CS 458 — Module 4: Networks

1 Intro to Networks

- To create a network, you need 3 things:
  1. Devices able to receive and send signals
  2. A way to connect devices to each other
  3. Rules for communicating, or a protocol
- Some examples of protocols:
  - Token ring — a person can only talk if they have the token
  - CSMA/CD — all listen to the wire, if they hear no signal they try to transmit. If there is a collision, then they all stop and resend.
- Well what are some problems?
  - The Internet's design connects many computer networks together. It also assumes that participants are honest and will cooperate — they will not look at messages that don't belong to them, they will not delete your messages, etc. Everyone should mutually work together... right?
  - There's also no routing logic in the addressing scheme — given some IP address, who knows where it comes from? For example, a phone number has an area/country code. An IPV4 address like `136.192.63.0` could come from anywhere!
  - Nor can you control the path your message follows!
  - Your message can be broken up with each part following a different route.
  - There is no real hard stop limit to the number of nodes (at least everywhere).
  - It's really hard to conceptualize.
  - Nobody is in charge (both good and bad).

2 Daemons, Servers, Ports

- A server is a computer on a network to do tasks for other computers (clients).
- A daemon is like a servant that can only do one task within a server.
- We can think of a server like a huge apartment building, and each apartment can have one servant (daemon).
- For example, the mail sending daemon (SMTP) is 25.
- Some apartments (ports) can be empty. Many ports are actually empty!
- One could hide a service in a port it's not supposed to be in.
- For example, an HTTP daemon is in port 80. This is implied by default (ie: `https://www.uwaterloo.ca` implies `https://www.uwaterloo.ca:80`).
- But one could put a web service at, for example, port 8080.
- A "loose-lipped" system may reply to an attacker and advertise what services they are running *and* what at what port.

3 Port Scanning, Information Gathering, Wiretapping, Impersonation

3.1 Port Scanning

- A port scan checks every port in sequence.
- We would ideally want, at least for security, to not reply when ports are checked.
- Unfortunately, this isn't really possible as we need this replying for actual use.
- Tools like `nmap` would give many details about a machine.
- A command like `finger` allows you to look up a user in a machine. If this is not closed, one could do it from outside a machine...
- But this is just the beginning... maybe you could get some info, but port scanning is not really malicious on its own yet.

3.2 Intelligence Gathering

- Social Engineering is attacking people via exploiting other humans, which can give valuable information.
- Pretending to be part of an organization they're not, exploiting the helpful nature of people ("I forgot my password"), distractions to grab information somehow, etc.
- Other ways you could get info?
  - Dumpster diving
  - Eavesdropping
  - Lots of things placed online that shouldn't be there — Google, social media, etc.
- Wiretapping
  - Two types: passive and active.
  - Passive wiretapping is basically just eavesdropping. When a message is sent, a node could read the destination data — but there is *nothing* stopping Eve from looking at the data!
  - The analogy is an envelope with an address and a non-sealed back.
  - Active wiretapping will require modification/fabrication of communication.
  - For example, Mallory could modify a message sending money from one account to another. That is, Mallory is usually a MITM during an active wiretap attack.
  - One can also eavesdrop while communication is flowing through a link; we call "promiscuous sniffing".
  - We should *always* assume someone is eavesdropping the data!
  - The degree of vulnerability would depend on the communication media:
    - For example, copper cables mean that a physically close attacker could eavesdrop without making physical contact, or just cut the cable open/splice in another cable.
    - Coaxial cables help shield some of this signal from leaking out compared to twisted pair cables.
    - Optical fibre would be harder, as there is no inductance and signal loss caused by splicing would be noticeable.
    - Unbound transmission is through the air — WiFi, microwaves, radio, etc.
    - This is versus bounded, like cables.
    - How could we protect something like WiFi? Problems are:
      - It is easy to intercept with anything that can use WiFi.

        - It's easy to read packet info like destination and source IP addresses — even at a distance!
        - Physical barriers are useless for a wireless network.
        - Wireless APs can also be faked; one could use a router that is not actually owned by the network you are connecting to to steal credentials.
    - When we transmit data, how do we choose what medium?
        * Is it sensitive? If it is, we probably don't want to use an unbounded medium.
        * Are there *segments* of the network carrying sensitive data?
        * Would one notice of an intruder is eavesdropping? For example, using barriers that would make an attack obvious due to damage to said barrier.
        * Are backbone segments accessible? Can an intruder actually attack said parts of the network?

3.3 Impersonation

- A person could try to log into a machine that does not belong to them by pretending to be an owner.
- Steal passwords, guess, social engineering, sniff password, etc.
- Or pretend to act like a machine itself.

4 Spoofing, Session Hijacking, Traffic Analysis, and Integrity Attacks

4.1 Spoofing

- Pretending to be another object. For example, `www.goggle.com` leading to a malicious site.
- What one can do to defend against it as an owner is to register typos of the site and redirect it to the correct site.
- Another way to spoof is via the evil twin attack for wifi access points.
- Or ISMI catchers which cause all mobiles in the area to be attracted to the ISMI catcher rather than a real cell phone tower.
- Traffic can also be spoofed — it may let packets through because it is coming through port 80 and looks like HTTP even if is not safe traffic.

4.2 Session Hijacking

- The TCP protocol sets up a state which the sender and receiver nodes can use to send packets in a secure manner.
- The server will keep some session cookie in order to not have to redo the reidentification process from scratch.
- But, someone could theoretically figure this out and pretend to be that client!
- HTTPS prevents this.

4.3 Traffic Analysis

- More of a privacy issue.
- Sometimes, the mere existence of communication between two parties is sensitive and should be hidden!
- Even encryption of this traffic isn't enough if the communication existing itself is important enough.
- For TCP/IP packets, one could just sniff the packet to see the source/destination since this is required to check.
- Some things like Tor can deal with this to at least hide some of this data.

4.4 Integrity Attacks

- Attackers can modify packets while they are being transmitted.
- A solution is to use a checksum (though this can obviously be easily modified to still work or even forged).
- Another idea is to use a more complicated checksum that makes it unfeasible to do.
- Another integrity attack is DNS cache poisoning.
- A DNS links hosts names to numerical addresses.
- This means an attacker could create the wrong mappings between a host and its IP address causing the server to save incorrect entries, which causes traffic to be rerouted to an IP of an adversary's choosing (poisoning).
- Another way to do this is to just use a DNS server that responds to a DNS request with a fake response. Your browser will use this response!

4.5 Protocol Failures

- TCP/IP assumes all nodes implement protocols faithfully.
- For example, TCP includes a mechanism that asks a sender node slow down if the network is congested.
- An attacker could ignore this.
- Or MS servers ignored congestion when pushing out updates.
- Protocols are also complex and some behaviour may not be properly defined.
- Some protocols have security features baked in — this is not so great though if said feature is flawed.
- For example, WEP.

4.6 Website Vulnerabilities

- Accessing a URL causes the web server to return the actual HTML code needed to render the page... and attackers can examine this code to find vulnerabilities.
- For example, what if the attacker sends a malicious URL with unsafe characters?
- Parsing the URL request could lead to unfortunate things!
- Website defacements are another example of exploiting website vulns.
- Some website defacements are legit — for example, when domain names are seized.
- Another way website vulns can be done is by exploiting the HTTP protocol.
  - Since it is stateless, web servers will ask clients to keep state info sometimes — perhaps, for example, storing the client ID in the URL.
  - This can easily be exploited.
- Other examples? XSS/CSRF. SQL injection.

4.7 DOS Attacks

- A physical attack — just jam a signal/cut a wire.
- For a more general one, flooding a node by overloading its internet connection or processing priority is a way to deny service to legitimate users.
- One attack is a ping flood — a node receives a lot of ping packets.
- Another is a ping of death — a malformed ping packet to crash a computer.
- Another theoretical attack is sending a non-existent IP address. This is prevented via a TTL (time-to-live), which sets a max lifetime for a packet. This prevents it from travelling forever.
- A smurf attack is when an attacker spoofs the source address of a sender in the ping packet by setting it to the victim's address — so it relies on many returning pings to overload someone.
- An attacker can exploit the knowledge of implementation details about a node to make the node perform poorly.
- SYN flooding:
  - Normally, in TCP, SYN sets up a communication to the server, SYN-ACK is when the server sends an acknowledgement to the client, then the client sends back an ACK.
  - The server queues a SYN from the client and removes it when the ACK is received.
  - SYN flooding ties up the server by sending many SYNs and never an ACK.
- Another attack is sending packet fragments that can't be reassembled... which ties up the server as it waits for or looks for (for example), a "missing" packet or something.
- A black hole/packet drop attack discards any traffic destined for the victim.
- DNS cache poisoning can also be used to deny service.
- DDOS-ing is using many machines (often taken over by malware) to attack.

4.8 Botnets

- Many machines (usually taken over) under the control of some centralized computer that will attack when told to.
- Usually the owner of the infected machine will not know.

4.9 Active Code

- A server might ask a client to execute code on the server's behalf... obviously this could be dangerous if the active code triggers some attack.
- There might be a sandbox.
- Word macros, Java and ActiveX are some examples.

4.10 Script Kiddies

- It's very easy to download and run scripts/attacks for cheap or free.
- Hell, you can buy people's services to attack people.

5 Network Controls

- Remember that using TCP/IP means that there is no actual permanent connection between the sender and receivers.
- Always check inputs, never trust an input from a client.
- One should use a whitelist of allowed characters/sites, *not* a blacklist of forbidden ones. This is obvious — it's easier to specify what we want to allow to do and not worry about things we didn't think of!
- Segment/separate your servers; don't use one machine if possible.
- Your web servers (or anything that should be accessible outside) should not be trusted by other servers of the company.
- Avoid single points of failure.
- Have redundancy/syncing.

6 Firewalls

- All traffic has to go through a small number of gates/choke points.
- These should examine traffic.
- Note firewalls cannot do anything for things inside its protection interacting with other things interacting within the firewall — you can't keep out what's already in!
- Note a firewall is an attractive target for attackers — protecting something does paint a bit of a target on what you're protecting!
- Four general ways to implement firewalls:
    1. Packet filtering gateways/screening routers (stop certain packets from entering/exiting if they have specific addresses, ports, etc.)
    2. Stateful inspection firewalls (keeps state to identify packets that belong together; looks to see if traffic is a new connection, replying to an existing connection, or not part of any connection and thus dropped)
    3. Application proxies (client talks to proxy, proxy talks to server)
    4. Personal firewalls (protect attacks on services running on the computer)
- Another approach is using a demilitarized zone (DMZ), which is a subnetwork that contains an organization's external services, accessible to the internet.
- The external firewall protects DMZ, the internal firewall protects internal network from attacks lodged in the DMZ.

7 Honeypots

- A machine that looks vulnerable and attract attackers.
- Allow us to realize we are being attacked, and prepare/analyze their attacks.
- Must be careful that an attacker doesn't realize they are being tracked.

8 Intrusion Detection Systems (IDS)

- Host-based IDS: run on a host to protect that host.
- This can monitor a lot of information about that host, but may miss information available to other hosts.
- If the host gets subverted, the IDS likely gets subverted as well.
- A network-based IDS runs on a node to protect all hosts attached to a network.
- This also means it will have to rely on information available in monitored packets, but also that it's harder to subvert.
- Signature-based IDSs can look for patterns of known attacks or behaviours that suggest an intrusion — however this also means it can't be used to find unknown or new attacks.
- Heuristic/anomaly-based IDSs look for behaviour that is out of the ordinary; and raising an alert when system activity doesn't resemble this model any more.