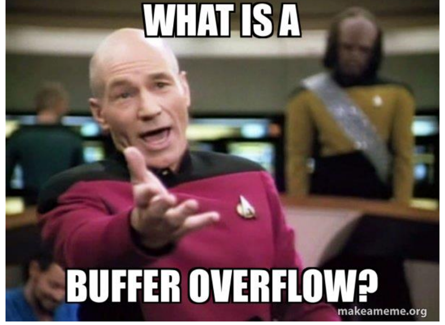
BORING BUFFER OVERFLOWS

CYBER SECURITY!

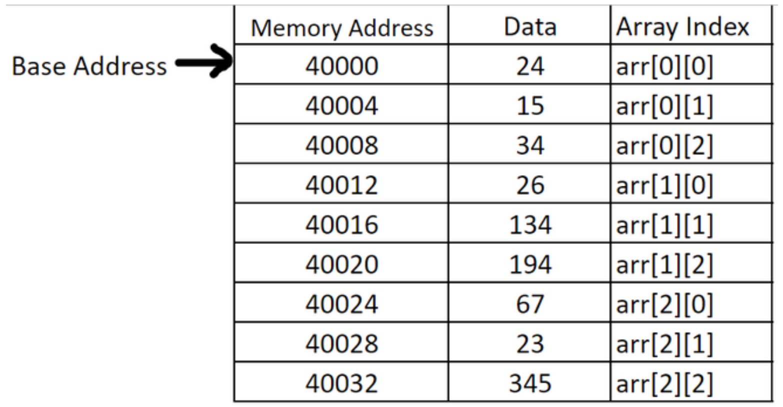
OUTLINE



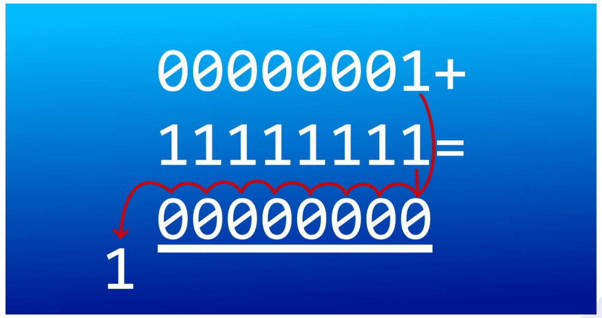
WHAT IS AN OVERFLOW

BASICS

- Buffers tend to be some form of **Array**
  - The structure, and storage location of an array is important in this case!
- This is not referring to an Integer Overflow!
  - Or an Integer Underflow
- What is an Overflow in the case of a Buffer?
  - This is often a **out of bounds** write
    - There are out of bounds reads too…
  - So where does this memory access occur?


Base Address →

Memory Address	Data	Array Index
40000	24	arr[0][0]
40004	15	arr[0][1]
40008	34	arr[0][2]
40012	26	arr[1][0]
40016	134	arr[1][1]
40020	194	arr[1][2]
40024	67	arr[2][0]
40028	23	arr[2][1]
40032	345	arr[2][2]



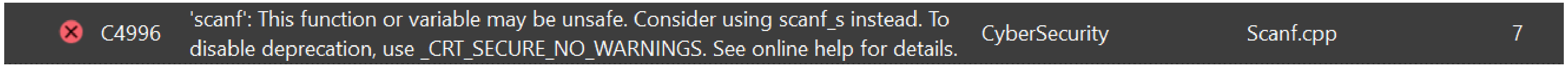
OUTLINE

- What is an Overflow
- Unsafe Instructions
- A Program in Memory
- Basic Exploitation
- Defenses
- Return Oriented Programming (ROP)
- BROP
- Riscy ROP

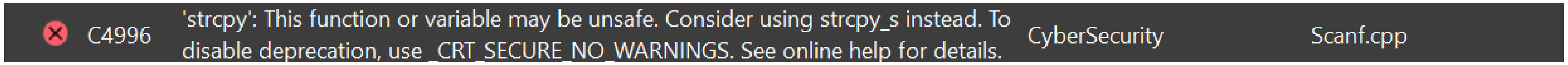
When you are the reason for the company safety video

UNSAFE INSTRUCTIONS

- There are more unsafe instructions then I would know
- We can also use common conventions in an unsafe manner
  - That is accessing memory out of bounds manually



```
C4996   'scanf': This function or variable may be unsafe. Consider using scanf_s instead. To    CyberSecurity    Scanf.cpp    7
        disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.
```



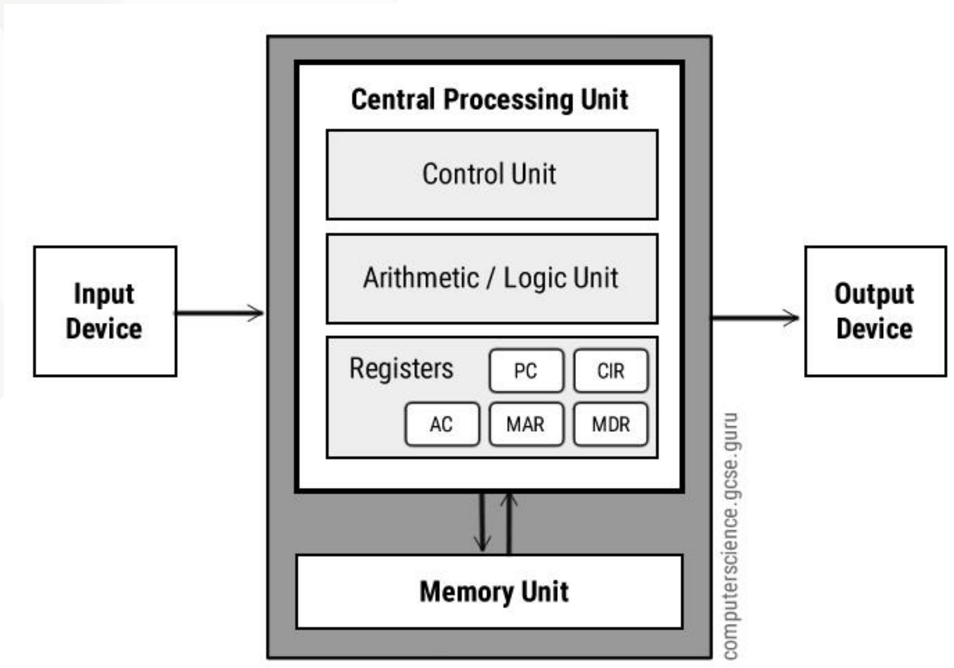
```
C4996   'strcpy': This function or variable may be unsafe. Consider using strcpy_s instead. To    CyberSecurity    Scanf.cpp
        disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.
```

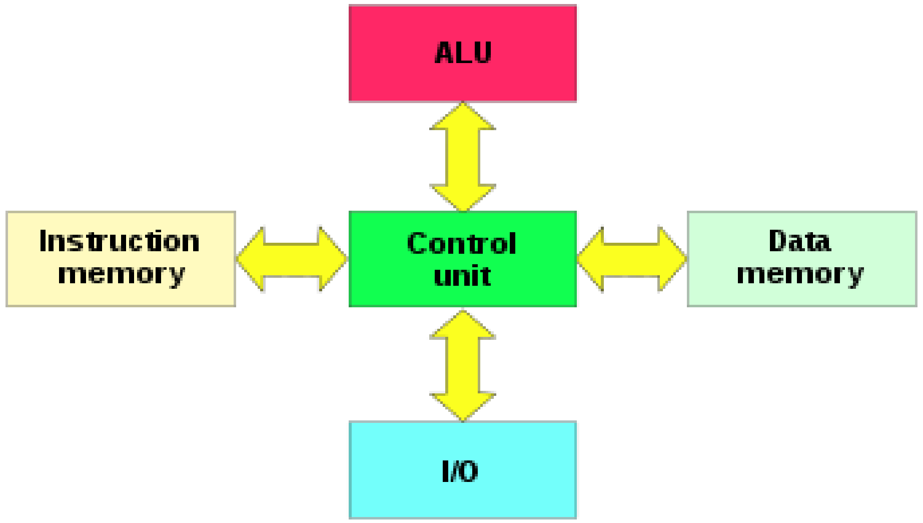
OUTLINE

- What is an Overflow
- Unsafe Instructions
- A Program in Memory
- Basic Exploitation
- Defenses
- Return Oriented Programming (ROP)
- BROP
- Riscy ROP

A PROGRAM IN MEMORY

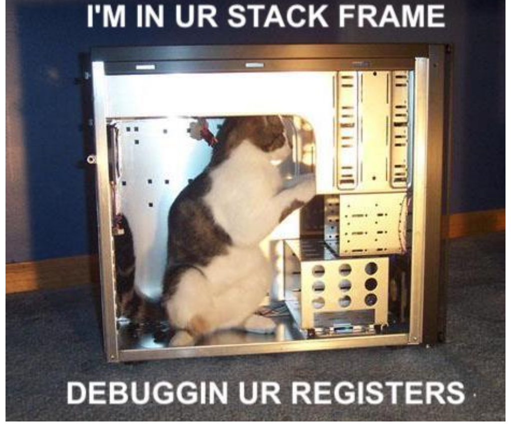
HARDWARE

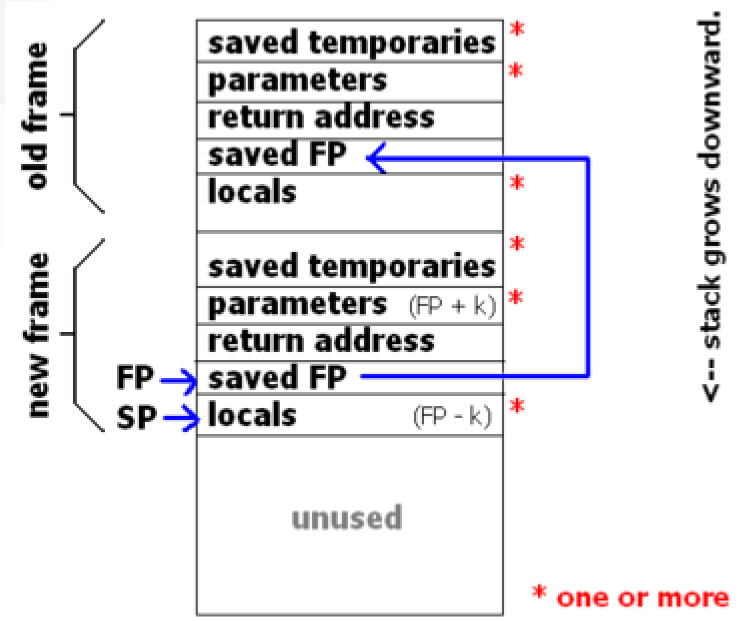


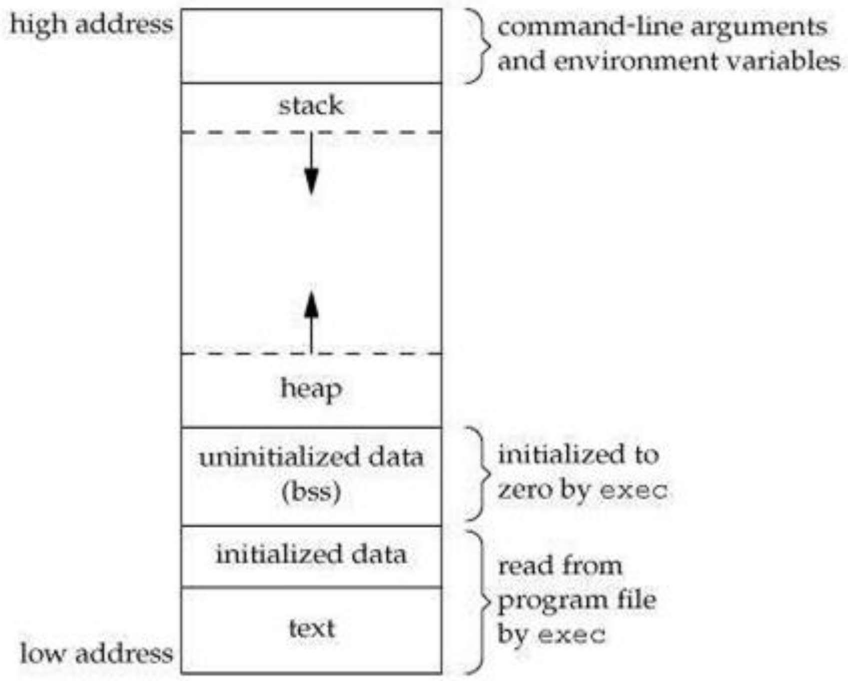


A PROGRAM IN MEMORY

LOADED IN MEMORY





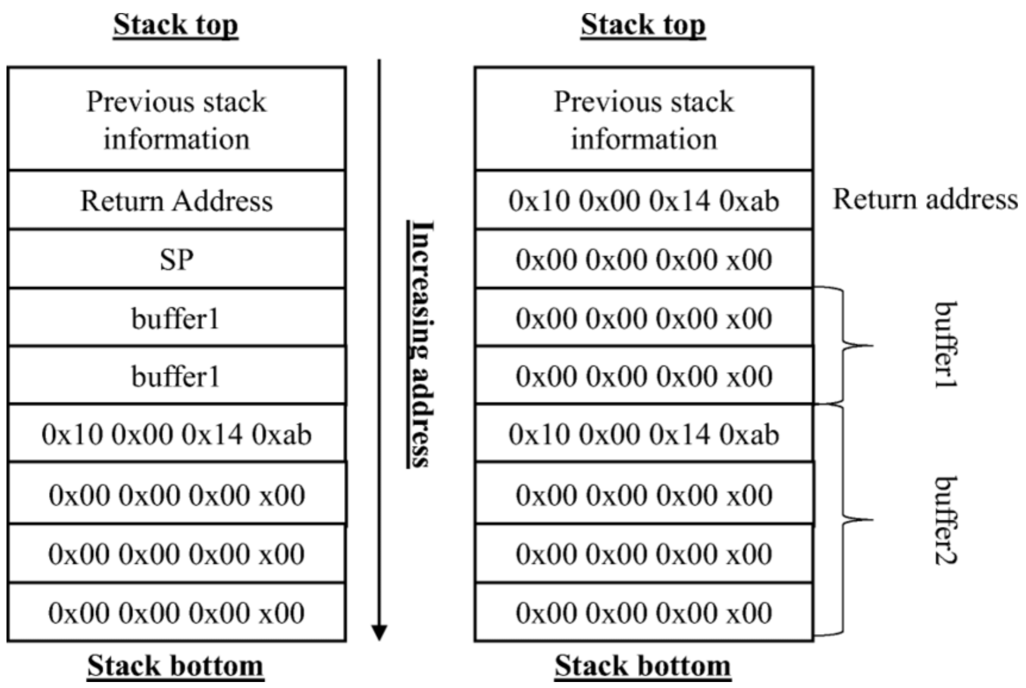

high address
command-line arguments and environment variables
stack
heap
uninitialized data (bss)
initialized to zero by exec
initialized data
text
read from program file by exec
low address

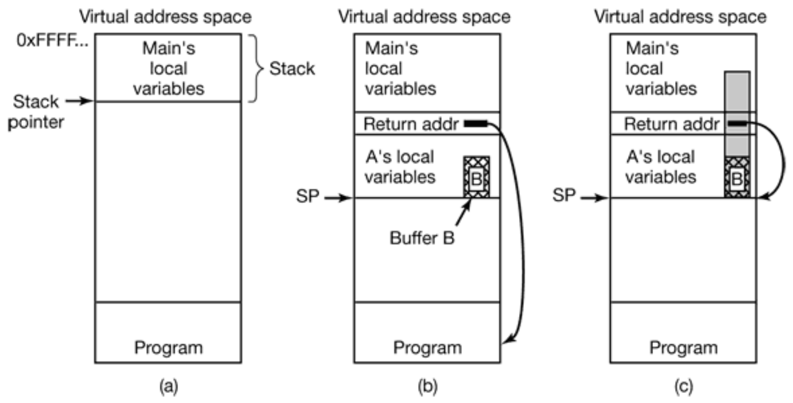
OUTLINE

- What is an Overflow
- Unsafe Instructions
- A Program in Memory
- Basic Exploitation
- Defenses
- Return Oriented Programming (ROP)
- BROP
- Riscy ROP

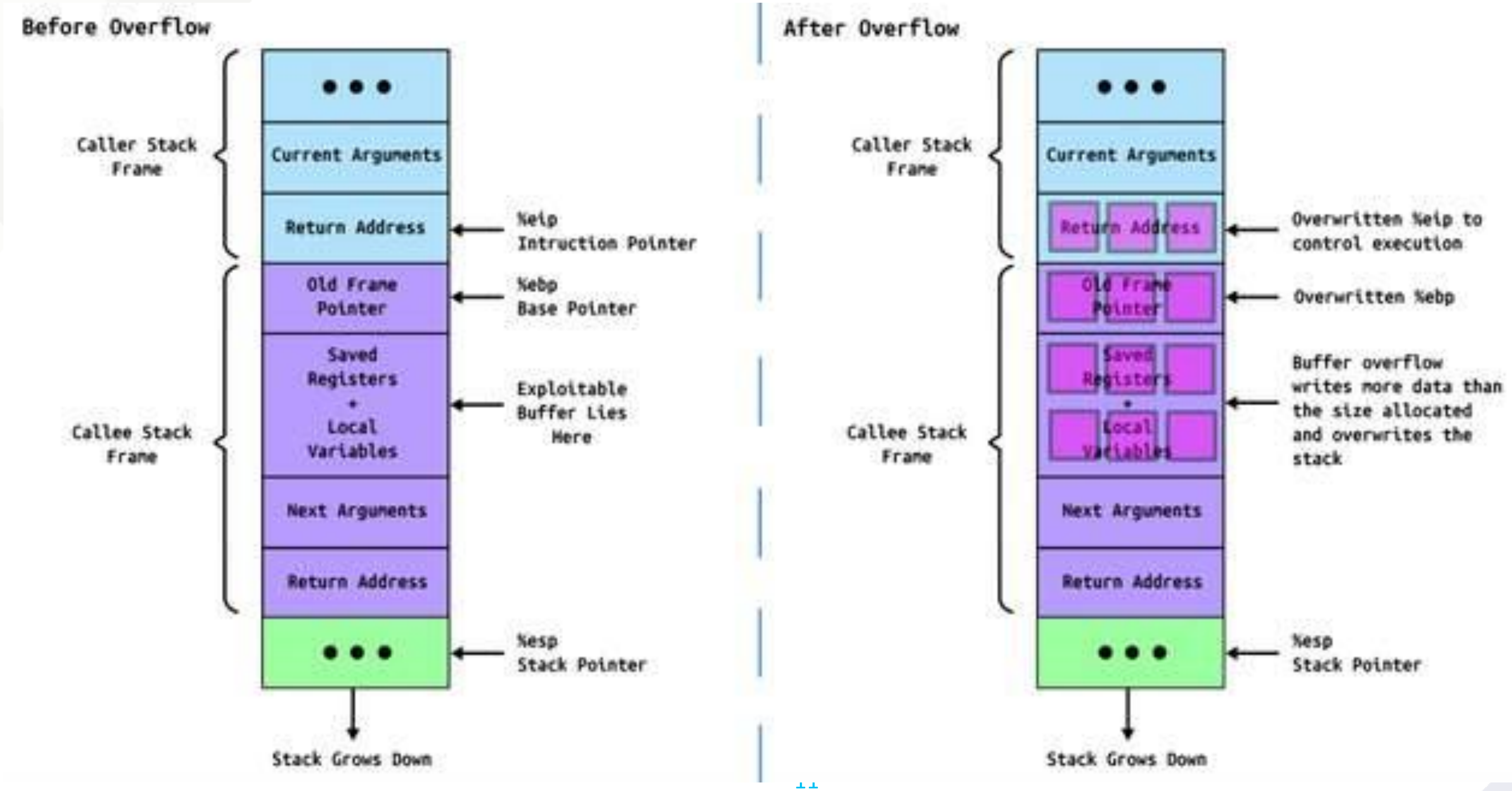
BASIC EXPLOITATION

- Sadly I will just talk about this ☹
- How can the Layout of memory help us?
- Where are local variables stored?
- What is the orientation of a Buffer?
- What do you put in the buffer?
  - Msfvenom
  - ROP





ANOTHER IMAGE



OUTLINE

- What is an Overflow
- Unsafe Instructions
- A Program in Memory
- Basic Exploitation
    - Shell Code Generation
- Defenses
- Return Oriented Programming (ROP)
- BROP
- Riscy ROP

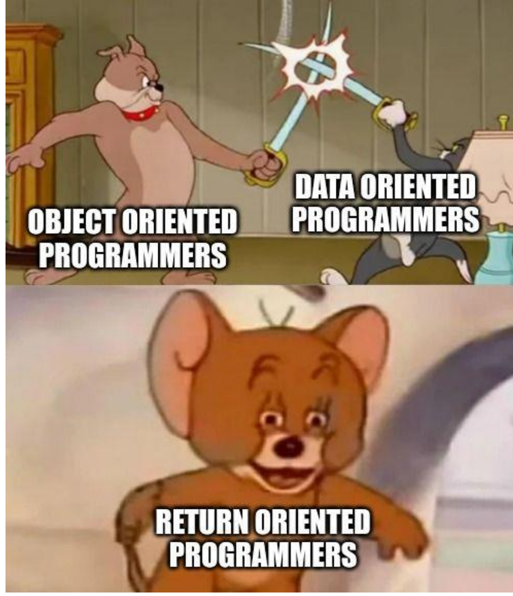
DEFENSES

- ASLR
  - Address space layout randomization
  - Mainly affects the use of library functions
    - Stack based attacks can use NOP slides to negate some of the challenge
    - Libraries need exact addresses to call
- Canaries
  - Terminator
  - Random
  - Random XOR
- NX Memory Pages
  - If you can't Execute Shell Code, What can you do?

OUTLINE

- What is an Overflow
- Unsafe Instructions
- A Program in Memory
- Basic Exploitation
- Defenses
- Return Oriented Programming (ROP)
- BROP
- Riscy ROP

ROP


OBJECT ORIENTED PROGRAMMERS
DATA ORIENTED PROGRAMMERS
RETURN ORIENTED PROGRAMMERS

- This is what we can do when NX Memory is used
  - That is we find *gadgets* in the existing code
- We use those gadgets to force a program to preform the actions we want!
  - Usually this is a system or library call to make the program do some action.
- This often requires access to the source binary/code for analysis
  - There are some methods to bypass this.

OUTLINE

- What is an Overflow
- Unsafe Instructions
- A Program in Memory
- Basic Exploitation
  - Shell Code Generation
- Defenses
- Return Oriented Programming (ROP)
- BROP
- Riscy ROP

BROP

- Blind!
  - No info about target system
- Requires special circumstances

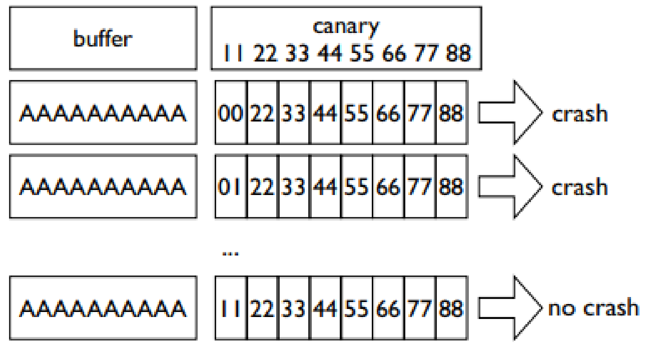


Figure 5. Stack reading. A single byte on the stack is overwritten with guess X. If the service crashes, the wrong value was guessed. Otherwise, the stack is overwritten with the same value and no crash occurs. After at most 256 attempts, the correct value will be guessed. The process is then repeated for subsequent bytes on the stack.

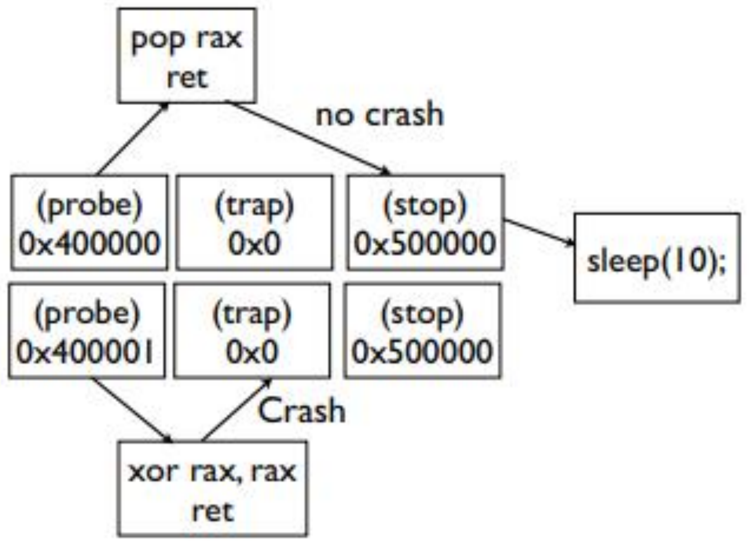


Figure 10. Scanning for pop gadgets. By changing the stack layout, one can fingerprint gadgets that pop words from the stack. For example, if a "trap gadget" is executed rather than popped, the program will crash.

OUTLINE

- What is an Overflow
- Unsafe Instructions
- A Program in Memory
- Basic Exploitation
  - Shell Code Generation
- Defenses
- Return Oriented Programming (ROP)
- BROP
- Riscy ROP

RISCY ROP

- Interesting paper, we are out of time…
  - Uses symbolic execution on a known binary to determine ROP chains in a RISC-V architecture

CONCLUSION

DONE, THERE IS NONE