Exam 2 Review

Stephen Checkoway
CSCI 343
Spring 2026

Format

- Short answer questions around 3 topics which can cover the whole course but with an emphasis on the second half of the course
- No notes
- Work alone

Potential topics

- Attacks
- Defenses
- Malware
- Finding vulnerabilities
- Passwords & authentication
- Access control
- Web & browser
- Cryptography and secure communications

Threat models

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

Example attacks

- Goto fail
- Shellshock
- Samy worm

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
  - eax: 0xb
  - ebx: pointer to "/bin/sh"
  - ecx: pointer to NULL-terminated array of pointers to arguments
  - edx: pointer to NULL-terminated array of pointers to environment variables
- 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

Format string

- Using %n and %x
- %hhn
- Where do you put shell code?

Code-reuse attacks

- Return-to-libc
- Chaining return-to-libc calls
- Return-oriented programming (ROP)
- Constructing gadgets

Defenses

- Stack cookies (a.k.a. stack canaries)
- Data execution prevention (DEP)
- Address space layout randomization (ASLR)

Malware

- Infection type
  - virus
  - worm
  - trojan
  - etc
- Attack
  - wiper
  - dropper
  - bot
  - ransomware

Finding vulnerabilities

- White box vs. black box
- Manual vs. automated
- Fuzzing
- Reverse engineering

Passwords & authentication

- What makes a good password
  - Length, mostly
- Salt
- Rainbow tables
- Password managers
- One-time passwords
- Two-factor authentication

Access control

- Difference between authentication and authorization
- Mandatory access control (MAC)
- Discretionary access control (DAC)
- Role-based access control (RBAC)

Web & browser

- Threats to the web server
  - Code injection (e.g., SQL injection)
- Threats to the browser
  - Running untrusted code in a sandbox
- Threats to one page from another
  - Same origin policy (SOP)
- Cross-origin attacks
  - CSRF
  - XSS
  - Defenses

Attacks and defenses

- Buffer overflows
- Code reuse attacks
- Stack canaries
- Data execution prevention (DEP)
- Address space layout randomization (ASLR)

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

Web and browser

- Browser's same origin policy for JavaScript
  - Origin = (scheme, host, port)
  - Browser blocks scripts in one origin from reading or modifying documents in other origins (this includes iframes and cross-origin XMLHttpRequests)
- Cross-site request forgery (CSRF)
  - Attacker's site instruct's victim's browser to make a request to an honest site; browser sends cookies along
  - Defenses: hidden tokens embedded in HTML forms, Origin header
- Cross-site scripting (XSS)
  - Reflected, stored

Injections

- Command/shell injection: use attacker-controlled text as part of a shell command run on the server
- SQL injection: use attacker-controlled text as part of SQL query
- SQL injection defense: use prepared statements which separates the form of the query from the parameters of the query
- Blind SQL injection

Message integrity

- Message Authentication Code (MAC)
- Transmit a message along with an authentication tag: M || MAC(key, M)
- Requires a shared key
- Prevents tampering
- HMAC $\mathrm{HMAC}(K, m) = H\Big((K' \oplus opad) \| H\big((K' \oplus ipad) \| m\big)\Big)$

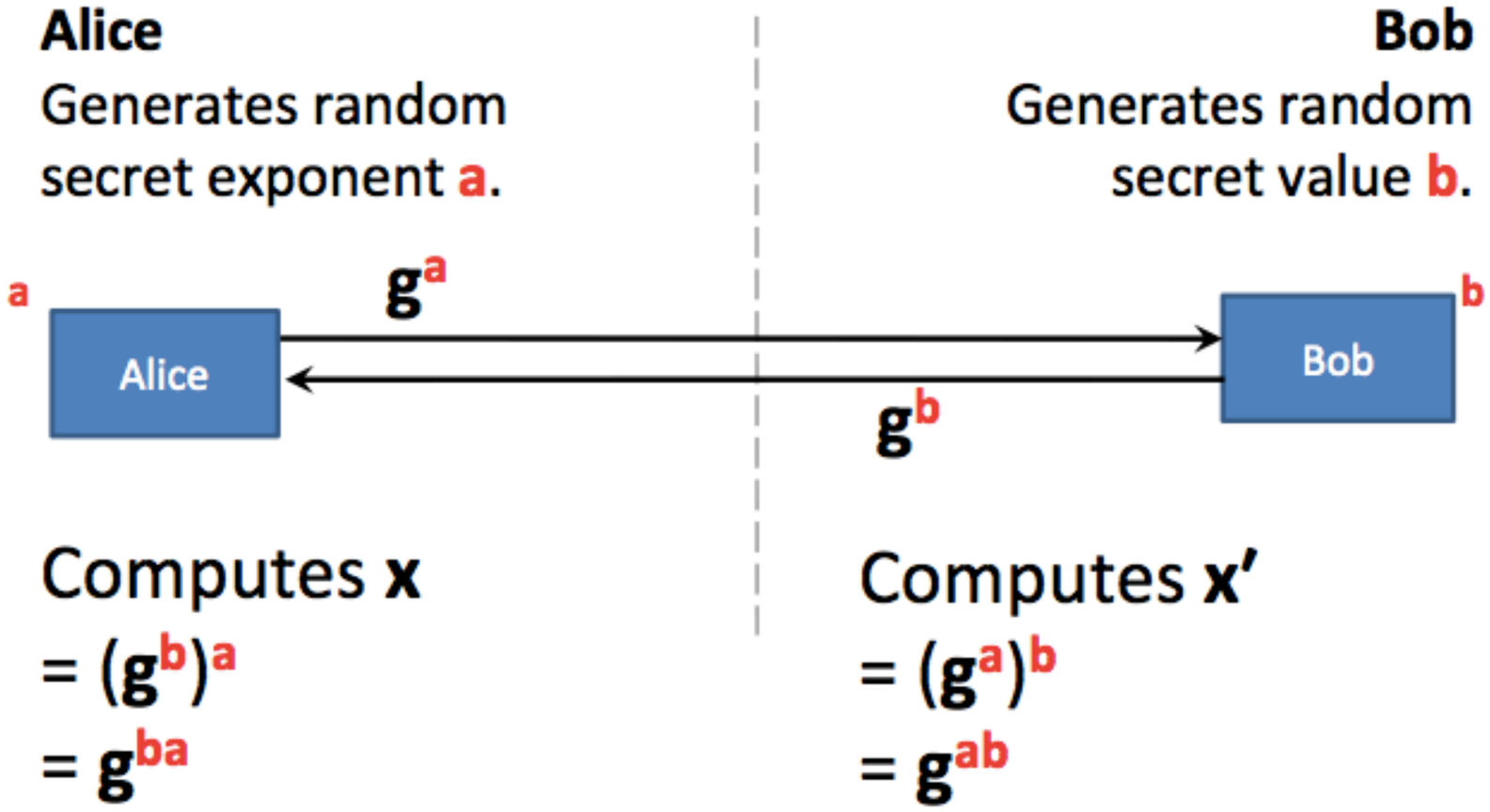
Pseudorandom numbers

- Computationally indistinguishable from true random (desired property)
- Pseudorandom generator: Expands a small number of "true" random bits into a large number of pseudorandom bits
- Useful wherever random numbers are needed (e.g., keys)
- Also useful when unpredictable numbers are needed (e.g., nonces)
- Difference between /dev/random and /dev/urandom

Confidentiality/secrecy

- Kerckhoff's Principles, really just the important one (rephrased): the only thing that should be sensitive in a crypto system is the key
- One-time pad (OTP): long, shared string of random bits; xor with message
  - Must *never* reuse the random string
- Stream cipher: Replace the shared stream of bits in a OTP with a pseudorandom generator with a shared key
  - Must *never* reuse the key
- Block cipher: Process message in fixed-size blocks
- Block cipher modes: ECB, CBC, Counter (turns block cipher into a stream cipher)
- AES (that it exists and is a block cipher, not how to implement it)

Diffie–Hellman key agreement



(Notice that **x** = **x′**)

Can use **k** = hash(**x**) as a shared key.

Digital signatures

- Public-key analogue to MAC
- Sign with private key
- Verify with public key
- RSA: public key (e, N), private key (d, N), N = p*q, e*d = 1 mod (p-1)(q-1)
  - Sign(m) = m^d mod N
  - Verify(m, s) = if s^e mod N == m, then YES else NO
- In real usage, messages are hashed and padded appropriately first

Public-key encryption

- Public-key analogue to symmetric encryption (block/stream ciphers)
- Encrypt with public key
- Decrypt with private key
- RSA: public key (e, N), private key (d, N), N = p*q, e*d = 1 mod (p-1)(q-1)
  - Enc(m) = m^e mod N
  - Dec(c) = c^d mod N
- In real usage, messages are padded first
- Hybrid encryption: Encrypt a symmetric key using the public key, use the symmetric key to encrypt the message (e.g., using AES). Transmit encrypted key and encrypted message

Secure channel construction

- Both sides exchange random values (for replay protection), DH public keys, and supported crypto algorithms
- Derive shared, unidirectional traffic keys (e.g., encryption and MAC keys for Alice -> Bob and Bob -> Alice) from DH shared secret and random values
- Exchange hashes of handshake messages (to prevent an adversary downgrading the connection)
- Protect traffic with traffic keys
- In TLS, server proves identity by signing DH parameters; in IPsec preshared keys are frequently used; in SSH "leap of faith" or "trust on first use" (TOFU) authentication

Certificates and CAs

- Certificates contain public keys and identity information, signed by the issuer
- Certificate authority has root keys that are trusted by browser/OS
- Certificate chain: server cert (signed by intermediate CA cert)* signed by root CA cert
- Browsers verify each cert in the chain until reaching a trusted cert
- Identity validation:
  - Domain validation (DV) cert: prove you control the domain by setting a DNS record or hosting a file with a secret at a well-known location
  - Extended validation (EV) cert: expensive, CA is supposed to really verify identity, doesn't provide any greater cryptographic protection

Anonymity

- Nymity spectrum: verinymity, pseudonymity, linkable anonymity, unlinkable anonymity
- Metadata: data about the communication, not including the content
- VPN: proxies your traffic, but not really designed for privacy/anonymity
- Attackers will just use compromised machines
- Tor
  - Build a circuit through nodes (usually three nodes)
  - Each node in circuit knows previous node and next node
  - No node knows both ends
  - No encryption between exit node and destination server, use HTTPS