While You Wait

- Start Kali
- Check still installed. If not, download and install
  - PwnDbg https://github.com/pwndbg/pwndbg
  - Cutter https://cutter.re/
  - pwntools (sudo pip3 install pwntools && pip3 install pwntools)
  - A text editor

Stack Smashing

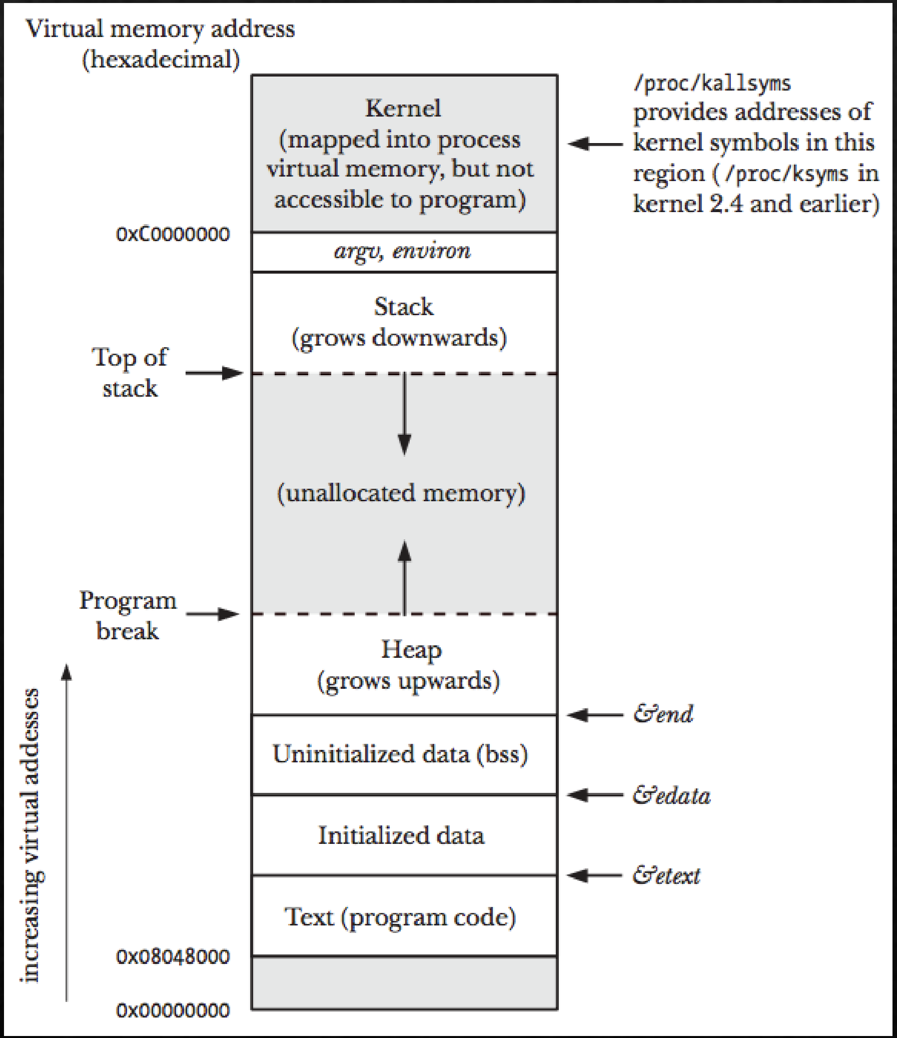
Part 3 of Binary Exploitation

4 Week Plan

- Week 1: Assembly & Shellcoding
  - Writing our own assembly, writing some shellcode. Getting used to debugging tools
- Week 2: Reverse Engineering
  - Learning some basic reversing techniques, getting used to reversing frameworks
- Week 3: Stack smashing
  - Basic program exploitation. How to exploit programs that have little/no protections
- Week 4: Return Oriented Programming
  - Exploiting programs with some modern protections enabled

Quick Recap: The Stack

- In programming, the stack is a logical data structure that obeys LIFO (Last in First Out) principle
- In processes, the stack is a region of memory where data is stored
- Each new function of a process gets it's own stack frame
- The stack starts at a high memory location and grows down


Virtual memory address (hexadecimal)
Kernel (mapped into process virtual memory, but not accessible to program)
/proc/kallsyms provides addresses of kernel symbols in this region (/proc/ksyms in kernel 2.4 and earlier)
0xC0000000
argv, environ
Stack (grows downwards)
Top of stack
(unallocated memory)
Program break
Heap (grows upwards)
&end
Uninitialized data (bss)
&edata
Initialized data
&etext
Text (program code)
0x08048000
0x00000000
increasing virtual addresses

A note on Protections

OS Protection

- Address Space Layout Randomization (ASLR)
  - Addresses on stack is randomised - harder to guess addresses in overflow

Compiler Protections

- Position Independent Execution (PIE)
  - Binary compiled using offsets for code location. Actual locations calculated on runtime
- Stack Canaries (Fortify)
  - Sets values that are monitored. If value changes the program stops
- Non Executable Stack (NX Stack)
  - Remove executable permissions from the stack
- Half/ Full Relocation Read Only (RelRO)
  - Global Offset Table Protections

What is Stack Smashing

- Originally coined by Phrack http://phrack.org/issues/49/14.html
- Stack smashing is the term used for reading malicious shellcode onto the stack and then executing it
- Prerequisites:
  - We have a buffer overflow vulnerability
  - The stack is executable
  - If ASLR is enabled, we need a stack leak
- If following along for this demo you should disable ASLR
  - echo 0 | sudo tee /proc/sys/kernel/randomize_va_space

Disclaimer: I never actually learned stack smashing, I learnt ROP first and so I may not have the answer to your every question, but ask anyway

Drawbacks of Stack Smashing

1. Very system specific
2. Clunky – the stack gets clobbered
3. Relies on no ASLR or a stack leak, even without PIE
4. Relies on NX Stack, or a way to bypass it
5. Inaccurate – nop sleds etc

Demo

Challenges

Challenge 1: Overwrite the return address correctly

Challenge 2: Complete a stack smash with ASLR

Challenge 3: Ret2?

Demo 2

1 method to defeating a canary