Lecture 2: Vulnerabilities

CS 181S

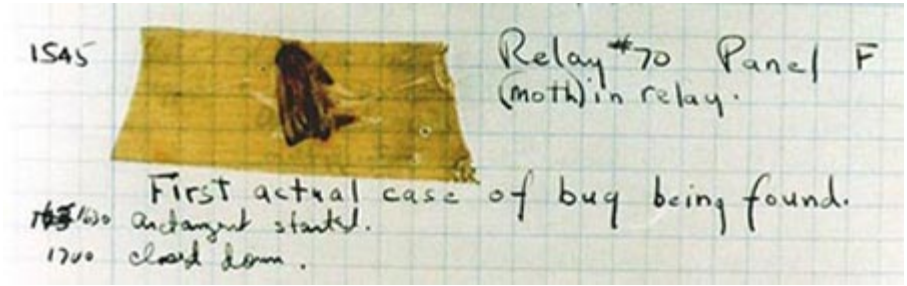
Fall 2020

The Big Picture

Attacks
are perpetrated by
threats
that inflict
harm
by exploiting
vulnerabilities
which are controlled by
countermeasures.

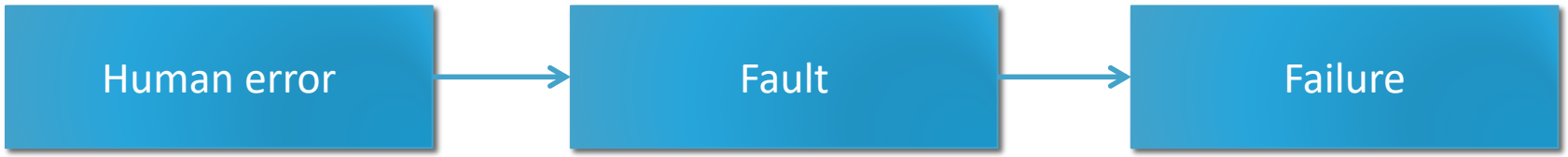
Bugs

"bug": suggests something just wandered in



[IEEE 729]

- Fault: result of human error in software system
  - E.g., implementation doesn't match design, or design doesn't match requirements
  - Might never appear to end user
- Failure: violation of requirement
  - Something goes wrong for end user

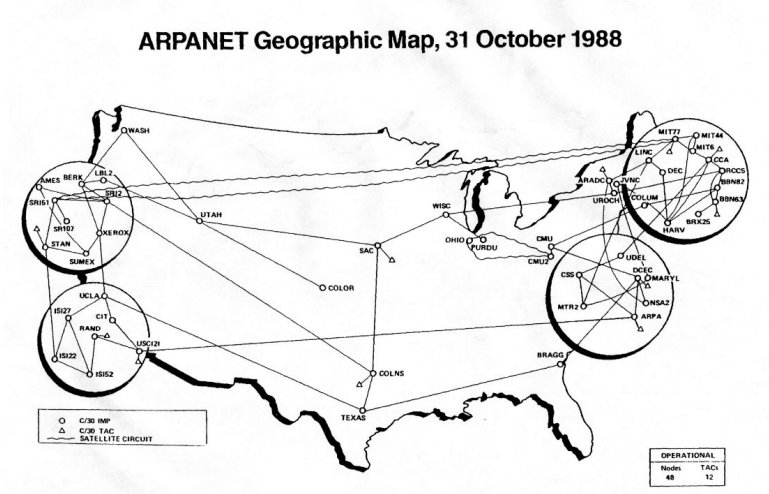


Vulnerability

An unintended aspect of a system (design, implementation, or configuration) that can cause the system to do something it shouldn't, or fail to do something it should

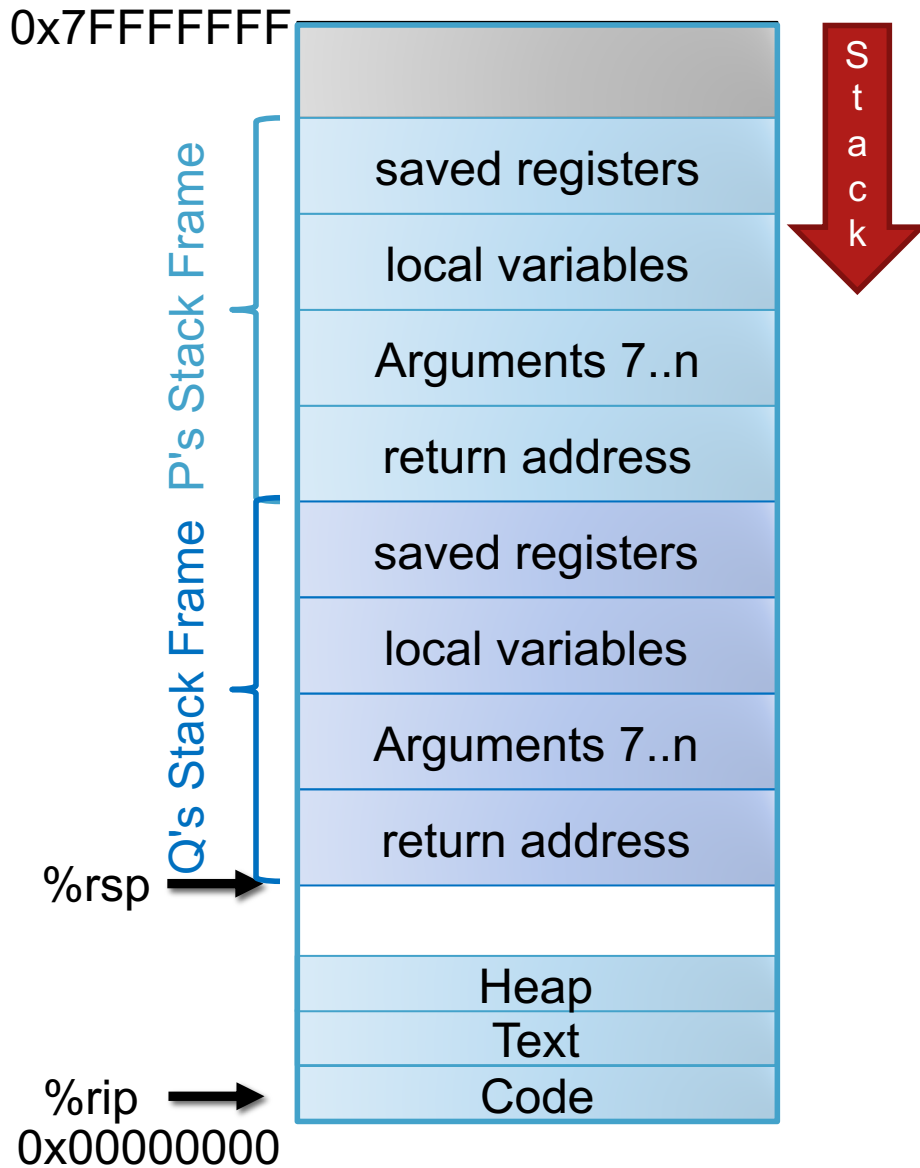
- E.g., buffer overflows, code injection, cross-site scripting, missing authentication or access control, misconfiguration
- National databases: CVE, NVD
- Ignoring vulnerabilities is risky
  - Too often: "no one would/could ever exploit that"
  - *Weakest link* phenomenon
- **Assumptions are vulnerabilities**

Buffer Overflow



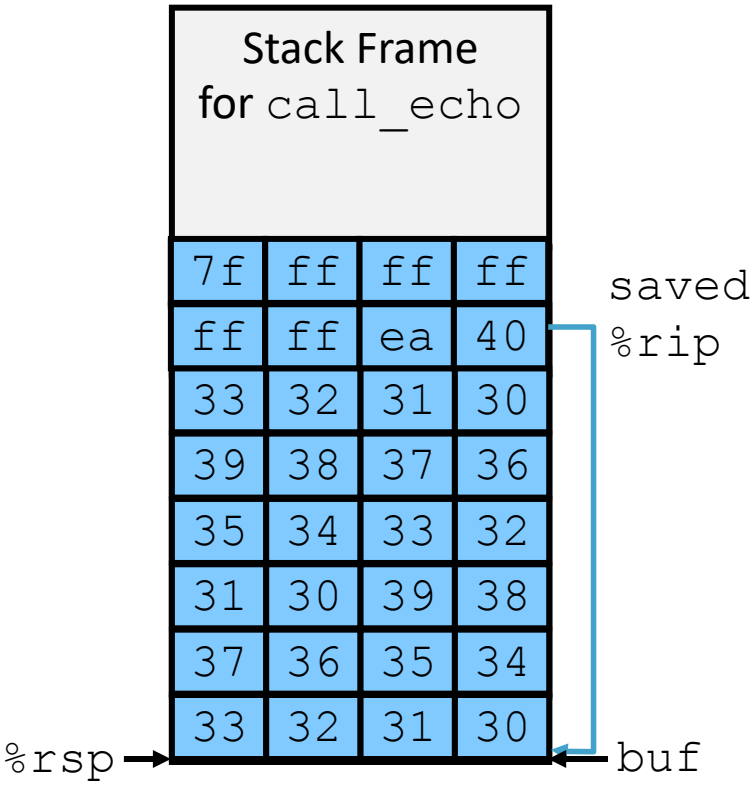
Review: Stack Frames

- Each function called gets a stack frame
- Passing data:
  - calling procedure P uses registers (and stack) to provide parameters to Q.
  - Q uses register %rax for return value
- Passing control:
  - **`call <proc>`**
    - Pushes return address (current **`%rip`**) onto stack
    - Sets **`%rip`** to first instruction of proc
  - **`ret`**
    - Pops return address from stack and places it in **`%rip`**
- Local storage:
  - allocate space on the stack by decrementing stack pointer, deallocate by incrementing



Review: Stack Smashing

- Idea: fill the buffer with bytes that will be interpreted as code
- Overwrite the return address with address of the beginning of the buffer

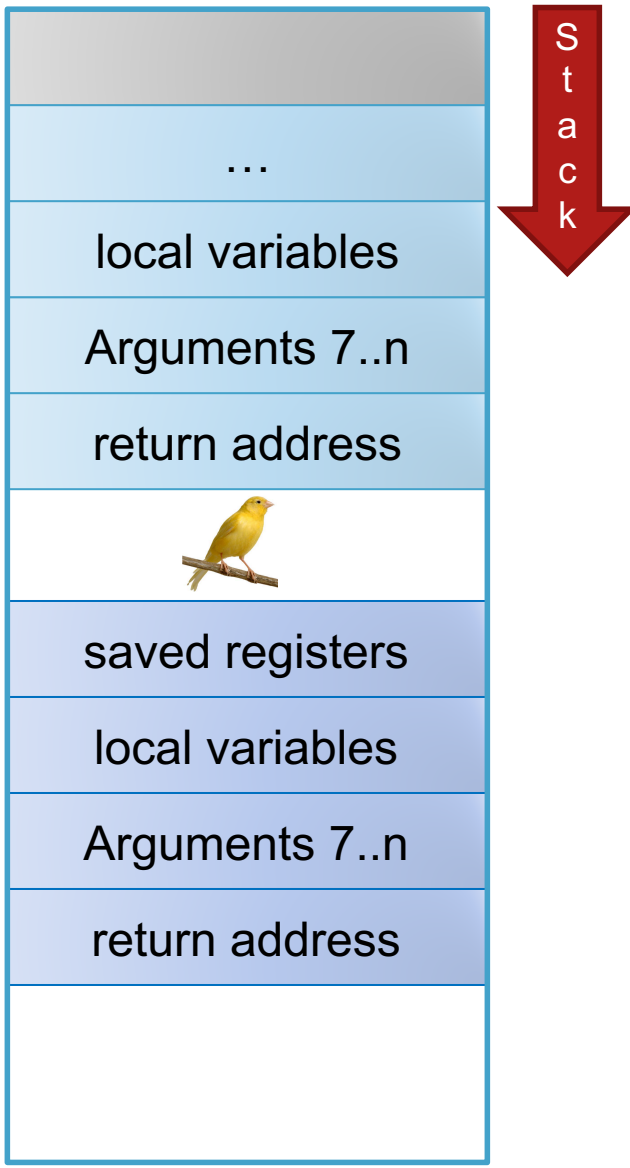



```
/* Echo Line */
void echo()
{
    char buf[4];
    gets(buf);
    puts(buf);
}
```

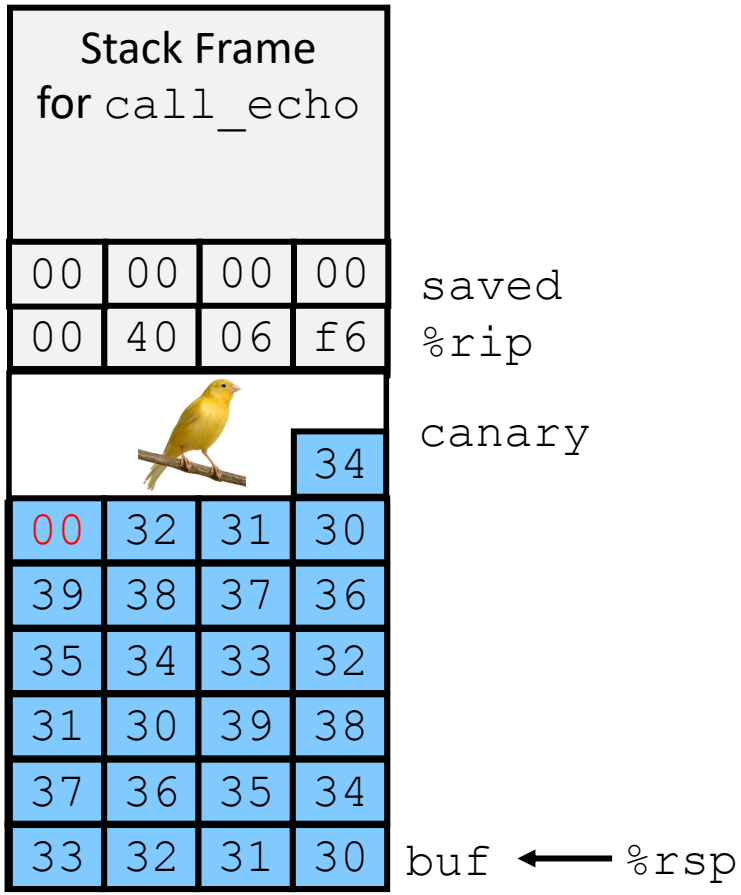
```
echo:
  subq  $18, %rsp
  movq  %rsp, %rdi
  call  gets
  call puts
  addq  $18, %rsp
  ret
```

Review: Stack Canaries

- Idea
  - Place special value (“canary”) on stack just beyond buffer
  - Check for corruption before exiting function
- GCC Implementation
  - `-fstack-protector`
  - Now the default (disabled earlier)



Review: Stack Canaries



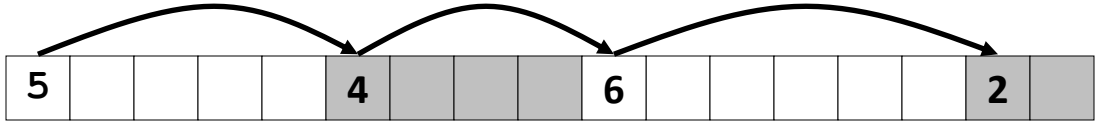

```
authenticate:
  pushq   %rbx
  subq    $16, %rsp
  movq    %rdi, %rbx
  movq    %fs:40, %rax
  movq    %rax, 8(%rsp)
  xorl    %eax, %eax
  movq    %rsp, %rdi
  call    gets
  movq    %rsp, %rsi
  movq    %rbx, %rdi
  call    strcmp
  testl   %eax, %eax
  sete    %al
  movq    8(%rsp), %rdx
  xorq    %fs:40, %rdx
  je      .L2
  call    __stack_chk_fail
.L2:
  movzbl  %al, %eax
  addq    $16, %rsp
  popq    %rbx
  ret
```

Exercise 1: Stack Canaries

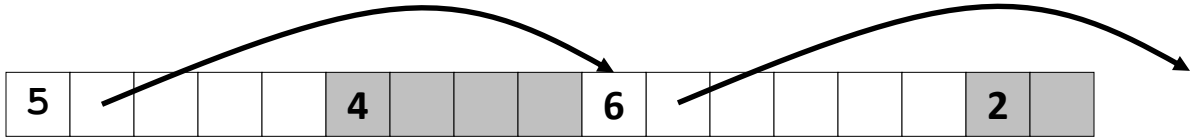
- Which of the following would make a good stack canary?
  1. A secret, constant value
  2. The current process ID
  3. A fixed sequence of common terminators (\0, EOF, etc.)
  4. A random number chosen each time the program is run

Review: Tracking Free Blocks

- Method 1: ***Implicit list*** using length—links all blocks



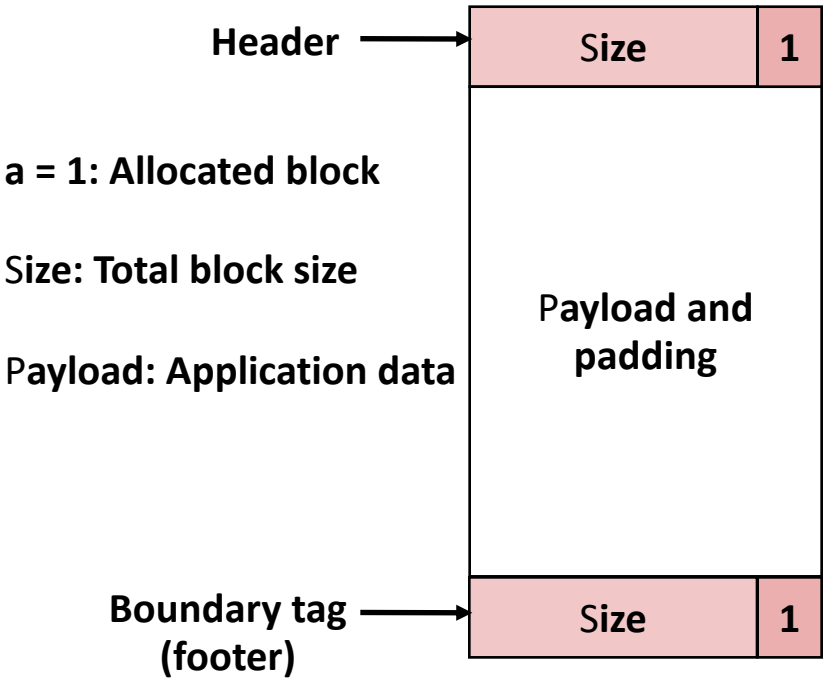
- Method 2: ***Explicit list*** among the free blocks using pointers



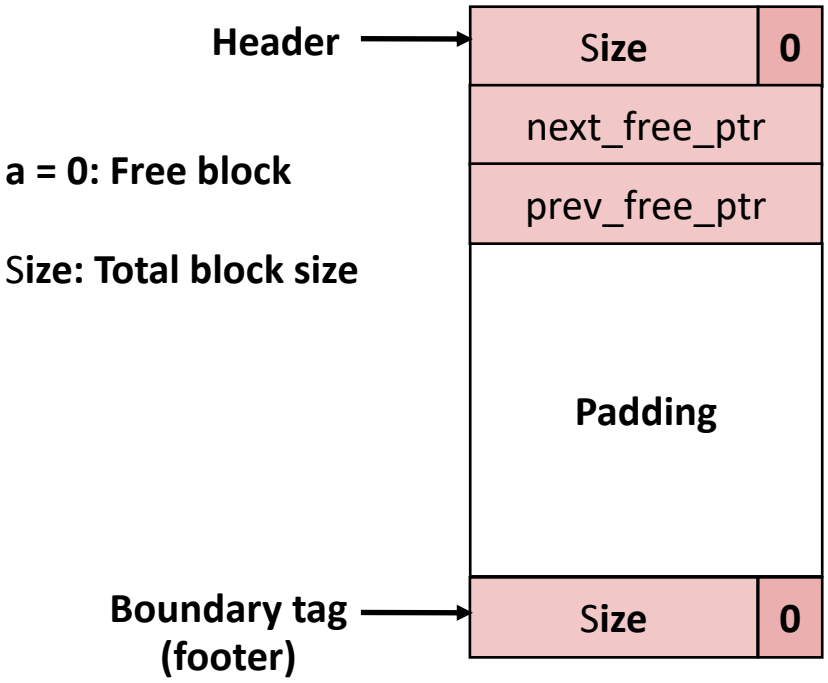
- Method 3: ***Segregated free list***
  - Different free lists for different size classes
- Method 4: ***Blocks sorted by size***
  - Can use a balanced tree (e.g. Red-Black tree) with pointers within each free block, and the length used as a key

Review: Block Format

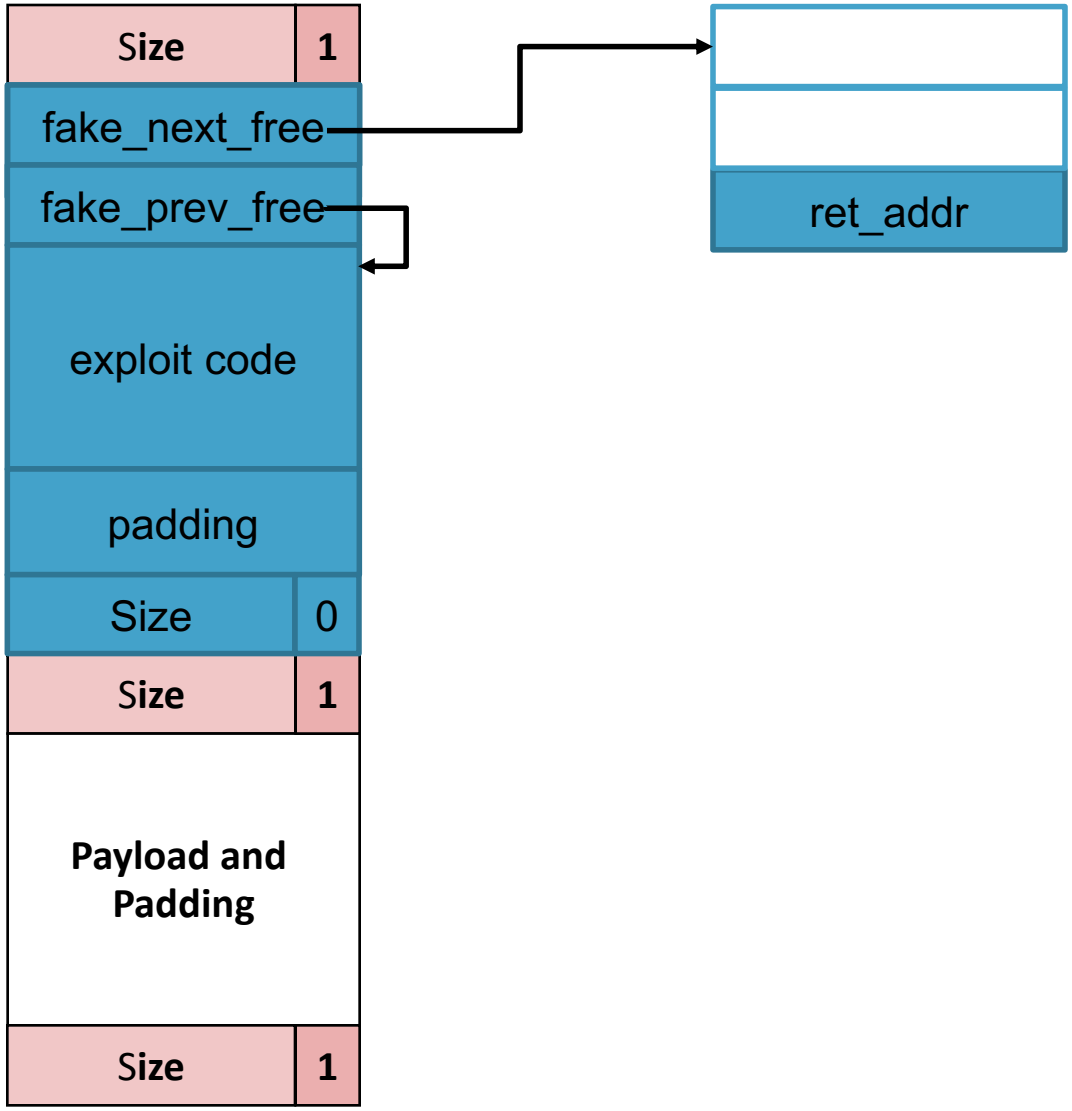
Allocated Blocks



Free Blocks

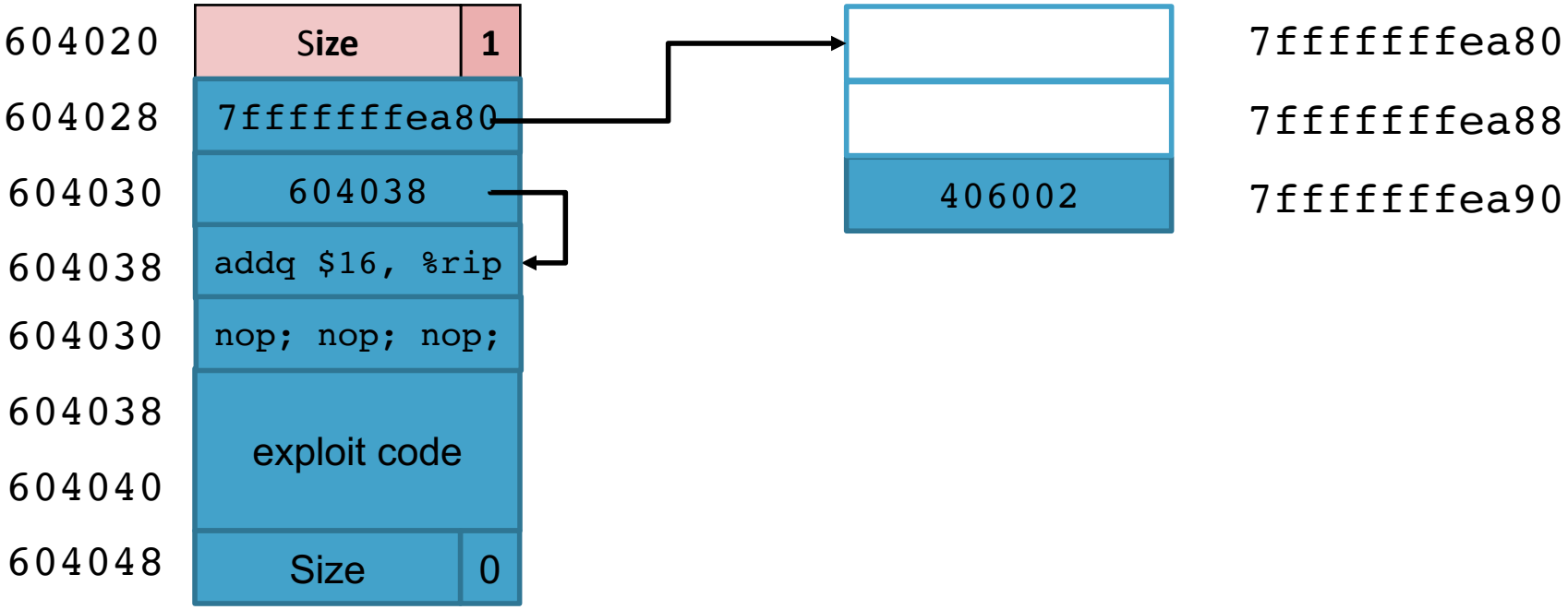


Heap Smashing



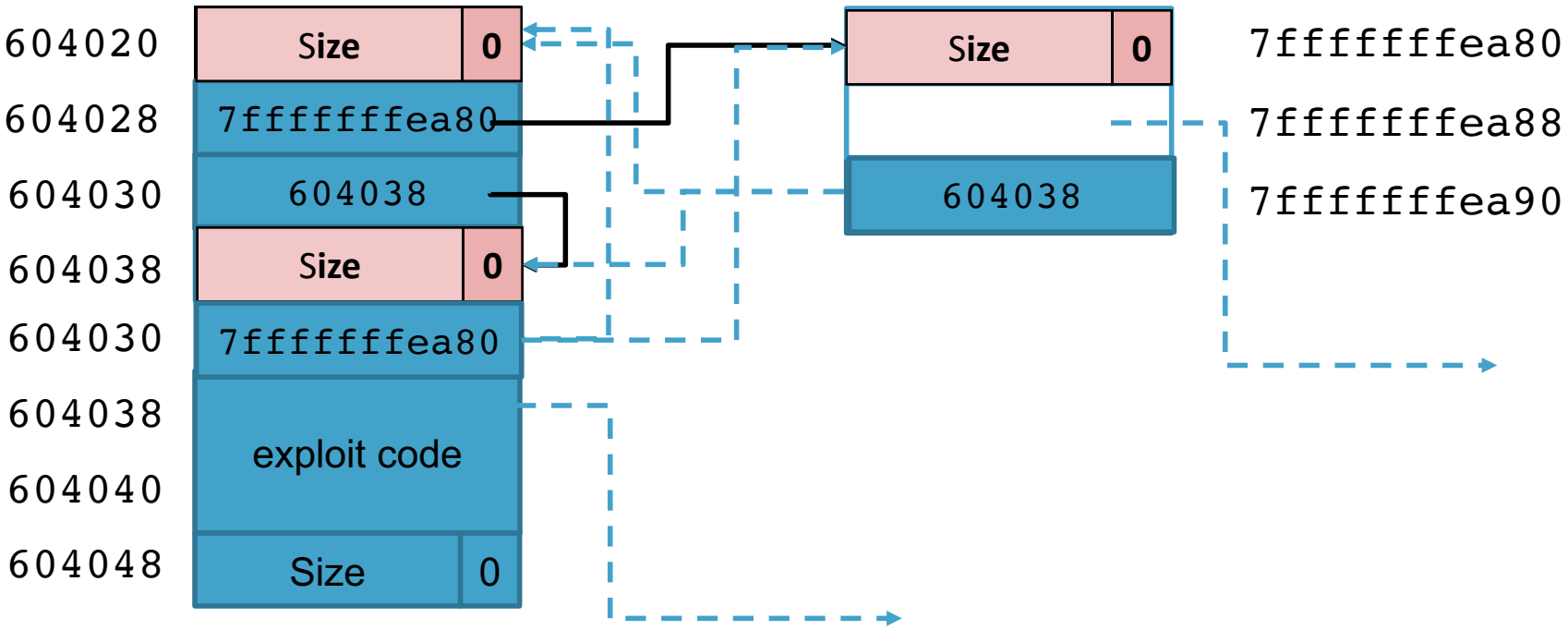
Exercise 2: Heap Smashing

- What would happen when the block after this one gets freed? (Assume that the coalesced block needs to be moved into a new linked list.)



Exercise 2: Heap Smashing

- What would happen when the block after this one gets freed? (Assume that the coalesced block needs to be moved into a new linked list.)



Data Execution Prevention (DEP)

Code Reuse Attacks

- Key idea: execute instructions that already exist
- Defeats memory tagging defenses
- Examples:
  1. return to a library function (e.g., return-into-`libc`)
  2. return to some other instruction (return-oriented programming)

Return-into-libc

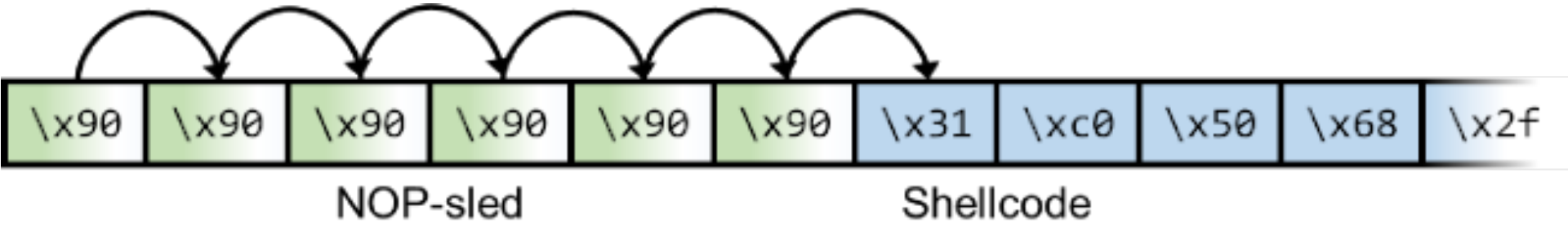
Sr.No.	Function & Description
1	**double atof(const char *str)** Converts the string pointed to, by the argument *str* to a floating-point number (type double).
2	**int atoi(const char *str)** Converts the string pointed to, by the argument *str* to an integer (type int).
3	**long int atol(const char *str)** Converts the string pointed to, by the argument *str* to a long integer (type long int).
8	**void free(void *ptr** Deallocates the memory previously allocated by a call to *calloc, malloc,* or *realloc*.
9	**void *malloc(size_t size)** Allocates the requested memory and returns a pointer to it.
10	**void *realloc(void *ptr, size_t size)** Attempts to resize the memory block pointed to by ptr that was previously allocated with a call to *malloc* or *calloc*.
15	**int system(const char *string)** The command specified by string is passed to the host environment to be executed by the command processor.
16	**void *bsearch(const void *key, const void *base, size_t nitems, size_t size, int (*compar)(const void *, const void *))** Performs a binary search.
17	**void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void*))** Sorts an array.
18	**int abs(int x)** Returns the absolute value of x.
22	**int rand(void)** Returns a pseudo-random number in the range of 0 to *RAND_MAX*.
23	**void srand(unsigned int seed)** This function seeds the random number generator used by the function **rand**.

ASCII Armoring

- Make sure all system library addresses contain a null byte (0x00).
- Can be done by placing this code in the first 0x01010101 bytes of memory

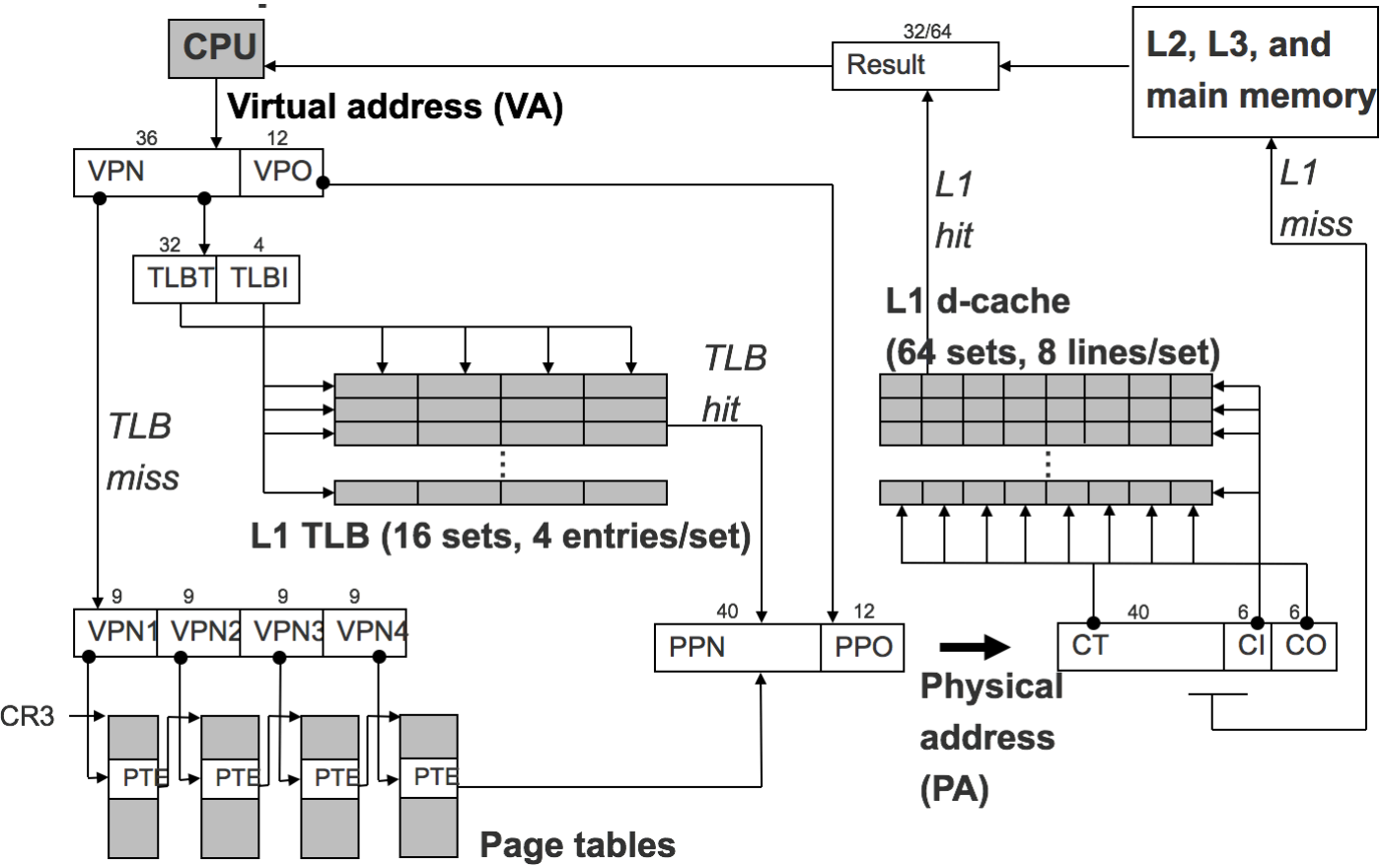
Address Space Layout Randomization

JIT Spraying



Memory Disclosure Vulnerabilities

- ASLR ⊕ Cache



Properties of x86 Assembly

- Intel Instruction Set Architecture (ISA)
- Introduced 1978, still supported
- As of 2020, most common architecture on servers, PCs, and laptops

- variable length instructions
- not word aligned
- dense instruction set

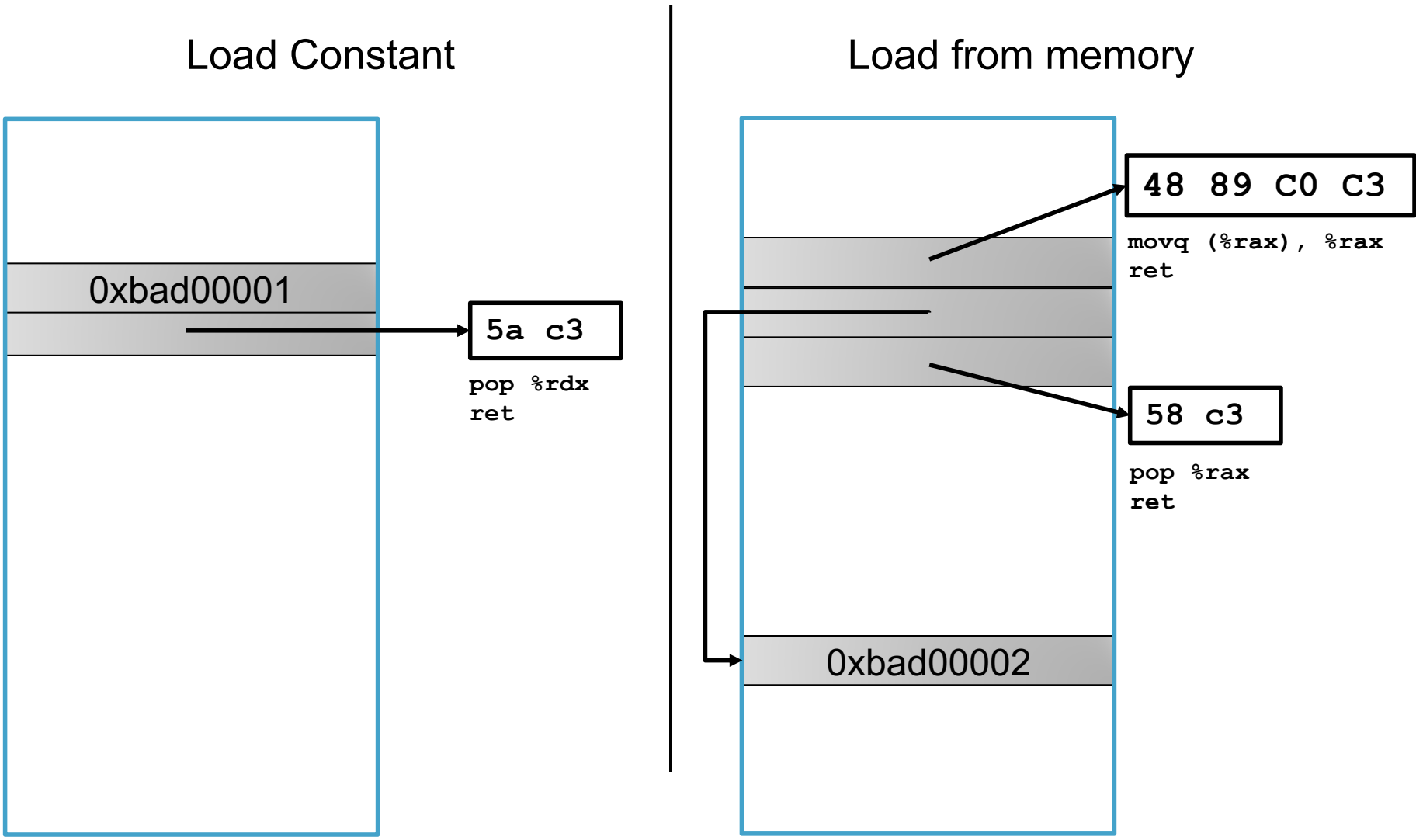
Gadgets

```
void setval(unsigned *p) {
  *p = 3347663060u;
}
```

```
<setval>:
4004d9: c7 07 d4 48 89 c7   movl $0xc78948d4,(%rdi)
4004df: c3                  ret
```

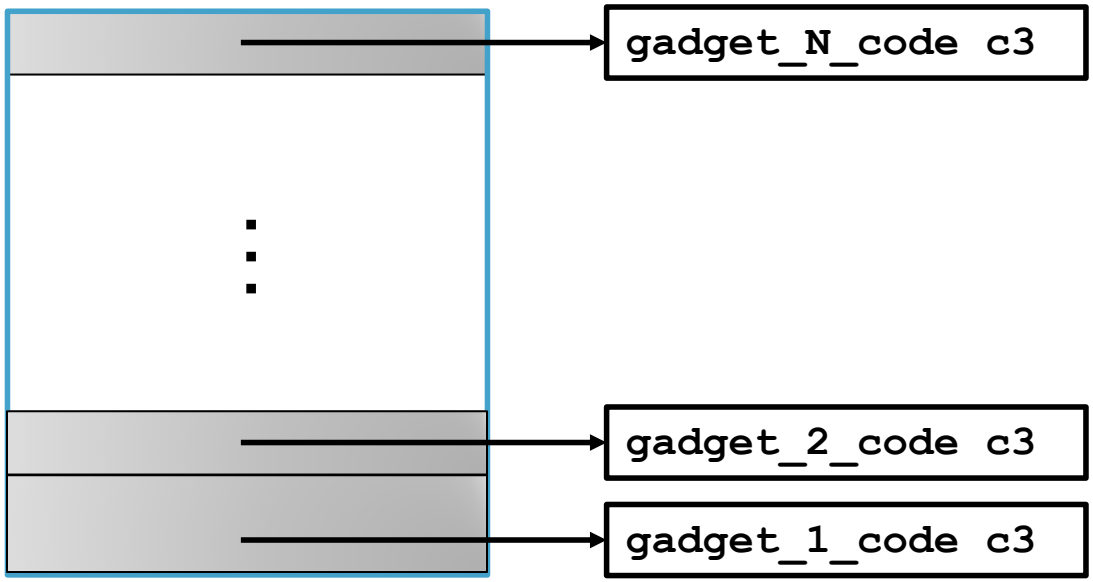
gadget address: `0x4004dc`
encodes: `movq %rax, %rdi`
`ret`
executes: `%rdi <- %rax`

Example Gadgets



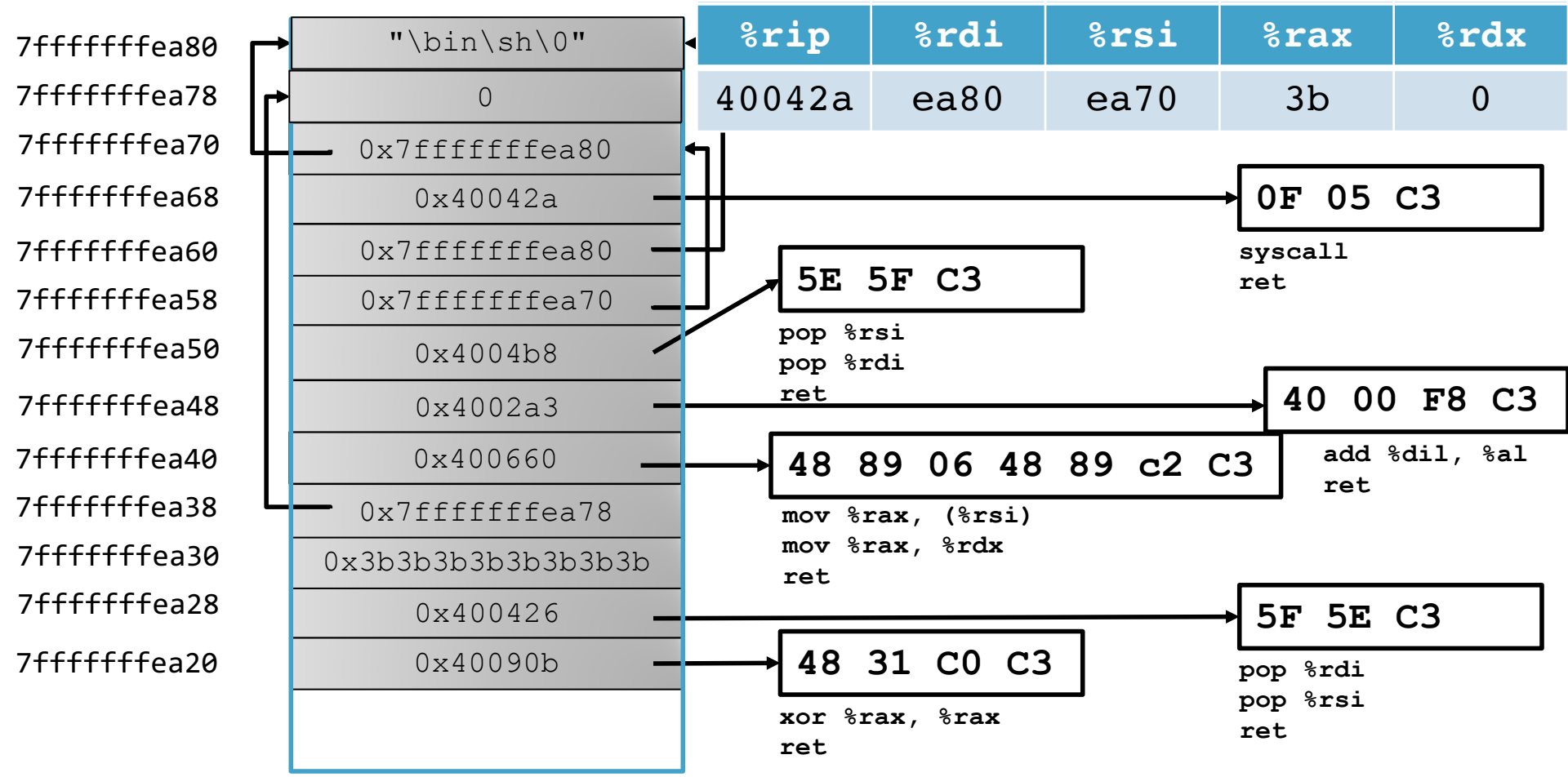
Return-oriented Programming

Return-oriented Programming



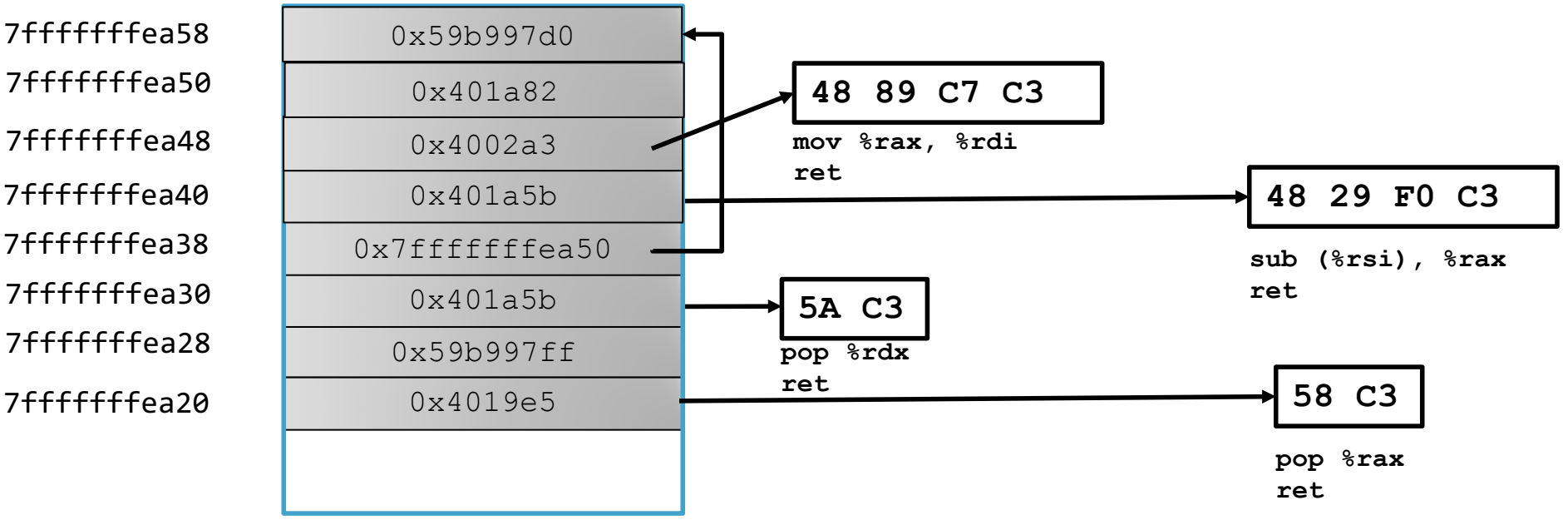
Final ret in each gadget sets pc (%rip) to beginning of next gadget code

Return-Oriented Shellcode



Exercise 3: ROP

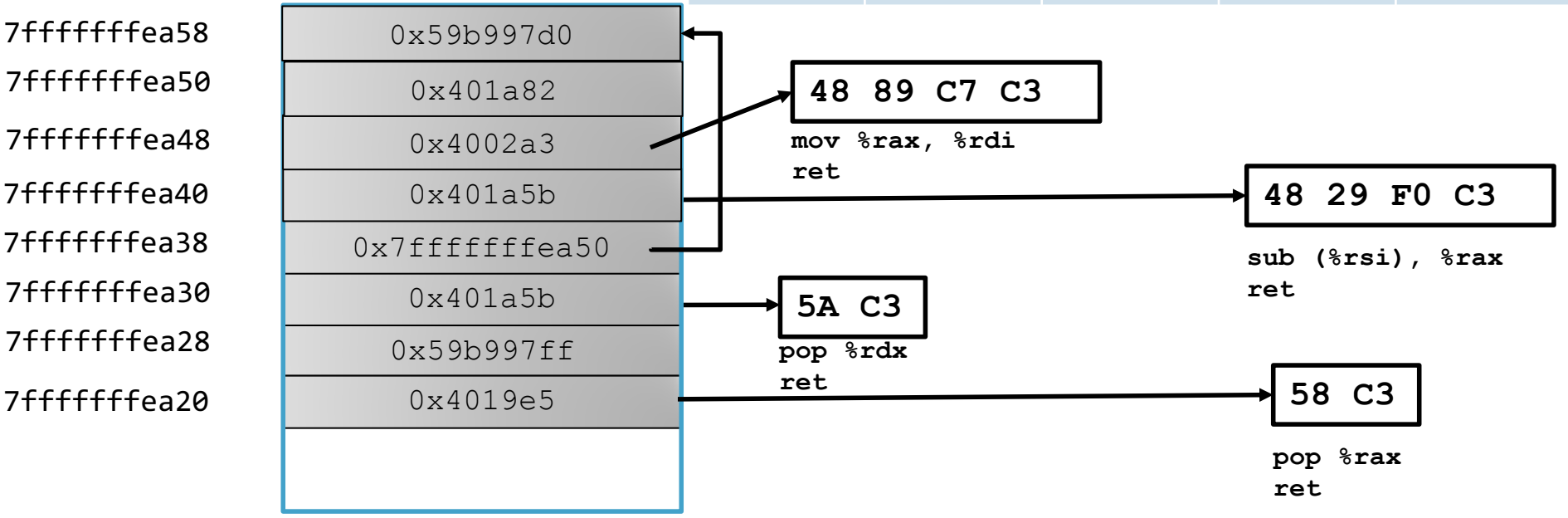
- What are the values in the registers when the function at address 0x401a82 gets called?



Exercise 3: ROP

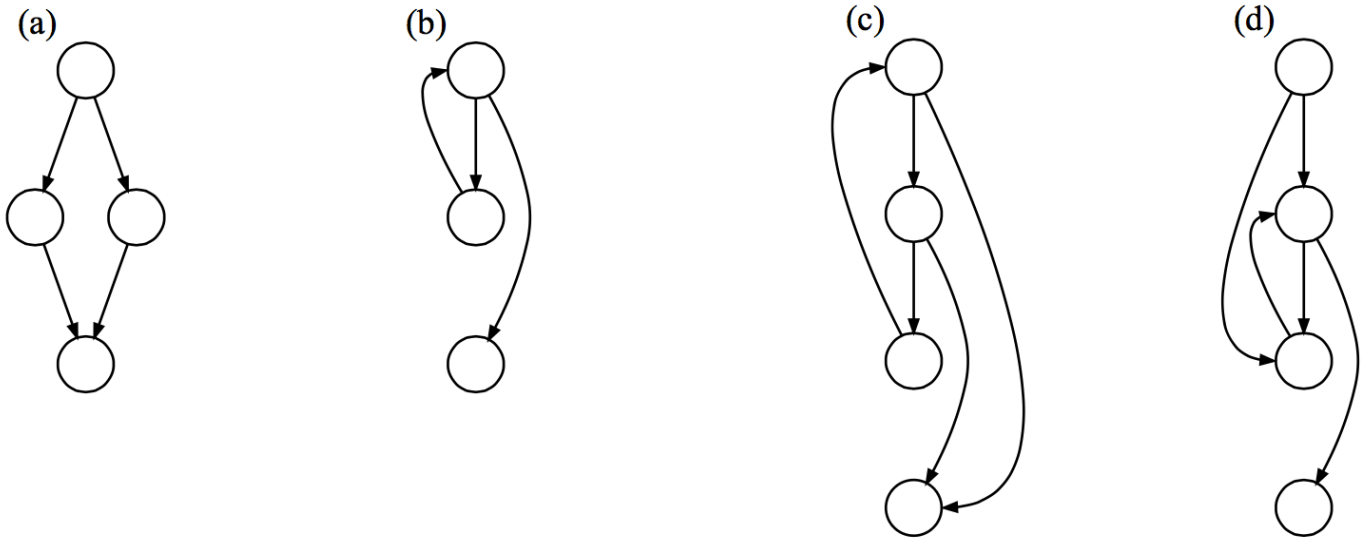
- What are the values in the registers when the function at address 0x401a82 gets called?

%rip	%rdi	%rsi	%rax	%rdx
401a82	2f		2f	..ea58

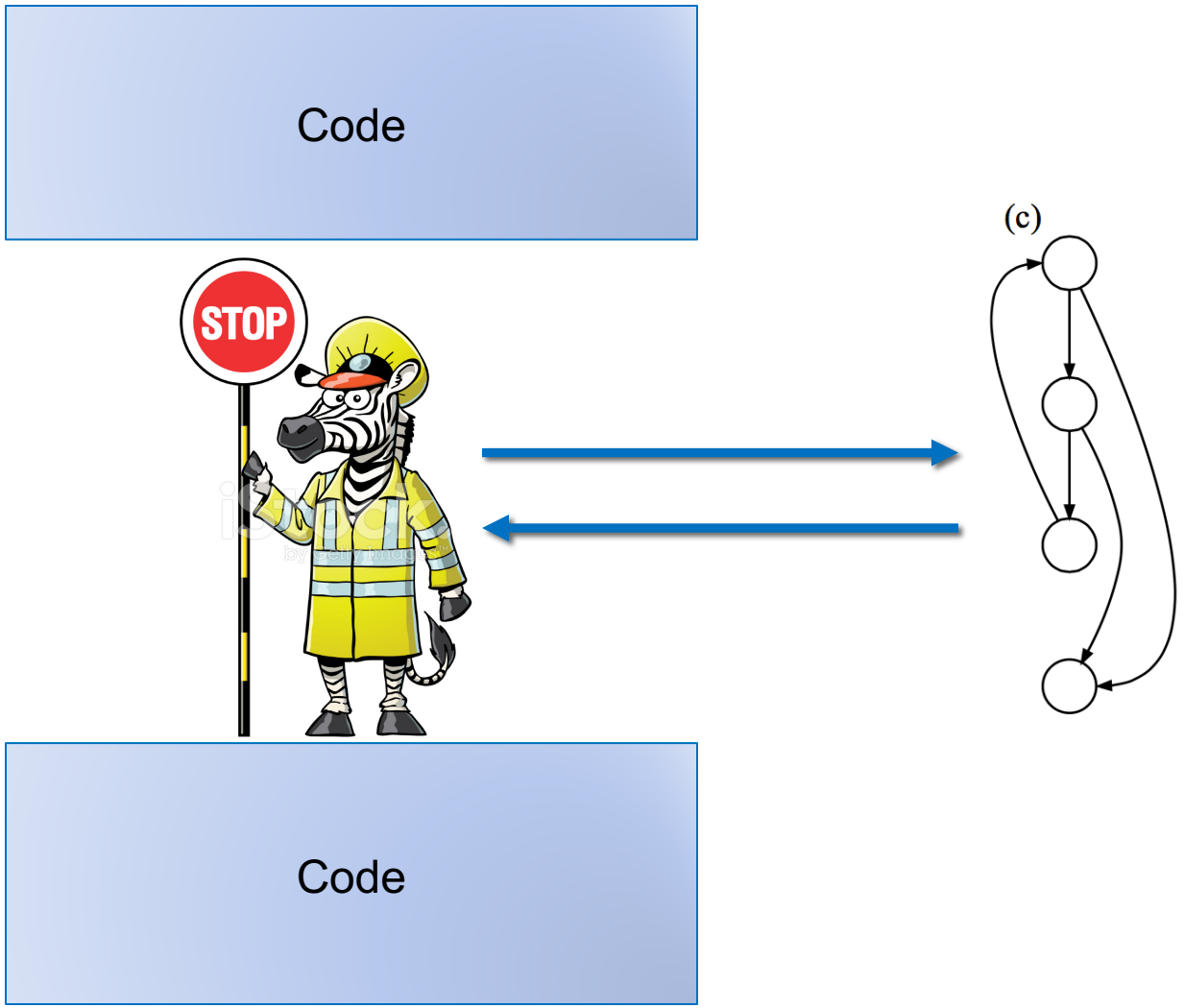


Gadget Elimination

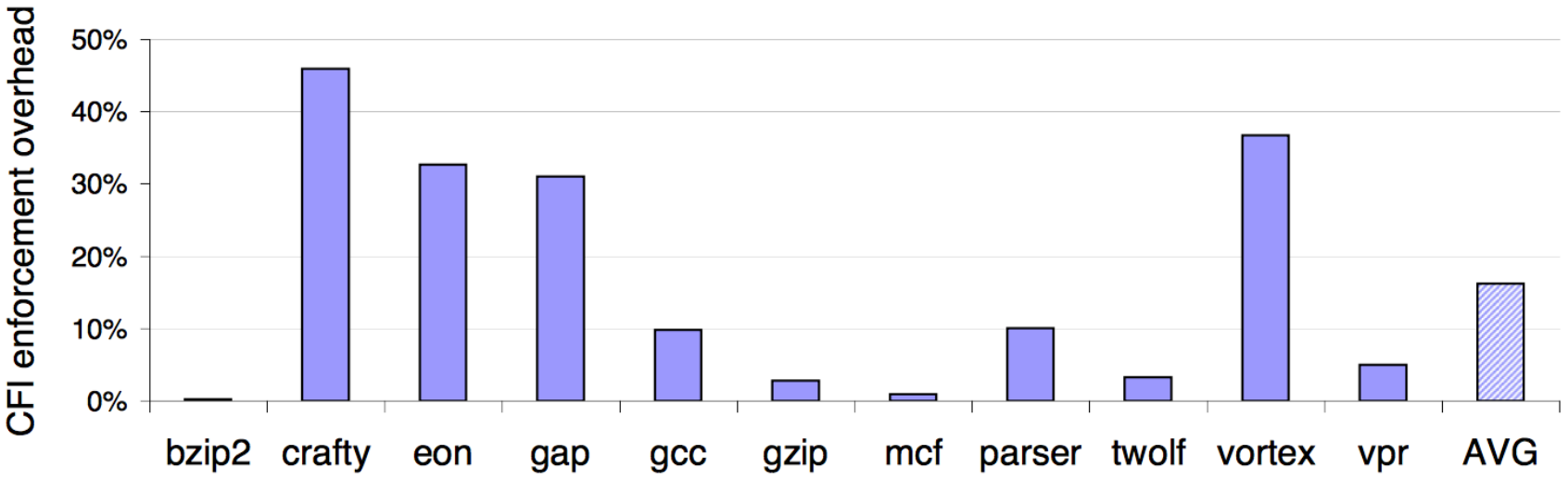
Control Flow Integrity



CFI = Insert Monitors

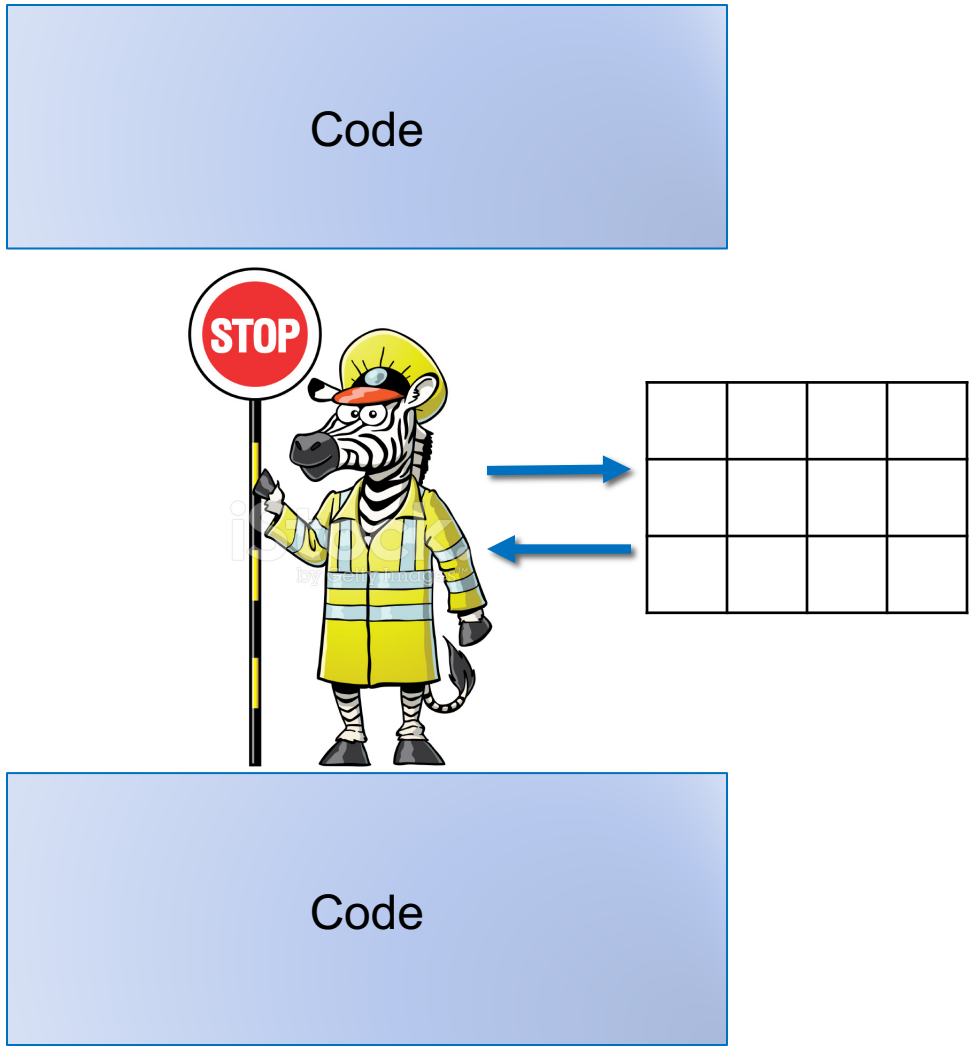


CFI Overhead

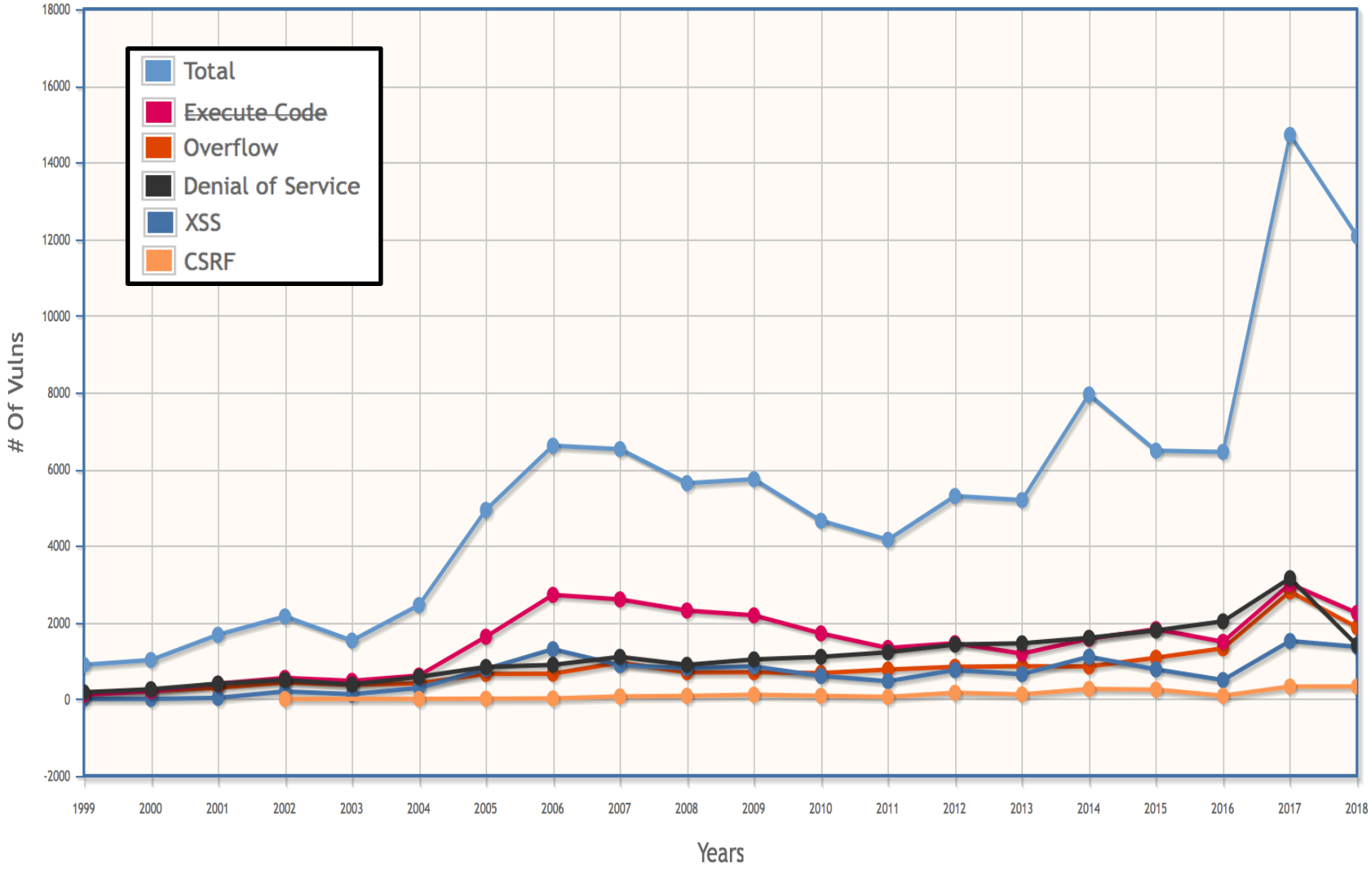


Control Flow Guard

- Approximate CFI implementation in Windows 8.1, 10
- Jump is valid if it beginning of function
  - Granularity: 8 bytes
- Check implemented as bitmap



Vulnerabilities by Year



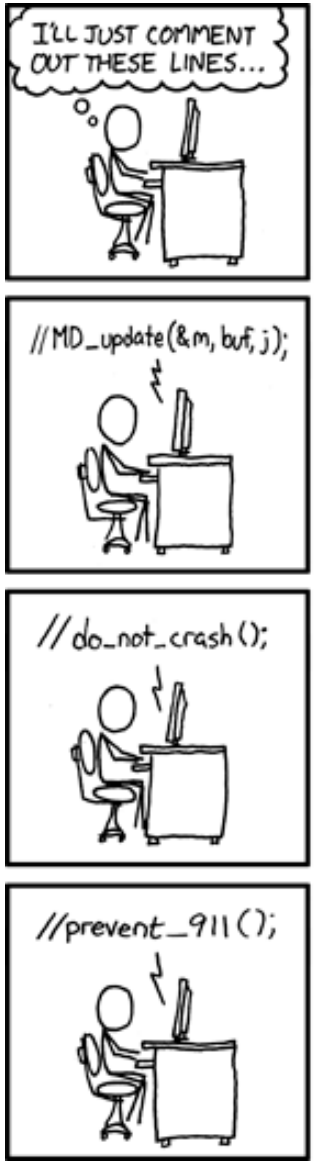
Exercise 4: Feedback

1. Rate how well you think this recorded lecture worked
    1. Better than an in-person class
    2. About as well as an in-person class
    3. Less well than an in-person class, but you still learned something
    4. Total waste of time, you didn't learn anything

2. How much time did you spend on this video lecture (including time spent on exercises)?

3. Do you have any comments or suggestions for future classes?

Vulnerabilities



IN THE RUSH TO CLEAN UP THE DEBIAN-OPENSSL FIASCO, A NUMBER OF OTHER MAJOR SECURITY HOLES HAVE BEEN UNCOVERED:

AFFECTED SYSTEM	SECURITY PROBLEM
FEDORA CORE	VULNERABLE TO CERTAIN DECODER RINGS
XANDROS (EEE PC)	GIVES ROOT ACCESS IF ASKED IN STERN VOICE
GENTOO	VULNERABLE TO FLATTERY
OLPC OS	VULNERABLE TO JEFF GOLDBLUM'S POWERBOOK
SLACKWARE	GIVES ROOT ACCESS IF USER SAYS ELVISH WORD FOR "FRIEND"
UBUNTU	TURNS OUT DISTRO IS ACTUALLY JUST WINDOWS VISTA WITH A FEW CUSTOM THEMES