Defending against buffer overflows

STACK CANARIES AND STACKGUARD

Simple Example Program

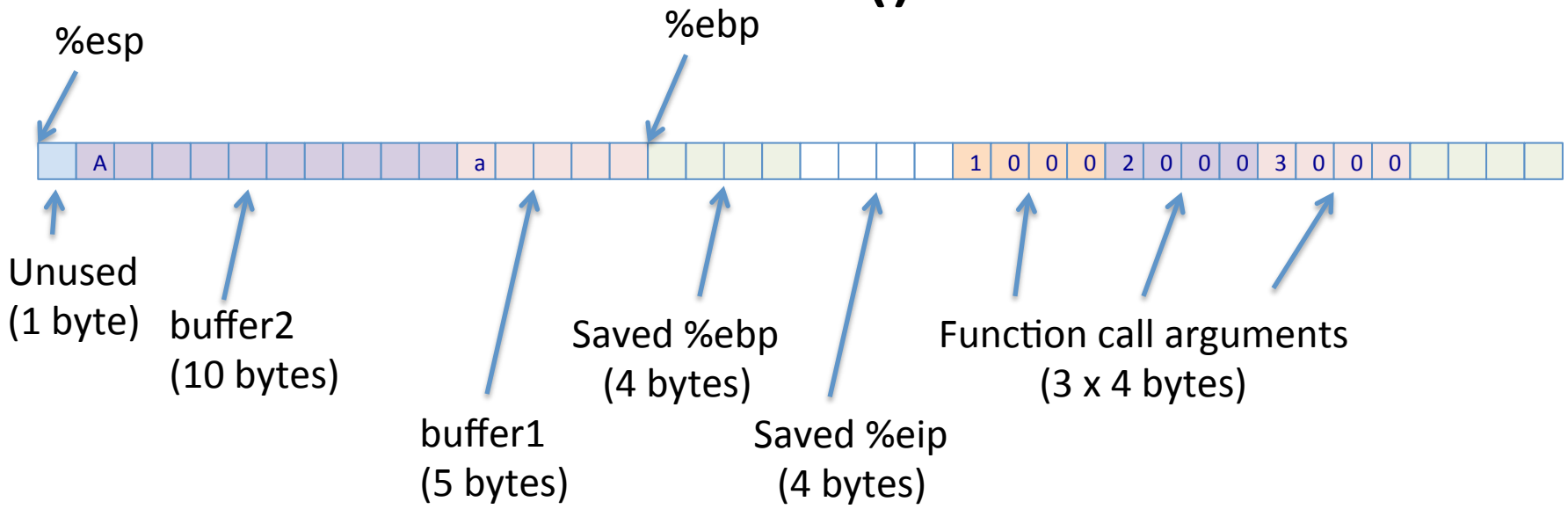
example1.c

```
void function(int a, int b, int c) {
  char buffer1[5];
  char buffer2[10];
  buffer1[0] = 'a';
  buffer2[0] = 'A';
}

void main() {
  function(1,2,3);
}
```

In Assembly: example1.s

```
function:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $16, %esp
        movb    $97, -5(%ebp)
        movb    $65, -15(%ebp)
        leave
        ret
```

Example1 Stack Frame Layout: function()



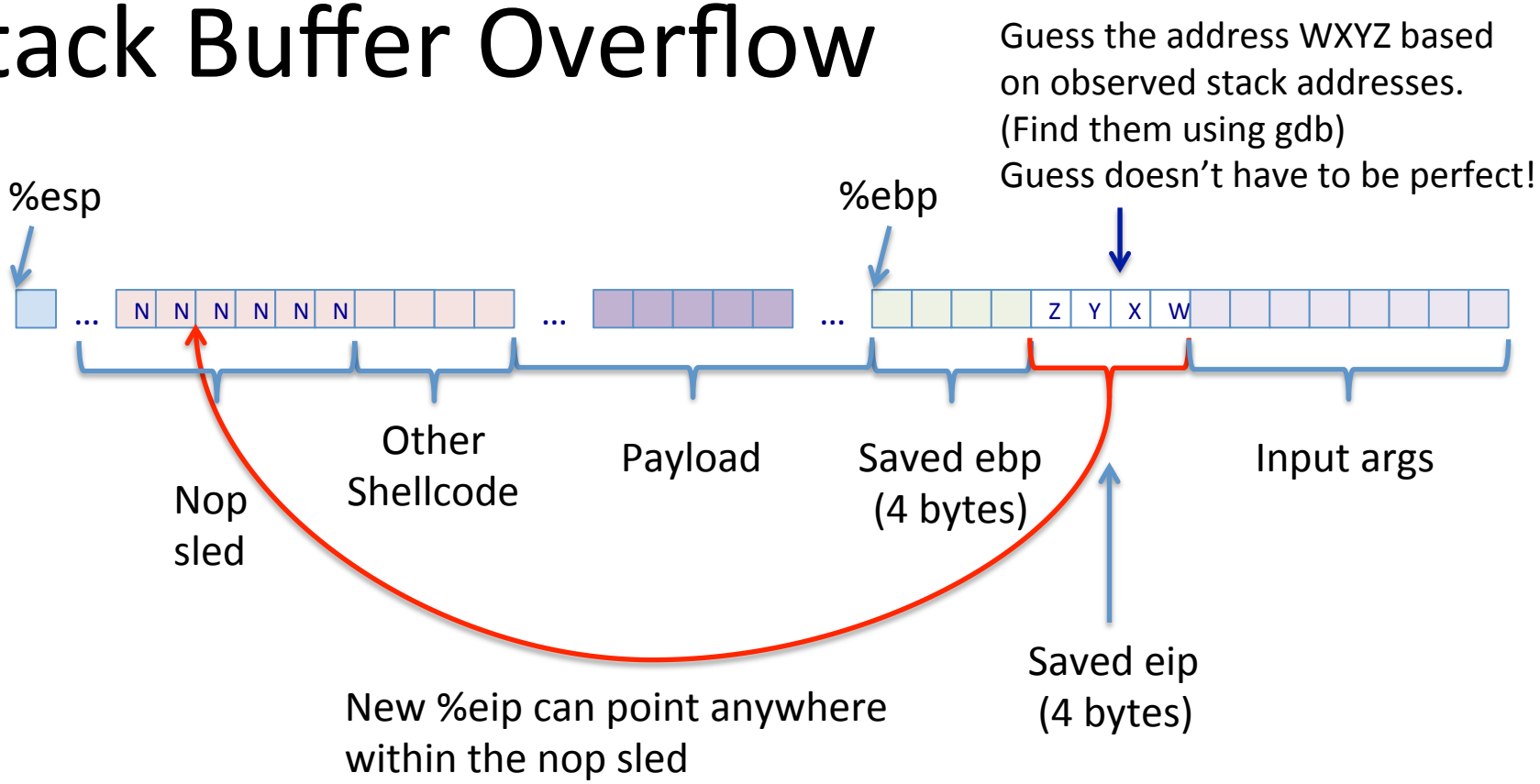
Example 4: Stack Overflow Vulnerability

```
#include <stdio.h>

void function(int a, int b, int c) {
  char buffer[16];

  scanf("%s", buffer);
}

void main() {

  int x;
  x = 0;
  function(1,2,3);
  x = 1;
  printf("  x = %d\n", x);
}
```

scanf takes arbitrary input from stdin and copies it onto the stack starting at buffer.

Now we can take control of %eip from outside the program!

Stack Buffer Overflow



Defense: What can we do?

1. Get programmers to write better code?

2. Get programmers to use a safe language?

3. Modify the compiler?

4. Modify the OS and hardware?

Secure Coding Practices

- Be very careful with memory in C
- Use safer versions of library functions
  - strcpy → strncpy
- Manual code reviews
  - Many eyes make bugs shallow?

Defense: Don't use C/C++ ?

- Advantages
  - Memory is managed automatically in many langs
  - Bounds checking is built in

- Disadvantages
  - Requires complete overhaul of source code
  - Performance ?
  - What language to use instead?

Defense: What can we do?

1. Get programmers to write better code?
   - Good luck with that…
2. Get programmers to use a safe language?
   - Maybe someday, not today…
3. Modify the compiler?
4. Modify the OS and hardware?

Defense: What can we do?

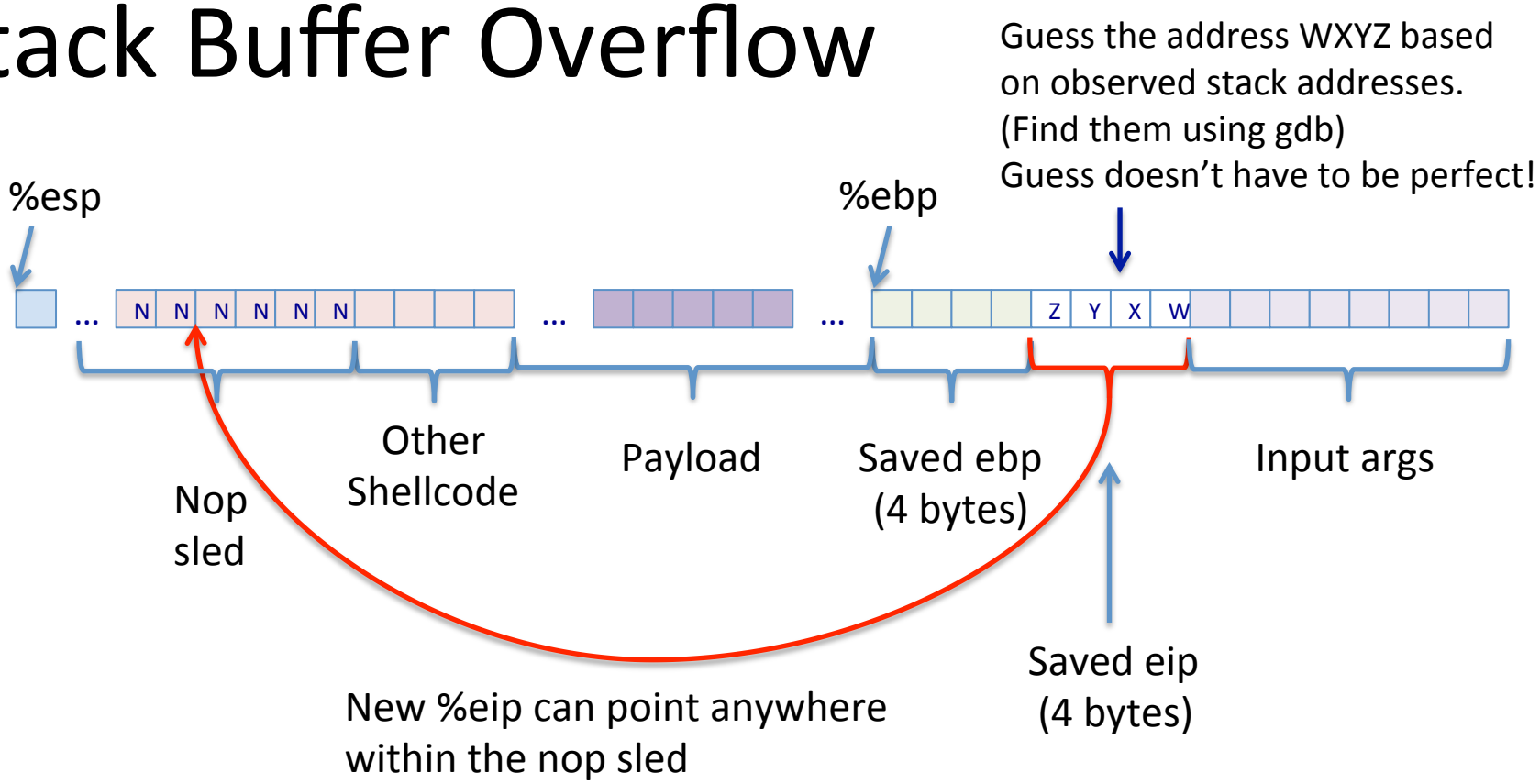
1. Get programmers to write better code?
   - Good luck with that…

2. Get programmers to use a safe language?
   - Maybe someday, not today…

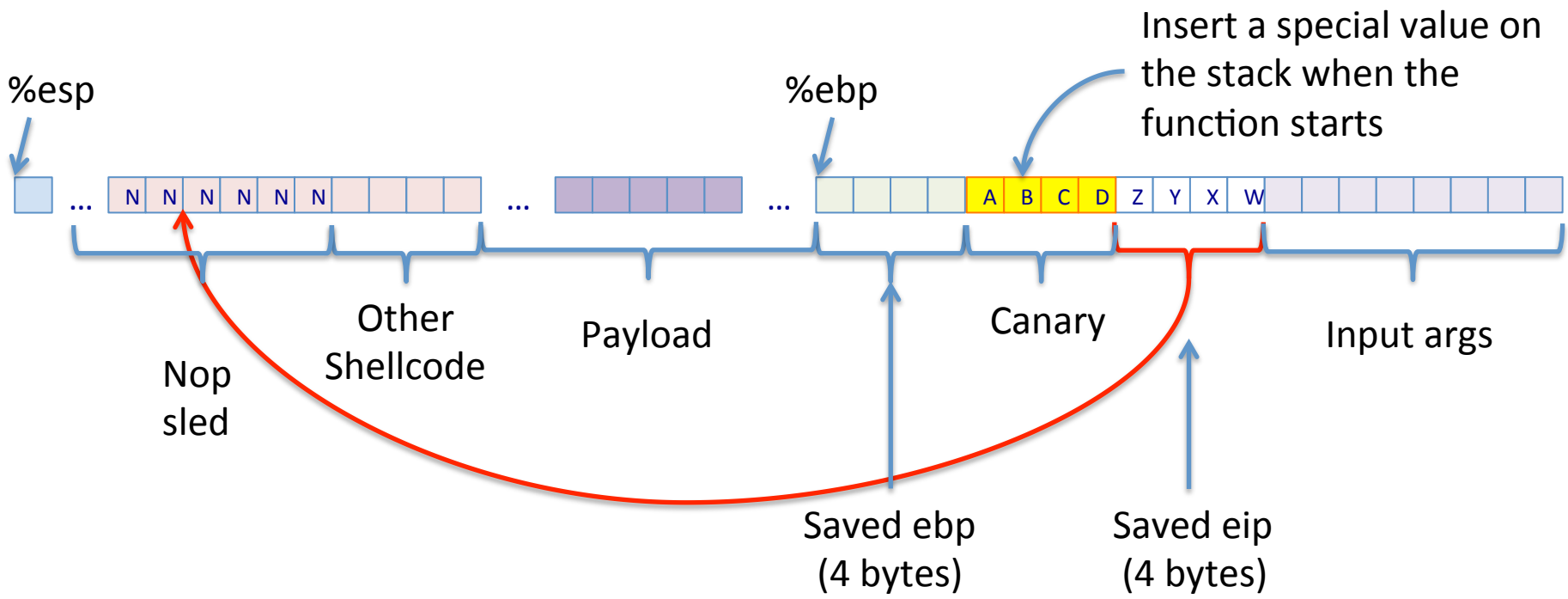
3. Modify the compiler?

4. Modify the OS and hardware?

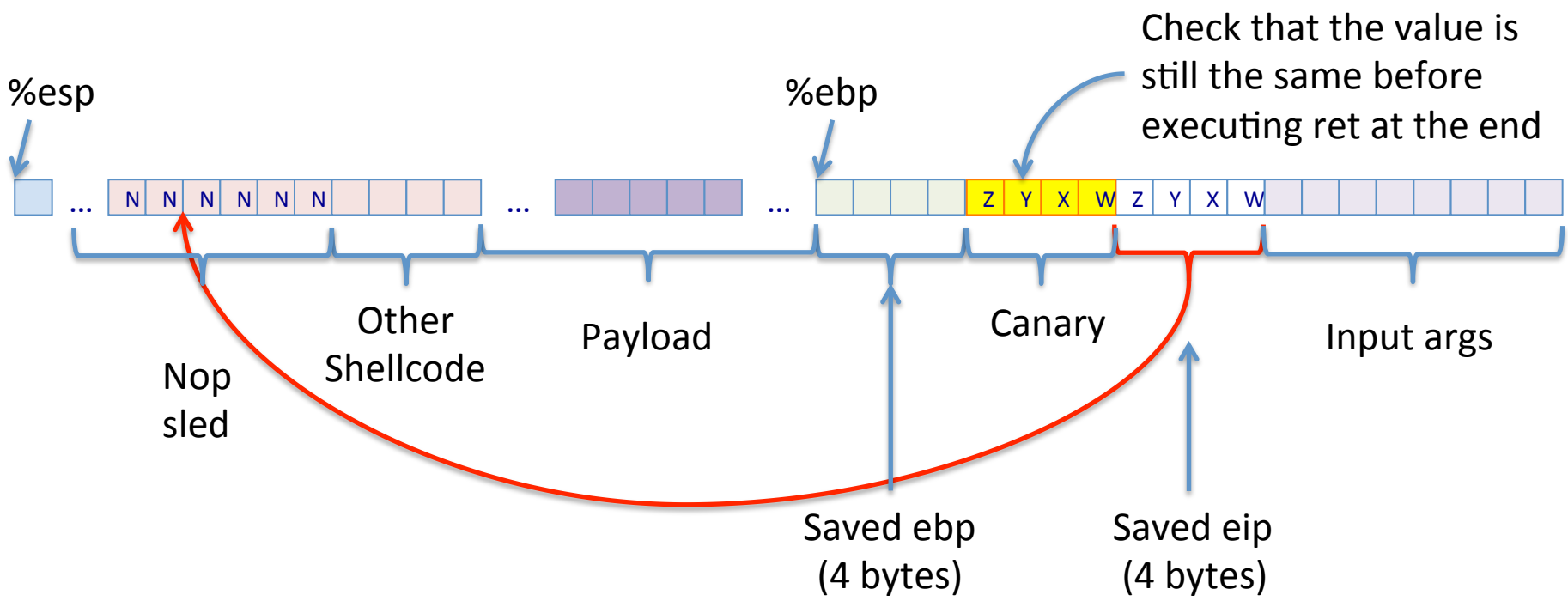
Stack Buffer Overflow



Stack Canaries (1)



Stack Canaries (2)



StackGuard

- StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks
  - by Crispin Cowan et al (including Dave Maier and Jon Walpole, who are now CS faculty at PSU!)
    In Proceedings of USENIX Security Symposium, 1998.
  - Original paper:
  - http://usenix.org/publications/library/proceedings/sec98/full_papers/cowan/cowan.pdf
  - Retrospective:
  - http://courses.cs.washington.edu/courses/cse504/10sp/Slides/lecture3.pptx

StackGuard Code

Without StackGuard

```
function:

    pushl   %ebp
    movl    %esp, %ebp
    …
    leave
    ret
```

With StackGuard

```
function:
    pushl   $0x44434241
    pushl   %ebp
    movl    %esp, %ebp
    …
    leave
    cmpl    $0x44434241, (%esp)
    jne canary_changed
    addl    $4, %esp
    ret
```

Breaking Naïve StackGuard

- Any idea how to do it?

- Hint: Remember Narnia0?

Breaking Naïve Stackguard

- Attack shellcode:

```
nop                             // nop sled
nop
…
nop
pushl       $0x68732f           // payload
pushl       $0x6e69622f
…
.string “ABCD”                  // overwrite canary with expected value
<return address>                // overwrite saved %eip
…
```

Randomized StackGuard

Without StackGuard

```
function:

    pushl   %ebp
    movl    %esp, %ebp
    …
    leave
    ret
```

With StackGuard

```
function:
    pushl   <random>
    pushl   %ebp
    movl    %esp, %ebp
    …
    leave
    cmpl    <random>, (%esp)
    jne canary_changed
    addl    $4, %esp
    ret
```

“Terminator” StackGuard

- Idea: Want to make it really hard for an attacker to “fake” the canary

- Strategy: What’s something that’s really hard to put in the injected code?

Terminator StackGuard

Without StackGuard

```
function:

    pushl   %ebp
    movl    %esp, %ebp
    …
    leave
    ret
```

With Terminator StackGuard

```
function:
    pushl   $0x000aff0d
    pushl   %ebp
    movl    %esp, %ebp
    …
    leave
    cmpl    $0x000aff0d, (%esp)
    jne canary_changed
    addl    $4, %esp
    ret
```

Terminator StackGuard

String terminators!

- 0x00 – NULL
- 0x0a – Line feed "\n"
- 0xff – Negative One (-1)
- 0x0d – Carriage return "\r"

Wi

fun

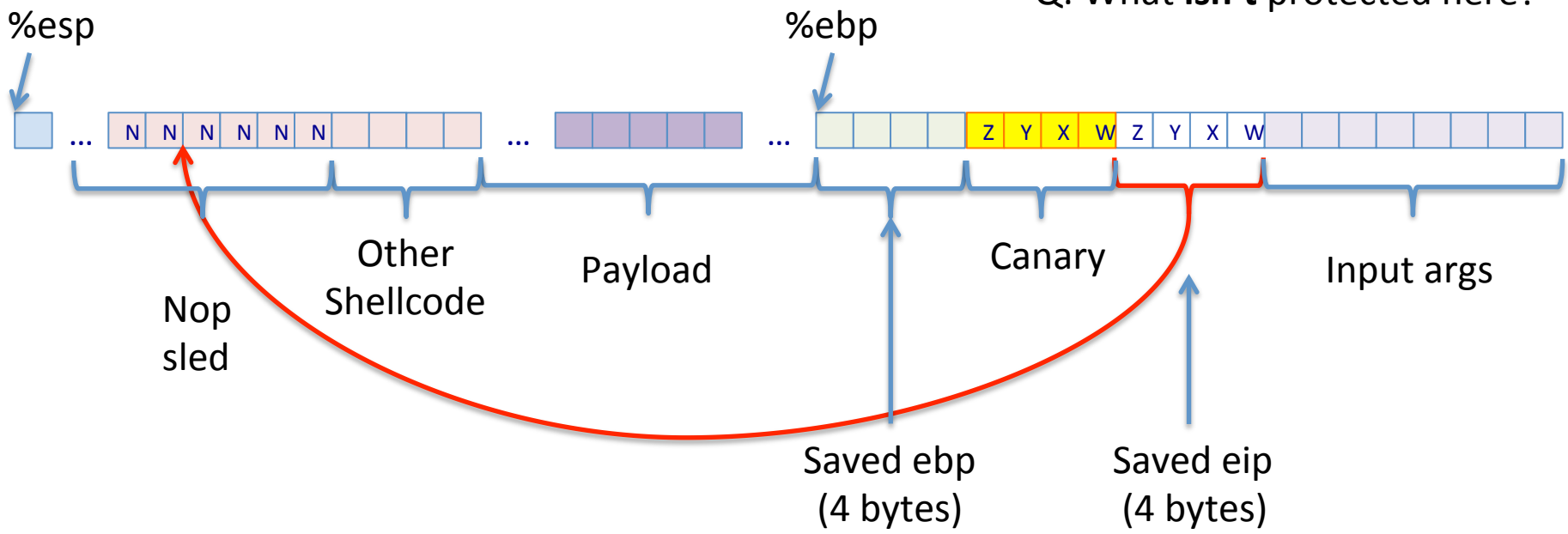
With Terminator StackGuard

function:

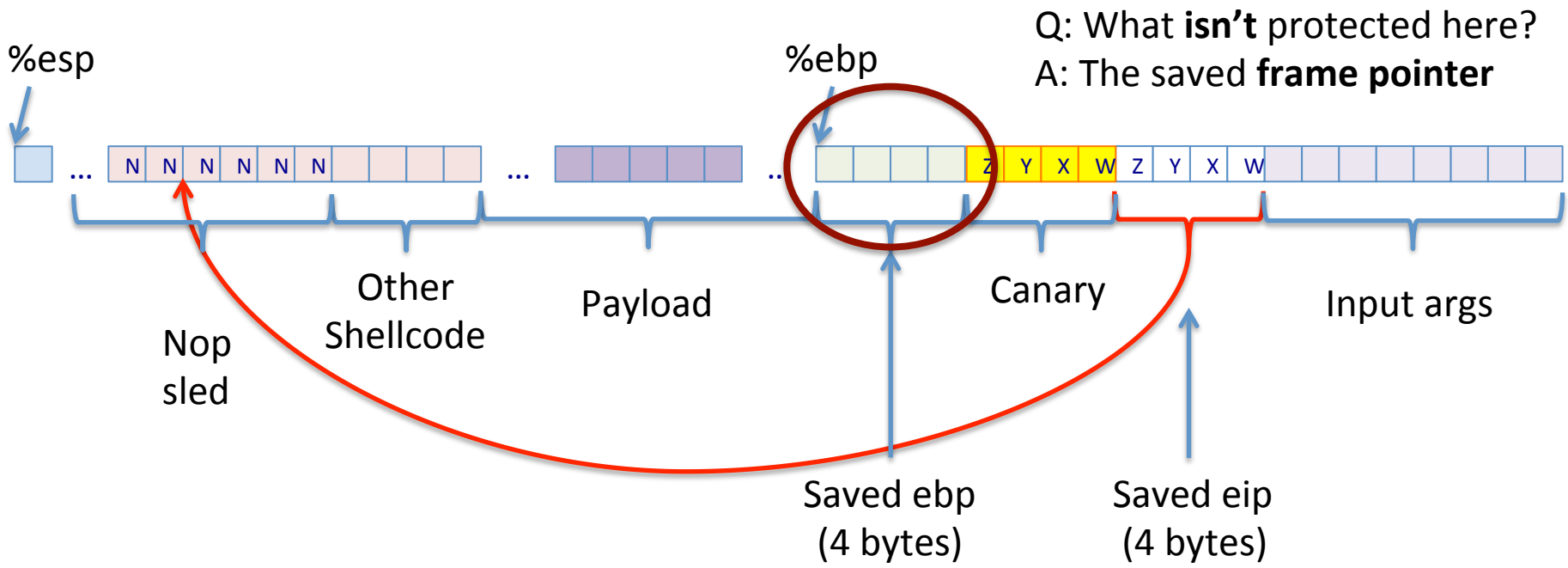
```
pushl   $0x000aff0d
pushl   %ebp
movl    %esp, %ebp
…
leave
cmpl    $0x000aff0d, (%esp)
jne canary_changed
addl    $4, %esp
ret
```

Defeating Stack Canaries

Q: What **isn't** protected here?



Defeating Stack Canaries



Function Calls in x86

- "Leave" instruction
  - **`leave`**

  - Sets %esp to the 32-bit address in %ebp
  - Loads the saved frame pointer from the stack
    - Sets %ebp to the value stored at address %esp
    - Sets %esp to %esp + 4

Attacking StackGuard

- More hacker gymnastics…
  - "If you give 'em an inch…"
- Basic attack idea:
  - Inject a "fake" frame on the stack
  - When the caller returns, it gets its %eip from your fake stack frame

StackGuard / Stack Canaries Summary

- Stack protection isn't perfect
- Is it still worth doing?
  - Microsoft says yes
    - /GS flag for their C compiler
  - GCC and Linux distributions say yes
    - Using ProPolice version of the stack protector
  - For best effect, use it with DEP and ASLR