Web

Same Origin Policy

- enforced by browser
- Protocol, domain, port

Cookies & CSRF

- Cookies stored in the browser (not server), cookie jar
- Name-value pair
  - Name, value, domain, path, secure (Https), HttpOnly (JS cannot use cookie), expires

Cookie Policy

- Enforced by browser
- Set
  - X can set cookie with Y if 1) X ends in Y, 2) X <= Y on hierarchy, 3) X is more specific than Y, 4) Y is not TLD
- Send
  - Cookie domain & path are suffix of server

CSRF (Cross-Site Request Forgery)

- Attack cookie-based authentication
- 1. User auth to server, receive cookie w/ session token
- 2. Attacker trick user to make a request to server
  - Trick GET: 1) click, 2) inject HTML
  - Trick POST: 1) click, redirect to a page that sends a POST request. 2) put JS on website
- 3. Server accepts malicious request

CSRF Defenses

- By server, not browser
- The request must cause server-side action

CSRF Tokens

- A secrete value to server, not send via cookie
- Attacker cannot trick user to make request bc no token
- Help defend against HTML forms POST request

Referer Header

- A header in an HTTP request that indicates which webpage made the request
- Allow same-site requests, prevent cross-site requests
- Remove RH before request reaches server, prevent private info leak. Optional → less secure

SameSite Cookie Attribute

- Flag on cookie, domain of cookie needs to watch domain of origin
- Not implemented by all browsers

XSS, UI Attacks

XSS (Cross-Site Scripting)

- Inject JS, receiving
- The response must be parsed as HTML for JS to run

Stored XSS

- Stored on server victim load the page

Reflected XSS

- Causes victim to input JS into request, content is copied in response from server
- requires the victim to make a request with injected JavaScript
- Click link, embed iframe, redirect to other URL…

Defenses

- HTML sanitization → escape special chars
- HTM.Escaping, templating (human factor)
- Content security policy (CSP) → restrict browser to only use resources loaded from specific places (defense-in-depth)

UI Attacks

Clickjacking

- 1) Invisible iframe (doesn't break same-origin
- 2) Temporal attack
- 3) Cursorjacking
- Defenses
  - Enforce visual integrity
  - Enforce temporal integrity (time to read)
  - Require confirmation
  - Frame-busting (forbid embed iframe)

Phishing

- 1) Homograph attack ("same" url)
- 2) Browser-in-browser attack
- Defenses:
  - 2FA (auth tokens, security keys-prevent phishing)
  - Subvert 2FA: relay attacks, social engineering

SQL Injection, CAPTCHA

SQL

- INSERT INTO table VALUES (...)
- UPDATE table SET column=value WHERE condition
- DELETE FROM table WHERE condition
- CREATE TABLE table {column type}
- DROP TABLE table

SQL Injection

- SELECT item, price FROM items WHERE name = '' OR '1' = '1' → OR TRUE, return every item
- '; DROP TABLE items --'

Defenses

- Input sanitization (escape special chars)
- Prepared statements
  - SQL start with "?". Parse SQL first, then insert data. db.QueryRow("SELECT name, price FROM items WHERE name = ?", itemName)
  - Must rely on the API to correctly convert

Command Injection

Defenses: use safe APIs (execv instead of system)

Intro to Networking

OSI Model

* lower layer, more headers. Send high to low, receive low to high

1)Physical - send bits (wires)

2)Link - LAN (ethernet)

3)Network (source, destination, data) - send packets

- IP (not reliable)

4)Transport

- Reliability: send in order
- Ports
- TCP, UDP

7)Application

- HTTP

Lower Level Network Attacks

ARP (Address Resolution Protocol)

- MAC identifies addr in layer 2, IP in layer 3
- ARP: translate L3 IP to L2 MAC
- Check cache to see whether have MAC addr. If not, broadcast to everyone on LAN, receive response, cache.
- If not on LAN, router respond with its MAC, router forward to Bob.
- All received ARP responses are cached, even no request

ARP Spoofing

- race condition - Before Bob's response can arrive, Mallory sends a malicious response, cached
- Defense:
  - Use Switches to avoid broadcast (large cache)
  - If not in cache, switch broadcast

DHCP (Dynamic Host Configuration Protocol)

- Initial network configuration

Steps

- Client discover
- DHCP offer
- Client request (client broadcast the choice)
- DHCP acknowledgment

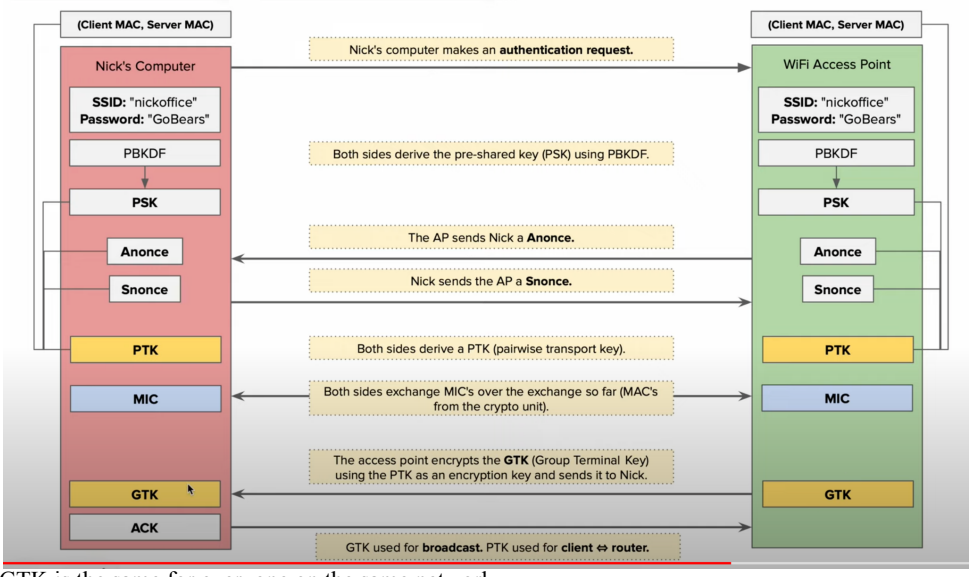
DHCP Attacks

- Spoofing
- Requires Mallory in same LAN
- Mallory can be MITM

Wi-Fi

- Layer 2, connect LAN
- Access point, SSID, password

WPA2 (layer 1)



GTK is the same for everyone on the same network.

WPA-PSK Attacks

Rogue AP

- know the password/PSK, you can complete the 4-way handshake with the client and become a MITM

Offline Bruteforce Attack

- Guess password (generate PSK, then PTK)

No Forward-Secrecy

- An eavesdropper who records the values of ANonce and SNonce can derive the key if they later learn the password or PSK. Unlike Diffie-Hellman

Solution: WPA Enterprise

- Vulnerable to higher level spoofing (ARP, DHCP)

BGP, TCP, UDP

BGP (Border Gateway Protocol)

IP routes by subnets (16 bits common prefix for IPv4)

Routing Packets

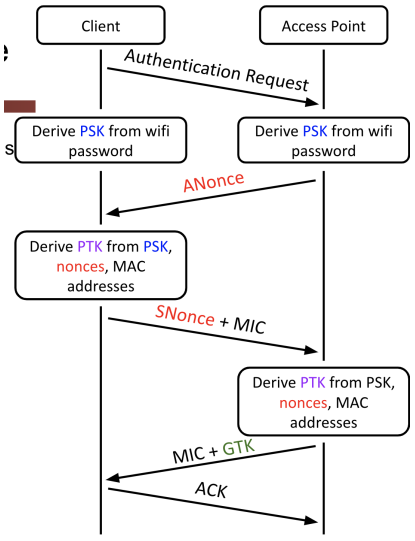
- Send within local network: Verify that the destination IP is in the same subnet. Use ARP (or contact a switch) to get the destination MAC address. Send the packet directly to the destination using the destination MAC address
- Send outside of local network: Send the packet to the gateway. Past the gateway, the packet goes to the Internet. It's the gateway's job to deliver the packet closer to the destination

Autonomous System (AS)

- The Internet is a network of networks, comprised of many **autonomous systems** (**AS**)
  - Each AS handles its own internal routing
  - Each AS is uniquely identified by its **autonomous system number** (**ASN**)
  - Each AS is comprised of one or more LANs
  - The AS can forward packet to other connected ASes
- The protocol for communicating between different Autonomous Systems is **Border Gateway Protocol** (**BGP**)
  - Each router announces what networks it can provide and the path onward from the router
  - The most precise route with the shortest path and no loops is the preferred route

IP Spoofing

- Malicious clients can send IP packets with source IP values set to a spoofed value. Higher level prevention



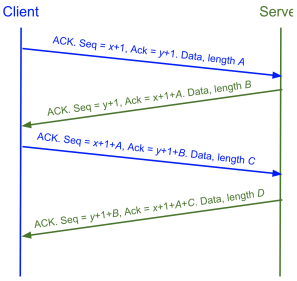
BGP Hijacking

- A malicious autonomous system can lie and claims itself to be responsible for a network which it isn't
- Higher level prevention needed

TCP (Transmission Control Protocol)

- Break into segments, ordering, reliability (ACKs, resend), ports.
- 2 initial sequence numbers (ISNs): client-server, server-client. Client and server must agree on this before TCP connection
- 3-way handshake
- Connection identified by 5 values: Source IP, destination IP, source port, destination port, protocol

1. Client chooses an initial sequence number *x* its bytes and sends a SYN (synchronize) packet to the server
2. Server chooses an initial sequence number *y* for its bytes and responds with a SYN-ACK packet
3. Client then returns with an ACK packet
4. Once both hosts have synchronized sequence numbers, the connection is "established"



Sending and receiving data

- Byte *i* of the bytestream is represented by sequence number *x* + *i*
  - The first byte is byte *i* = 1, since sequence number *x* was used for the SYN packet and *y* for the SYN-ACK packet
- A packet's sequence number is the number of the first byte of its data
  - This number is from the sender's set of sequence numbers
- A packet's ACK number, if the ACK flag is set, is the number of the byte immediately after the last received byte
  - This number is from the receiver's set of sequence numbers
  - This would be (sequence number) + (length of data) for the last received packet

Retransmission

- If a packet is dropped (lost in transit):
  - The recipient will not send an ACK, so the sender will not receive the ACK
  - The sender repeatedly tries to send the packet again until it receives the ACK
- If a packet is received, but the ACK is dropped:
  - The sender tries to send the packet again since it didn't receive the ACK
  - The recipient ignores the duplicate data and sends the ACK again
- When packets are dropped in TCP, TCP assumes that there is congestion and sends the data at a slower rate

End connection: FIN-FINACK, RST

TCP Hijacking

- **Data injection**: Spoofing packets to inject malicious data into a connection
  - Need to know: The sender's sequence number
  - Easy for MITM and on-path attackers, but off-path attackers must guess 32-bit sequence number (called **blind injection/hijacking**, considered difficult)
  - For on-path attackers, this becomes a race condition since they must beat the server's legitimate response
- **RST injection**: Spoofing a RST packet to forcibly terminate a connection
  - Same requirements as packet injection, so easy for on-path and MITM attackers, but hard for off-path attackers
  - Often used in censorship scenarios to block access to sites

TCP Spoofing

- Spoofing a TCP connection to appear to come from another source IP address

- Need to know: Sequence number in the server's response SYN-ACK packet
- Easy for MITM and on-path attackers, but off-path attackers must guess 32-bit sequence number (called **blind spoofing**, also considered difficult)
- For on-path attackers, this is a race condition, since the real client will send a RST upon receiving the server's SYN-ACK!

* TCP provides no confidentiality or integrity (rely on higher layer)
* Choose random sequence numbers to defense against off-path attacks (hard to brute force)

UDP (User Datagram Protocol)

- No reliability, no ordering, have ports
- Faster than TCP, only 3-way handshake

Attacks

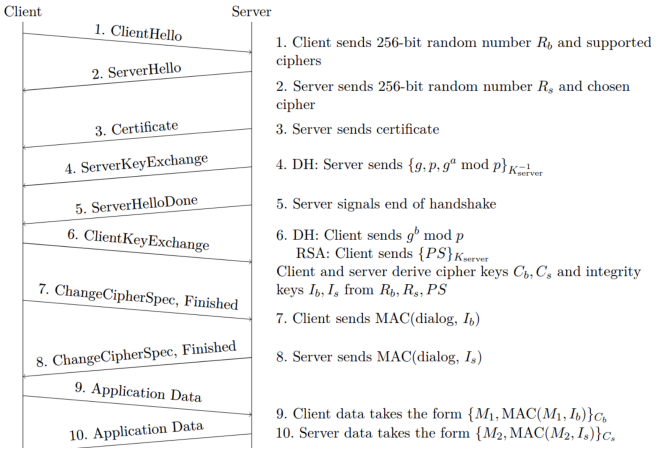
- Same injection and spoofing attacks as TCP, but easier

TLS (Transport Layer Security)

- Built upon TCP, provides confidentiality and integrity. Prevent replay attack.

- Security properties
  - DHE TLS: Forward secrecy
  - RSA TLS: No forward secrecy
  - End-to-end security: Secure even if all intermediate parties are malicious
  - Not anonymous: Attackers can determine who you're talking to
  - No availability: Connections can be dropped or censored
- Can be used by the application layer (e.g. HTTPS)
- Trusting certificate authorities can be hard

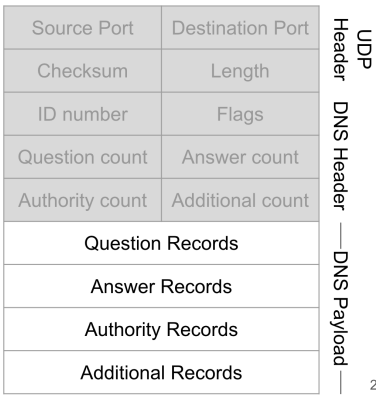
TLS does not defend against application-layer vulnerabilities
- Example: SQL injection, XSS, CSRF, and buffer overflow vulnerabilities in the application are still exploitable over TLS



DNS

- Use UDP for speed

- **Stub resolver**: The resolver on your computer
  - Only contacts the recursive resolver and receives the answer
- **Recursive resolver**: The resolver that makes the actual DNS queries
  - Usually one recursive resolver per local network
  - Benefits: The recursive resolver can cache common requests for the network



DNS Records Format

- A (answer) type records: domain → IP
- NS (name server) type: point to another NS
- TTL

- Question section: What is being asked
  - Included in both requests and responses
  - Usually an A type record with the domain being looked up
- Answer section: A **direct response** to the question
  - Empty in requests
  - Used if the name server responds with the answer
  - Usually an A type record with the IP address of the domain being looked up
- Authority section: A **delegation of authority** for the question
  - Empty in requests
  - Used to direct the resolver to the next name server
  - Usually an NS type record with the zone and **domain** of the child name server
  - Additional:

Cache Poisoning Attack

- Malicious name server returns malicious record to client, cache them, becomes MITM
- Defense: **Bailiwick checking**: the resolver only accepts records if they are in the name server's zone

MITM Attacker

- Can poison cache by change DNS response

On-Path Attacker

- Send spoofed response

Off-Path Attacker

- Brute force ID # (16bits) to spoof a response
- ID number needs to be random

Kaminsky Attack

- Query non-existent domains and put the poisoned record in the additional section (which will still be cached). Lets the off-path attacker try repeatedly until succeeding
- Defense: **randomize source port**. Attacker needs to guess destination port and query ID (32bits)
- Defense: **glue validation**. Don't cache glue records as part of DNS lookups. Not implemented by all DNS softwares.

DNSSEC

- Need integrity, not confidentiality
- DNS over TLS: slow, only provides channel security, but need object security to secure DNS
- Records, signature + certs, public key

Sign Records

- Digital signatures provide integrity
  - Only the name server with the private key can generate signatures
  - Everybody can verify signatures with the public key
- Digital signatures defeat network attackers
  - An off-path, on-path, or MITM attacker can no longer tamper with records
  - The recursive resolver can no longer tamper with records
- Signatures can be cached with the records for object security
  - Any time we fetch a record from the cache, we can verify its integrity

Public Key Infrastructure (PKI)

- Name servers are arranged in a hierarchy, as in ordinary DNS
- Parents can delegate trust to children
  - The parent signs the child's public key to delegate trust to the child
  - If you trust the parent name server, then now you trust the child name server
- Trust anchor: We implicitly trust the root name server
  - The root name server's public key is hard-coded into resolvers
- PKI defeats malicious name servers
  - A malicious name server (assuming they don't have access to the private key, only the signatures) won't have a valid chain of trust back to the root

Resource Record Sets (RRSETs)

- A group of DNS records with the same name and type form a **resource record set** (**RRSET**)
  - Example: All the AAAA records for a given domain
- RRSETs will be useful for simplifying signatures
  - Instead of signing every record separately, we can sign an entire RRSET at once
- We need new record types to send cryptographic information in DNSSEC packets
  - RRSIG (resource record signature): encode signatures on records
  - DNSKEY: encode public keys
  - DS (delegated signer): encode the child's public key (used to delegate trust)

New DNS Record Type: DS

- DNSSEC delegates trust with two records:
  - A DS type record with the hash of the signer's name and the child's public key
  - An RRSIG type record with a signature on the DS record

I don't know, but you should ask the **`berkeley.edu`** name server.
- NS record: Domain of the **`berkeley.edu`** name server
- A record: IP address of the **`berkeley.edu`** name server

Here is a signature on the public key of the **`berkeley.edu`** name server. If you trust me, then now you trust them too.
- DS record: Hash of the **`berkeley.edu`** name server's public key
- RRSIG DS record: Signature on the DS record

Here is my public key so you can verify the signature.
- DNSKEY record: The **`.edu`** name server's public key

NSEC: Signing Non-Existent Domains

Authenticated Denial of Service

- Prove nonexistence of a record type
  - Sign a record stating that no record of a given type exists
  - Useful for proving that a domain doesn't support DNSSEC ("No DS records exist")
- Prove nonexistence of a domain
  - Provide two adjacent domains alphabetically, so that you know that no domain in the middle exists
  - Example: If I query for **`nonexistent.google.com`**, I can receive a signed NSEC response saying "No domains exist between **`maps.google.com`** and **`one.google.com`**."
  - We can sign all pairs of adjacent records ahead of time and keep them as NSEC records, along with their RRSIGs

Issues: domain enumeration to find the correct one

NSEC3

- Instead of storing pairs of adjacent domain names, store pairs of adjacent hashes

Issues

- Domain enumeration still possible. The only real way to prevent enumeration is online signature generation with the private key

Denial of Service and Firewalls

DoS Attack Strategies

- Exploit program flaws
- Resource exhaustion
- Exhaust the bottleneck

DoS Targets

- Application level, network level

Application Level DoS

- Some attacks rely on **asymmetry**: A small amount of input from the attack results in a large amount of consumed resources. (storage, RAM, threads, disk I/O)
- Algorithmic complexity attack
- Defenses:
  - Identification, isolation, quotas, proof-of-work, overprovisioning

Network Level DoS

- Overwhelm network bandwidth, overwhelm packet processing capacity

DDoS: Distributed DoS

- **Botnet**: A collection of compromised computers controlled by one attacker

Amplified DoS

- Some services send large response when given small request. Ex. DNS amplification

Network Level DoS Defenses

- **Packet filter**: Discard any packets that are part of the DoS attack
  - Discard packets where the source IP is the attacker's IP address
  - Find some pattern in the content of the DoS packets to distinguish DoS packets from legitimate packets
  - The packet filter must be before the bottleneck
- Subverting packet filters
  - Spoof DoS packets so that packets look like they're coming from many IP addresses
    - Packet filters can't use IP addresses to filter packets anymore!
    - Hard to defend against
    - Rely on anti-spoofing mechanisms on the network
  - Distributed DoS actually send packets from many IP addresses
    - Packet filters need to be much more sophisticated to defend against DDoS attacks

* overprovisioning

SYN Flooding and SYN Cookies

- Attacker sends SYN
- Defenses:
  - Overprovisioning
  - Filtering
  - SYN cookies: don't store state

- Relies on the client to store the server's state
- The client returns the state to the server in the ACK packet of the handshake

Firewalls

- Outbound, inbound policy

Default Security Policies

- Default allow (blacklist)
- Default deny (whitelist), best practice

Stateless Packet Filters

- Have no history, only rely on packet info

- Issue: How do we know what inbound traffic is in response to an outbound connection?
  - TCP: Can be implemented with a hack
    - Allow inbound traffic with the ACK flag set
    - Deny inbound traffic without an ACK flag set
    - If the internal computer sees an ACK packet without having formed a connection, it will ignore it or send a RST
  - UDP: Impossible to implement
    - UDP "connections" are typically implemented at the application layer, so we can't inspect much

Stateful Packet Filters

- The filter keeps track of inbound/outbound connections

: drop * 5.6.7.8 :*/ext -> 1.2.3.4 :443/int

Subverting Packet Filters

- Send keyword across packets
- Send keyword out of order
- TTL

Proxy Firewalls

- Instead of forwarding packets, form two TCP connections: One with the source, and one with the destination (MITM)

Application Proxy Firewall

- Example: HTTP proxies will make an HTTP request on behalf of the user then return the HTTP response to the client

Virtual Private Network (VPN)

- A set of protocols that allows direct access to an internal network via an external connection

Intrusion Detection

Path Traversal Attacks

- Defense: Check that user input is not interpreted as a file path

Detector - NIDS (Network Intrusion Detection System)

- A detector installed on the network, between the local network and the rest of the Internet. Monitor network traffic.

- NIDS has a table of all active connections and maintains state for each connection
- If the NIDS sees a packet not associated with any known connection, create a new entry in the table
  - Example: A connection that started before the NIDS started running
- NIDS can be used for more sophisticated network monitoring: not only detect attacks, but analyze and understand all the network traffic

- Benefits: cheap, easy to scale, simple management, end system not affected, small TCB
- Drawbacks: inconsistent interpretation between detector and end host (TTL attack), encrypted traffic (path traversal)

Evasion Attack

- Exploit inconsistency and ambiguity to provide malicious inputs that are not detected by the NIDS

- Problem: Imperfect observability
  - What the NIDS sees doesn't match what the end system sees
  - Example: The packet's time-to-live (TTL) might expire before reaching the end host
- Problem: Incomplete analysis (double parsing)
  - Inconsistency: Inputs are interpreted and parsed differently between the NIDS and the end system
  - Ambiguity: Information needed to interpret correctly is missing

- Defenses:
  - Normalize all inputs
  - Analyze all possible interpretations
  - Flag potential evasions

Drawback: Encrypted Traffic

- Recall: TLS is end-to-end secure, so a NIDS can't read any encrypted traffic
- One possible solution: Give the NIDS access to all the network's private keys
  - Now the NIDS can decrypt messages to inspect them for attacks
  - Problem: Users have to share their private key with someone else

HIDS (Host-Based)

- Drawback: evasion attack still possible, expensive

Logging

- Benefits: cheap, less inconsistency
- Drawback: attack already happened, evasion attack still possible, attacker can change logs

Detection Accuracy

- False positive, false negative

Combining Detectors

- Parallel: Fewer false negatives, more false positives
- Series: Fewer false positives, more false negatives

Styles of Detection

- Signature-based
  - Flag any activity that matches the structure of a known attack (blacklisting)
  - Good at detecting known attacks, but bad at detecting unknown attacks
- Specification-based
  - Specify allowed behavior and flag any behavior that isn't allowed behavior (whitelisting)
  - Can detect unknown attacks, but requires work to manually write specifications
- Anomaly-based
  - Develop a model of what normal activity looks like. Alert on any activity that deviates from normal activity.
  - Mostly seen in research papers, not in practice
- Behavioral
  - Look for evidence of compromise
  - Can cheaply detect new attacks with few false positives, but only detects after the attack

Vulnerability scanning

Honeypot (like stack canaries)

Forensics

Intrusion prevention system (IPS) - block attack

Malware

Self-replicating code

Virus: need user action to propagate

Detection:

- signature-based, replicating same code
- Antivirus
- Attacker look for evasion strategies

Polymorphic code

- Each time the virus propagates, it inserts an encrypted copy of the code. Also include key and decryptor. Used for obfuscation, not confidentiality.

Metamorphic code

- Each time the virus propagates, it generates a semantically different version of the code. Include code writer

- Behavioral detection
  - Need to analyze behavior instead of syntax
  - Look at the effect of the instructions, not the appearance of the instructions
  - Antivirus company analyzes a new virus to find a behavioral signature, and antivirus software analyzes code for the behavioral signature
- Subverting behavioral detection
  - Delay analysis by waiting a long time before executing malcode
  - Detect that the code is being analyzed (e.g. running in a debugger or a virtual machine) and choose different behavior
  - Antivirus can look for these subversion strategies and skip over them

Defense: flag unfamiliar code

Worm: do not need user action to propagate

- How does the worm find new users to infect?
  - Randomly choose machines: generate a random 32-bit IP address and try connecting to it
  - Search worms: Use Google searches to find victims
  - Scanning: Look for targets (can be limited by bandwidth)
  - Target lists
    - Pre-generated lists (hit lists)
    - Lists of users stored on infected hosts
    - Query a third-party server that lists other servers
  - Passive: Wait for another user to contact you, and reply with the infection
- How does the worm force code to run?
  - Buffer overflows for code injection
  - A web worm might propagate with an XSS vulnerability

Infection Cleanup and Rootkits

- Recovery method: Reset everything and start from scratch
- Rootkits: Malware in the operating system that hides its presence

Anonymity and Tor

Conceal identity, but not confidentiality

- The exit node is a man-in-the-middle attacker
  - If the user is not using TLS to connect to the end host (using HTTP), the exit node can see and modify the traffic
  - If the user is using TLS (using HTTPS), the exit node cannot see or tamper with the traffic
- **Collusion**: Multiple nodes working together and sharing information
  - Collusion is adversarial (dishonest) behavior
  - Honest nodes should never share information with other proxies
  - If *all* nodes in the circuit collude, anonymity is broken
  - If *at least one* nodes in the circuit is honest, anonymity is preserved
- Defense: **Guard nodes**
  - Guard nodes must have a high reputation and must have existed for a long time
  - Clients will always use a guard node as the entry node (by default) and the same guard node is used for a long period of time
    - Attackers' nodes are unlikely to become guard nodes
    - Because clients use the same guard nodes for a long period of time, there is only a low chance that the client will switch to an attacker's guard node

Tor Weaknesses: Distinguishable Traffic

Computer Science 161

- Defense: **Tor bridges**
  - Notice: Attackers can tell you are using Tor because they can see you are connecting to an entry node
    - Lists of entry nodes are publicly available
  - **Tor bridges** are entry nodes that are not available on any public list
    - Users request bridges from a separate directory, which will only give a few bridges to the user
    - There is no publicly available list of all bridges!
  - Censors can no longer block Tor based on IP addresses, but they can still distinguish traffic that looks like Tor traffic from normal traffic
- Defense: **Pluggable transports**
  - Pluggable transports change the appearance of the client's traffic to the entry node (only for bridges)
  - Obfuscates the encrypted traffic to make it "look" more like normal web traffic