# Code Vulnerabilities & Attacks

Marcus Botacin

# Topics

# Topics

# Producer-Consumer Problem

## Definition

- Circular Buffer
- How to guarantee order ?

## Code

```
producer()
while(1)
   if buffer_size == FULL
      sleep
consumer()
while(1)
   if buffer_size == EMPTY
      sleep
```

# Race Condition

## Definition

"Condição em que o resultado do processo é dependente da sequência ou sincronia de outros eventos".

## Problem

Leva a resultados incorretos.

# Topics

# Password Check

## Code

```
Checker()
   if allowed
      grant_access
Creator()
   create_user
   allowed=True
```

# TOCTOU

## Definition

"Um bug causado por uma *race condition*".

## Severity

"De computações incorretas a acessos indevidos".

# Topics

# Background

## Memory Management

- Page faults retries
- Dirty bits
- Copy-on-write

## Linux

- PTRACE POKE_DATA[a]
- mdavise syscall[b]

---

[a] man ptrace
[b] man madvise

# Dirty Cow

## Part I - Page Fault

```
handle_pte_fault
 do_fault <- pte is not present
  do_cow_fault <- FAULT_FLAG_WRITE
```

## Part II - Write Fault

```
do_wp_page
 PageAnon() <- this is CoWed page already
  reuse_swap_page <- page is exclusively ours
   wp_page_reuse
    maybe_mkwrite <- dirty but RO again
```

[0] https://dirtycow.ninja/

# Dirty Cow

## Part III - Read Fault

```
cond_resched -> different thread will now
unmap via madvise
 follow_page_mask  !pte_present && pte_none
  faultin_page
   handle_mm_fault
    __handle_mm_fault
     handle_pte_fault
      do_fault <- pte is not present
       do_read_fault <- this is a read
fault and we will get pagecache
```

# Topics

# ABIs and Calling conventions

- How parameters are passed.
- The order they are passed.
- Where they are passed.

# Calling conventions

- **cstd**: Arguments on the `stack`, on the reverse order.
- **x64**: Arguments on `registers`.
- **syscalls**: Syscall number on `eax`.

# Concept Definitions

## ABI

"Uma definição da interface entre módulos de *software*, tais como entre códigos e o sistema operacional. Uma ABI especifica uma *calling convention*."

## Calling Convention

"Uma definição de como trechos de código são chamados, incluindo a ordem, localização, e a restauração dos parâmetros."

Back to O.S. (again)

# Stack frame

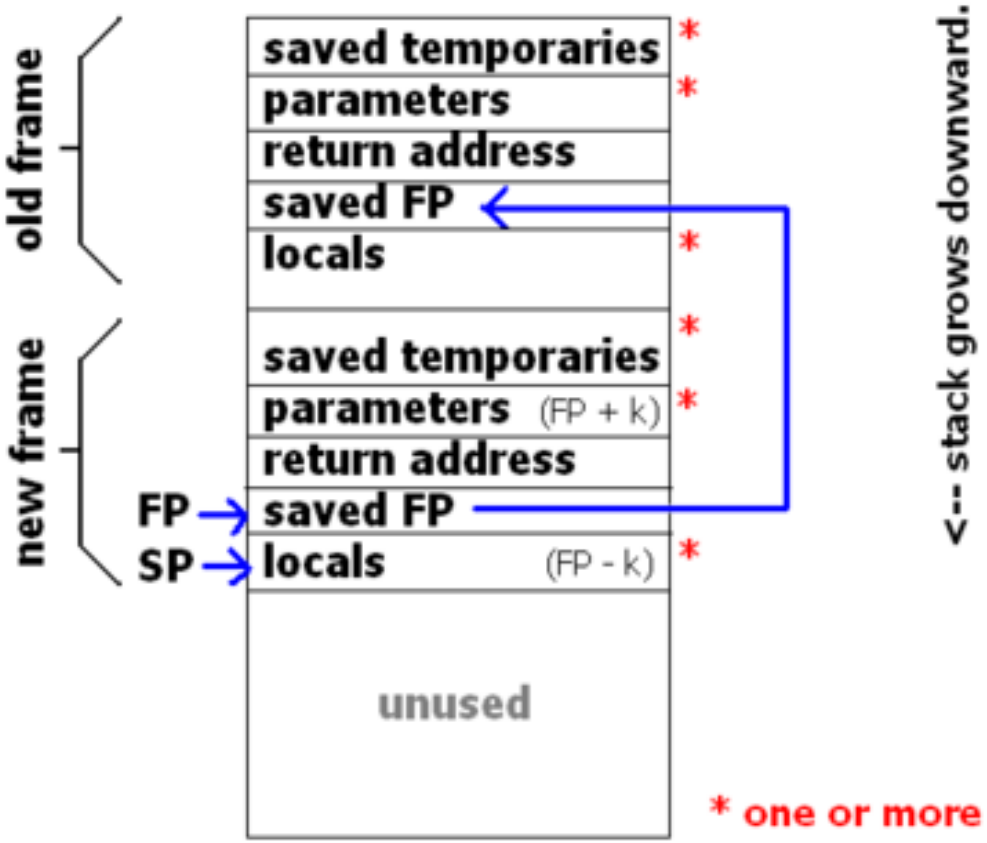


Figure: Stack Frame

[0] https://tinyurl.com/hhq934r

# Stack Frame

**Definition**

"Uma definição de contexto de função de forma temporária."

# Topics

# For fun and for profit

.oO Phrack 49 Oo.

Volume Seven, Issue Forty-Nine

File 14 of 16

BugTraq, r00t, and Underground.Org
bring you

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Smashing The Stack For Fun And Profit
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

by Aleph One
aleph1@underground.org

Figure: Aleph One's article.

[0] http://phrack.org/issues/49/14.html

# A typical payload

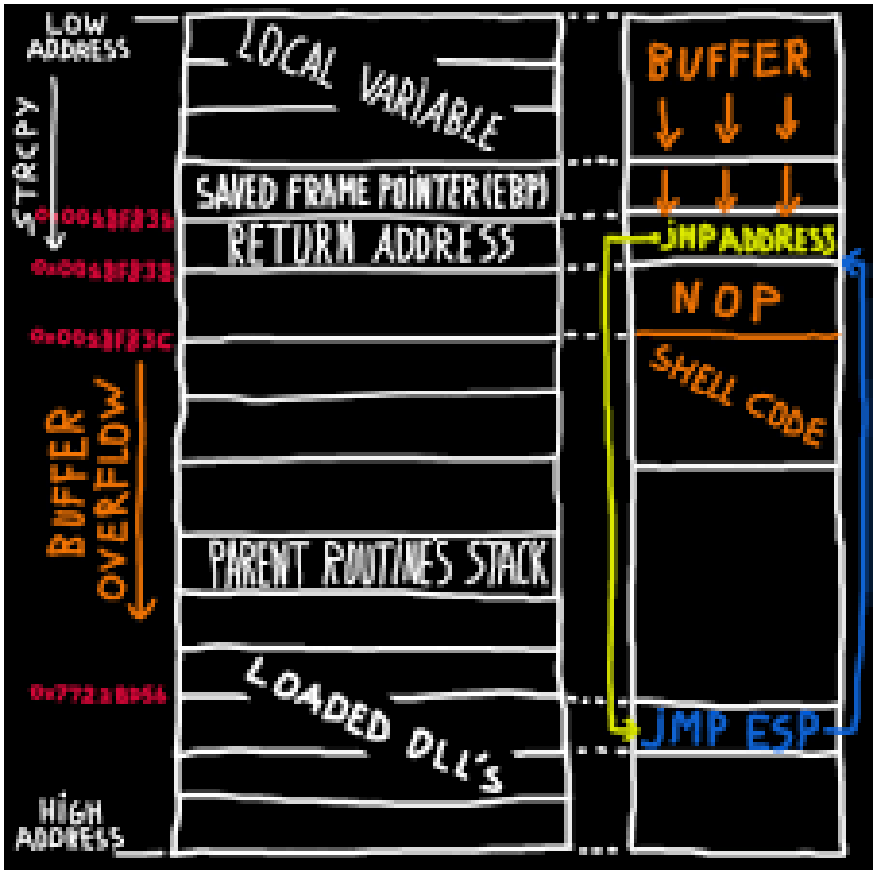


Figure: Buffer overflow payload structure

[0] https://tinyurl.com/y9dornpu

# Vulnerable code

Listing 1: Vulnerable code

```
int main(int argc, char *argv[])
{
  char str[32];
  strcpy(str,argv[1]);
  return 0;
}
```

# Shellcodes

Twitter Profile
GitHub Profile
Google+ Profile
Linkedin Profile
RSS feeds
Email

## Shellcodes database for study cases

- Windows - sp3 (Tr) calc.exe Shellcode 53 bytes *by ZoRLu*
- Windows - sp3 (Tr) cmd.exe Shellcode - 42 bytes *by ZoRLu*
- Windows - sp3 (Tr) cmd.exe Shellcode 52 bytes *by ZoRLu*
- Windows - Xp Pro SP3 Fr (calc.exe) - 31 Bytes *by agix*
- Windows - XP PRO SP3 - Full ROP calc shellcode *by b33f*
- Windows - xp pro sp3 (calc) - 57 bytes *by cr4wl3r*
- Windows - win32/xp pro sp3 MessageBox shellcode - 11 bytes *by d3c0der*
- Windows - download & exec shellcode - 226 bytes+ *by darkeagle*
- Windows - Shellcode Checksum Routine *by dijital1*
- Windows - IsDebuggerPresent ShellCode (NT/XP) - 39 bytes *by ex-pb*
- Windows - PEB method (9x/NT/2k/XP) - 29 bytes *by loco*
- Windows - connectback, receive, save and execute shellcode *by loco*
- Windows - Bind Shell (NT/XP/2000/2003) - 356 bytes *by metasploit*
- Windows - Create Admin User Account (NT/XP/2000) - 304 bytes *by metasploit*
- Windows - Vampiric Import Reverse Connect - 179 bytes *by metasploit*
- Windows - PEB method (9x/NT/2k/XP) *by oc192*
- Windows - eggsearch shellcode - 33 bytes *by oxff*
- Windows - XP-sp1 portshell on port 58821 - 116 bytes *by silicon*
- Windows - XP SP3 addFirewallRule *by sinn3r*

Figure: Shellcode Repository.

[0] `http://shell-storm.org/shellcode/`

# No Executable pages

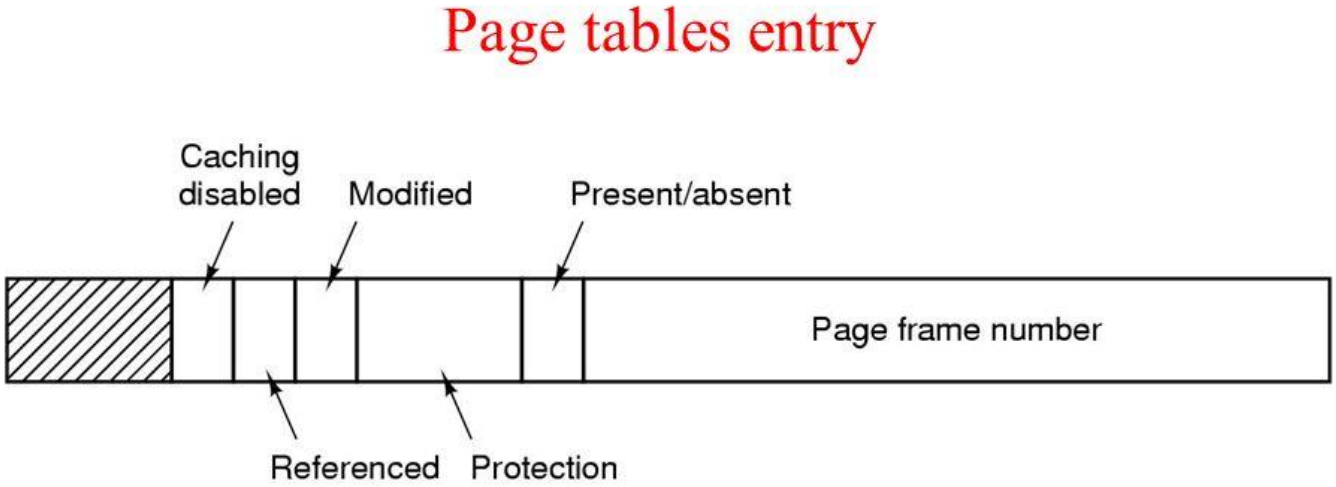


Figure: Page protection bits

[0] `https://tinyurl.com/y8wodhn2`

# Still overflowing

Listing 2: variable overflow

```
int main(int argc, char *argv[])
{
        int a = 10;
        char str[32];
        scanf("%s",str):
        printf("%d\n",a);
        return 0;
}
```

# Concept Definitions

## Buffer

"Região **contígua** de memória, de tamanho **limitado**, utilizada para armazenamento **temporário**."

## Buffer Overflow

"Exceder a capacidade de armazenamento de um *buffer*."

## Buffer Overflow Attack

"Utilizar-se de um *buffer overflow* para alterar **intencionalmente** o estado de um programa."

# Concept Definitions

## Program Hijacking

- **Variable Hijacking**: “Alterar uma variável que controla um estado”.
- **Instruction Hijacking**: “Inserção de instruções para alterar o estado”.
- **Control Flow Hijacking**: ”Uso das técnicas anteriores para alterar o estado”.

# Concept Definitions

## Payload

"Conteúdo de preenchimento do *buffer* em um ataque do tipo *overflow*"

## Shellcode

"Conjunto de instruções que compõem um *payload*"

## NOP

"Uma operação que não altera estados"

## NOP Sled

"Sequência de NOPs que compõem um *payload*"

# Buffer Overflow Classification

## Types

- Stack-based.
- Heap-based.

# Topics

# Back to O.S (again!!)

## Libs and permissions

```
marcus@malware-lab:~/Documentos/aula-ROP$ cat /proc/self/maps
00400000-0040c000 r-xp 00000000 08:05 11022740                           /bin/cat
0060b000-0060c000 r--p 0000b000 08:05 11022740                           /bin/cat
0060c000-0060d000 rw-p 0000c000 08:05 11022740                           /bin/cat
00d56000-00d77000 rw-p 00000000 00:00 0                                  [heap]
7f3411e47000-7f3412121000 r--p 00000000 08:05 9052298                    /usr/lib/locale/locale-archive
7f3412121000-7f34122e1000 r-xp 00000000 08:05 20189578                   /lib/x86_64-linux-gnu/libc-2.23.so
7f34122e1000-7f34124e1000 ---p 001c0000 08:05 20189578                   /lib/x86_64-linux-gnu/libc-2.23.so
7f34124e1000-7f34124e5000 r--p 001c0000 08:05 20189578                   /lib/x86_64-linux-gnu/libc-2.23.so
7f34124e5000-7f34124e7000 rw-p 001c4000 08:05 20189578                   /lib/x86_64-linux-gnu/libc-2.23.so
7f34124e7000-7f34124eb000 rw-p 00000000 00:00 0
7f34124eb000-7f3412511000 r-xp 00000000 08:05 20189562                   /lib/x86_64-linux-gnu/ld-2.23.so
7f34126ca000-7f34126ef000 rw-p 00000000 00:00 0
7f3412710000-7f3412711000 r--p 00025000 08:05 20189562                   /lib/x86_64-linux-gnu/ld-2.23.so
7f3412711000-7f3412712000 rw-p 00026000 08:05 20189562                   /lib/x86_64-linux-gnu/ld-2.23.so
7f3412712000-7f3412713000 rw-p 00000000 00:00 0
7ffea5db9000-7ffea5dda000 rw-p 00000000 00:00 0                          [stack]
```

Figure: Memory mapping and protection

# The attack

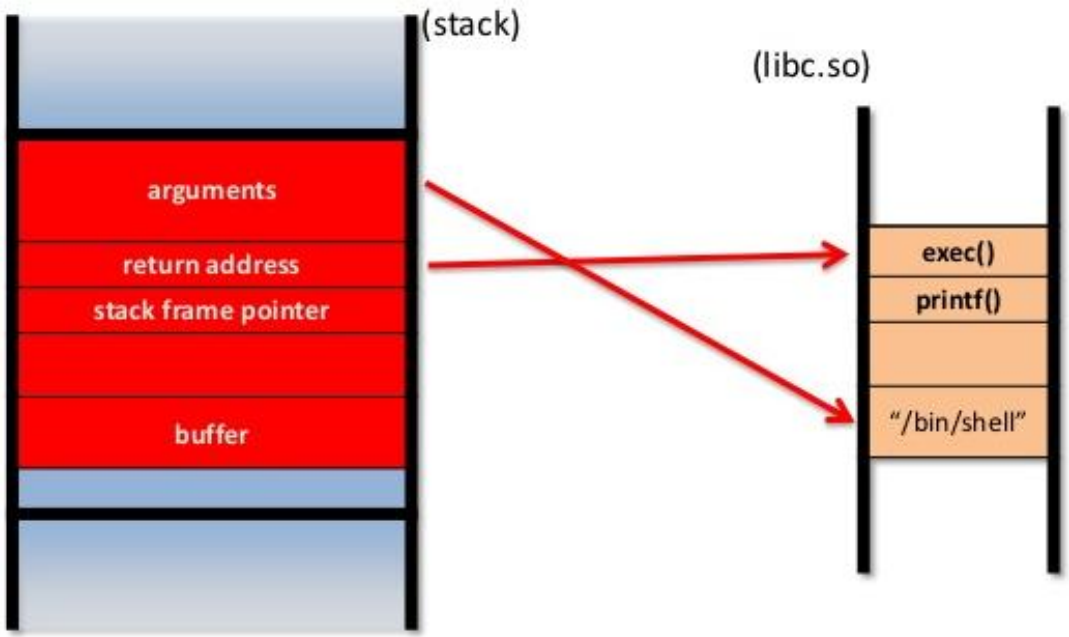


Figure: Return to libc

[0] https://tinyurl.com/yd8r36q4

# Payload building

## Finding strings

objdump -s ret2libc
(gdb) find 0x400000, 0x401000, "/bin"

## Finding gadgets

ropper –file ret2libc –search "% ?di"

# Ret2LibC Concepts

## Definition

"Ataque de *Control Flow Hijacking* através da exploração de um *buffer overflow* tendo o endereço de funções e argumentos da `libc` como *payload*"

# Topics

# The attack

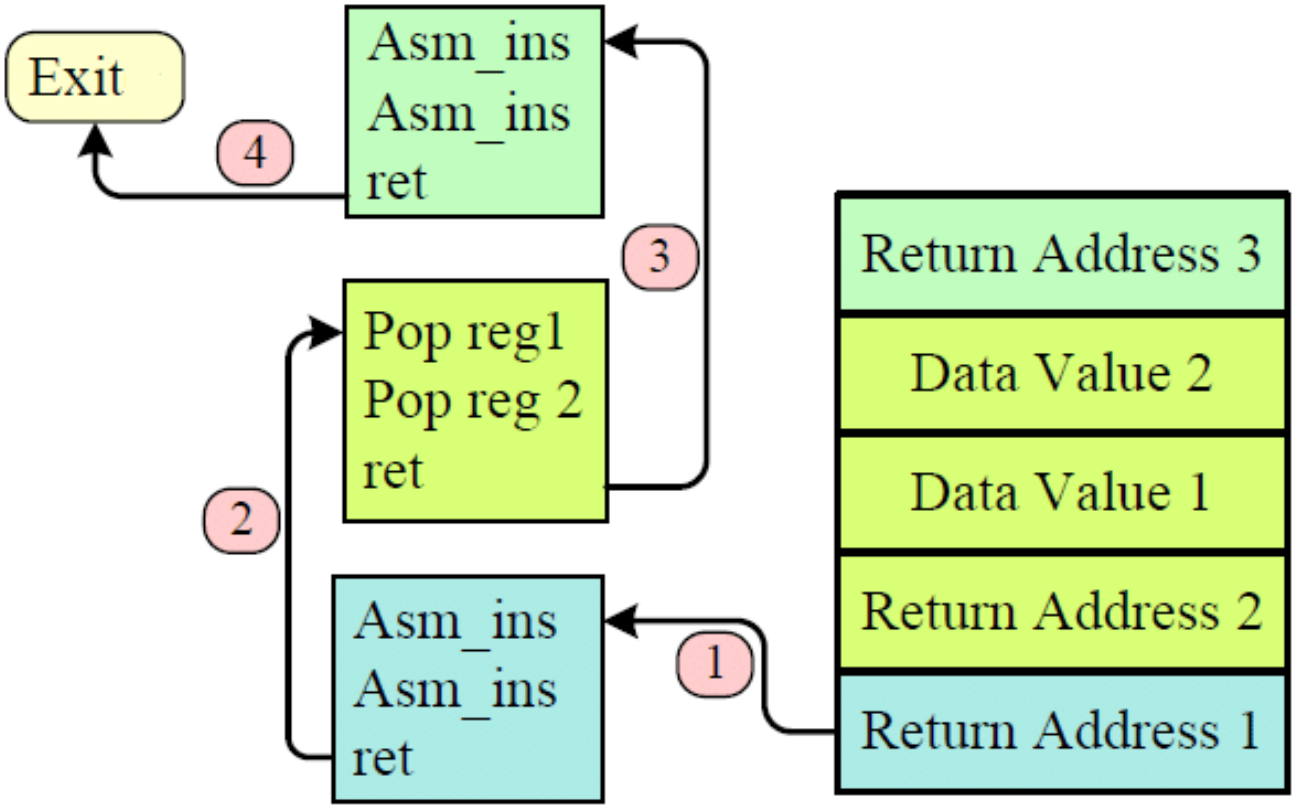


Figure: ROP gadget chaining

[0] `https://tinyurl.com/y7p3hrkk`

# Questions

- What is the relation between ROP attacks and the x64 calling convention ?
- How to find gadgets ?

Return Oriented Programming (ROP)

# More background

```
marcus@malware-lab:~/Documentos/aula-ROP$ objdump -d /bin/ls | head -50

/bin/ls: formato do arquivo elf64-x86-64


Desmontagem da seção .init:

00000000004022b8 <_init@@Base>:
  4022b8:       48 83 ec 08             sub    $0x8,%rsp
  4022bc:       48 8b 05 35 bd 21 00    mov    0x21bd35(%rip),%rax        # 61dff8 <_fini@@Base+0x20a39c>
  4022c3:       48 85 c0                test   %rax,%rax
  4022c6:       74 05                   je     4022cd <_init@@Base+0x15>
  4022c8:       e8 23 07 00 00          callq  4029f0 <__sprintf_chk@plt+0x10>
  4022cd:       48 83 c4 08             add    $0x8,%rsp
  4022d1:       c3                      retq
```

Figure: Binary disassembly

# Unaligned Instructions

Listing 3: Static disassembly of the `MSVCR71.dll` library.

```
c08: f2 0f 58 c3          addsd  %xmm3,%xmm0
c0c: 66 0f 13 44 24 04    movlpd %xmm0,0x4(%esp)
```

Listing 4: ROP Gadget.

```
c0a: 58 pop rax
c0b: c3 ret
```

# ROP Concepts

## Gadgets

"Sequência independente de instruções terminadas por `RET`"

## Gadget Chain

"Encadeamento de *gadgets* com o objetivo de realizar uma computação."

## ROP Attack

"Ataque de *control flow hijacking* através da exploração de um *buffer overlow* tendo um *gadget chain* como parte do *payload*."

# Topics

# Safe functions/languages

Listing 5: Popular strcpy prototype

```
char* strcpy(char* dst, char* src);
```

Listing 6: Not so popular strncpy prototype

```
char* strncpy(char* det, char* src, size_t num);
```

# Safe functions/languages

Listing 7: String definition example

```
typedef struct str{
  char str[MAX];
  uint size;
  uint max=MAX;
}string;
```

Listing 8: Concat implementation example

```
concat(str1,str2)
  if str1->size + str2->size < str1->max
```

# Safe String Functions

**Definition**

"Função que verifica os tamanhos dos *buffers* a fim de evitar um *overflow*."

# Stack Canaries

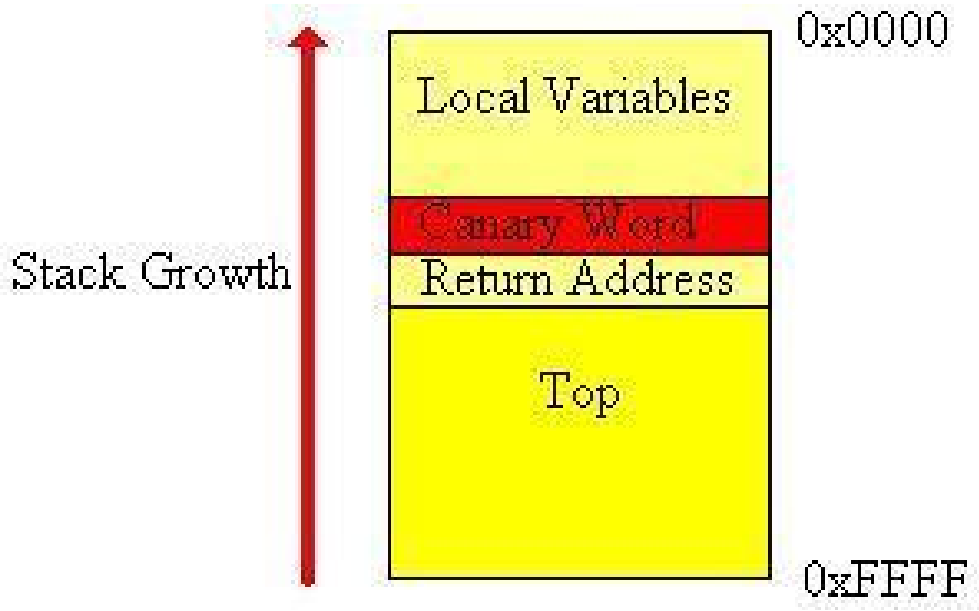


Figure: Stack canary

[0] https://tinyurl.com/yd9947ol
[0] Disable with: `gcc -fno-stack-protector`

# Stack Canaries

**Definition**

"Marcador da continência de um *buffer*."

# Question

## Compile-time solutions

- Advantages ?
- Disadvantages ?

# ASLR

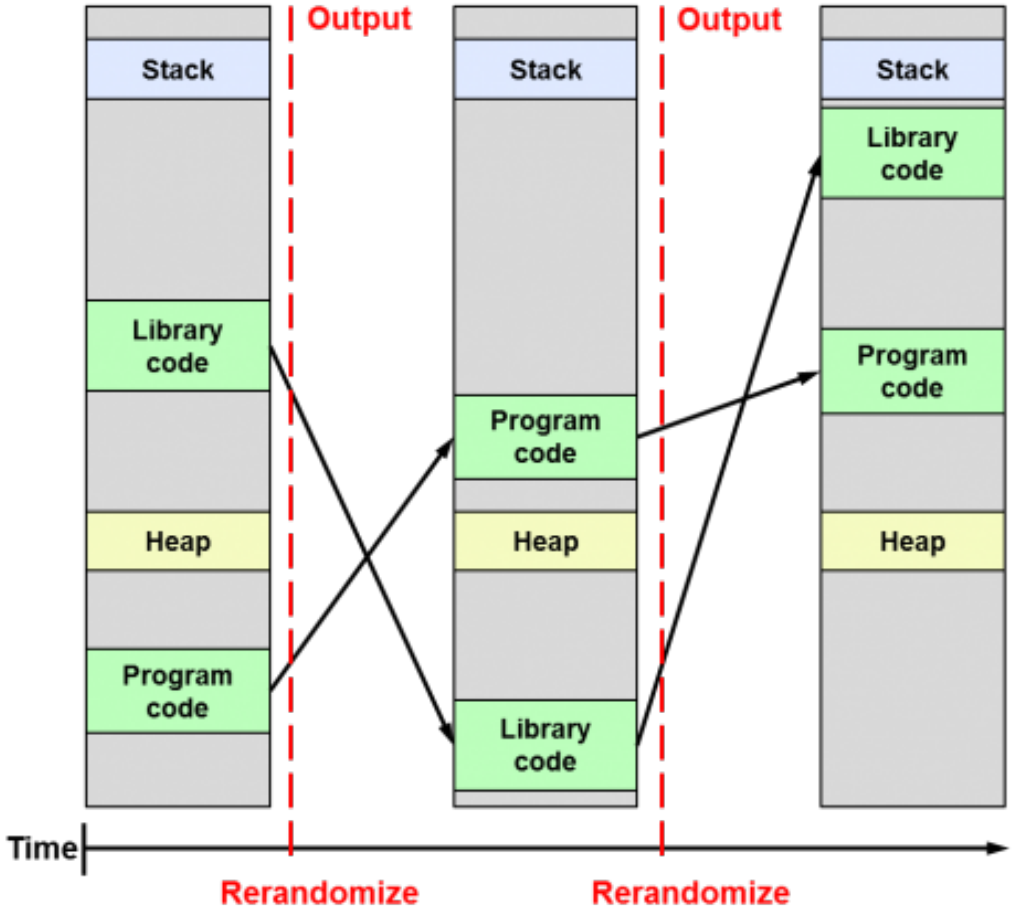


Figure: Address Space Layout Randomization


https://tinyurl.com/ycg2dlvo

# ASLR

Table: aslr - library placement after two consecutive reboots.

| **library** | ntdll | kernel32 | kernelbase |
|---|---|---|---|
| **address 1** | 0xbaf80000 | 0xb9610000 | 0xb8190000 |
| **address 2** | 0x987b0000 | 0x98670000 | 0x958c0000 |

# ASLR

**Definition**

"Aleatorização do posicionamento de conteúdos de memória a fim de evitar ataques de *offset* fixo."

# ASLR

## Position Independent Code

- GCC `-fpic`

## Jump Tables

```
00000000004022e0 <__ctype_toupper_loc@plt-0x10>:
  4022e0:       ff 35 22 bd 21 00       pushq  0x21bd22(%rip)        # 61e008 <_fini@@Base+0x20a3ac>
  4022e6:       ff 25 24 bd 21 00       jmpq   *0x21bd24(%rip)        # 61e010 <_fini@@Base+0x20a3b4>
  4022ec:       0f 1f 40 00             nopl   0x0(%rax)

00000000004022f0 <__ctype_toupper_loc@plt>:
  4022f0:       ff 25 22 bd 21 00       jmpq   *0x21bd22(%rip)        # 61e018 <_fini@@Base+0x20a3bc>
  4022f6:       68 00 00 00 00          pushq  $0x0
  4022fb:       e9 e0 ff ff ff          jmpq   4022e0 <_init@@Base+0x28>

0000000000402300 <__uflow@plt>:
  402300:       ff 25 1a bd 21 00       jmpq   *0x21bd1a(%rip)        # 61e020 <_fini@@Base+0x20a3c4>
  402306:       68 01 00 00 00          pushq  $0x1
  40230b:       e9 d0 ff ff ff          jmpq   4022e0 <_init@@Base+0x28>

0000000000402310 <getenv@plt>:
  402310:       ff 25 12 bd 21 00       jmpq   *0x21bd12(%rip)        # 61e028 <_fini@@Base+0x20a3cc>
  402316:       68 02 00 00 00          pushq  $0x2
  40231b:       e9 c0 ff ff ff          jmpq   4022e0 <_init@@Base+0x28>
```

Figure: Jump Table

# ASLR - Solution ?

**Breaking Kernel Address Space Layout Randomization with Intel TSX**

Yeongjin Jang, Sangho Lee, and Taesoo Kim
*Georgia Institute of Technology*

Figure: Breaking ASLR 1

**CAIN: Silently Breaking ASLR in the Cloud**

Antonio Barresi
*ETH Zurich*

Kaveh Razavi
*VU University Amsterdam*

Mathias Payer
*Purdue University*

Thomas R. Gross
*ETH Zurich*

Figure: Breaking ASLR 2

# Write XOR Execute Policy

## Write Xor Execute

- How to implement ?

Listing 9: /proc/cpuinfo

```
flags: fpu vme de pse tsc msr pae mce apic nx
```

# Write XOR Execute Policy

**Definition**

"Política que assegura que páginas de memória executáveis não podem ser escritas."

# Control Flow Integrity

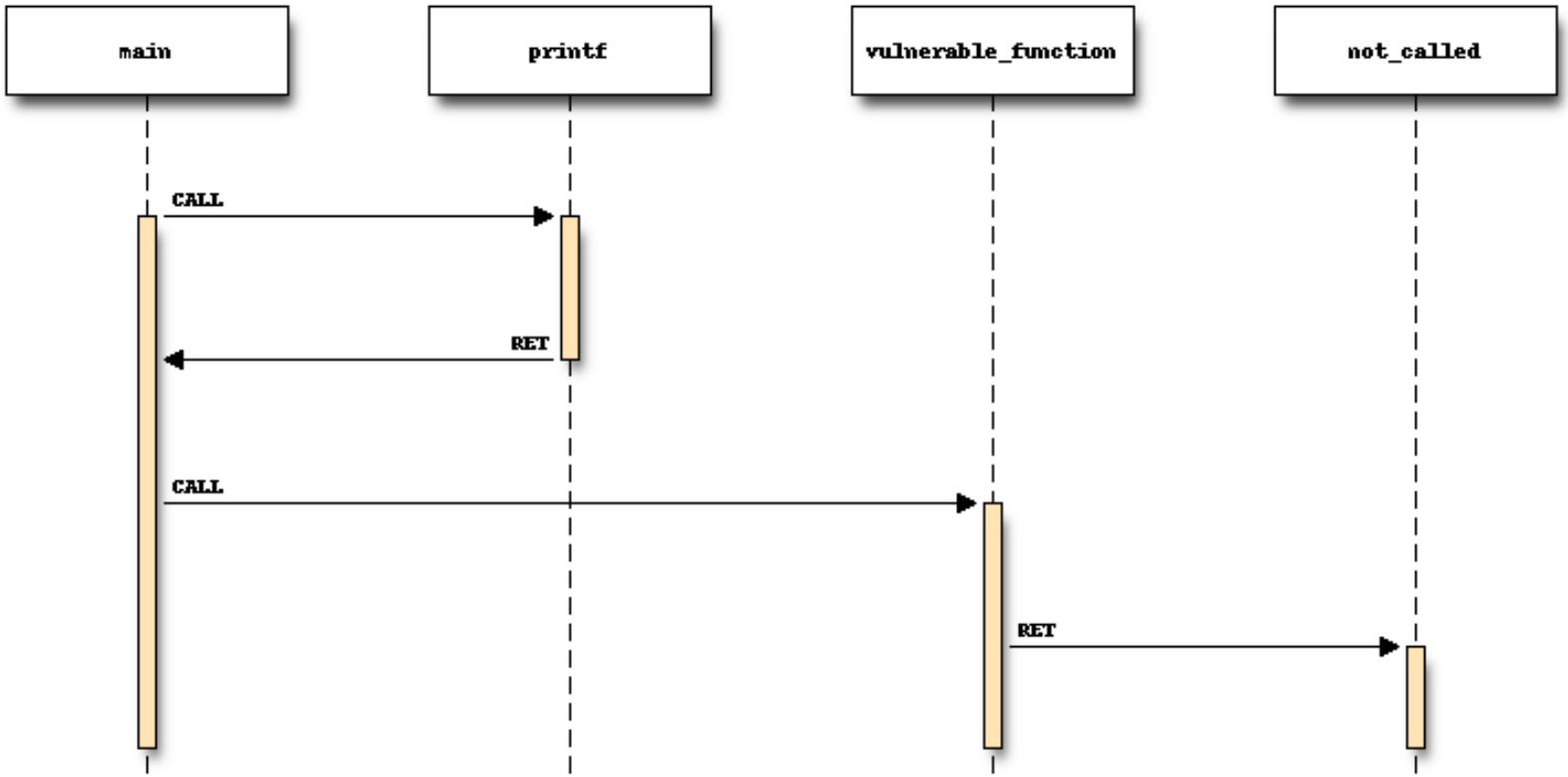


Figure: CALL-RET policy

# Shadow Stacks

**LBR Stack**

| | Branch | Target | |
|---|---|---|---|
| 00 | 73802745 | 738028D7 | |
| 01 | 05F015CA | 05F00E17 | |
| 02 | 7C348B06 | 7C34A028 | G01 |
| 03 | 7C34A02A | 7C34252C | G02 |
| 04 | 7C34252D | 7C36C55A | G03 |
| 05 | 7C36C55B | 7C345249 | G04 |
| 06 | 7C34524A | 7C3411C0 | G05 |
| 07 | 7C3411C1 | 7C34B8D7 | G06 |
| 08 | 7C34B8D8 | 7C366FA6 | G07 |
| 09 | 7C366FA7 | 7C3762FB | G08 |
| 10 | 7C3762FC | 7C378C81 | G09 |
| 11 | 7C378C84 | 7C346C0B | G10 |
| 12 | 7C346C0B | 7C3415A2 | G11 |
| 13 | 7C3415A2 | 74F64347 | |
| 14 | 74F64908 | 752AD0A1 | |
| 15 | 752D6FC8 | 752AD0AD | |

Figure: Branch Stack

# CFI - solution ?

**Control Jujutsu:**
**On the Weaknesses of Fine-Grained Control Flow Integrity***

Isaac Evans
MIT Lincoln Laboratory
ine@mit.edu

Fan Long
MIT CSAIL
fanl@csail.mit.edu

Ulziibayar Otgonbaatar
MIT CSAIL
ulziibay@csail.mit.edu

Howard Shrobe
MIT CSAIL
hes@csail.mit.edu

Martin Rinard
MIT CSAIL
rinard@csail.mit.edu

Hamed Okhravi
MIT Lincoln Laboratory
hamed.okhravi@ll.mit.edu

Stelios Sidiroglou-Douskos
MIT CSAIL
stelios@csail.mit.edu

Figure: CFI Bypass.

# CFI

## CFI Definition

"Imposição de que o fluxo de execução siga uma determinada política."

## CALL-RET Definition

"Política que exige que todas as instruções `RET` sejam precedidas por um `CALL`."

# Heuristics

## Questions

- What gadget feature ?
- Limitations ?

# Gadget Size Heuristic - solution ?

**Size Does Matter: Why Using Gadget-Chain Length to Prevent Code-Reuse Attacks is Hard**

Enes Göktaş, *Vrije Universiteit Amsterdam;* Elias Athanasopoulos, *FORTH-ICS;*
Michalis Polychronakis, *Columbia University;* Herbert Bos, *Vrije Universiteit Amsterdam;*
Georgios Portokalidis, *Stevens Institute of Technology*

Figure: Heuristic Bypass.

# Conclusion

- Questions ?