CS161 Final cheat sheet

Eknoor Kaur
12/10/20

· Networking

- Background
  - primary goal of internet is to move data from one location to another
  - can connect a group of local machines in a local area network (LAN)
    - ↳ all machines are connected to all other machines
      - ↳ router connects multiple LANs (since infeasible to connect every machine in world)
  - with enough routers and LANs, we can connect the entire world in a wide area network which forms the basis of the internet
- Internet Layering (OSI 7-layer model)
  - Layer 1: physical Layer → moves bits across space
  - Layer 2: Link Layer → uses layer1 as a building block to connect local machines in a LAN
  - Layer 3: Internetwork Layer → connects many layer 2 LANs
  - Layer 4: Transport Layer
  - Layer 6.5: Secure Transport
  - Layer 7: Application

  ethernet is an example of a implementation of LAN where all machines are connected by wires
- each layer has its own set of protocols (a set of agreements on how to communicate)
  - ↳ how communication is structured, how machines should behave while communicating, and how errors should be handled
    - ↳ to support this, messages are sent with a header (placed at beginning of msg and contains some metadata such as sender, recipient, length of msg, ID #s, etc.)
      - ↳ multiple headers on a packet (one per level: Link, Internetwork, Transport)
- each msg begins w/ human readable text, then we go through layers (high → low) and add header to top of msg provided from layer directly above
  - ↳ when msg reaches destination, recipient must unpack the msg and decode it back into human-readable text
- Addressing (machine can be referred to by diff addresses depending on layer) → both source & destination addresses are contained in the header of a msg
  - ↳ Layer 2: uses 48-bit (6 byte) MAC addresses to uniquely identify each machine on LAN
    - ↳ written as 6 pairs of hex numbers
    - ↳ special address (broadcast address = ff:ff:ff:ff:ff:ff)
      - ↳ "send this msg to everyone on local network"
    - like apartment #, only useful in the context of apartment complex (LAN)
  - ↳ Layer 3: uses 32-bit (4 byte) IP addresses to uniquely identify each machine globally
    - ↳ written as 4 integers between 0 and 255 (IPv4)
  - ↳ Higher Layers: assign each process on a machine a unique 16-bit port number
    - ↳ allows each machine to have multiple processes communicating across the network
- packets versus connections
  - at lower levels (1-3), there is no concept of a connection
    - ↳ call individual msgs packets (usually limited to a fixed length)
  - at higher levels, we actually create a 2 way connection
    - ↳ levels maintain connection by breaking up longer msgs into individual packets & sending them through lower layer protocols
    - ↳ can implement cryptographic protocols for additional security
  - ↳ packets can be corrupted in transit or even fail to send entirely
    - ↳ IP (layer 3) only guarantees best-effort delivery
      - ↳ rely on higher layers for correctness & security
- Network Adversaries
  - offpath adversary → cannot read or modify any packets sent over the connection (weakest)

- onpath adversary → can read, but not modify packets
- inpath adversary (MITM) → can read, modify, and block packets (strongest)
- all adversaries can send packets of their own (including faking/spoofing the packet headers to appear like the msg is coming from someone else)

- Wired Local Networks: ARP (protocol used by IP)
  - ARP → on layer 2
    - → translates IP addresses to MAC addresses (layer 3 address to layer 2 address)
    - ↳ vulernability: on-path attackers can see requests & send spoofed malicious responses
      - ↳ Defense: switches, arpwatch
      - ↳ cuz no way to verify that reply in step 2 is actually from Bob (so, just race condition w/ Bob & Mallory)
  - ethernet (LAN implementation)
    - started as broadcast only network
      - ↳ each node on network could see msgs sent by all other nodes cuz there was a common wire or a network hub (a simple repeater that took every packet it received & rebroadcast it to all outputs)
    - Receiver is supposed to ignore all packets not sent to receiver's MAC or broadcast address
      - ↳ but ethernet devices can enter promiscuous mode (receive all packets — sniffing packets)

Alice wants to send Bob a msg (Bob's IP: 1.1.1.1.)
1. Broadcast to everyone on LAN: "MAC address of 1.1.1.1.?"
2. Bob responds with msg to only Alice: "My IP is 1.1.1.1. & my MAC address is ca:fe:f0:0d:be:ef"
3. Alice caches IP address to MAC address mapping for Bob

- Wired Local Networks: WPA2 (wifi protected access)
  - WPA2 → on layer 2
    - → communicate securely in a wireless local network
    - ↳ vulernability: on-Path attackers can learn the encryption keys from the handshake and decrypt msgs (includes brute forcing pwd) → in handshake, everything but GTK, is sent unencrypted & if attacker is on network, they know PSK and can eavesdrop for nonces & MACs and thus get PTK (becomes MITM)
      - ↳ Defense: WPA2-Enterprise → each user has unique source of secrecy PMK instead of shared PSK
  - WIFI (another implementation of Link layer)
    - ↳ to connect computer to WIFI: computer establishes a connection to the network's AP (Access Point) which is continuously broadcasting packets signifying its existance and announcing the name of the network (SSID - service set Identifier); after connecting to a WIFI network, it will broadcast a request to join the network
    - ↳ if network is configured without a pwd, your computer immeadiately joins the network and all data is transmitted w/out encryption
      - ↳ anyone on same network can see traffic & inject packets
    - WPA2-PSK
      - one password for all users (wifi password)
      - AP derives PSK (pre-shared key) by applying PKDF on SSID & pwd → slower by a large constant factor to make brute force attacks more difficult
      - when computer wants to connect to a network protected with WPA2-PSK, the user types in wifi pwd, then computer uses same PKDF to generate PSK
        - ↳ now each user needs unique encryption key (otherwise others on network can still decrypt your traffic)
          - ↳ AP and computer have handshake to generate shared encryption keys

can be optimized to only 4 msgs:
1. computer & AP exchange random nonces (unencrypted)
2. computer & AP independtly derive PTK (pairwise transport key) as function of nonces, PSK, and MAC addressess of computer & AP
3. computer & AP exchange message integrity codes (MICs) to ensure no one tampered with nonces
4. AP encrypts GTK (group temporal key) & sends it to computer
5. computer sends ACK (acknowledgement msg) to indicate it received GTK

Now, all future communication between computer & AP is encrypted with PTK (GTK is same for everyone on network, so used for broadcast msgs)

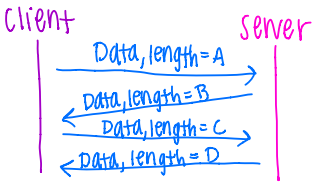
- DHCP (Dynamic Host Configuration Protocol)
  - DHCP → on layer 2-3
    → get configurations when first connecting to a network (these settings enable communication over LANs and the internet)
    ↳ vulernability: on-path attackers can see requests & send spoofed malicious responses ↳ race condition!
      ↳ Defense: Accept as fact of life & rely on higher layers
  - to connect to a network, need: IP address, IP address of DNS server, IP address of router (own)
  - DHCP Handshake (client (you) & server (has needed IP addresses))
    ① Client Discover: client broadcasts a request for configuration
    ② Server offer: any server able to offer IP addresses responds w/ some config. settings (DHCPlease)
    ③ Client Request: client broadcasts which configuration it has chosen
    ④ Server Acknowledge: the chosen server confirms that its configuration has been chosen
  - NAT (network address translation) → allows multiple computers on local network to share an IP address
- IP Routing: BGP (Border Gateway Protocol)
  - BGP → on layer 3
    → send msgs globally by connecting lots of local networks
    ↳ vulernability: malicious local networks can read msgs in intermeadiate transit & forward them to wrong place
      ↳ Defense: Accept as a fact of life & rely on higher layers
  - subnets
    ↳ IP routes by subnets (groups of addresses with a common prefix)
    - general internet (between LANs) is composed of many AS (autonomous systems)
      ↳ each AS consists of one or more LANs managed by an organization
  - determines routing between ASs on the Internet
    - has each AS advertise which networks it is responsible for to its neighboring ASs
    - each neighbor advertises that they can process packets to that network and provides info about the AS path that the packets would follow
    - continue until entire Internet is connected into a graph with many paths between ASs
- Transport Layer: TCP, UDP
  - TCP/UDP → on layer 4
    → establish connections between individual processes on machines (TCP & UDP)
    guarantee that packets are delivered successfully and in the correct order (TCP only)
    ↳ vulernability: On-path & MITM attackers can inject data or RST packets; off-path attackers must guess the 32-bit sequence number to inject packets
      ↳ Defense: Rely on cryptography at a higher layer (TLS); use randomly generated sequence numbers to stop off-path attackers → force them to guess the correct sequence # with very low probability
  - TCP guarantees reliable, in order packet delivery while UDP does not
    ↳ both use port numbers to support communication between processes
  - UDP (user datagram protocol)
    - best effort transport layer protocol
    - applications send & receive discrete packets but packets are not guaranteed to arrive
    - Header

16 bits	16 bits
source port	destination port
Length	checkSum

  - TCP (transmission control protocol)
    - reliable, in order connection based stream protocol
    - client first establishes a connection to the server by performing a handshake
      - TCP handles resending dropped packets until they are received on the other side and rearranging any packets received out of order
      - breaks long msgs into individual packets
    - Header
      - ACK (keeps track of missing/out of order packets)
      - flags: SYN, ACK, FIN are special TCP packets

16 bits	16 bits
source port	destination port
sequence number	
Acknowledgement Number	
Flags	checksum

- unique TCP connection = (client IP address, client port, server IP address, server port, protocol = TCP)
- TCP connection consists of 2 bytestreams of data
  ① client → server
  ② server → client
  - data in each stream is indexed using sequence numbers
    ↳ need 2 sets of sequence numbers



sequence	ACK
X+1	Y+1
Y+1	X+1+A
X+1+A	Y+1+B
Y+1+B	X+1+A+C

- sequence # in header = index of first byte sent in that packet
- to ensure packets are successfully delivered, when one side receives a TCP packet, it must reply with an acknowledgement saying it received the packet
  ↳ each TCP packet can contain both data and an acknowledgement that a previous packet was received
- ACK # in header = index of last received byte + 1
- TCP Handshake → to exchange random initial sequence #s
  ① client sends SYN packet (no data, SYN flag set) to server w/ sequence # field set to random 32-bit initial sequence # (ISN)
  ② if sever accepts request, it sends back a SYN-ACK packet (no data, SYN & ACK flags set) w/ sequence # field set to its own random 32-bit initial sequence # and acknowledgement # set to client's ISN + 1
  ③ client responds with ACK packet w/ sequence # = client's ISN+1 & acknowledgement # = server's ISN + 1

  To end a connection, one side sends a FIN packet & other side replies w/ FIN-ACK
  ↳ side who sent FIN will not send anymore but can still accept
  ↳ when other side also sends FIN & gets FIN-ACK reply, then connection closed
  Alternatively, if one side sends a RST packet w/ proper sequence #, this terminates connection unilaterally (usually indicates smthg went wrong)

- TCP versus UDP
  - TCP is slower but has better correctness guarantees
  - UDP is generally used when speed is a concern
    ↳ better to miss packet than to delay

- TLS (Transport Layer Security) → provides an end-to-end encrypted communication channel (guarantees that even if any one part of the communication chain is compromised, no one except sender/receiver is able to read/modify data being sent)
  ↳ layer 6.5 protocol, built on top of layer 4 TCP
    ↳ provides secure communications to layer 7 applications

- TLS Handshake
  - starts with TCP Handshake
  - in first msg to server, client sends a random # R_B and a list of encryption protocols it supports
  - in second msg (response from server to client), server gives its own random # R_S (protect against replay attacks), the selected encryption protocol & the server's certificate (contains copy of server's public key signed by a certificate authority (CA)) → attacker can spoof response w/ correct server's certificate if they went through TLS handshake with this server themselves
    ↳ client now has trusted copy of server's public key
  - generate a random premaster secret (PS) known to only the client & the server
    ① RSA version
      - client generates random PS, encrypts it with server's public key, and sends it to the server, which decrypts using its private key (only actual server can decrypt PS cuz only they have decryption key)
    ② DH key exchange
      - regular DH but server signs its half of exchange with its secret key (- server sends $(g, p, g^a \bmod p)$ w/ signature → can authenticate
      - client sends $g^b \bmod p$)