Exam 1 Review

Stephen Checkoway
Oberlin College
CSCI 343

Format

- Fifty minutes
- Work alone (copying or sharing answers *will* result in failing the course)
- Three questions
  - Multiple choice
  - Short answer
  - Attack construction

Topics

- Threat models
- Memory layout
- Stack
- Buffer overflows
- Constructing shell code
- Integer overflow
- Format string attacks
- Code-reuse attacks
- Defenses
- Malware
- Finding vulnerabilities
- Passwords & authentication
- Access control

Threat models

- Who are the attackers?
- What are their capabilities?
- What is their motivation?
- What is their level of access?

Memory layout

- Stack (including argv and envp)
- Heap
- Libraries
- Code
- Data

Stack

- Grows down (on most architectures)
- Stack pointer
- Frame pointer
- Return address (pushed to stack or stored in a register)
- Function arguments (on stack or in registers)
- Local variables

Buffer overflows

- Overwrite control data or code pointers
  - On the stack
  - On the heap
- Overwriting data used for control

Constructing shell code

- Want to call execve
  - rax: 59
  - rdi: pointer to "/bin/sh"
  - rsi: pointer to NULL-terminated array of pointers to arguments (or just NULL itself on Linux)
  - rdx: pointer to NULL-terminated array of pointers to environment variables (or just NULL itself on Linux)
- Avoiding zero bytes
  - Sometimes you need to, sometimes you don’t

Integer overflow

- Truncations
- Using the same data as both signed and unsigned
- Comparing signed and unsigned
- Arithmetic causing overflow

Format string

- Using %n and %x
- %hhn
- Positional arguments like %25$lx

Code-reuse attacks

- Return-to-libc
  - overwrite return address with address of function in libc
- Chaining return-to-libc calls
  - Inserting stack incrementing code between addresses of functions to call
- Return-oriented programming (ROP)
  - sequence of pointers to short code sequences (usually) ending in return and data those sequences act on
  - stack pointer as the ROP analogue of the instruction pointer
- Constructing higher-level gadgets
  - e.g., loading values from memory, storing values to memory, performing arithmetic
  - principles of control flow: conditionally change the stack pointer

Defenses

- Stack cookies (a.k.a. stack canaries)
  - words written to the stack that get checked prior to function return
- Data execution prevention (DEP)
  - hardware-enforced rules about which pages of memory can be executed as code
  - in short: code is executable but not writable; data is (potentially) writable but not executable
- Address space layout randomization (ASLR)
  - stack, heap, programs, and libraries all loaded at random addresses in memory

Malware

- Infection type
  - virus: replicates by writing new instances of itself in other programs; generally requires humans to run the infected code
  - worm: self-propagating; generally copies itself to new systems to run without human involvement
  - trojan: appears to perform some useful function but actually malicious, e.g., fake video codex
  - rootkit: modifies operating system (or even lower-levels of system code) to hide its existence
- Attack
  - wiper: erases data
  - dropper: downloads and runs additional malware
  - bot: part of a network (botnet) used for coordinated attacks like DDoS
  - ransomware: encrypt files until the victim pays the attacker

Finding vulnerabilities

- White box vs. black box testing
  - white box: you have full visibility into and control over the system; e.g., you can make source code modifications
  - black box: you have no (or limited) visibility/control; e.g., you interact with the system through its defined interfaces
- Manual vs. automated
- Fuzzing: supplying random inputs to a program
- Reverse engineering: disassembling/decompiling binary programs to discover their internal workings

Passwords & authentication

- What makes a good password: length, mostly
- Hashing: Ideally slow cryptographic function that turns arbitrary bytes (e.g., a password) into a fixed-length sequence of bytes
- Salt: random values that go into the password hashing algorithm; stored along with the hash value itself in the password database
- Rainbow tables: precomputed hash chains to speed up dictionary attacks
- Password managers: enables unique passwords used for every website; single point of failure
- One-time passwords: TOTP systems use a shared secret along with a time-dependent number (often number of minutes since registration) to compute a (usually) 6 digit number
- Two-factor authentication: two of something you know (e.g., a password), something you have (e.g., a hardware token), and something you are (e.g., fingerprint)
- U2F: password + device like a security key; challenge/response protocol that cryptographically binds the response to the website's origin (unlike TOTP which can be subject to a MitM attack)
- Passkeys: Similar to U2F but usually without a password

Access control

- Difference between authentication and authorization
  - authentication is about establishing who the **subject** is
  - authorization is about does the **subject** have permission to access an **object** in a particular way
- Mandatory access control (MAC)
  - central authority governs access decisions
  - subjects and objects have security labels = classification + categories
- Discretionary access control (DAC)
  - owners of objects control access to objects rather than a central authority
  - UNIX file permissions as a paradigmatic example
- Role-based access control (RBAC)
  - access to an object is dependent on a subject's role rather than individual identity