CMSC 426
Principles of Computer Security

Lecture 06

Overflow Defenses and Variations

Last Class We Covered

- How the shellcode works
  - In excruciating detail
- Stack buffer overflow exploit demo
  - Partial
  - Setup, failures

Any Questions from Last Time?

Today's Topics

- Defenses against stack overflow attacks
  - ASLR
  - Stack canaries
  - Preventing stack execution

- Buffer overflow variations
  - return-to-libc
  - Return-oriented programming

Stack Overflow Defenses

ASLR

- Address Space Layout Randomization
  - Stack memory region is moved around between executions
- Shellcode must contain an absolute address to jump to
  - We've made use of gdb to get that information out of the executable
- How much does it need to be moved around by?
  - Minimum: enough to prevent vulnerable buffers from having overlap
- Workarounds?
  - Brute forcing, partial EIP overwrites, direct RET overwrite
  - Outside of the scope of this class

Stack Canaries (Stackguard)

- Named after coal mine canaries
- Write a “canary” value to the stack before allocating space for local variables
- Function checks canary value has not changed before exiting
- How is canary value chosen?
  - Must be random/unpredictable… why?
  - Otherwise attacker could simply write a static value in their overflow

Prevent Stack Execution

- Blocks the execution of code located in the stack
- What would this affect?
  - Shellcode can be written to the stack, but will not be executed
- There are certain programs that require placing executable code on the stack (JIT compilation)
  - Special provisions must be made for these to work
- Called DEP (Data Execution Prevention) on Windows

Buffer Overflow Variations

return-to-libc

- Refers to the C standard library (**`libc`**)

- Instead of jumping to shellcode on the stack, jump to useful library functions
  - **`system()`**
  - Calls host environment's command processor with specified command (for example, **`/bin/sh`**)

- No longer requires executable stack

Return-Oriented Programming (ROP)

- Video (with transcript):
  - https://www.rapid7.com/resources/rop-exploit-explained/

- When the stack is no longer executable, jump to other parts of the program that are executable to have those pieces run
  - Piece (“gadget”) must end with a return, so it’ll jump back
  - Chaining enough gadgets together with allow tasks to be performed

- More complicated than just writing the shell code, but still very doable, and difficult to protect against

Daily Security Tidbit

- Shipping companies were hit in August 2018 by ransomware tied to a popular accounting software
- Maersk, in Ukraine, is responsible for about 15% of the world's shipping network
  - Country's network was down for days
  - Resorted to using WhatsApp on private phones to conduct business

Image from https://twitter.com/wimremes/status/1041039369484861440

Information taken from https://www.theregister.co.uk/2017/08/16/notpetya_ransomware_attack_cost_us_300m_says_shipping_giant_maersk/

Image Sources

- Canary:
  - http://pngimg.com/download/20108