CS161 Midterm 2 Review

Midterm 2: April 29, 18:30-20:00
Same room as lecture
Overview

• Static analysis and program verification
• Security architecture and principles
• Web security
• Crypto
• Network security
Static Analysis

• Syntactic analysis
  – Does not interpret the statements

• Semantic analysis
  – Interpret statements
Syntactic Analysis

- **Specify Error Patterns**
- **Parse Program**
- **Detect Patterns**
- **Prune False Alarms**

**Error patterns**: Heuristically observed common error patterns in practice

**Parsing**: generates data structure used for error detection

**Detection**: match pattern against program representation

**Pruning**: Used to eliminate common false alarms
Semantic Analysis

• Sign analysis
• Zero propagation
• Interval analysis
• Product analysis
  – Disjunctive refinement
Architecture of a Static Analyzer

The behavior of a program can be approximated by separately approximating variable values, statements and control flow.
Quiz: Sign Analysis Transformers

Which of the following is the right transformer for $x=x-1$?

Answer: C
Program Verification

• E.g., how to prove a program free of buffer overflows?
• Precondition
  – An assertion that must hold at input to $f(\ )$
• Postcondition
  – An assertion that holds when $f(\ )$ returns
• Loop invariant
  – An assertion that is true at entrance to a loop, on any path through the code
  – Prove by induction
Security Architecture and Principles

• Access control
  – ACL/Capability
  – Role-based access control
  – Reference monitor
• Principle of least privilege
• Defense in depth
• Consider human factors
• Separation of responsibility
• Don’t rely on security through obscurity
• Fail safe
• Design security in from the start
• Ensure complete mediation
• Detect if you cannot prevent
Malware

• Virus
  – Propagation requires human intervention
• Polymorphic virus
  – Creates a random encryption of the virus body
• Metamorphic virus
  – Mutate the virus body, too
  – Code obfuscation/mutation
Malware

• Worm
  – Propagates automatically without human intervention

• Botnet
  – A network of programs capable of acting on instructions
    • Bot master and bots
  – Used for spamming, click fraud, and DDoS
Web Security

- Same-origin Policy (SOP)
- Command injection
- SQL injection
- Cross-site Scripting (XSS)
- Cross-site Request Forgery (CSRF)
- Session hijacking
Same-origin Policy (SOP) (for javascript and DOM)

Two documents have the same origin if:

- **Same protocol**: (https, http, ftp, etc)
- **Same domain**: (safebank.com, etc)
- **Same port**: (80, 23, 8080, etc)

Results of same-origin checks against “http://cards.safebank.com/c1/info.html”

**Same origin:**
- “http://cards.safebank.com/”

**Different origin:**
- “http://www.cards.safebank.com”
- “http://catville.com”
- “https://cards.safebank.com”
- “http://cards.safebank:8080”
Command Injection

• Inject malicious code into data
  – Malicious code in the parameters of URLs

• Defenses
  – Input validation
    • Backlisting
    • Whitelisting
  – Input escaping
  – Use of less powerful APIs
SQL Injection

• Caused when attacker controlled data interpreted as a (SQL) command
  – Goal is to manipulate a SQL database

• Defenses
  – Input validation
    • Backlisting
    • Whitelisting
  – Input escaping
  – Use of less powerful APIs
    • Prepared statements
Cross-site Scripting (XSS)

• Vulnerability in web application that enables attackers to inject client-side scripts into web pages viewed by other users.

• Three types
  – Persistent or stored
    • Malicious code is stored at the server
  – Reflected
    • Malicious code is reflected back by the server
  – DOM based
    • The vulnerability is in the client side code
Cross-site Request Forgery (CSRF)

• An attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated.
• Caused because browser automatically includes authorization credentials such as cookies.
• Defenses
  – Origin headers
  – Nonces
Session Hijacking

• Get the user’s session token and act on behalf of the user

• How to get session tokens?
  – Session token theft
    • Eavesdropping network communication, e.g., http
    • XSS
  – Session fixation
    • Attacker sets the user’s session token
    • Defense: issue a new session token when logging in
Crypto

- Symmetric-key crypto
  - Blocker cipher
  - Modes of operation
  - HMAC
- Public-key crypto
  - Encryption
  - Digital signature
  - Digital certificate
  - Diffie-Hellman key exchange
  - Shamir secret sharing
  - Secure multi-party computation
  - Zero-knowledge proof
Block Cipher

- Encrypt/Decrypt messages in fixed size blocks using the same secret key
  - k-bit secret key
  - n-bit plaintext/ciphertext

Examples: DES, AES
Modes of Operation

• Electronic Code Book (ECB)
  – Blocks are encrypted independently

• Cipher Block Chaining (CBC)
  – Encryption of one block depends on the ciphertext of the previous block

• Counter (CTR)
  – Encrypts counter value
Cryptographic Hash Functions

• Preimage resistance
  – Given $h$, intractable to find $y$ such that $H(y)=h$

• Second preimage resistance
  – Given $x$, intractable to find $y\neq x$ such that $H(y)=H(x)$

• Collision resistance
  – Intractable to find $x, y$ such that $y\neq x$ and $H(y)=H(x)$
Message Integrity: MACs

- Goal: provide message integrity. No confidentiality.
  - ex: Protecting public binaries on disk.

Alice $k$ 

Message $m$ tag

Bob $k$

Generate tag: $tag \leftarrow S(k, m)$

Verify tag: $V(k, m, tag) = \text{`yes' }$

(note: non-keyed checksum (CRC) is an insecure MAC !!)
HMAC (Hash-MAC)

Most widely used MAC on the Internet.

H: hash function.
example: SHA-256 ; output is 256 bits

Building a MAC out of a hash function:

opad, ipad: fixed strings

– Standardized method: HMAC

\[ S( k, m ) = H( k \oplus \text{opad} , H( k \oplus \text{ipad} , m ) ) \]
Public Key Encryption

**Def:** a public-key encryption system is a triple of algs. \( (G, E, D) \)

- \( G() \): randomized alg. outputs a key pair \((pk, sk)\)
- \( E(pk, m) \): randomized alg. that takes \( m \in M \) and outputs \( c \in C \)
- \( D(sk, c) \): det. alg. that takes \( c \in C \) and outputs \( m \in M \) or \( \perp \)

Consistency: \( \forall (pk, sk) \) output by \( G \):

\[
\forall m \in M: \quad D(sk, E(pk, m)) = m
\]
Building Block: Trapdoor Functions (TDF)

**Def:** a trapdoor function over $X$ is a triple of efficient algs. $(G, F, F^{-1})$

- $G()$: randomized alg. outputs a key pair $(pk, sk)$
- $F(pk, \cdot)$: deterministic alg. that defines a function $X \hookrightarrow Y$
- $F^{-1}(sk, \cdot)$: defines a function $Y \hookrightarrow X$ that inverts $F(pk, \cdot)$

\[
\text{for all } x \text{ in } X:\ F^{-1}(sk, F(pk, x)) = x
\]

**Security:** $(G, F, F^{-1})$ is secure if $F(pk, \cdot)$ is a “one-way” function:

given $F(pk, x)$ and $pk$ it is difficult to find $x$
Example TDF: RSA

- **alg. G()**: generate two equal length primes \( p, q \)
  
  set \( N \leftarrow p \cdot q \) \quad (3072 \text{ bits } \approx 925 \text{ digits})

  set \( e \leftarrow 2^{16}+1 = 65537; \quad d \leftarrow e^{-1} \pmod{\varphi(N)} \)

  \( pk = (N, e) \quad ; \quad sk = (N, d) \)

- **RSA\((pk, x)\)**: \( x \rightarrow (x^e \mod N) \)
  
  Inverting this function is believed to be as hard as factoring \( N \)

- **RSA\(^{-1}\)(sk, y)**: \( y \rightarrow (y^d \mod N) \)
Public Key Encryption with a TDF

G(): generate pk and sk

E(pk, m):
- choose random \( x \in \text{domain}(F) \) and set \( k \leftarrow H(x) \)
- \( c_0 \leftarrow F(pk, x) \), \( c_1 \leftarrow E(k, m) \) \( (E: \text{symm. cipher}) \)
- send \( c = (c_0, c_1) \)

D(sk, c=(c_0, c_1)):
\( x \leftarrow F^{-1}(sk, c_0) \), \( k \leftarrow H(x) \), \( m \leftarrow D(k, c_1) \)
Digital signatures

Example: signatures from trapdoor functions (e.g. RSA)

\[
\begin{align*}
sign(\ sk, \ m) & \ := \ F^{-1}(sk, \ H(m)) \\
\text{Verify}(pk, \ m, \ sig) & \ := \ \text{accept if} \quad F(pk, \ sig) = H(m) \\
& \quad \text{reject otherwise}
\end{align*}
\]
Digital Certificates

CA signs a user’s public key. The certificate includes both the public key and the CA’s signature.

Browser

Alice

PK_{CA}

verify cert

Server Bob

generate (sk,pk)

PK_{CA}

pk and proof “I am Bob”

issue Cert with SK_{CA}:

Certificate Authority (CA)

check proof

SK_{CA}

Bob’s key is pk

Bob’s key is pk
Diffie-Hellman Key Exchange

Alice: Prime $p$, number $g$, $0 < g < p$

Bob

$g^A \mod p$

$g^B \mod p$

$(g^A)^B \mod p$

$(g^B)^A \mod p$
SSL Session Setup (Simplified)

Client

- Pick random 48 byte PreK

Server

- RSA secret key

ClientHello: nonce\textsubscript{C}

ServerHello: cert, nonce\textsubscript{S}

ClientKeyExchange: \( c \leftarrow E(pk, \text{PreK}) \)

\( c \) decrypt to get PreK

session-keys \( \leftarrow \text{PRF( PreK, nonce}\textsubscript{C}, \text{nonce}\textsubscript{S} ) \)

Finished

Finished
Attacks to Passwords

• Online guessing attacks
• Social engineering and phishing
• Eavesdropping
• Client-side malware
• Server compromise
Shamir Secret Sharing

• Make a random polynomial curve $f(x)$ of degree $q-1$:
• Secret is $f(0)$
• Distribute $n$ points
• $q$ points determine the curve
• $q-1$ or less points do not determine the curve
• All calculations are mod $p$, where $p$ is a prime
More Crypto Tools

• Secure Multi-party Computation (SMC)
  – Suppose $n$ participants, each has a private data point $p_i$. SMC computes the value of a public function $F$ on the $n$ data points such that each participant does not learn others’ private data except what the result reveals.
  – Anything that can be done with a trusted authority can also be done without

• Zero-knowledge proof
  – Prove something without revealing the proof
An Example: ZKP for Discrete Logs

• Suppose a prover has an identity $x$, which is a number satisfying $B=A^x \pmod{p}$. $(A,B,p)$ is publicly available. The prover wants to prove he/she has $x$ but does not want to reveal $x$ to the verifier.
  – Prover chooses a random number $0 \leq r < p-1$ and sends the verifier $h=A^r \pmod{p}$
  – Verifier sends back a random bit $b$
  – Prover sends $s=(r+bx) \pmod{(p-1)}$ to verifier
  – Verifier computes $A^s \pmod{p}$ which should equal $hB^b \pmod{p}$
Network Protocol Stack

1. Physical
2. Link
3. (Inter)Network
4. Transport
5. Application
On-path vs. off-path

Topology with 4 nodes

A

B

C

D

Data flow

B is on path

D is off path
Threats to Link/Physical Layers

• Eavesdropping
  – Wireshark to collect public WiFi packets

• Disruption
  – Jamming signals
  – Routers & switches can simply “drop” traffic

• Spoofing
  – Create messages attackers like
Threats to IP Layer

- Can set arbitrary source address
- Can set arbitrary destination address
Threats to TCP

• An on-path attacker who can observe your TCP connection,
  – Forcefully terminate by forging RST packet.
  – TCP hijacking/spoofing: spoof data into either direction by forging packets
    – The key is to spoof the sequence number
Establishing a TCP Connection: 3-Way Handshaking

Different starting sequence numbers in each direction

Client (initiator)  Server

connect()  listen()

SYN, SeqNum = x

SYN + ACK, SeqNum = y, Ack = x + 1

ACK, Ack = y + 1

Different starting sequence numbers in each direction
DNS Blind Spoofing (Kaminsky 2008)

- Attacker spoofs the targeted user to generate a series of different DNS name lookups
  
  ```html
  <img src="http://random1.google.com" ...
  <img src="http://random2.google.com" ...
  <img src="http://random3.google.com" ...
  ...
  <img src="http://randomN.google.com" ...>
  ```

- Attacker sends many DNS replies with random identification IDs to the targeted user

- Modern DNS implementation: also include randomized SRC port as ID in the UDP packet
Denial-of-Service (DoS)

• Denial-of-Service (DoS)/DDoS
  – SYN flooding: send many SYNs to start 3-way TCP handshake with the server
    • Defense: SYN Cookies (only works for spoofed source IPs)
  – DNS amplification. Send forged DNS lookups with the targeted server’s IP as source address.
Firewall

• Firewall enforces an (access control) policy:
  – Who is allowed to talk to whom, accessing what service?
• Distinguish between inbound & outbound connections
  – **Inbound**: attempts by external users to connect to services on internal machines
  – **Outbound**: internal users to external services
• Default policies
  – Default allow
  – Default deny
  – Generally we use default deny
• Stateful Packet Filter
  – Checks each packet against security rules and decides to forward or drop it
  – Example: Permits TCP connection that is initiated by host 4.5.5.4