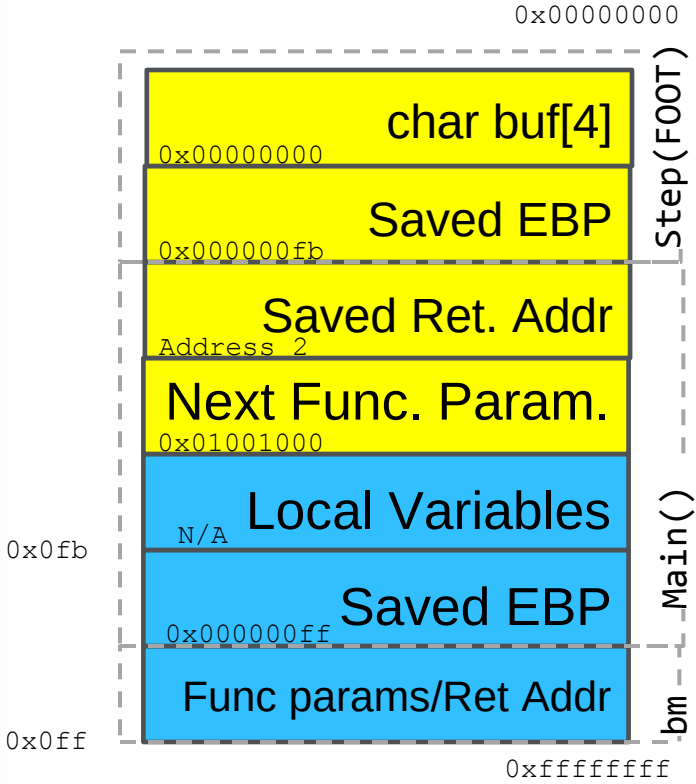
k

Baby IDA:

View->Disassembly

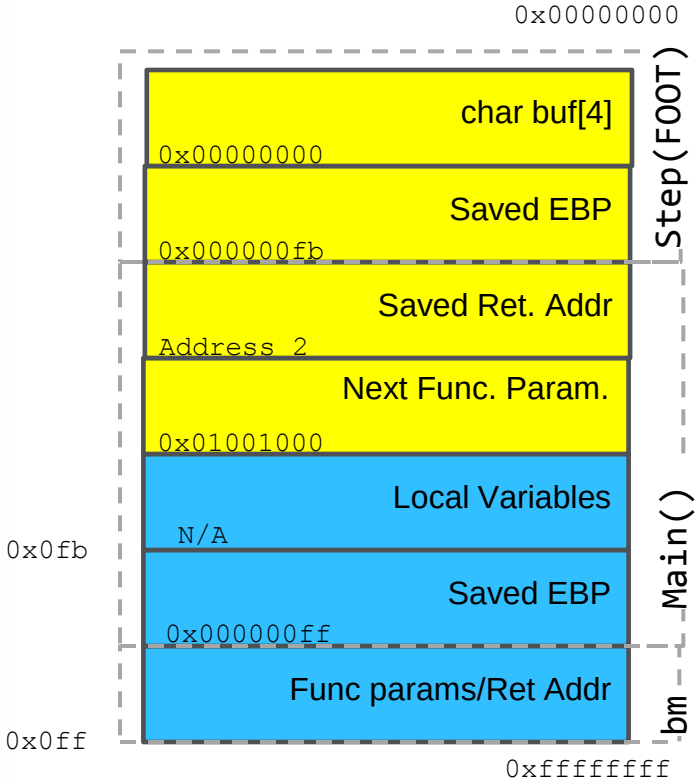
Exploit Round 1: Stack Overflow!

```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```



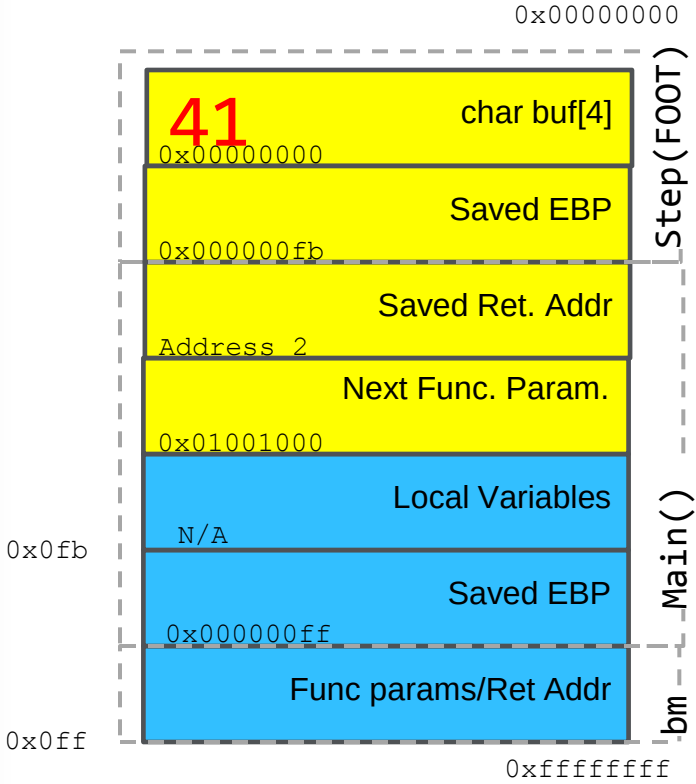
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → `memcpy()…`



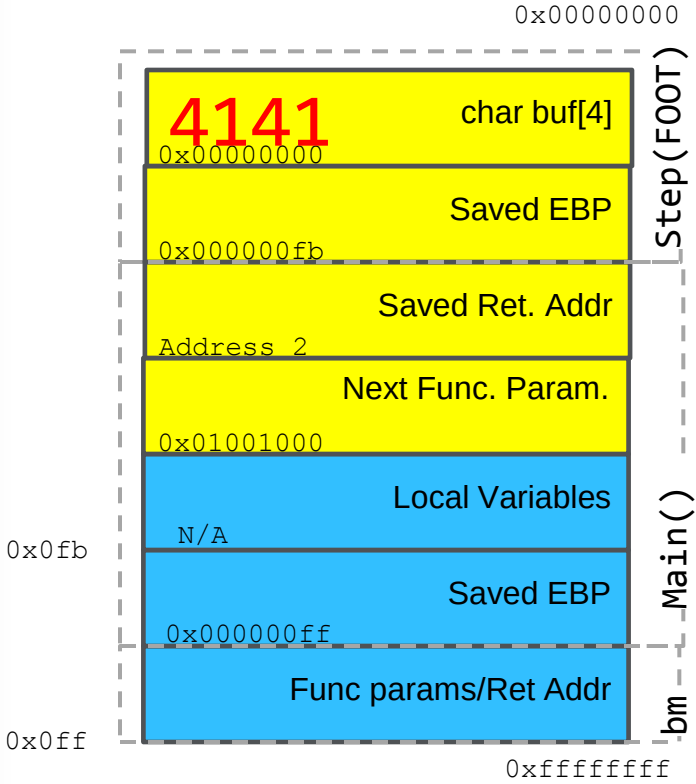
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



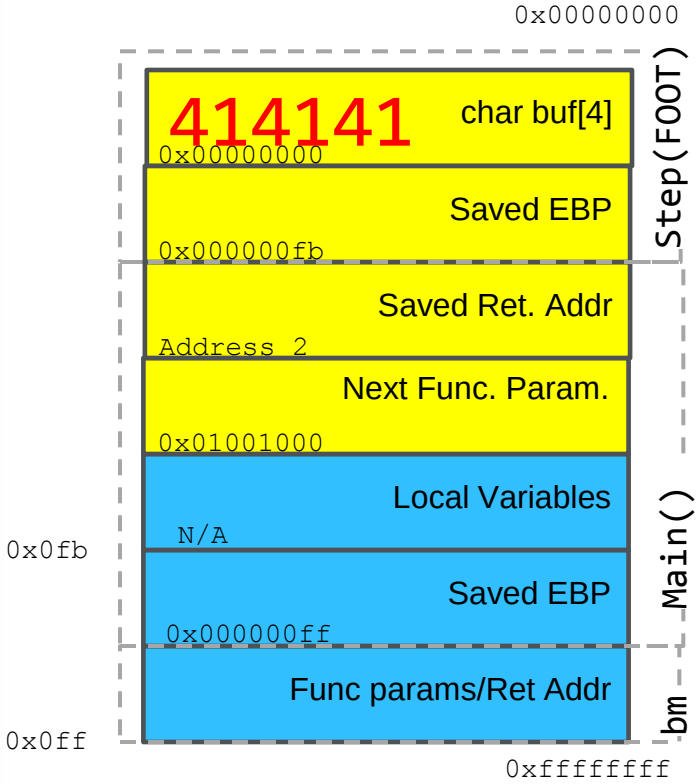
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



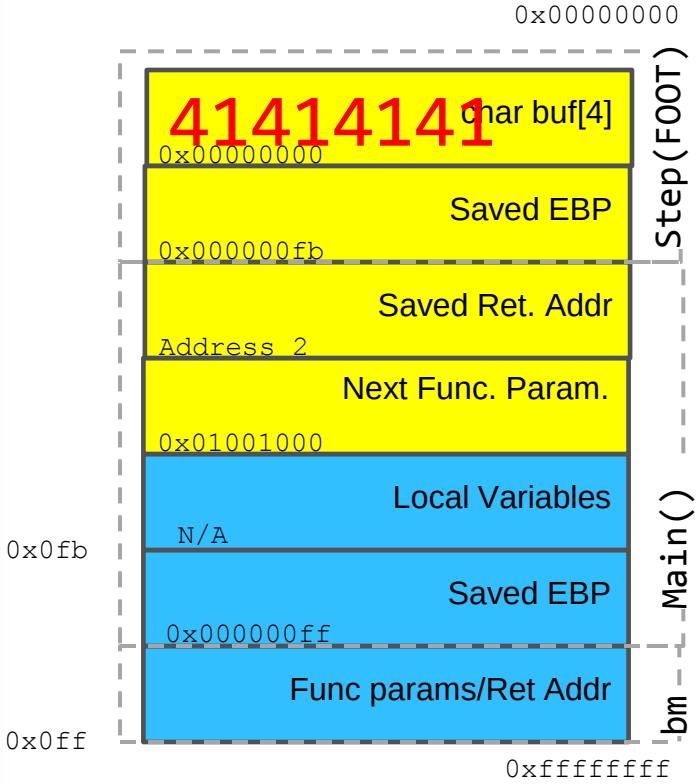
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



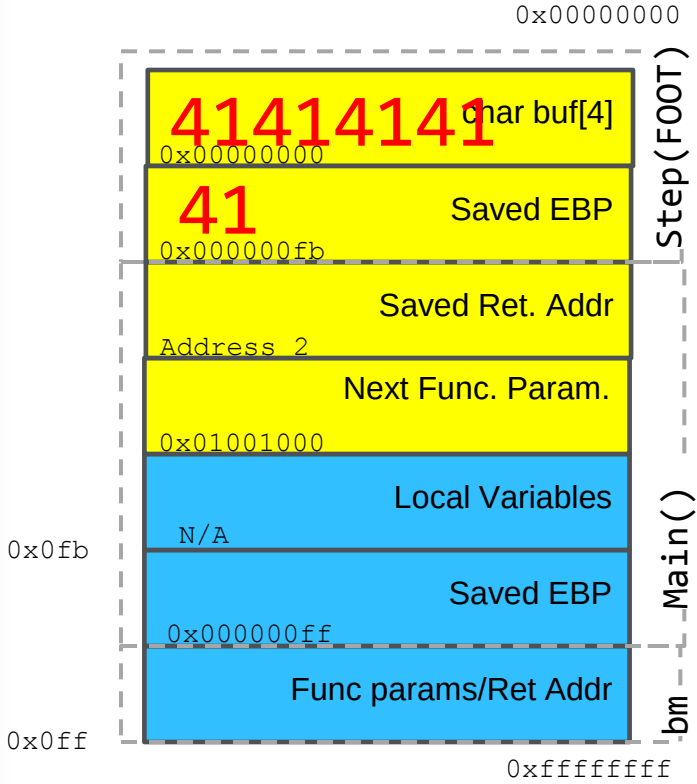
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



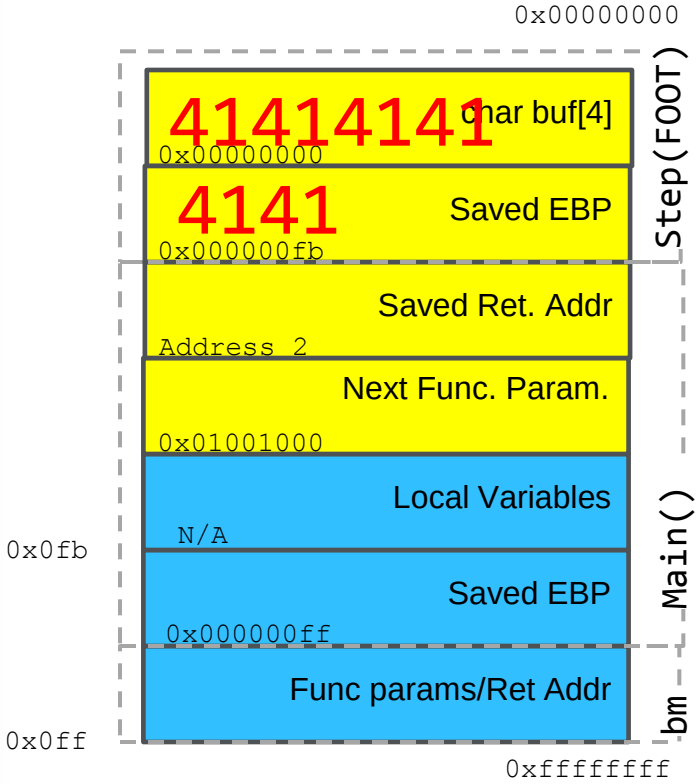
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



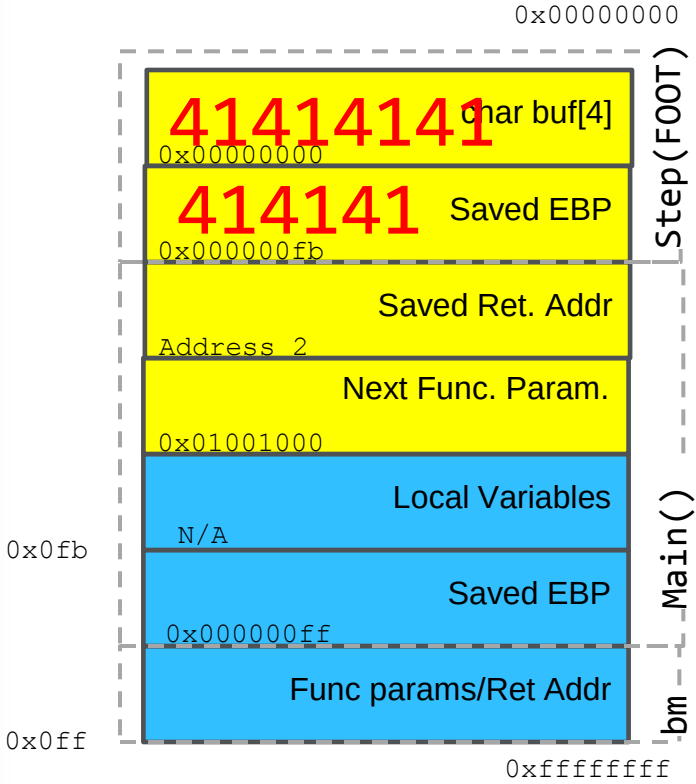
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



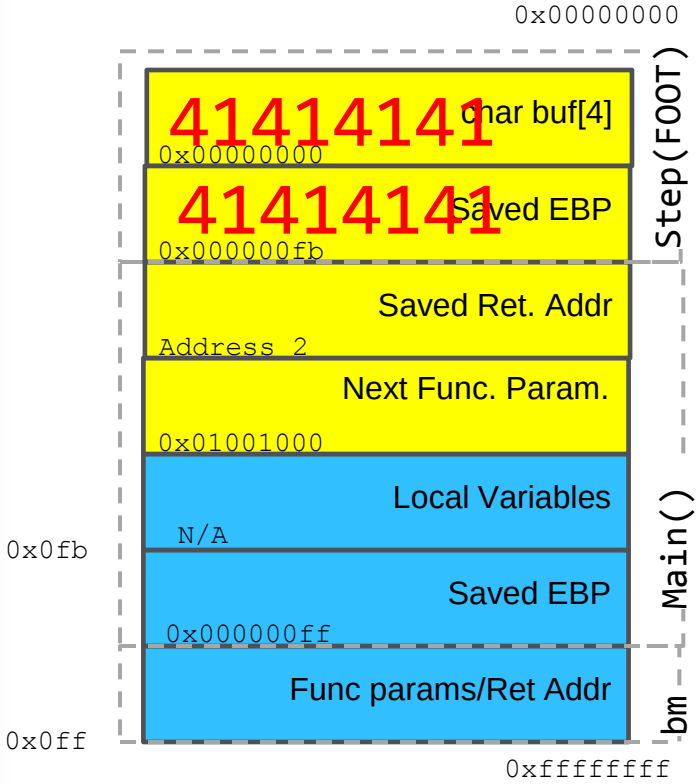
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



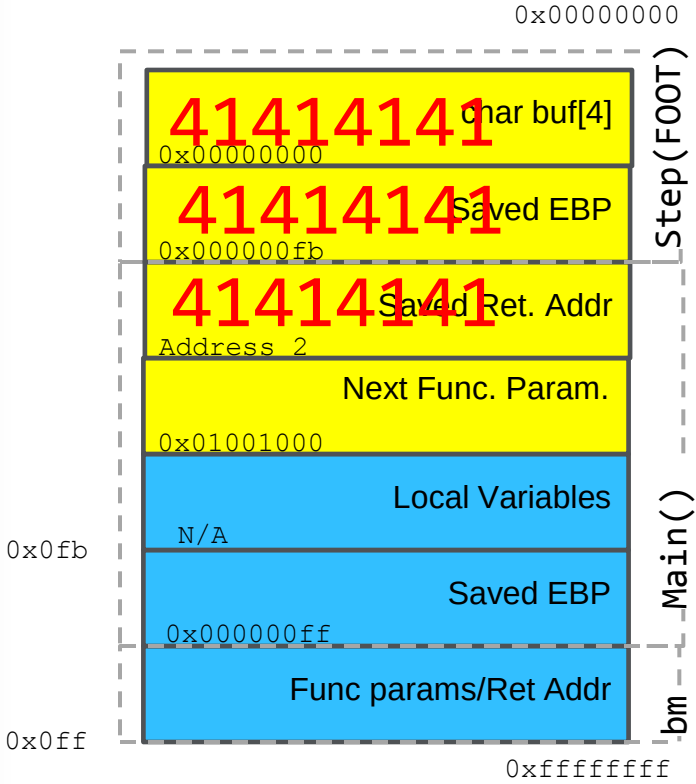
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



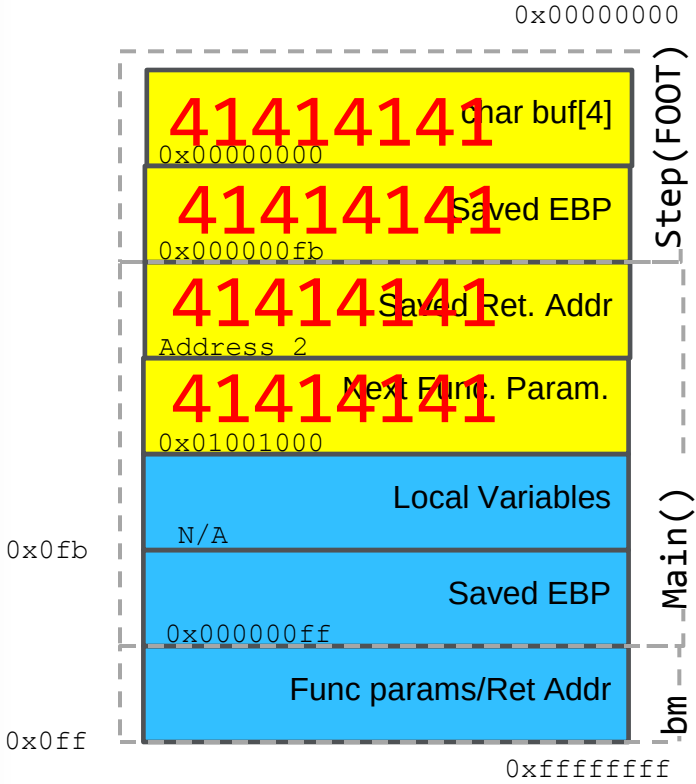
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



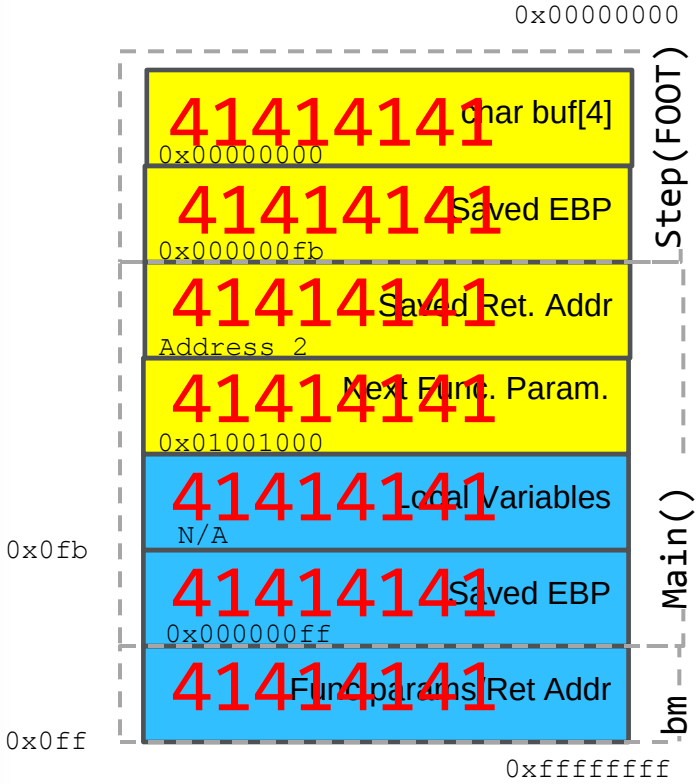
```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP → memcpy()…



```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

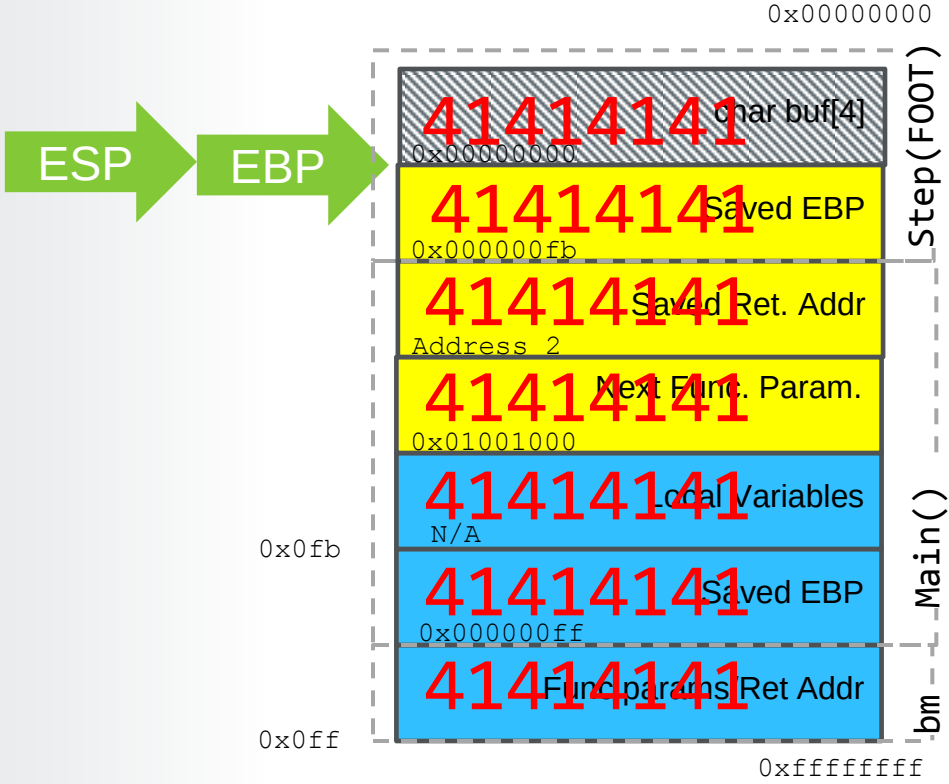
EIP → memcpy()…



```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP

```
pop ebp
ret 4
```



```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

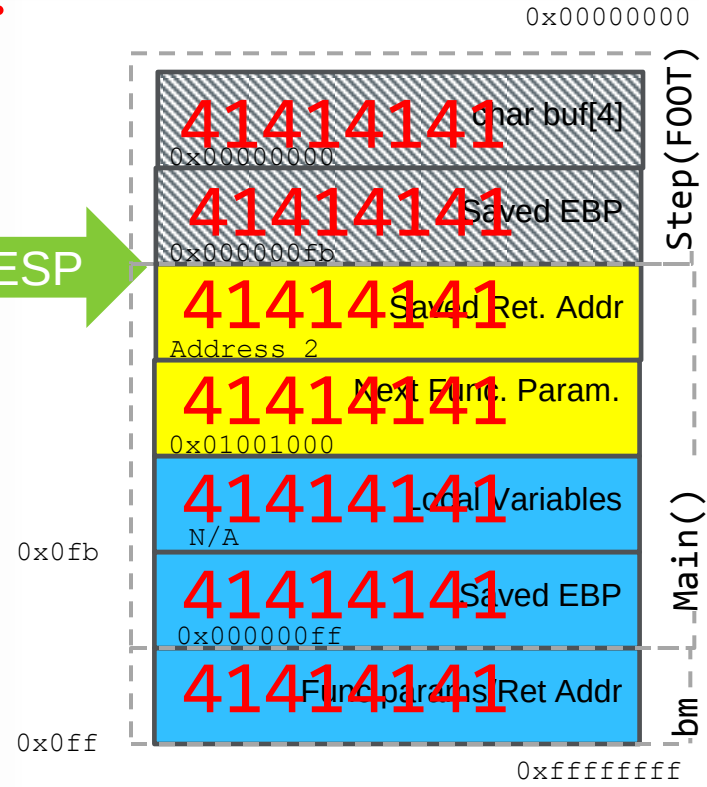
EIP

```
pop ebp
ret 4
```

EBP

41414141:????

ESP

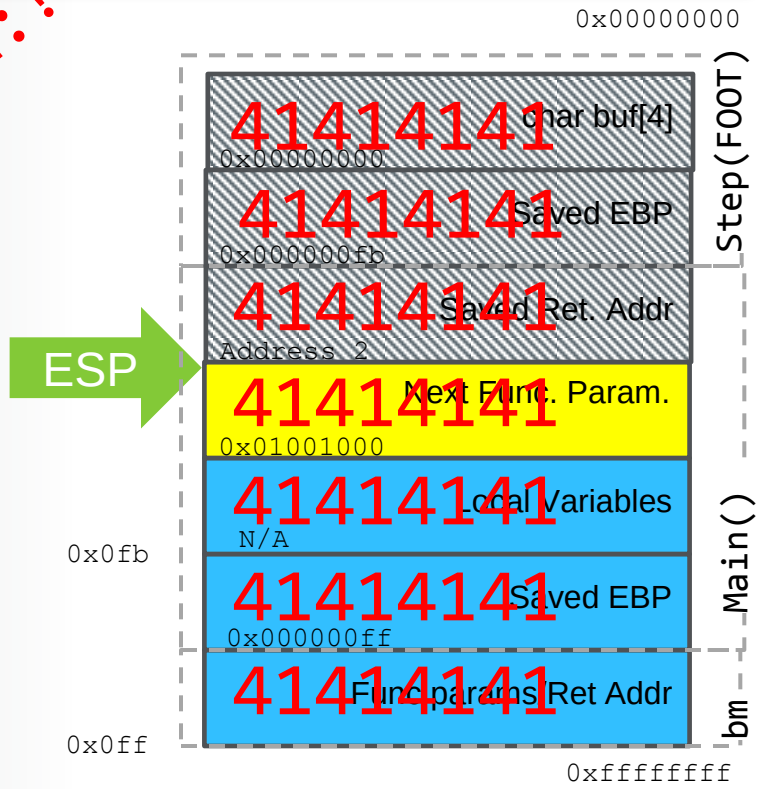


```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

```
pop ebp
ret 4
```

EIP → 41414141:????

EBP → 41414141:????



w000t!

we got program control!

But how do we get code execution?

1. Crash Triage
2. Determine the return address offset
3. Position our shellcode
4. Find the address of our shellcode

CRASH TRIAGE

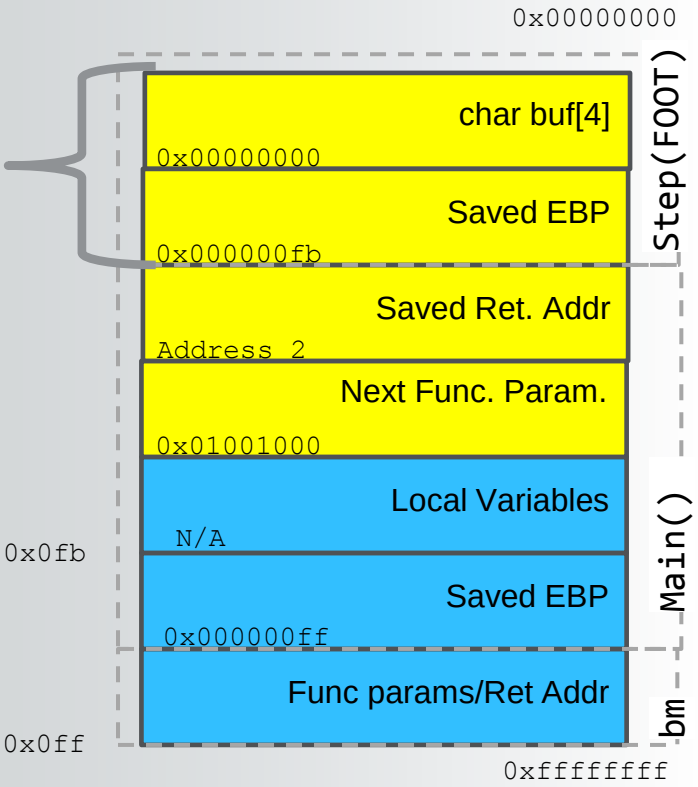
- What do we control?
  - What Registers contain attacker-controlled data?
  - What Registers point to attacker-controlled data?
  - Is attacker controlled data on the Stack or the Heap?
  - Do we control critical data such as stack frames?

- Where are we in the execution of the program?
  - Where is the vulnerability?
  - Was the crash caused by an exploit mitigation?

1. Crash Triage
2. Determine the return address offset
3. Position our shellcode
4. Find the address of our shellcode

OFFSET OF THE RETURN ADDRESS

How many bytes?

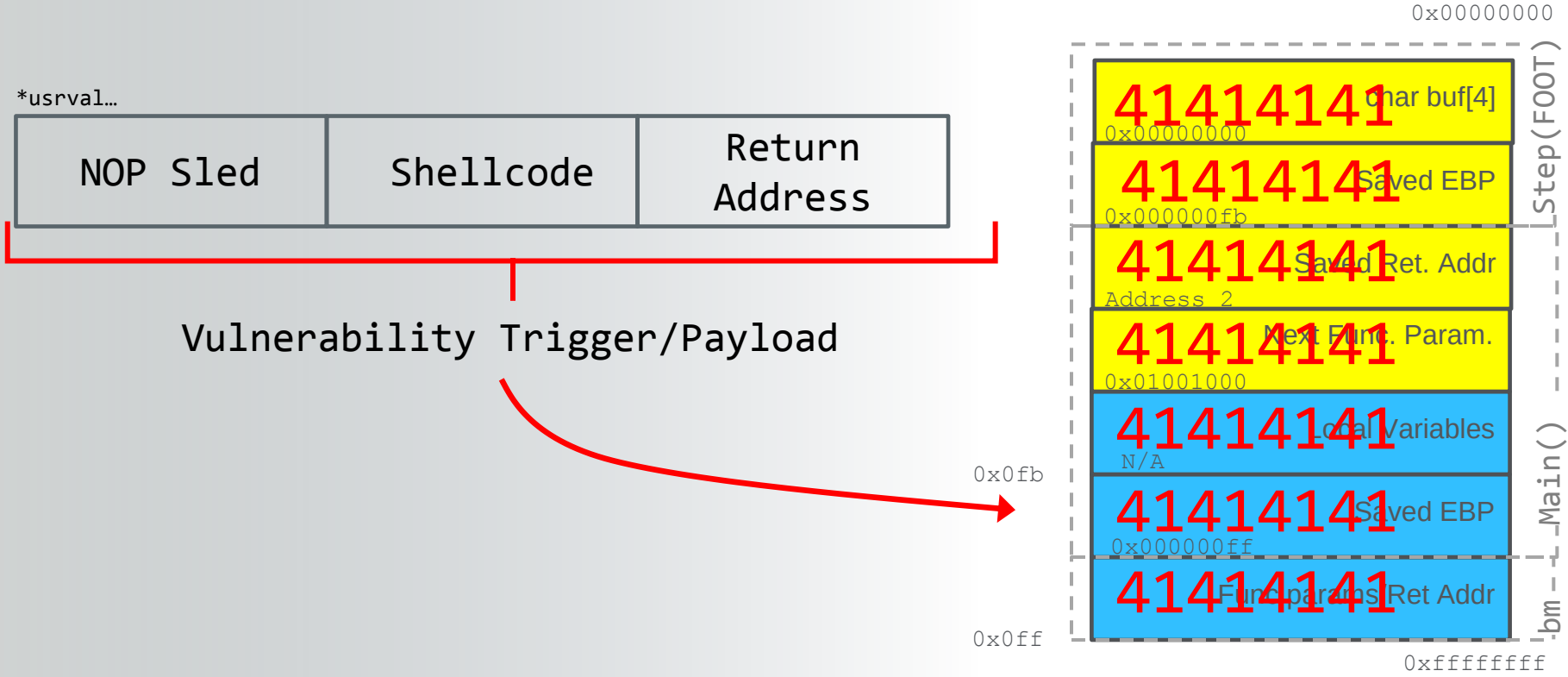


- Figure out the offset to EIP overwrite
- Don’t fear javascript :)
- Lab helpers:
  - Built in ‘`msfPatternString`’ variable
  - `From WinDBG:`
    - `!load byakugan, !pattern_offset 2000`

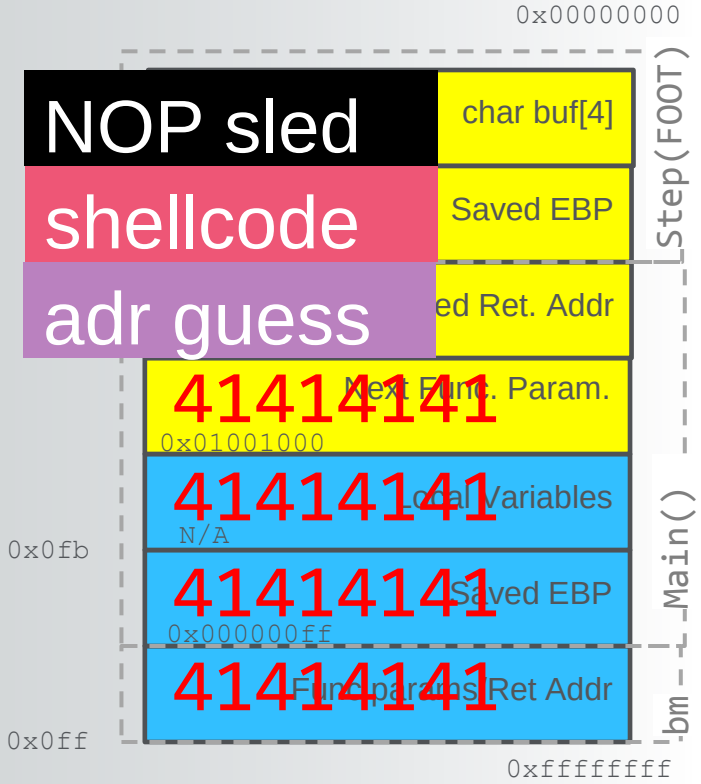
1. Crash Triage
2. Determine the return address offset
3. Position our shellcode
4. Find the address of our shellcode

Time Warp to 1996

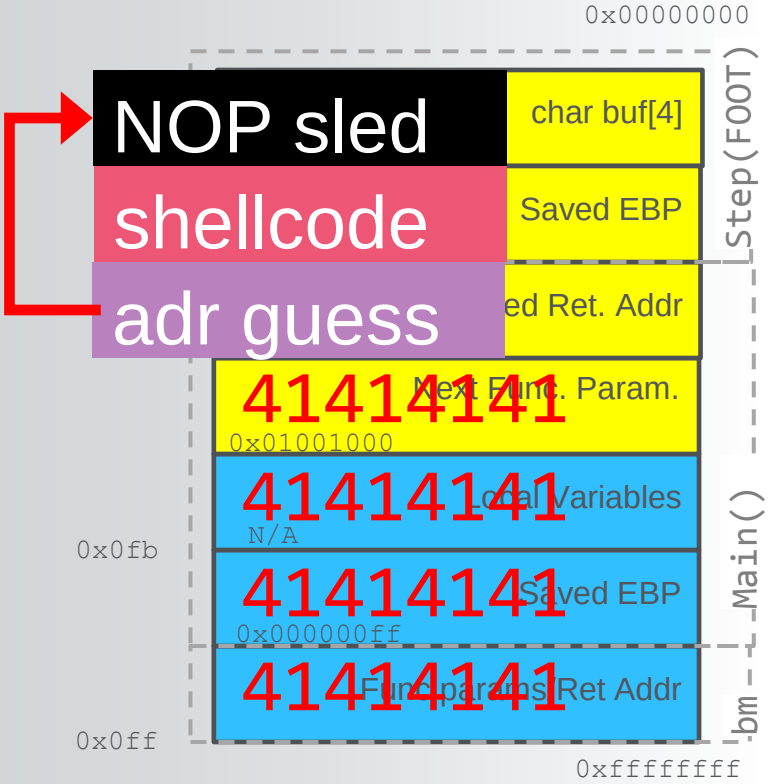
LINEAR STACK OVERFLOW



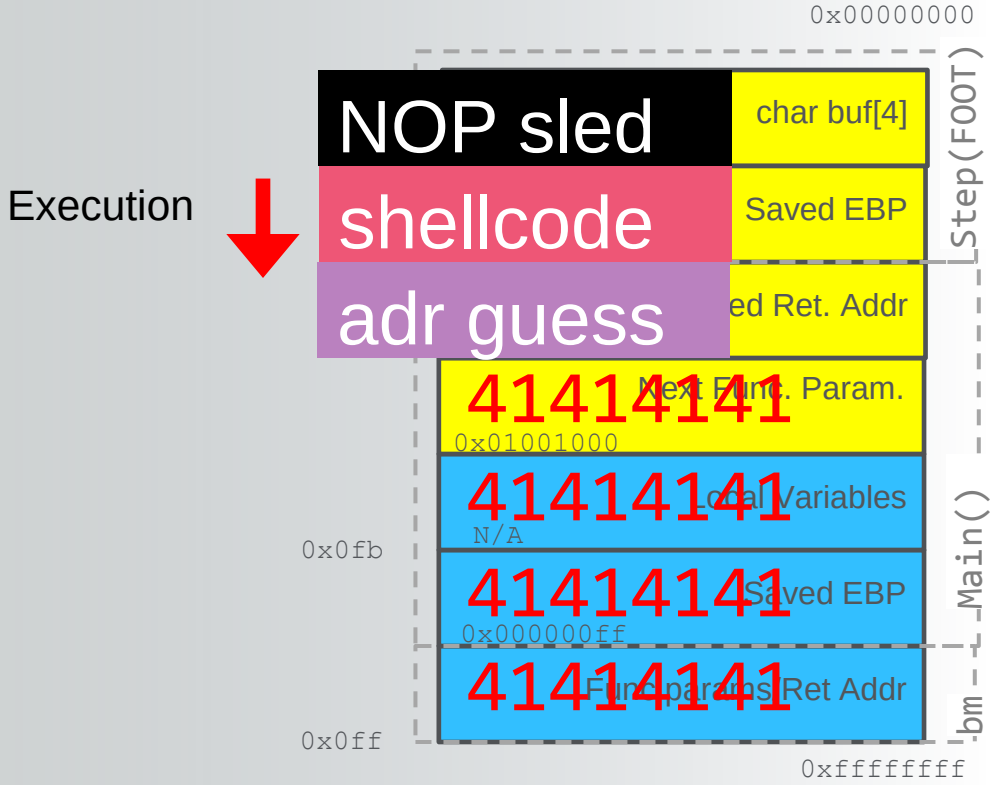
LINEAR STACK OVERFLOW

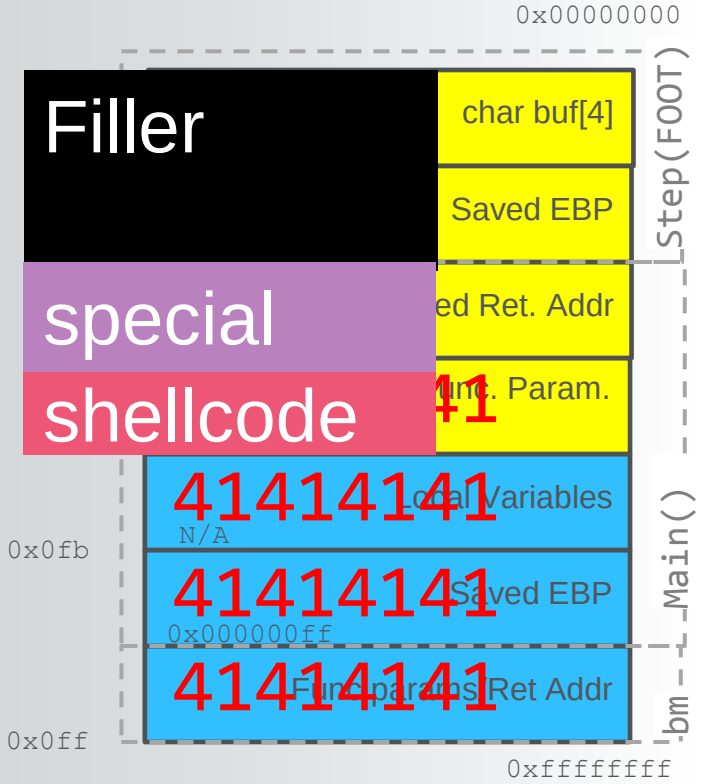


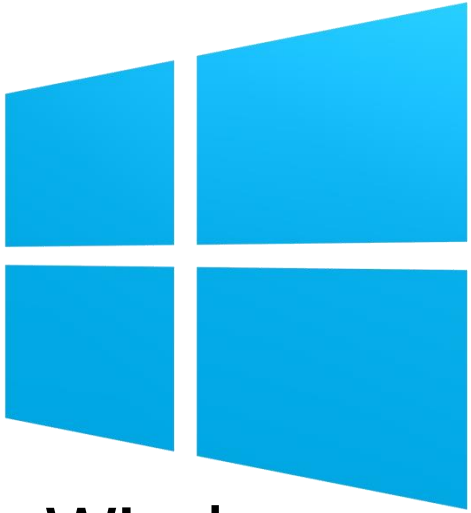
LINEAR STACK OVERFLOW



LINEAR STACK OVERFLOW



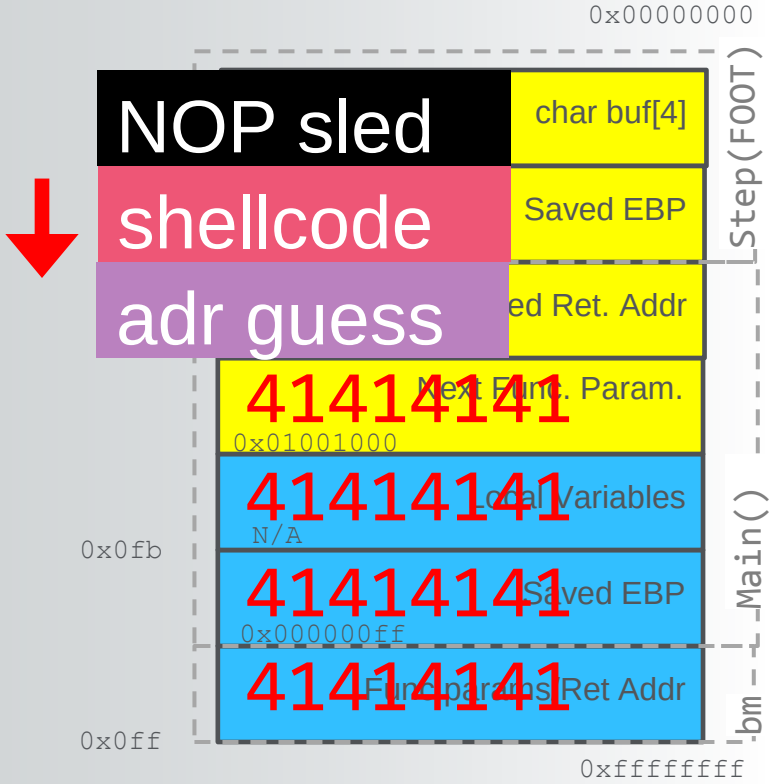




Windows
(and modern OS's)

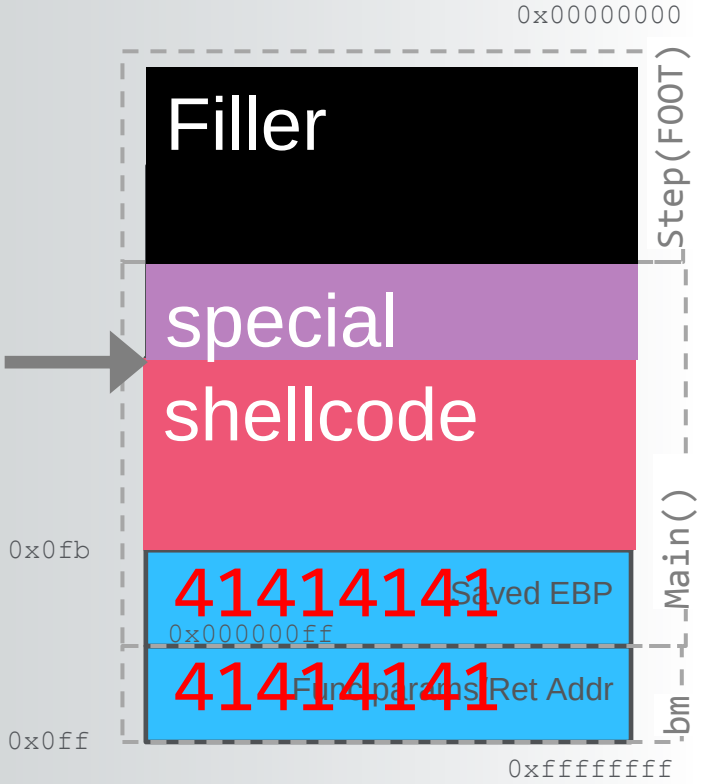
1. Crash Triage
2. Determine the return address offset
3. Position our shellcode
4. Find the address of our shellcode

LINEAR STACK OVERFLOW



Good for *nix
(consistent stack)

What memory address is this at?



```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

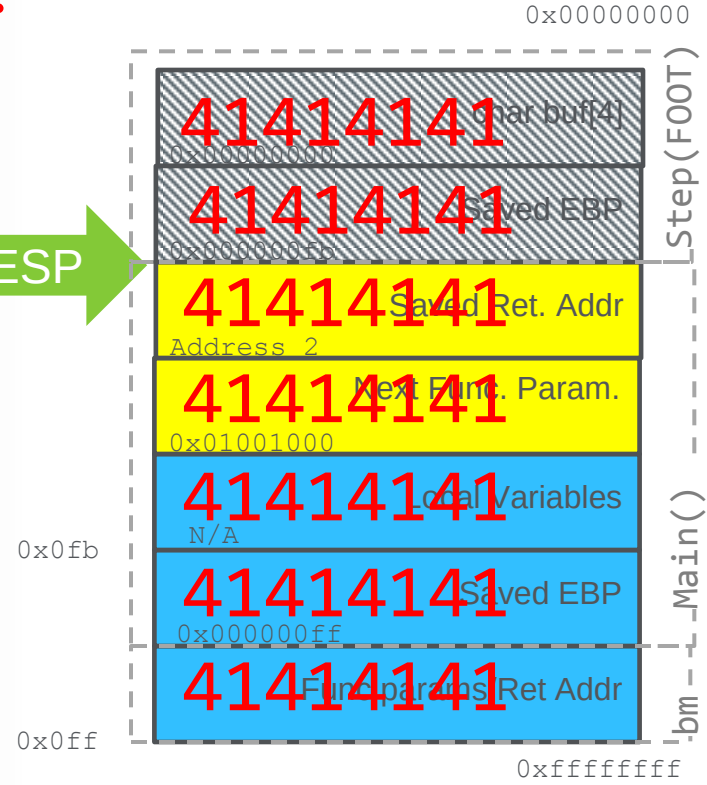
EIP

```
pop ebp
ret 4
```

EBP

41414141:????

ESP

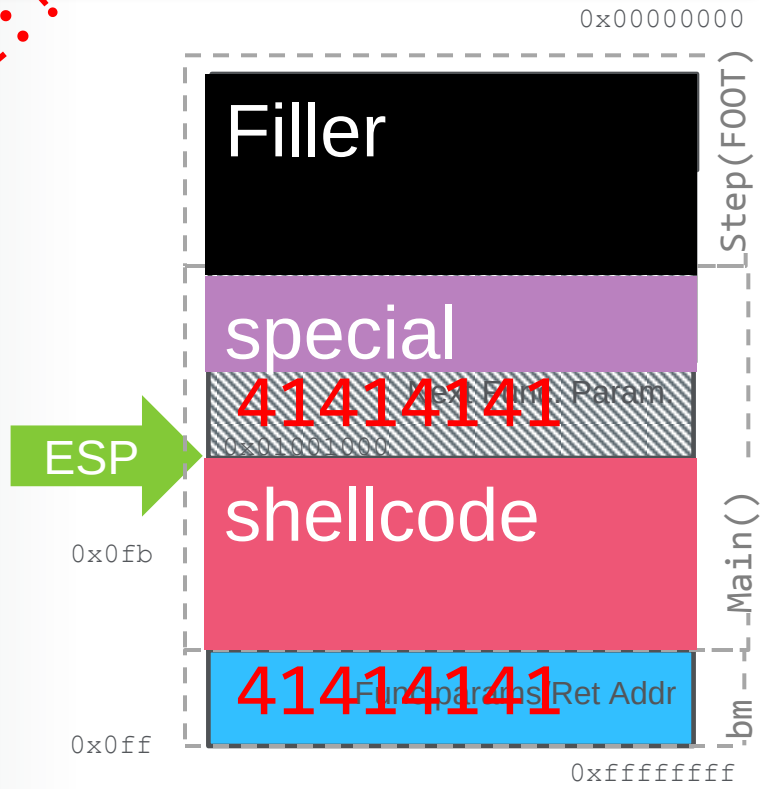


```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

```
pop ebp
ret 4
```

EIP → 41414141:????

EBP → 41414141:????



Trampoline!

Jump to a location which jumps to another (e.g. addr of a `jmp esp` instruction)

TRAMPOLINE

1. Find a module loaded at a static address
2. Find "`jmp esp`" (or similar instruction) within that memory space

TRAMPOLINE

```
0:012> u 77c373cb
ntdll!RtlpConsole+0xd4:
77c373cb 2a81ffe40400    sub     al,byte ptr [ecx+4E4FFh]
77c373d1 00742281        add     byte ptr [edx-7Fh],dh
```

TRAMPOLINE

```
0:012> u 77c373cb
ntdll!RtlpConsole+0xd4:
77c373cb 2a81ffe40400    sub     al,byte ptr [ecx+4E4FFh]
77c373d1 00742281        add     byte ptr [edx-7Fh],dh
```

TRAMPOLINE

```
0:012> u 77c373cb
ntdll!RtlpConsole+0xd4:
77c373cb 2a81ffe40400    sub     al,byte ptr [ecx+4E4FFh]
77c373d1 00742281        add     byte ptr [edx-7Fh],dh

0:012> u 77c373cd
ntdll!RtlpConsole+0xda:
77c373cd ffe4            jmp     esp
77c373cf 0400            add     al,0
```

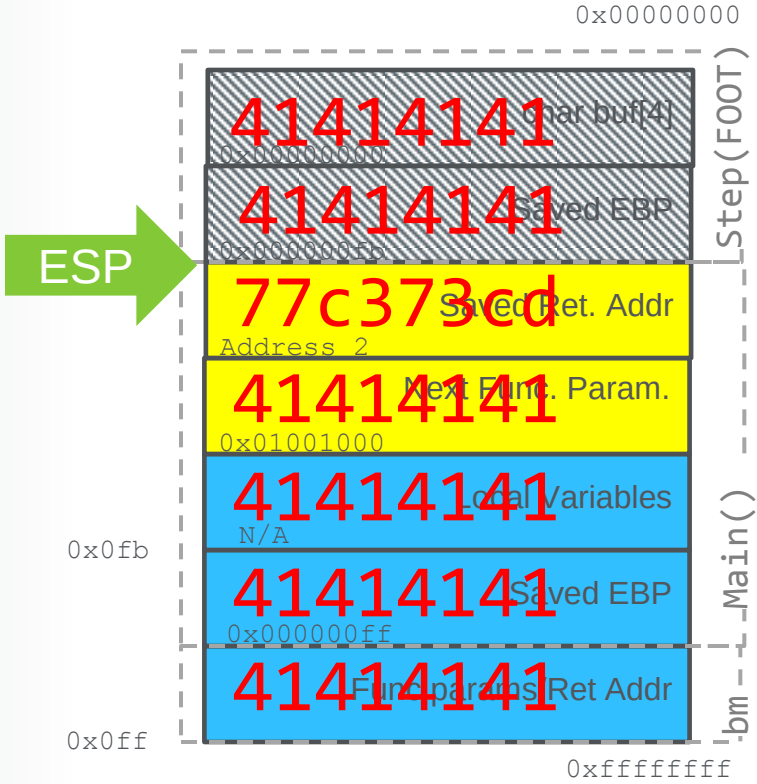
TRAMPOLINE

```
0:012> u 77c373cb
ntdll!RtlpConsole+0xd4:
77c373cb 2a81ffe40400    sub     al,byte ptr [ecx+4E4FFh]
77c373d1 00742281        add     byte ptr [edx-7Fh],dh

0:012> u 77c373cd
ntdll!RtlpConsole+0xda:
77c373cd ffe4            jmp     esp
77c373cf 0400            add     al,0
```

```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

EIP

```
pop ebp
ret 4
```

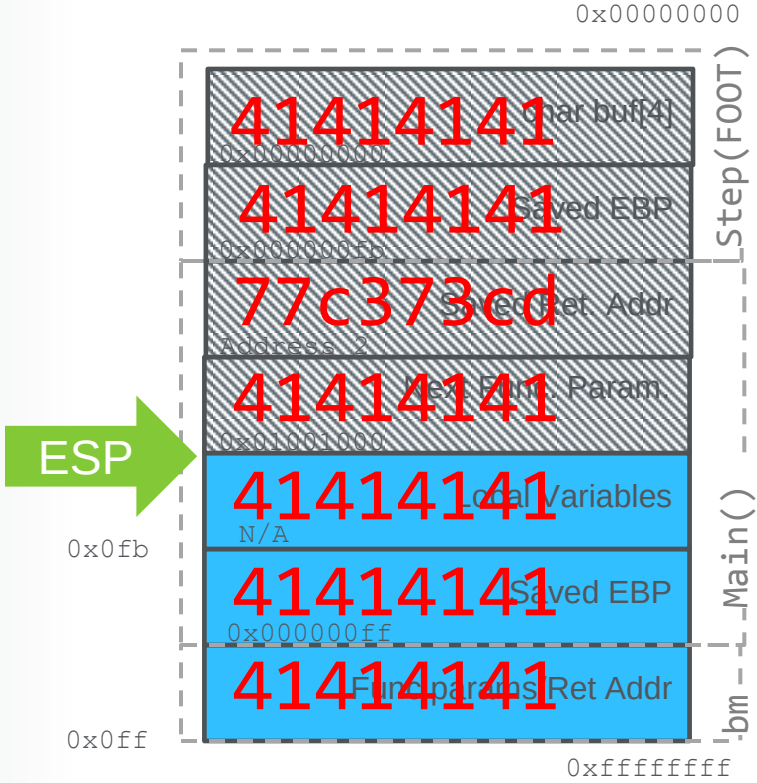


```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

```
pop ebp
ret 4
```

Ntdll.dll (0x77c373cd)

EIP → `jmp esp`

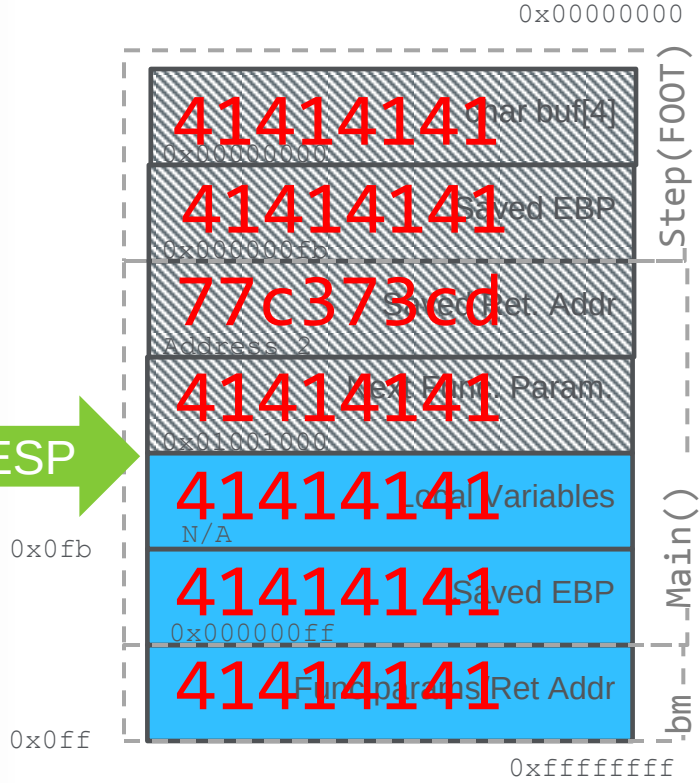


```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

```
pop ebp
ret 4
```

Ntdll.dll (0x77c373cd)

EIP `jmp esp`

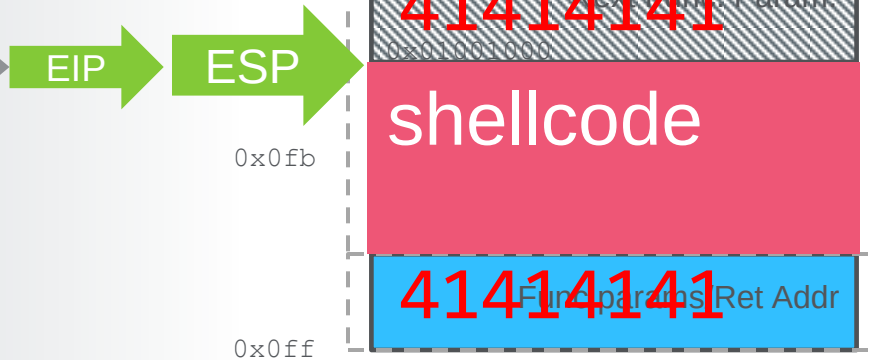


```
func(char *usrval, int len) {
    char buf[4];
    memcpy(buf, usrval, len);
}
```

```
pop ebp
ret 4
```

Ntdll.dll (0x77c373cd)

EIP `jmp esp`

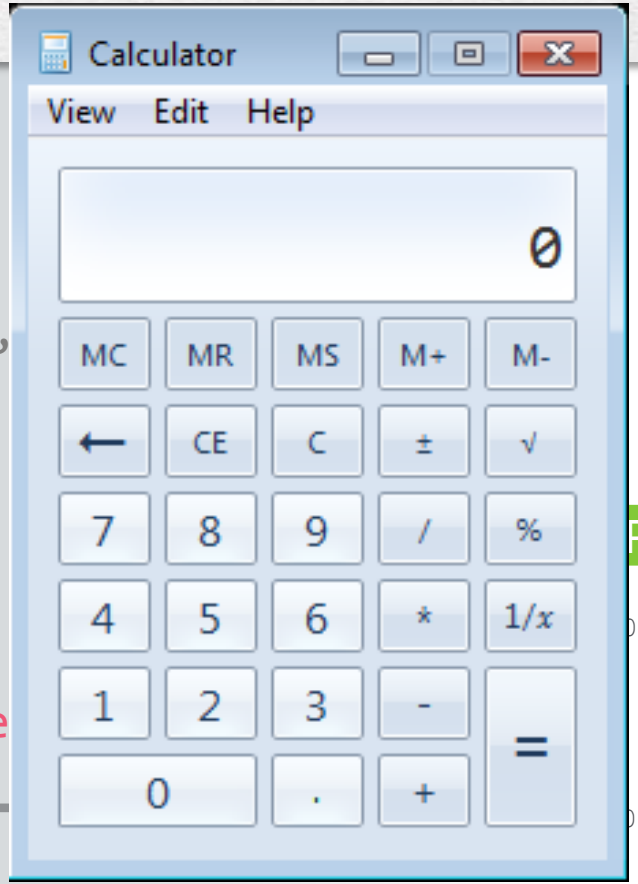


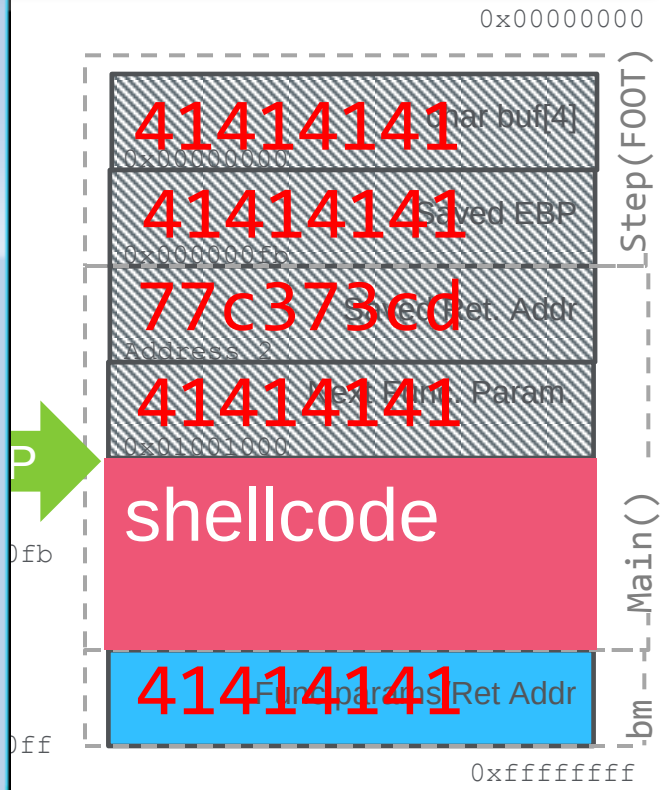
```
func(char *usrval, int
    char buf[4];
    memcpy(buf, usrval,
}
```

```
pop ebp
ret 4
```

Ntdll.dll

EIP jmp e





Lab 2: Smashing the Stack!

1. Triage:
   - ‘k’ for call stack and disassembly view
   - ‘bp <addr>’ for break points, ‘t’ to step into
2. Trigger (build the ‘s’ variable in the JS)
   - `MakeString(Amount); // 1 = 2 bytes`
   - `Remember order (12345678 = \u5678\u1234)`
3. Find address to jmp esp in windbg, add it to ‘s’
   - `s [start] [end] ff e4`
4. Add in ‘`shellcode`’ variable to ‘s’