CS 465 Computer Security

Kent Seamons
Buffer Overflow Attacks
Buffer Overflow

- The most common security vulnerability
- Root cause
  - Unsafe programming languages (C, C++)
- What areas of process memory are vulnerable to a buffer overflow?
  - Stack
  - Heap
  - Data
Stack Smashing Attack

- A specific type of buffer overflow attack
- How does it work?
  - During a function call, the return address is pushed on the stack
  - An attacker overflows a buffer (local variable)
  - The return address on the stack is overwritten to point to an existing function or to injected code
  - During function return, the instruction pointer is set to the new value stored on the stack, not the original stored value
Questions on Stack Smashing

- How does the stack normally operate during a function call/return?
- Describe how an attacker can inject code on the stack?
- What is a NOP sled and how/why is it used in a stack smashing attack?
  - Intrusion detection and ADMutate
- What are the requirements for the format of the injected code?
Defenses

- Write correct code
  - Avoid vulnerable functions
  - Audit code
- Non-executable buffers
  - Kernel patches make the stack non-executable
- Array Bounds Checking
  - Compile time or run-time checks
  - Use a type-safe language
- Code Pointer Integrity Checking
  - Detect when a pointer is corrupted
  - StackGuard and PointerGuard
StackGuard

- How does it work?
- What is a canary?
  - Terminator canary
  - Random canary
  - XOR canary
Questions

- What are the approaches to defend against a buffer overflow attack? What are the pros/cons of each?
- Is a buffer overflow only useful for a remote attack?
- Making the stack non-executable makes a stack smashing attack impossible? T/F
- If your web server is written in Java, it is not vulnerable to a stack smashing attack? T/F