Exam 2 Review

Stephen Checkoway
University of Illinois at Chicago
CS 487 – Fall 2017

Format

- Two parts
  - Part I:
    - Fifty minutes, in-class
    - Short answer questions
    - Probably an attack problem
  - Part II:
    - Ninety minutes, online
    - Twenty multiple choice
- No notes
- Work alone (copying or sharing answers *will* result in failing the course)

Topics from first half

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

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

Topics from second half

- Message Integrity
- Pseudorandom numbers
- Confidentiality/secrecy
- Diffie–Hellman key agreement
- Digital signatures
- Public-key encryption
- Secure channel construction (TLS/SSH/IPsec)
- Certificates and Certificate Authorities
- Cryptocurrencies
- Anonymity

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 $\text{HMAC}(K, m) = H\Big((K' \oplus opad) \| H\big((K' \oplus ipad) \| m\big)\Big)$

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

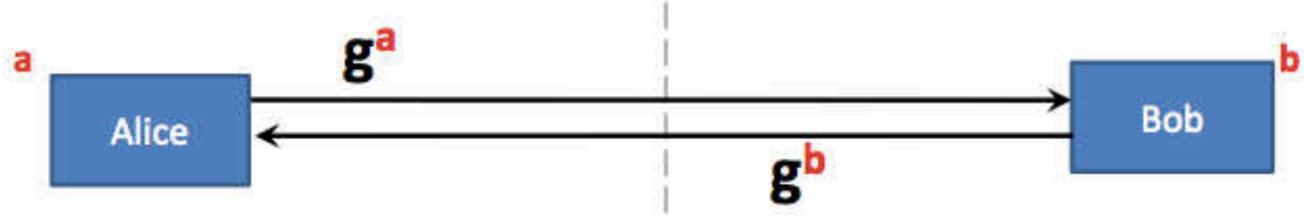
Confidentiality/secrecy

- 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

Alice
Generates random secret exponent **a**.

Bob
Generates random secret value **b**.



Computes **x**
$= (g^b)^a$
$= g^{ba}$

Computes **x′**
$= (g^a)^b$
$= g^{ab}$

(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)
  - $Enc(m) = m^e \bmod N$
  - $Dec(c) = c^d \bmod N$
- 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

Cryptocurrencies

- Pseudonymous digital currency
- Distributed transaction ledger
- Block chain: Each block links to the transactions in the block as well as to the previous block in the chain by hashing
- Miners mine blocks by looking for a nonce such that
  H(previous_block || transactions || nonce) = 0x00..0xx.x
  that is, it has the appropriate number of leading zeros
- Mining difficulty increases over time
- Longest chain is authoritative; orphan blocks

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