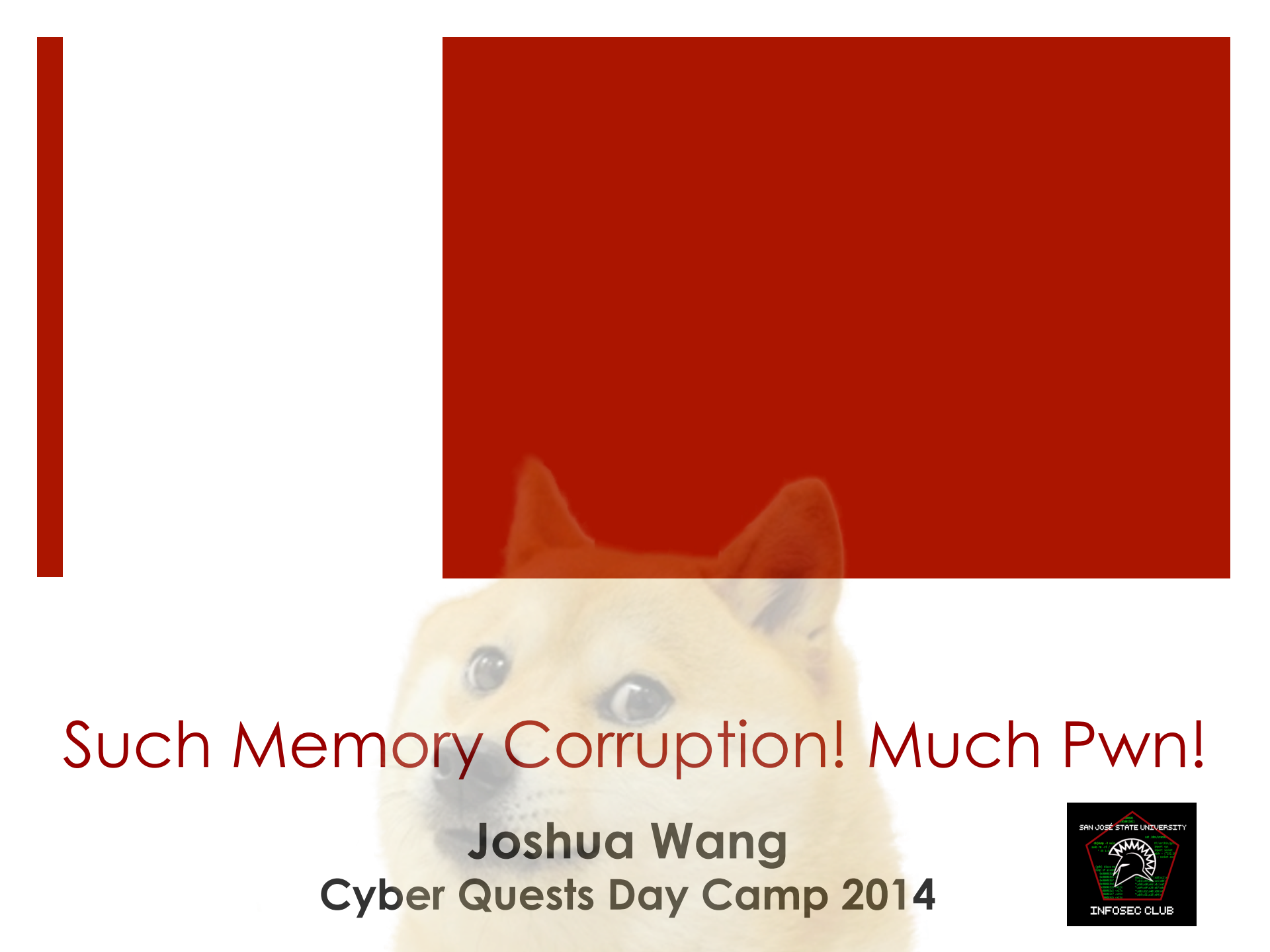


Such Memory Corruption! Much Pwn!

Joshua Wang

Cyber Quests Day Camp 2014

SAN JOSE STATE UNIVERSITY

INFOSEC CLUB

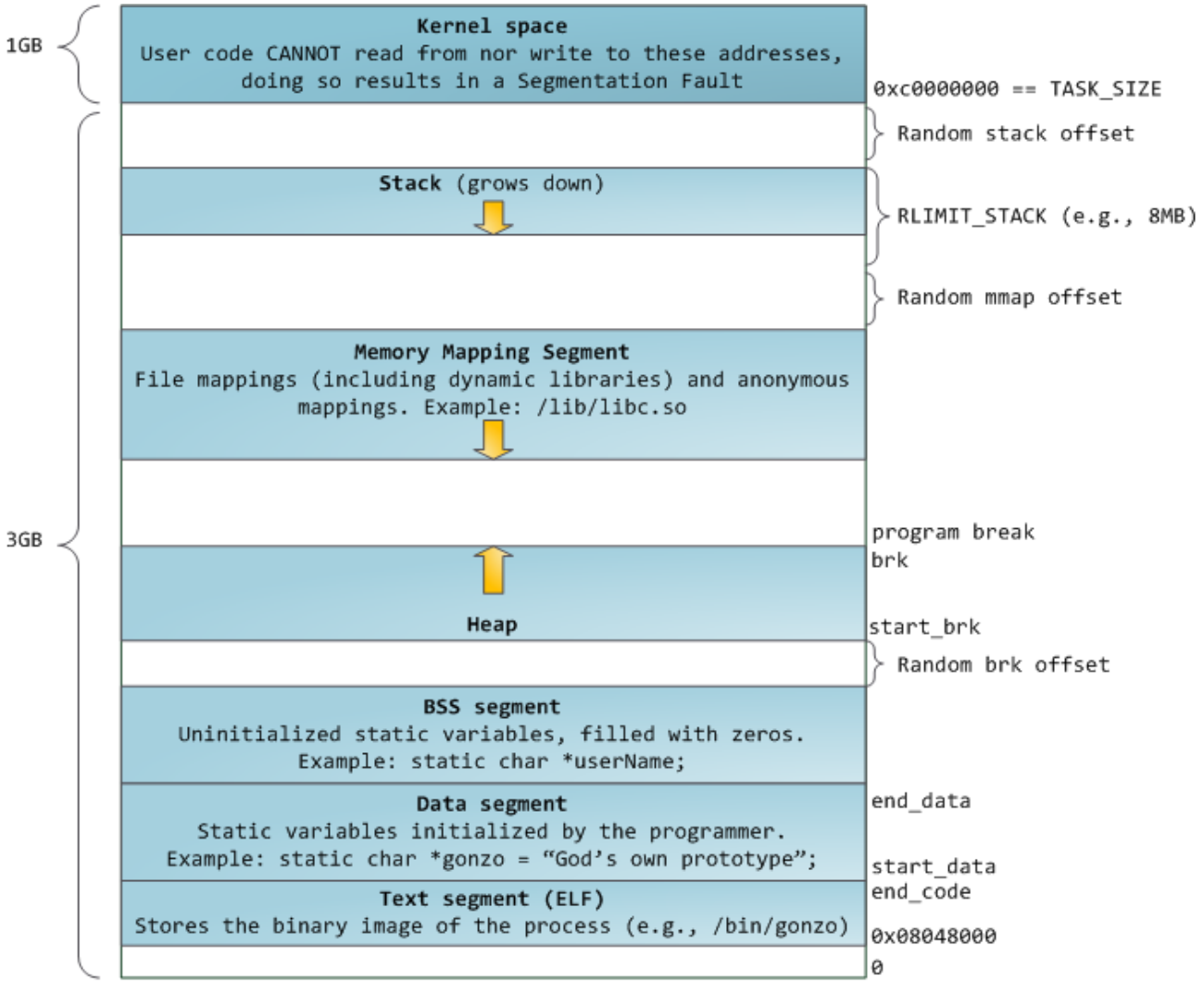
About Me

- Senior studying MIS, CS
- CTF, wargame enthusiast
- Like to read…a lot
- Intern @ iSEC Partners in SF
- http://conceptofproof.wordpress.com

Overview

- x86 ASM
- Vuln/exploitation trends
- Stack overflows
- Lab 1
- More stack overflows
- Lab 2
- Heap overflows

Virtual Address Space



Src: http://duartes.org/gustavo/blog/

Assembly 101

- Registers
  - EAX - return value
  - EBX
  - ECX
  - EDX
  - ESI
  - EDI
  - ESP - points to data at top of stack (stack pointer)
  - EBP - points to base of stack frame (frame pointer)
  - EIP - points to next instruction to execute (instruction pointer)

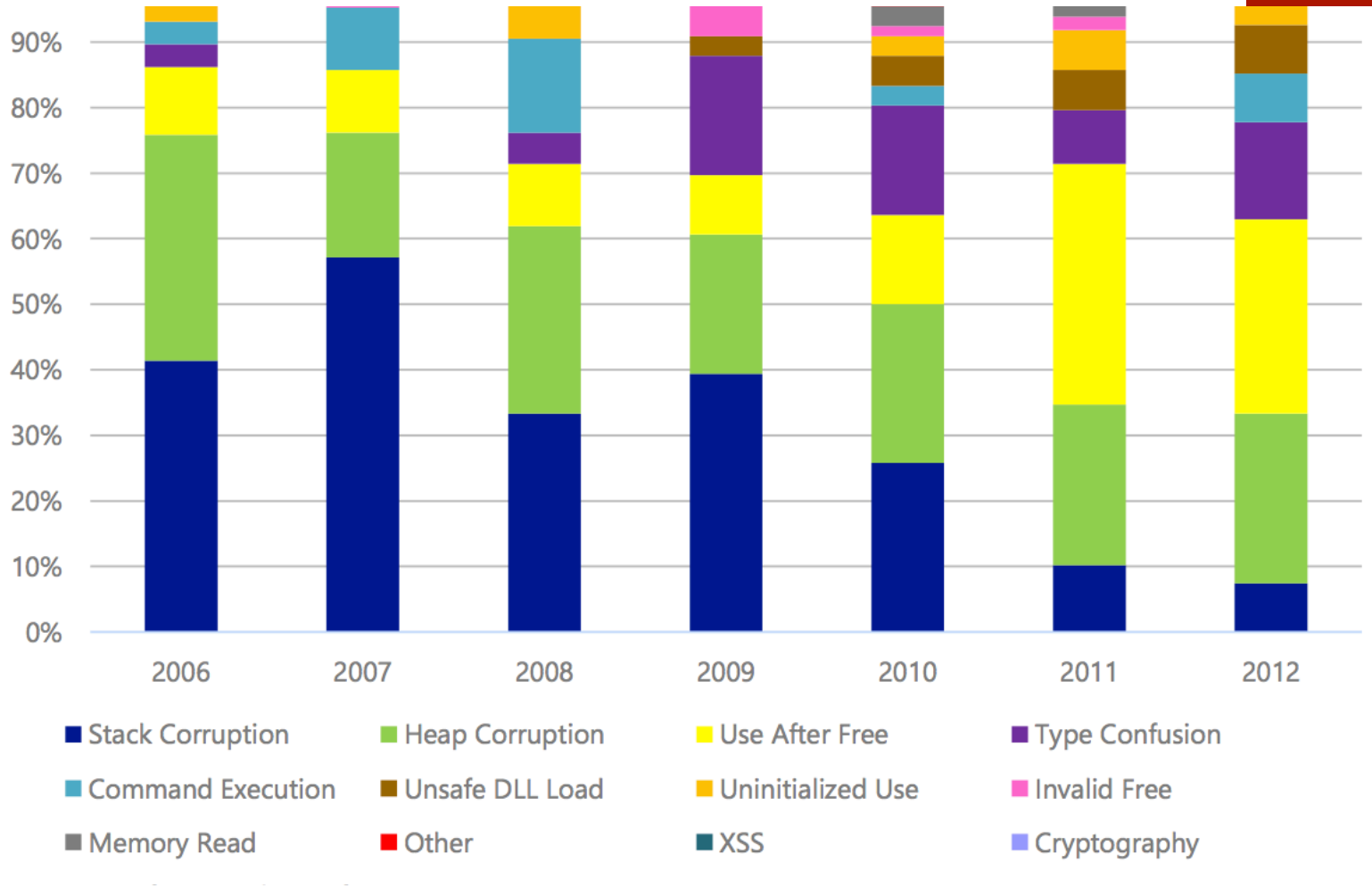
x86 Calling Conventions

- Cdecl
  - Function arguments pushed onto stack from right to left
  - EAX, ECX and EDX are caller-saved
  - Caller cleans up stack
- Stdcall
  - Function arguments also pushed onto stack from right to left
  - Callee cleans up stack

Calling a Subfunction & Returning

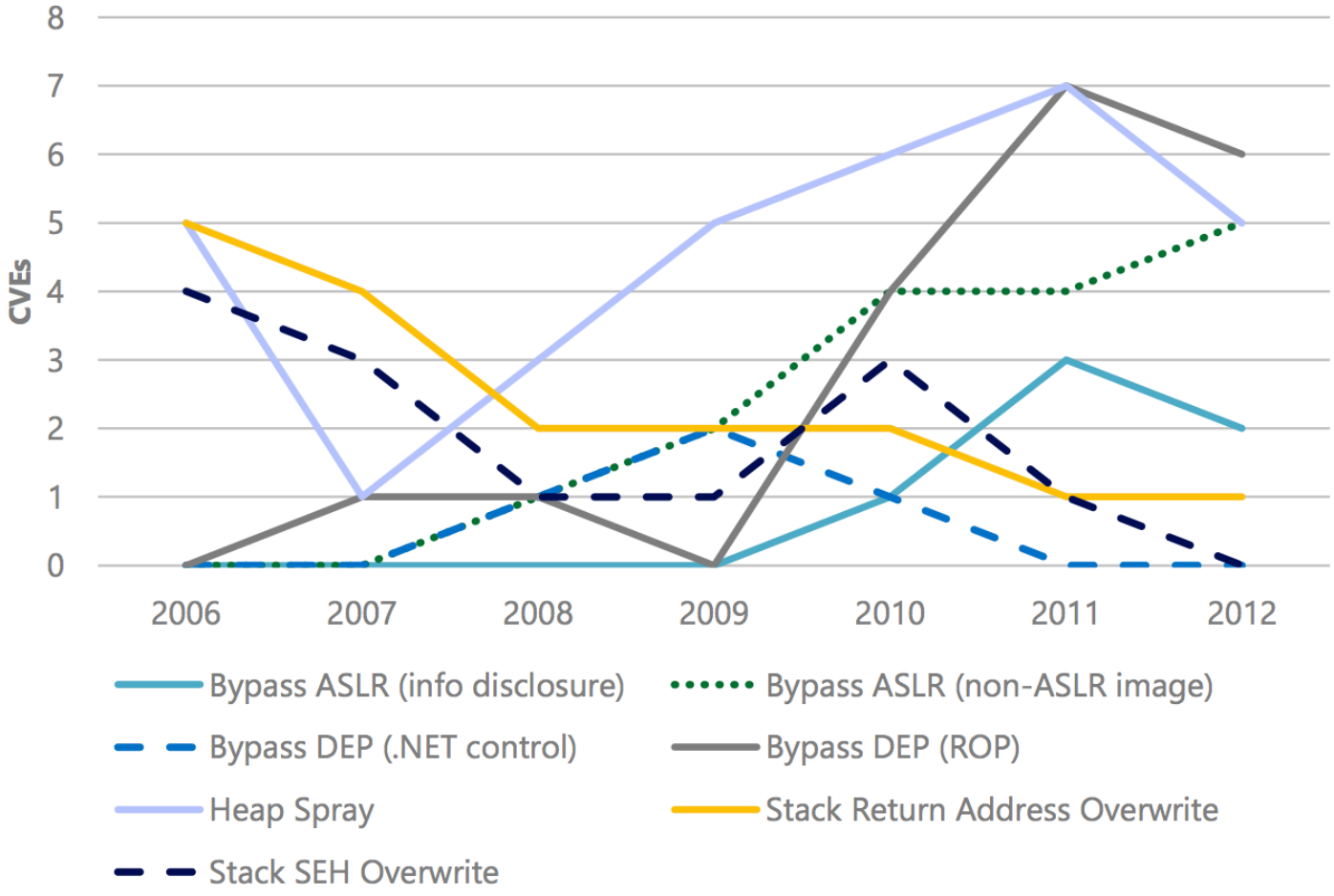
- Arguments pushed onto stack
- EIP pushed onto stack ~ Saved Return Address
- EBP pushed onto stack ~ Saved Frame Pointer
- ESP decremented
- Local variables POP-ed off stack
- Retn

Vulnerability Classes



Source: http://www.microsoft.com/en-us/download/confirmation.aspx?id=39680

Exploitation Trends



Source: http://www.microsoft.com/en-us/download/confirmation.aspx?id=39680

source: http://stackoverflow.com

Exploit Goal?

- Arbitrary Code Execution!

Vulnerable Functions

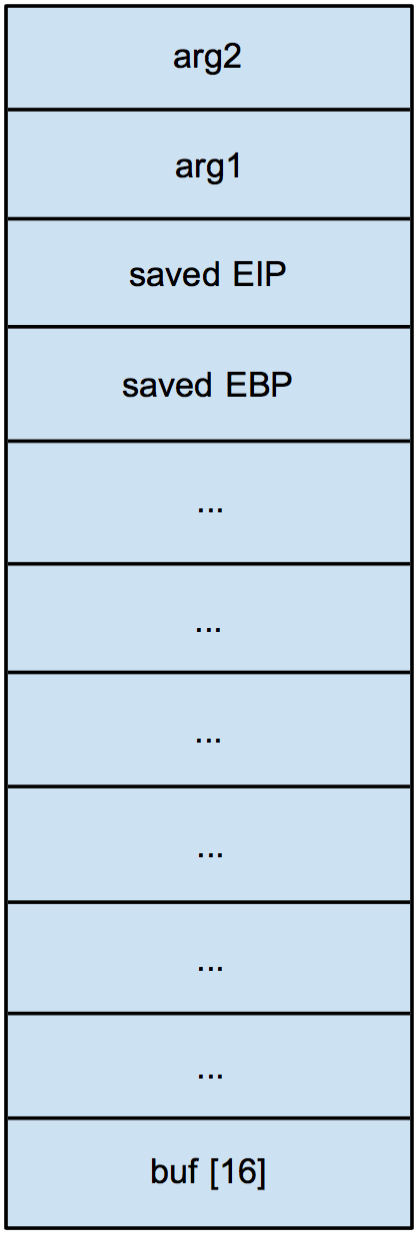
- sprintf()
- gets()
- strcpy()
- strcat()
- strcmp()
- memcpy()

Source: http://natashenka.ca/posters/

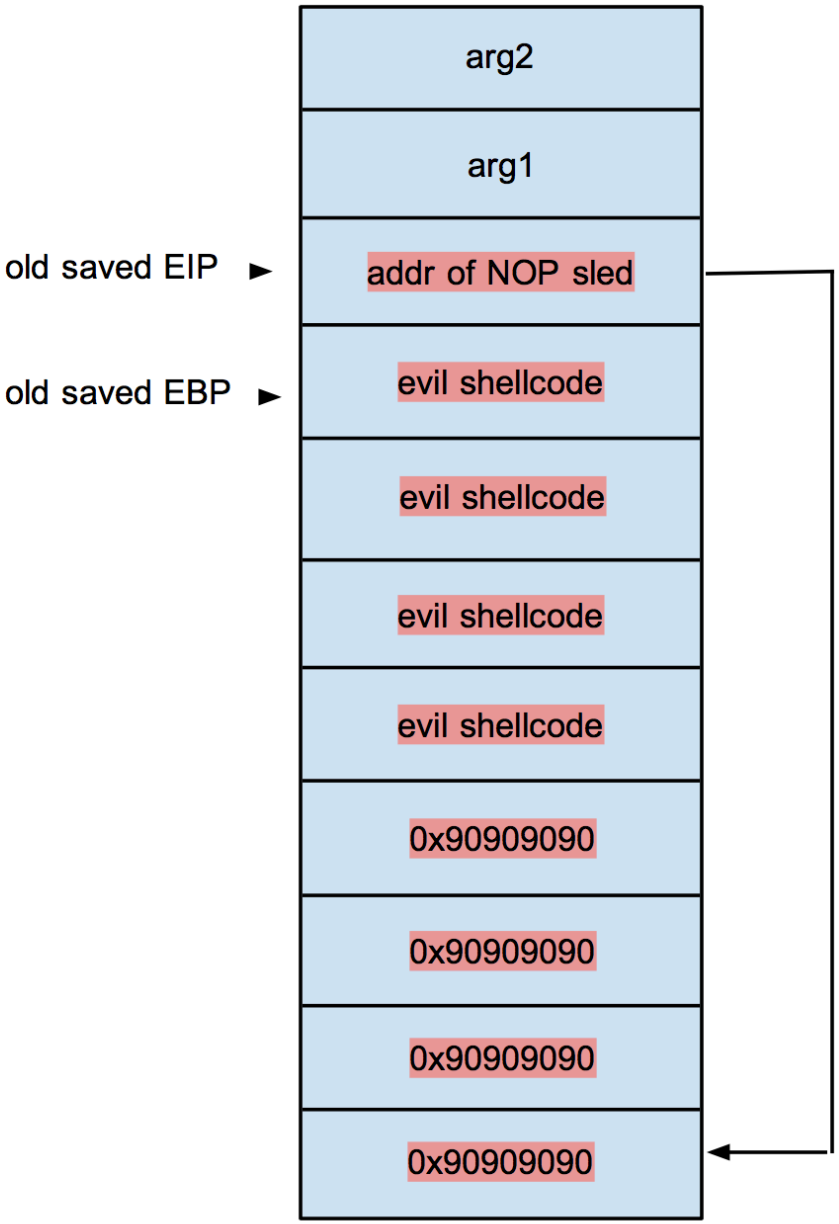
Basic Stack Smashing Algo

- Find vulnerable function
- Find offset of EIP
- Find address of NOP sled, shellcode
- Overwrite EIP with addr of NOP sled, shellcode

Normal Stack

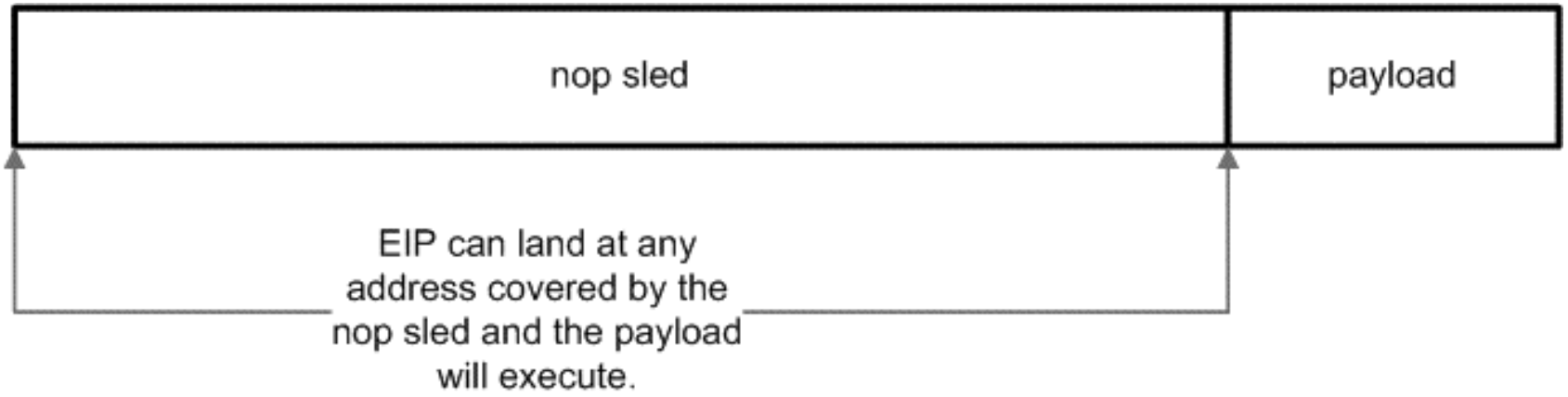


Pwned Stack


arg2
arg1
old saved EIP
addr of NOP sled
old saved EBP
evil shellcode
evil shellcode
evil shellcode
evil shellcode
0x90909090
0x90909090
0x90909090
0x90909090

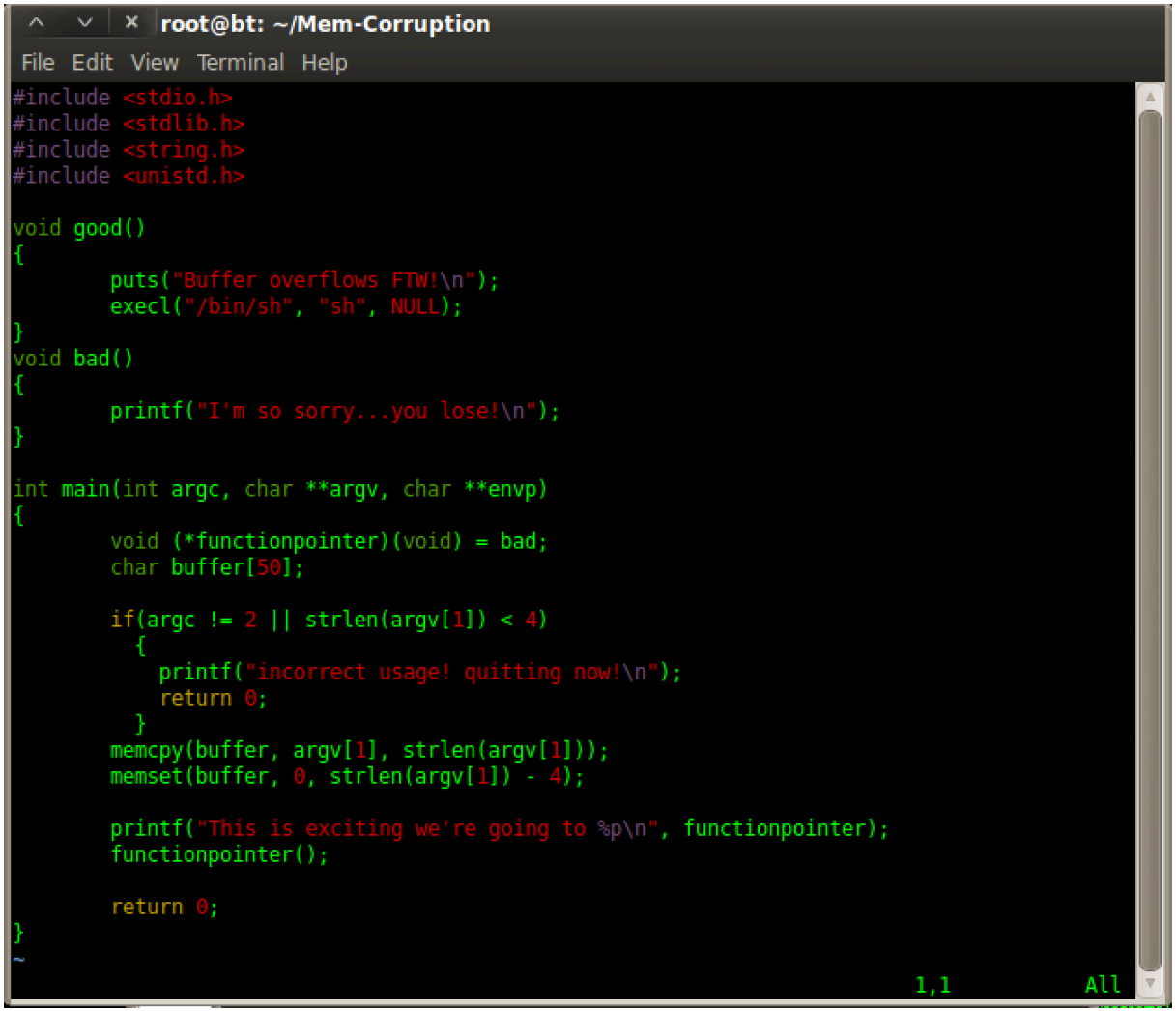
What the heck's a NOP Sled?

- Chain of 0x90 instructions
- Add before shellcode
- Allows for less precise attacks



source: http://www.securityfocus.com/excerpts/9/5

Lab #1 – Vuln1.c


root@bt: ~/Mem-Corruption
File Edit View Terminal Help
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
void good()
{
puts("Buffer overflows FTW!\n");
execl("/bin/sh", "sh", NULL);
}
void bad()
{
printf("I'm so sorry...you lose!\n");
}
int main(int argc, char **argv, char **envp)
{
void (*functionpointer)(void) = bad;
char buffer[50];
if(argc != 2 || strlen(argv[1]) < 4)
{
printf("incorrect usage! quitting now!\n");
return 0;
}
memcpy(buffer, argv[1], strlen(argv[1]));
memset(buffer, 0, strlen(argv[1]) - 4);
printf("This is exciting we're going to %p\n", functionpointer);
functionpointer();
return 0;
}
~
1,1 All

Well that was easy!

Exploit Mitigation

- ASLR
- DEP/NX
- SafeSEH
- SEHOP
- Stack canaries
- RELRO
- Safe Unlinking
- VTGuard
- ...

DEP

- Useless on its own
- Call to API to SetProcessDEPPolicy() or VirtualProtect()
- Marks stack pages or other memory regions non-executable

API / OS	XP SP2	XP SP3	Vista SP0	Vista SP1	Windows 7	Windows 2003 SP1	Windows 2008
VirtualAlloc	yes	yes	yes	yes	yes	yes	yes
HeapCreate	yes	yes	yes	yes	yes	yes	yes
SetProcessDEPPolicy	no (1)	yes	no (1)	yes	no (2)	no (1)	yes
NtSetInformationProcess	yes	yes	yes	no (2)	no (2)	yes	no (2)
VirtualProtect	yes	yes	yes	yes	yes	yes	yes
WriteProcessMemory	yes	yes	yes	yes	yes	yes	yes

(1) = doesn't exist

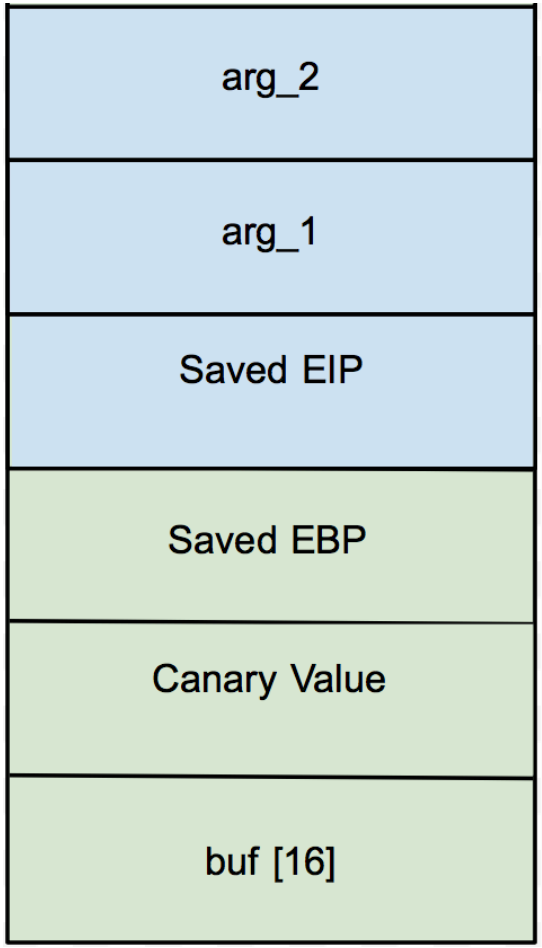
(2) = will fail because of default DEP Policy settings

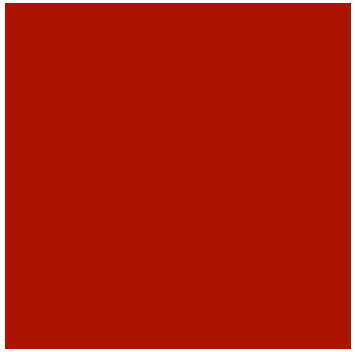
source: https://www.corelan.be/index.php/2010/06/16/exploit-writing-tutorial-part-10-chaining-dep-with-rop-the-rubikstm-cube/

ASLR

- Not really that random
  - Eg 0xb7eXXfd0
- Useless unless used with DEP
- Modules can opt-out
  - Perform register jmp rather than absolute value jmp
    - Eg) jmp ESP

Stack Canaries, /GS

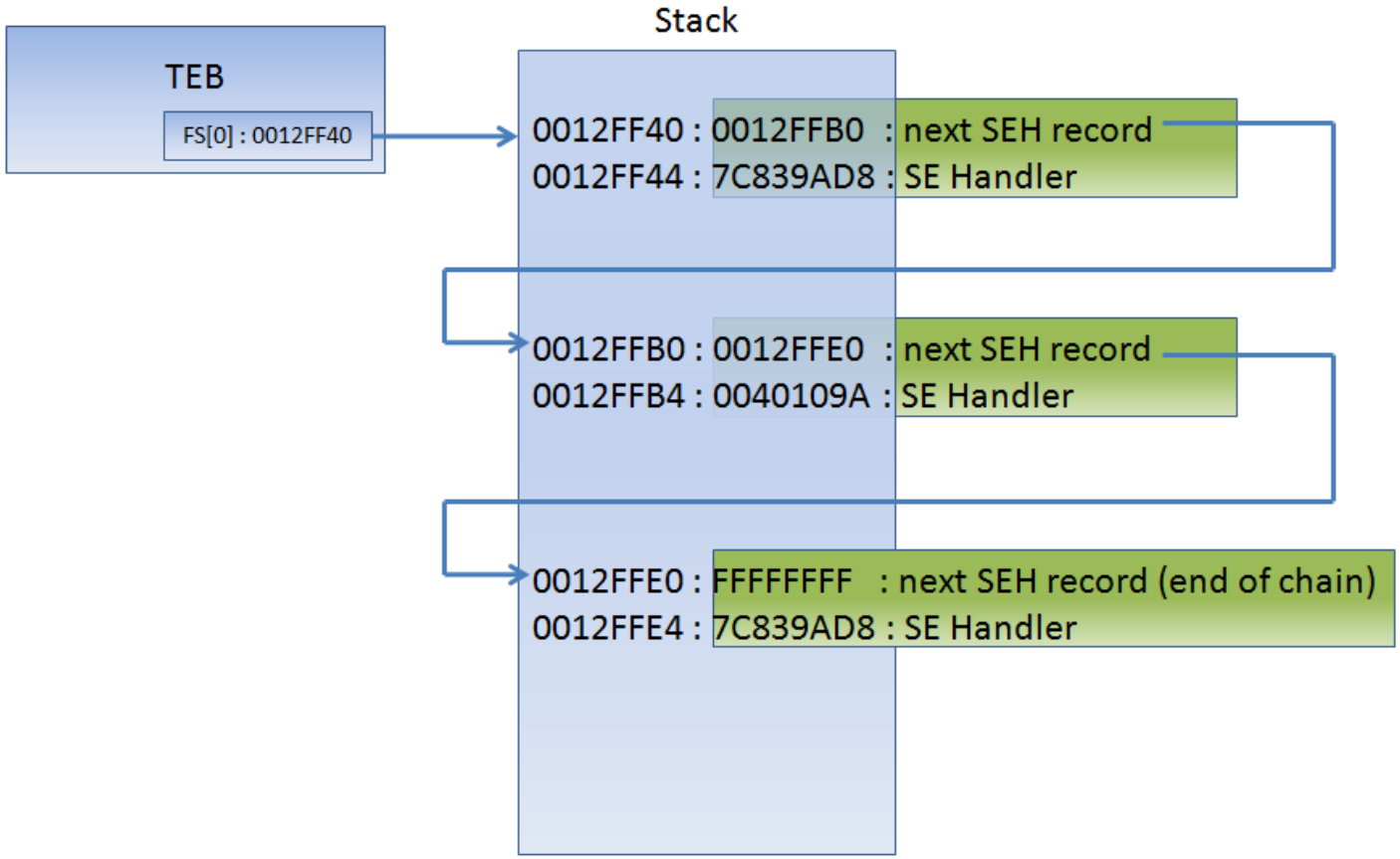




```
int __cdecl attack_mouse(int clientfd)
{
  int v1; // edx@3
  int v2; // ecx@3
  int len_of_str; // eax@4
  size_t v4; // ST5C_4@4
  int result; // eax@5
  int fd; // [sp+18h] [bp-20h]@1
  int buf; // [sp+22h] [bp-16h]@1
  int v8; // [sp+26h] [bp-12h]@1
  __int16 v9; // [sp+2Ah] [bp-Eh]@1
  int canary; // [sp+2Ch] [bp-Ch]@1

  canary = *MK_FP(__GS__, 20);
  buf = 0;
  v8 = 0;
  v9 = 0;
  fd = open("mouse.txt", 0);
  if ( fd < 0 )
    exit_msg((int)"open() error");
  write(clientfd, "Are you sure? (y/n) ", 0x14u);// 0x14 = 20
  read(clientfd, &buf, 0x6Eu);                  // 0x6E = 110
  if ( (_BYTE)buf == 'y' )
  {
    len_of_str = sprintf(s, "You choose '%s'!\n", &buf);
    write(clientfd, s, len_of_str);
    v4 = read(fd, s, 0x1388u);                  // 0x1388 = 5000
    write(clientfd, s, v4);
    write(clientfd, "\n\"MOUSE!!!!!!!!! (HP - 25)\"\n", 0x1Cu);// 0x1C = 28
    health -= 25;
  }
  result = *MK_FP(__GS__, 20) ^ canary;
  if ( *MK_FP(__GS__, 20) != canary )
    __stack_chk_fail(v2, v1);
  return result;
}
```

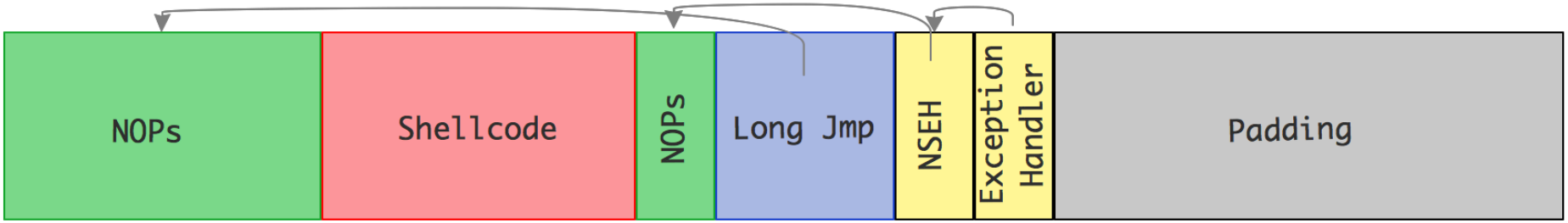
Structured Exception Handlers



Source: https://www.corelan.be/index.php/2009/07/25/writing-buffer-overflow-exploits-a-quick-and-basic-tutorial-part-3-seh/

SEH Overflow

- Abusing Windows SEH Chain
- Overwrite pointer to NSEH record with JMP instruction
- Overwrite pointer to handler code with addr of "POP POP RET" instruction in non-ASLR enabled module/dll



SafeSEH

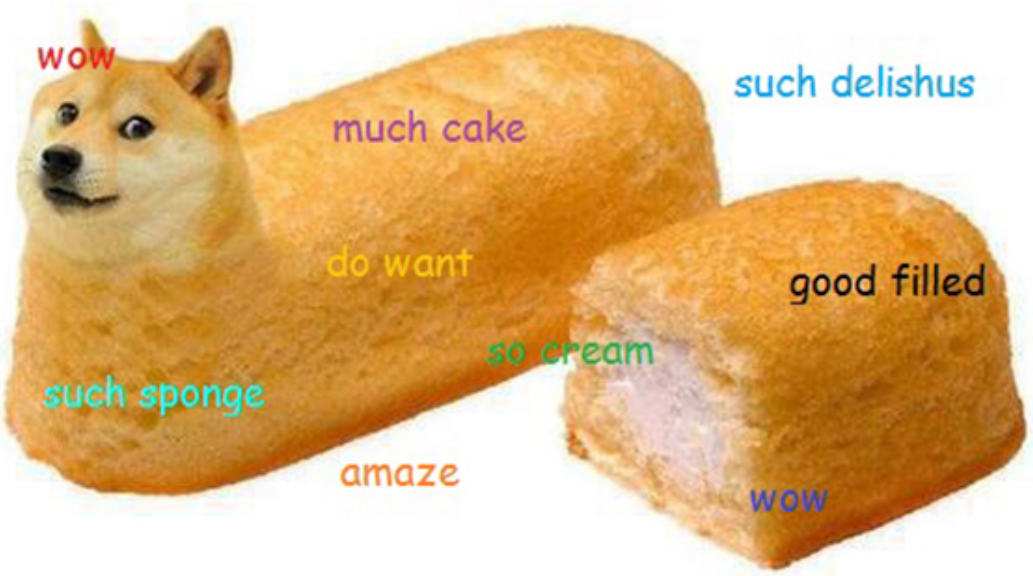
- Software DEP
- List of valid SEH records stored in PE header of loaded modules at compile-time
- Helps to mitigate SEH overflow attacks (kinda)
- Bypass
  - Not all modules compiled with SafeSEH

Lab 2: Ropasaurus Rex

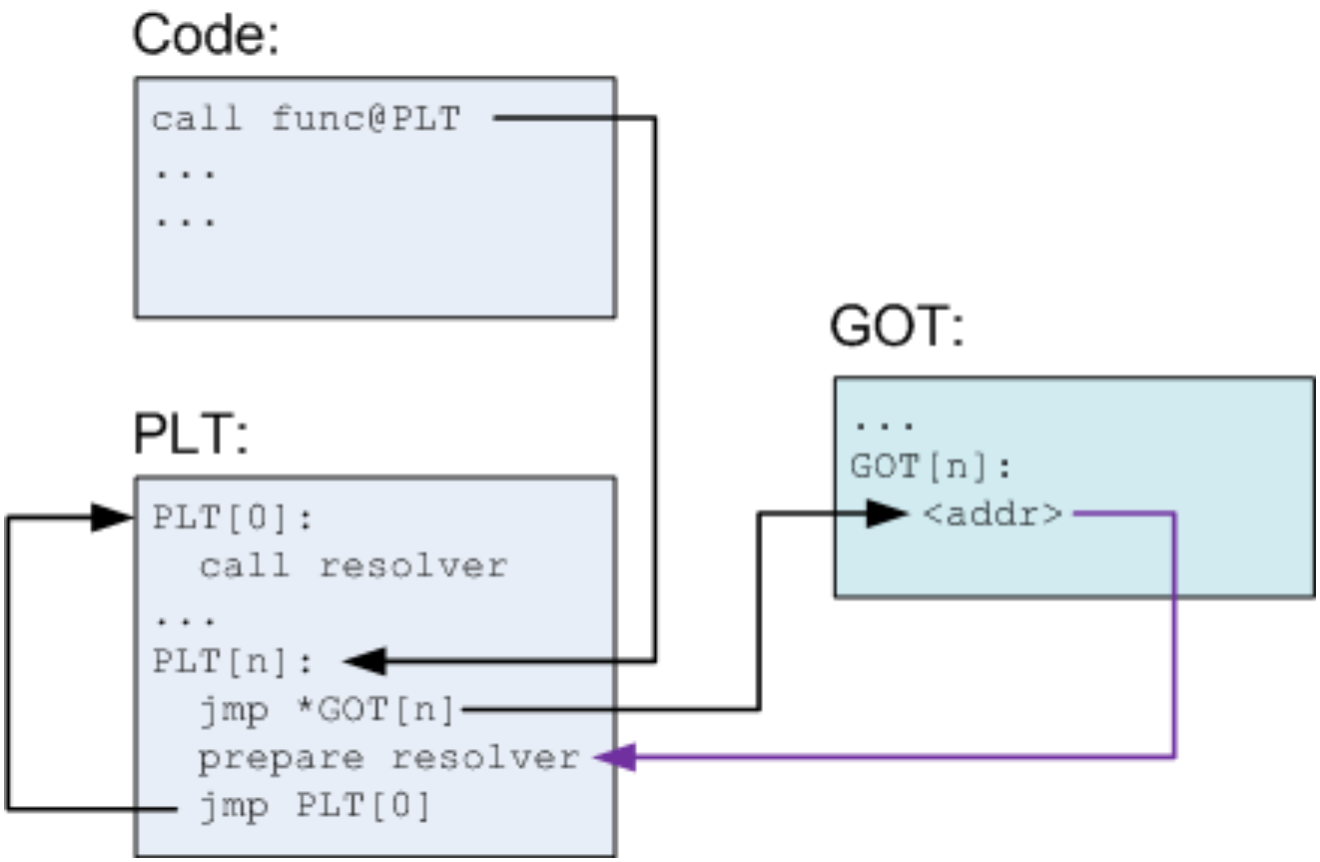
https://blog.skullsecurity.org/2013/ropasaurusrex-a-primer-on-return-oriented-programming

But first, let's take a look @

- GOT overwrite
- Ret-2-Libc
- ROP



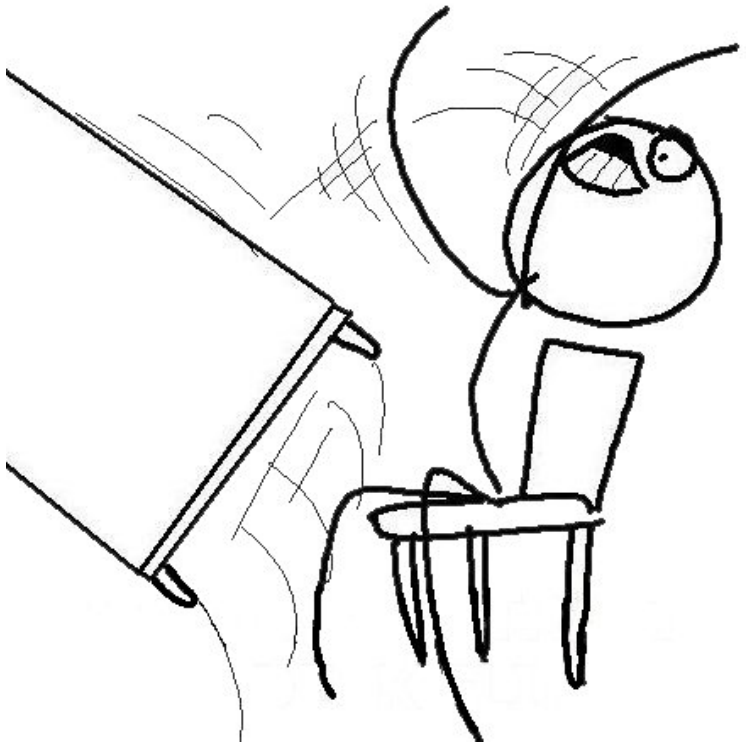
GOT Overwrite



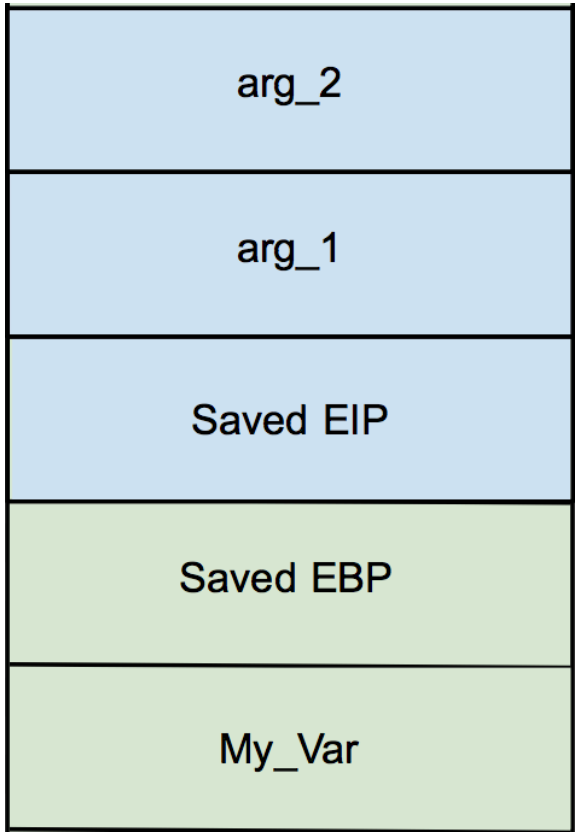
Source: http://eli.thegreenplace.net/2011/11/03/position-independent-code-pic-in-shared-libraries/

RELRO

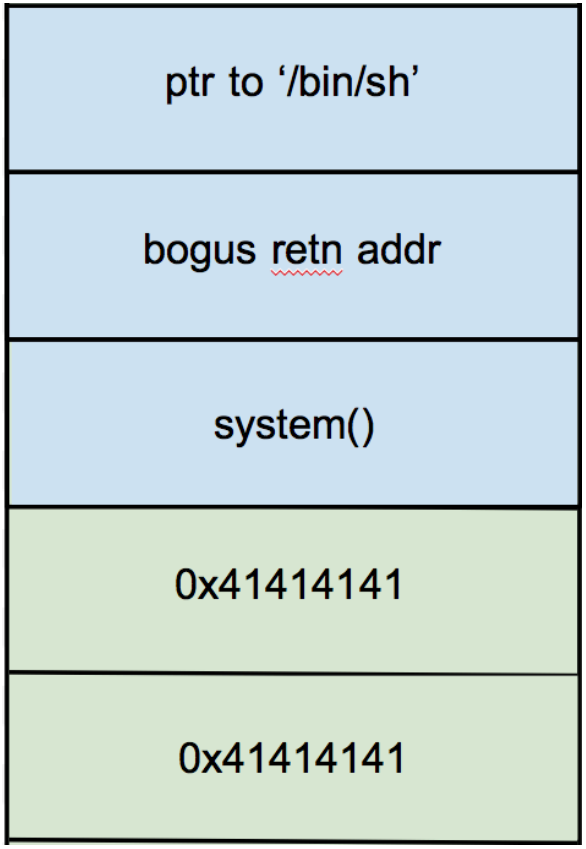
- RELocation Read Only
- Prevents GOT overwrites
- Marks GOT as Read Only



Return-to-libc



normal

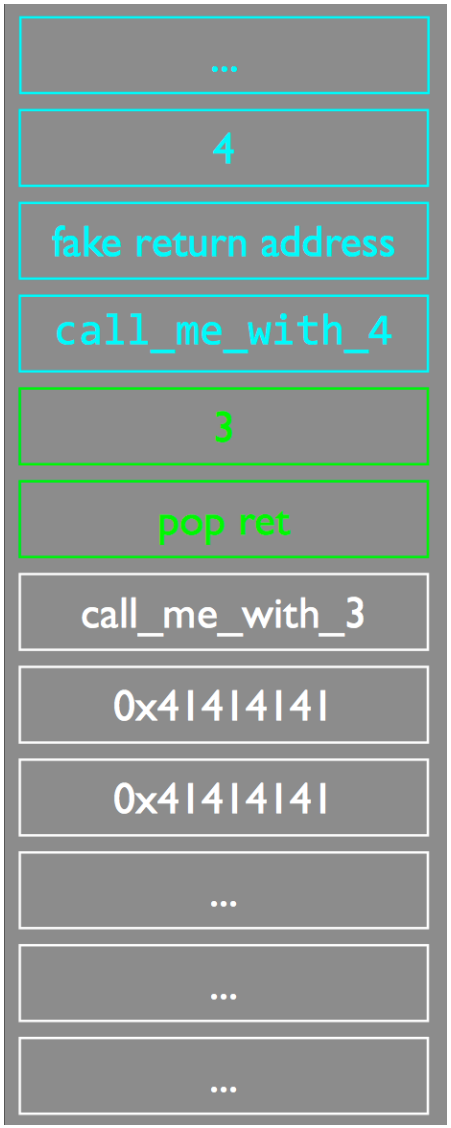


system("/bin/sh")

ROP

- Extension of Return-to-libc
- Used to bypass DEP
- Pop-pop-pop-ret
- Can chain together “gadgets” that end in ret
- Call VirtualProtect() or perform any other big picture operation

ROP Chaining



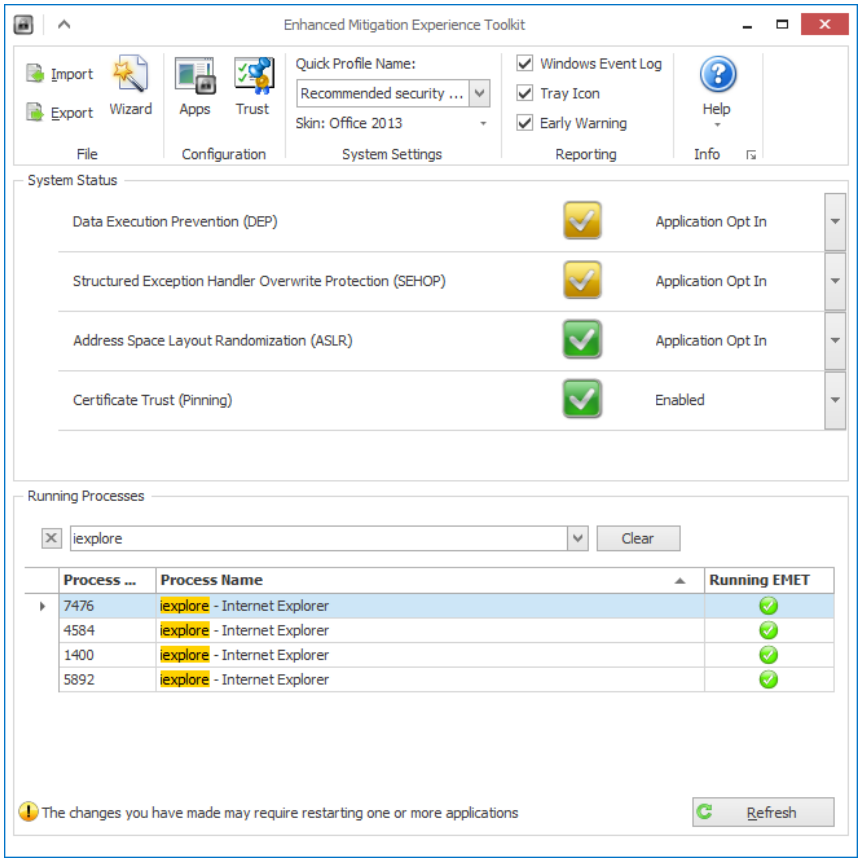
Source: http://codearcana.com/posts/2013/04/28/picoctf-videos.html

Actual Lab 2: Ropasaurus Rex

https://blog.skullsecurity.org/2013/ropasaurusrex-a-primer-on-return-oriented-programming

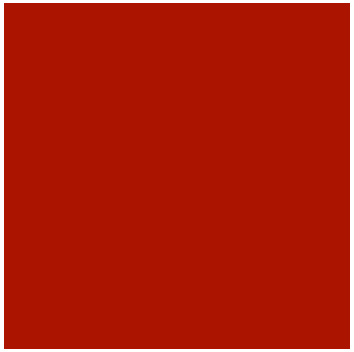
EMET

- Behavior based ~ prevents 0-days!
- DLL dynamically linked at runtime
- ROP mitigations
- Certificate pinning
- No recompilation necessary!



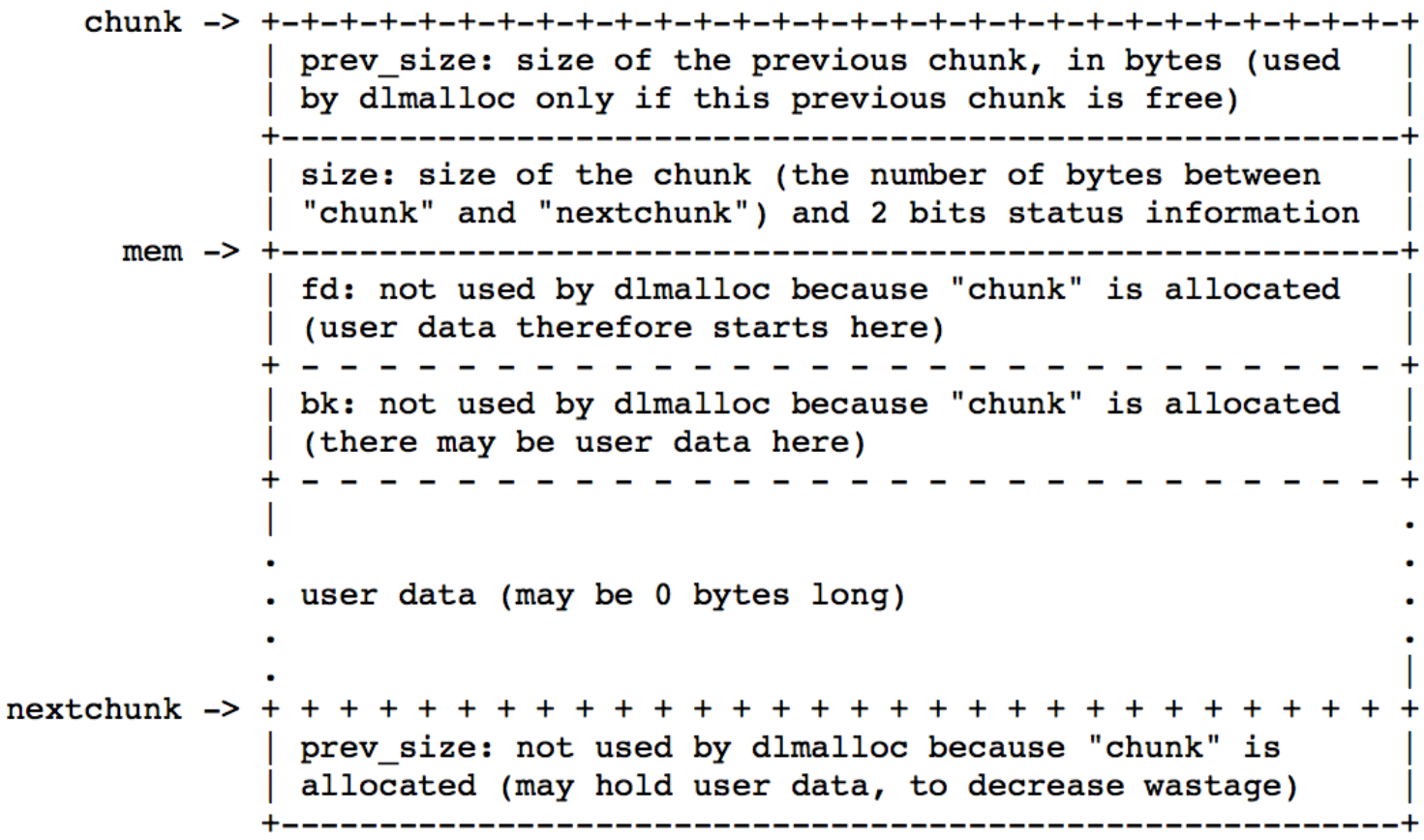
Pwning the HEAP

Doug Lea's Malloc



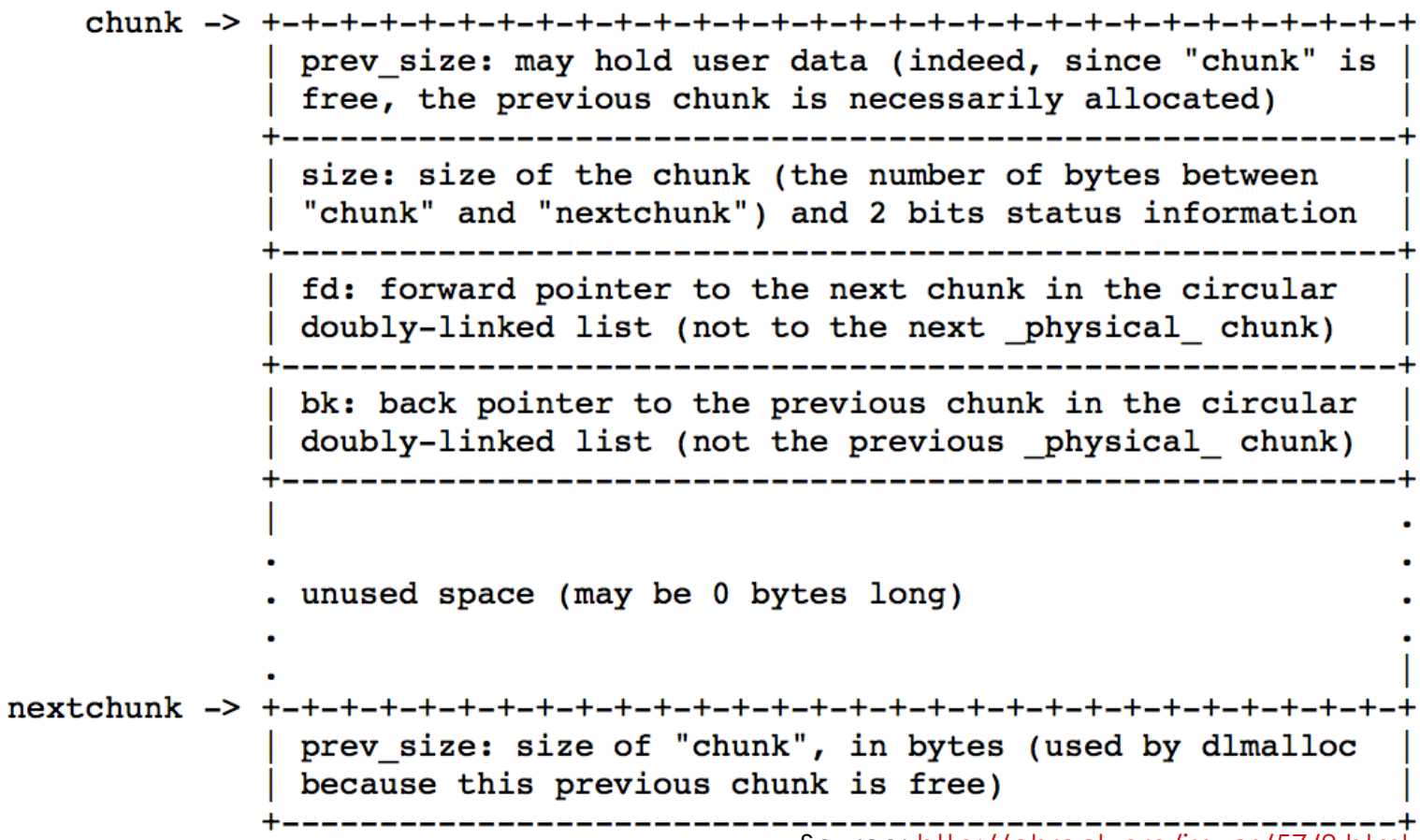
- Sbrk() and brk() calls to allocate more memory
- Heap chunks
- Reclaims free chunks to support defragmentation
- Free chunks stored in double linked list
- Free chunk coalescing

Allocated Heap Chunk



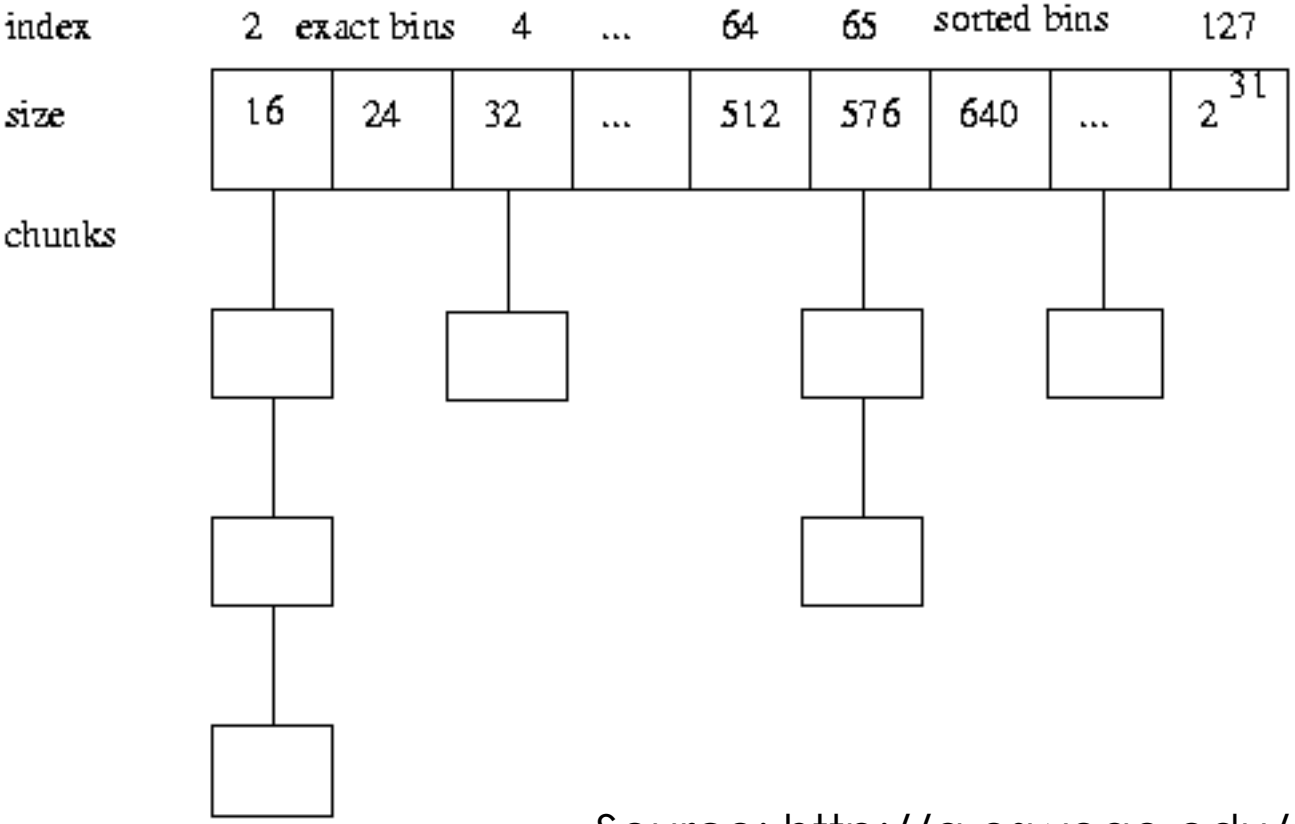
Source: http://phrack.org/issues/57/8.html

Free Heap Chunk



Source: http://phrack.org/issues/57/8.html

Binning



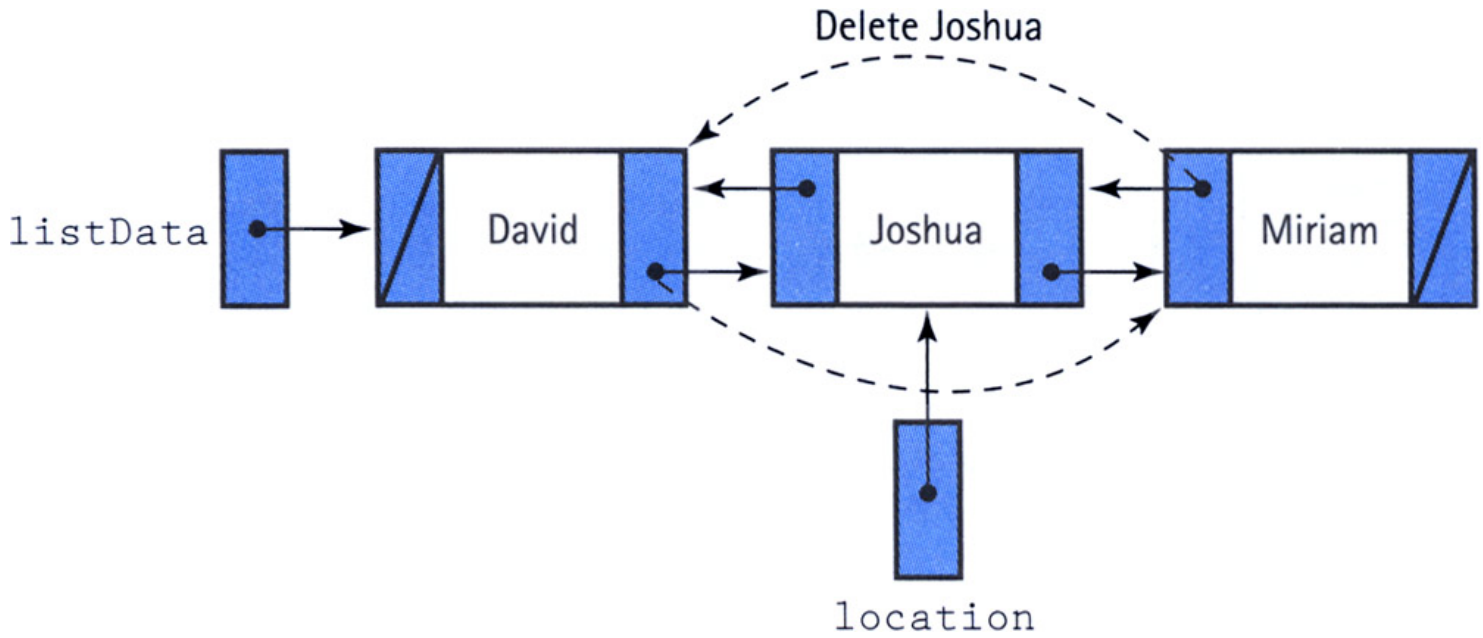
Source: http://g.oswego.edu/dl/html/malloc.html

Corrupting the Heap

- DWORD Overwrite
- Corrupting Metadata
- Abusing Free() function
- http://phrack.org/issues/57/8.html

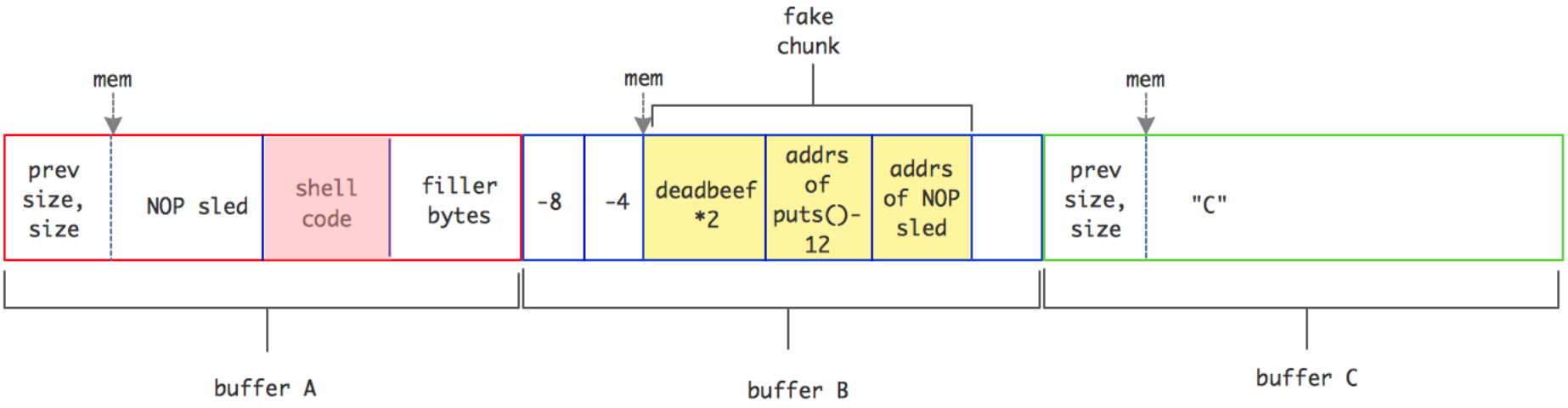
Freeing memory

- Chunk->prev->next = Chunk->next
- Chunk->next->prev = Chunk->prev

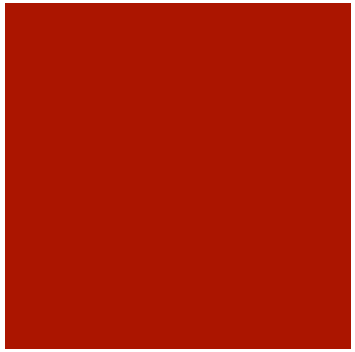


source: http://younginc.site11.com/source/5895/fos0052.html

Protostar Heap3

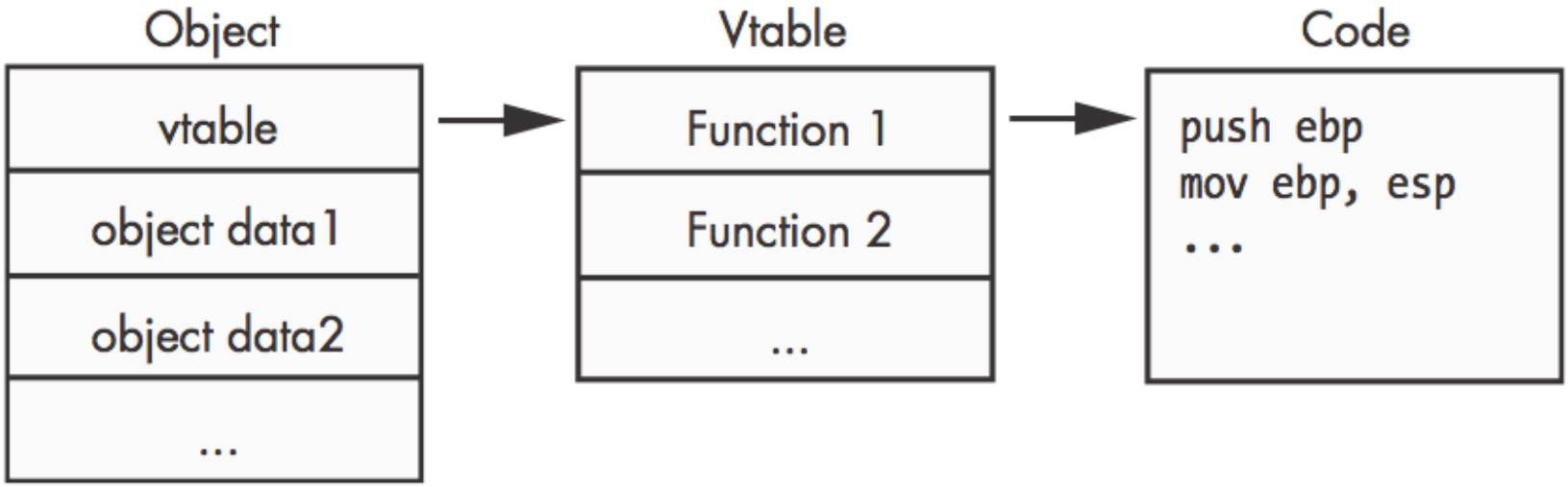


Use-After-Free/Heap Spray



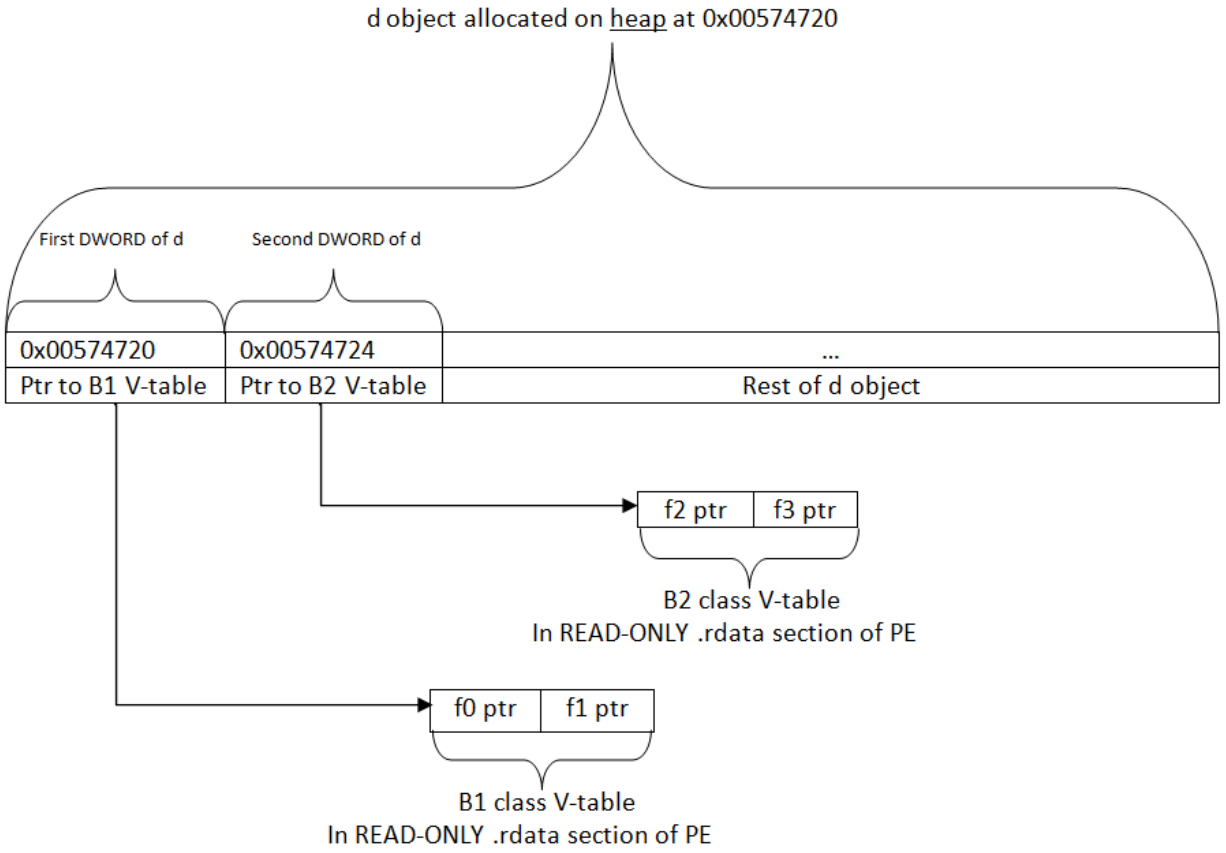
- Use After Free
  - Abusing C++ V-Tables, dangling pointers
- Heap spraying
  - Payload delivery method
  - Popular with web browser exploits

Virtual Functions



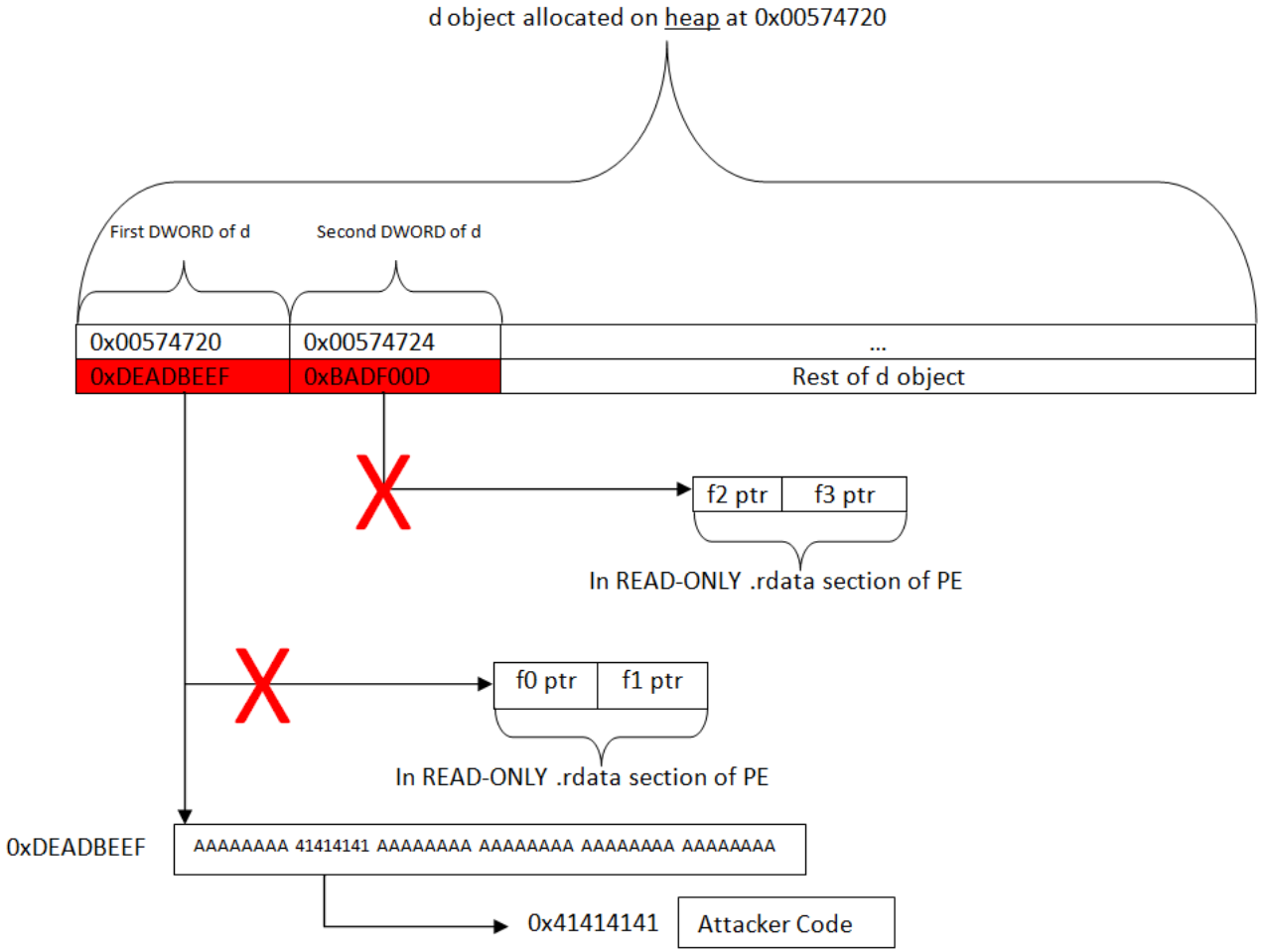
Source: Practical Malware Analysis by Mike Sikorski, Andrew Hoing

Expected Behavior of Virtual Functions



Source: http://neilscomputerblog.blogspot.com

Unexpected Behavior of Abused Virtual Function



Source: http://neilscomputerblog.blogspot.com

Heap Feng Shui

- Alex Sotirov
- HeapLib
- Increases precision
- Accurate, deterministic way of placing shellcode on the heap
- http://www.phreedom.org/research/heap-feng-shui/heap-feng-shui.html


That's all Folks!