Module 5 : Network Attacks II

1) Sniffing → threat of eavesdropping
2) Spoofing → IP address, host on n/w can impersonate as another host.
3) Session Hijacking → attacker fully controls connection b/w 2 entities (man in the middle attack)
4) DOS and DDOS → (Denial of service) (Distributed denial of service) services are brought down by attackers so that legitimate users are not able to access the service
5) Connection & BW flooding → specific DOS & DDOS flooding host with too many connection request / (BW/traffic) sent
6) DNS Attacks

Lecture 1 : Sniffing

Interconnection devices :

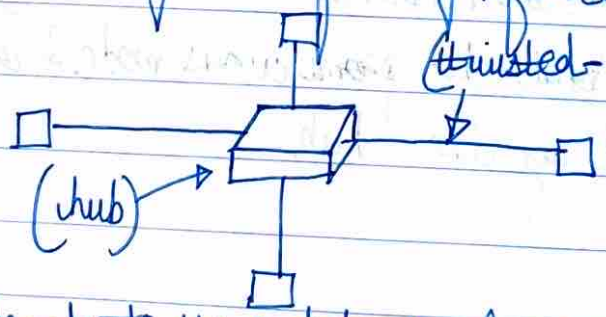
1) Hubs 2) Switches 3) Routers (Routing packets within a n/w & across a n/w)

- Hubs: often used in local-area network to transmit packets from local-servers to host within the n/w
- Routers: Routing package across to diff n/w's

Hubs → (dumb router) (preliminary type of router)

Hubs are essentially physical-layer repeaters :

* bits coming from one link go out to all other links. (no notion of routing table mapping etc)
* at the same rate host ports
* no frame buffering
* no CSMA/CD at hub : adapters detect collisions
* provides net management ~~facility~~ functionality.



* problem → broadcasts the packet, so also received by other hosts (intended only for 1 host, but goes to other 2 entities as well)

* the other two entities can act as sniffers because if they set their n/w cards in promiscuous mode, they receive all the messages/sniff messages b/w Alice & Bobs.

Sniffing

* Attacker is inside firewall.
* Requirements:
  - Attacker's host connected to shared medium.
  - NIC should be in "promiscuous mode". (Attacker has to set their n/w card to promiscuous mode, so it can capture all packets sent to their machine)
    - processes all frames that come to NIC
* Sniffer has two components:
  - o Capture
  - o packet analysis (pattern, info)
* Grab & file away:
  - * userids & passwords
  - * credit card numbers
  - * secret email conversations
  - * Island hopping attack:
    - Take over single machine (virus)
    - Install sniffer, observe passwords, take over more machines, install sniffers
* Sniffing easy when hubs are used in LAN, just have to set your card to promiscuous mode & capture all packets broadcasted by the hub.

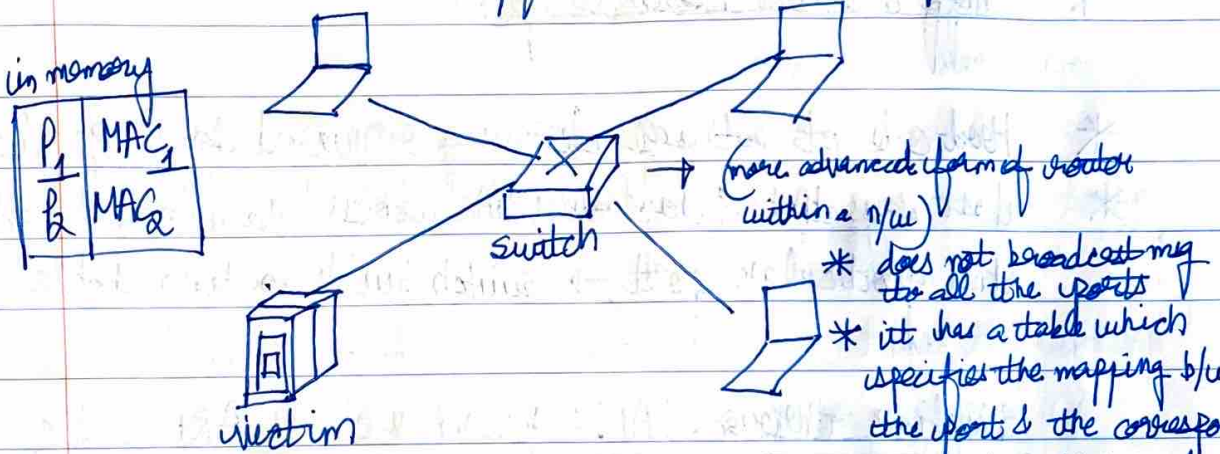
Sniffing itself is passive

Passive Sniffing → attacker sets card to promiscuous mode & listening on the communication

* Easy to sniff:
  - 802.11 traffic (wireless n/w as broadcasting is involved)
  - Ethernet traffic passing through the hub.
    - Any packets sent to hub is broadcasted to all interfaces
    - Not true for switch.
* Popular sniffers
  - wireshark, tcpdump (for unix), snort (sniffing & intrusion detection but also does (also for diagnostic purpose)

Active sniffing through a switch

How does attacker sniff packets sent to/from the victim?



→ (more advanced form of router within a n/w)

* does not broadcast msg to all the ports
* it has a table which specifies the mapping b/w the ports & the corresponding MAC address that is located at that port
* it receives packet knows its intented to which MAC & sends msg over there (knows which PORT corresponds to that MAC address & sends msg over there) rather than sending it in all directions which happens for a hub.

* Have to get victims packet

* Sniffing with switch is harder as msg is not getting broadcasted in all directions.
* Attacker has to do something more active/proactive to sniff conversation b/w 2 hosts that are connected by switch!

Active sniffing → attacker inserted fake addresses in switch's memory table, able to switch → HUB & eavesdrop communication.

Sniffing through a switch: flooding switch memory approach

→ exploit self-learning mechanism often employed in machines that use switch.

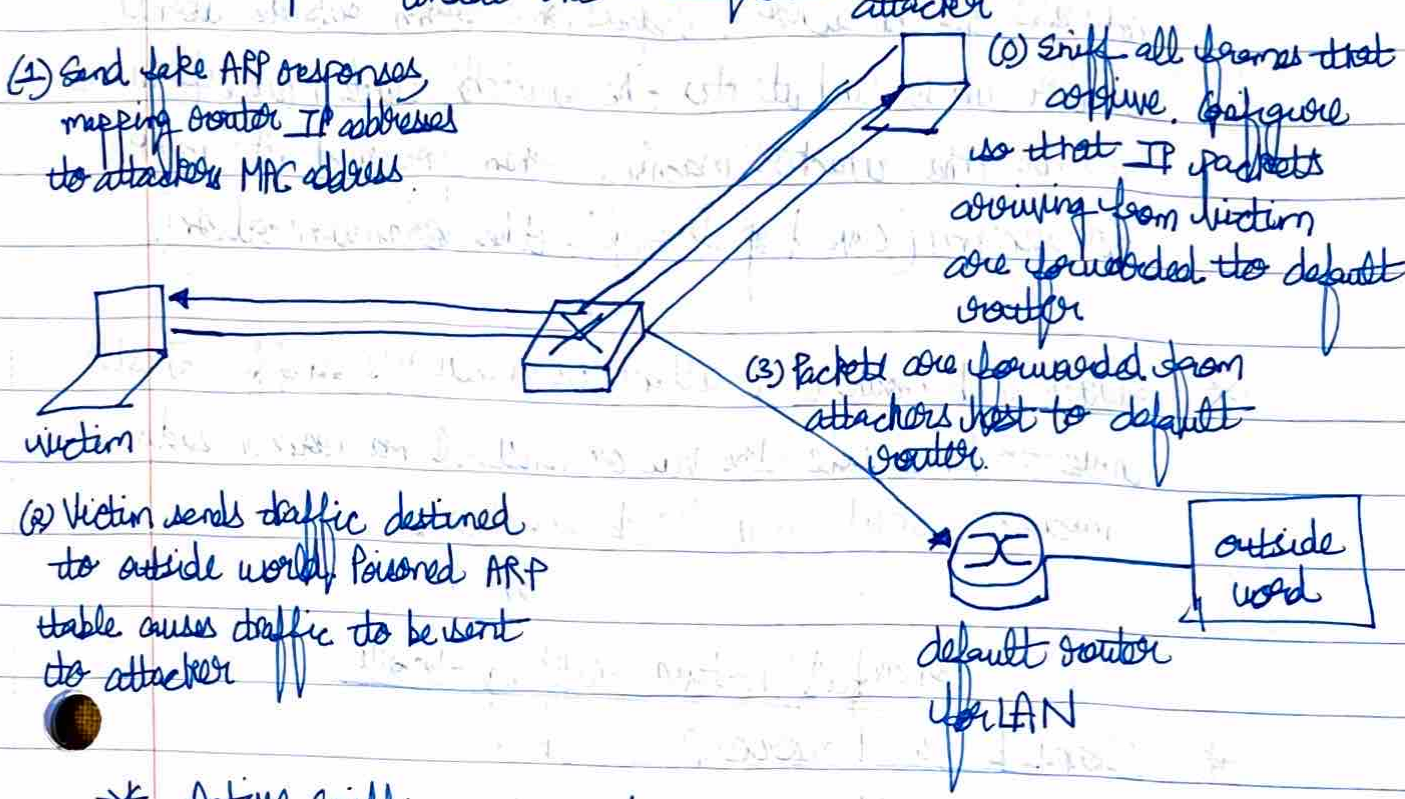
* Host sends flood of frames with random source MAC address
  - Switch's forwarding table gets filled with bogus MAC addresses
  - When "good packets arrive", destination MAC address not in switch memory ⇒ (falls into HUB mode)
  - Switch broadcasts real packets to all links
* Sniffs all the broadcast packets.

* Hard-code it but self-learning required for large n/w's
* Host says that I have this MAC address & I am located at this particular port. → switch builds routing table

Sniffing through LAN: poison victim's ARP table approach

Idea → have clients traffic diverted to attacker

* Route packets outside the n/w → Router
  a " inside the " → Switch



* Active sniffing approach.
* ARP → address resolution protocol. provides mapping b/w IP-addresses & corresponding MAC addresses.
* If victim has to send packet to outside n/w, he has to know what IP-address maps to MAC address of router.
* Poison the ARP ~~table~~ cache/table maintained by the victim ARP has entry corresponding to IP address of router & its corresponding MAC address. Attacker sends unsolicited msge saying IP-address of the router corresponds to MAC address of the attacker. victim updates these entries into the ARP table ($IP_{router} \leftrightarrow MAC_{attacker}$)

→ go through switch, to attacker. Attacker can forward this msg to the router & transmit to outside world. (eavesdrop)

(REVERSE ATTACK)

* Attacker can also poison the ARP of the router telling that IP address of victim corresponds to MAC address of attacker. replies come from outside world, router will send it to the switch, switch will forward it to the attacker's machine, then forward it back to victim (eavesdrop & sniff the communication)

* switch, hub, router → attacker can always sniff. Just have to be inside the n/w or install malware inside machine inside n/w to do sniffing.

Powerful Active Sniffing tools

* Dsniff & ettercap
→ Flooding switch memory
→ ARP poisoning

(IMP) Sniffing defenses:

1) Encrypt data: IPsec, SSL, PGP, SSH
2) use encryption for wireless (wireless is vulnerable to sniffing/eavesdropping as mgs are broadcasted)
3) Get rid of hubs: complete migration to switched network. (low cost but dumb routers)
4) Configure switches with MAC addresses
→ Turn off self-learning (knowing mapping b/w ports & MAC addresses)
→ Eliminate flooding problem

5) Intrusion detection systems:

→ Lookout for large numbers of ARP replies. (unsolicited) from host becoz that could be purposed for ARP spoofing/poisoning

6) Honeypot:

→ Create fake account & send password over n/w.

→ Identify attacker when it uses the password.

* prevention is best, use Encryption.

Lecture 2: Spoofing → host tries to spoof the address corresponding to other host in order to impersonate as another host (M as A)

(I) IP address spoofing

* Attacker doesn't want actions traced back.
* simply reconfigure IP address in windows or Unix.
* or enter spoofed address in an application.

eg → decoy packets with Nmap

* Change its IP-address (145.13.145.67) to whatever IP-address its targetting in the source address (SA: 36.220.9.59) field of the IP-packet and then send it over to the other guy. The other guy will think that this packet came from the S.A which the attacker spoofed.

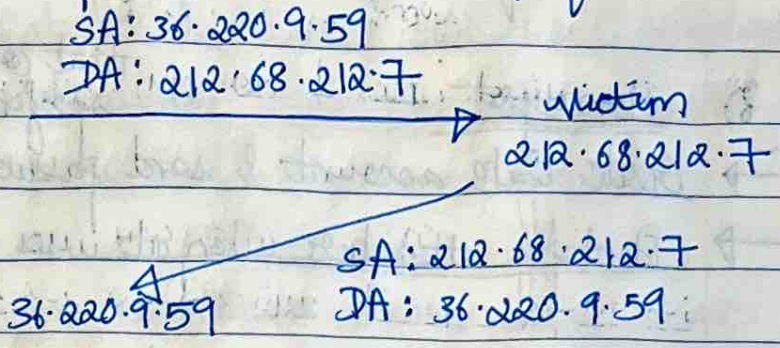
SA: 36.220.9.59

DA: 212.68.212.7

145.13.145.67 → (212.68.219.7)

(II)

* But, attacker cannot interact with the victim.
→ Unless attacker is on path b/w victim and spoofed address



Useful
* gain access control based on IP-address spoofing
* Not allows interactive communication (sends msg to spoofed IP address & not the attacker)
* Useful for one-side/one-time communication.

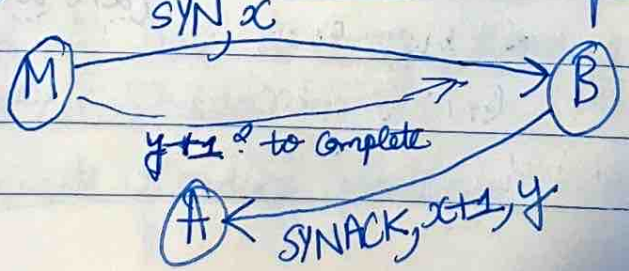
IP spoofing with TCP:

* Can an attacker make a TCP connected to server with a spoofed IP-address?
* Not easy: SYNACK and any subsequent packets sent to spoofed address. (not to the attacker, cannot know sequence no & ACK it back to the party to complete the handshake)
* If attacker can guess initial sequence number, the attacker can attempt to send commands.

Handshaking protocol will not be complete

→ send ACK with spoofed IP & corrected seq# say, one second after SYN.

* But, TCP uses random initial sequence number.



* IP address spoofing not sufficient to establish a TCP connection with the machine.
* good for non-interactive protocols like UDP but not good for interactive protocols like TCP.
* IP spoofing useful to hide IP-address of traces of attacker
  ↳ when you do Nmap

Defense: Ingress and egress filtering

Egress → firewall will analyze source address of all packets leaving the n/w.

Ingress → firewall is going to analyze all the packets & their source addresses (SA embedded in the packet) and then only forward legitimate packets inside to the n/w.

* Ingress filtering is more popular & highly used in practise

(I) Ingress filtering: upstream ISP

* a works by collaboration b/w different ISP's.
* specifying each other what are the different n/w's that each ISP is serving.

original
BGP (Border gateway protocol) update sent by ISP1 to.
Border gateway router in tier-I ISP and tell them that I am governing these particular networks

* Ingress filtering → filter all but 12.12/24 & 34.34/24
* similarly, BGP router communicating with original ISP2 will filter out all but 56.56/24 & 78.78/24
* Block this packet & lower impact of IP-spoofing.

[any packets that tier-I IP receives will have S.A of this form, then only forward it to the n/w]

* Allow attackers to spoof packets within these n/w's
* ∴ lower the impact & not fully solve the problem.

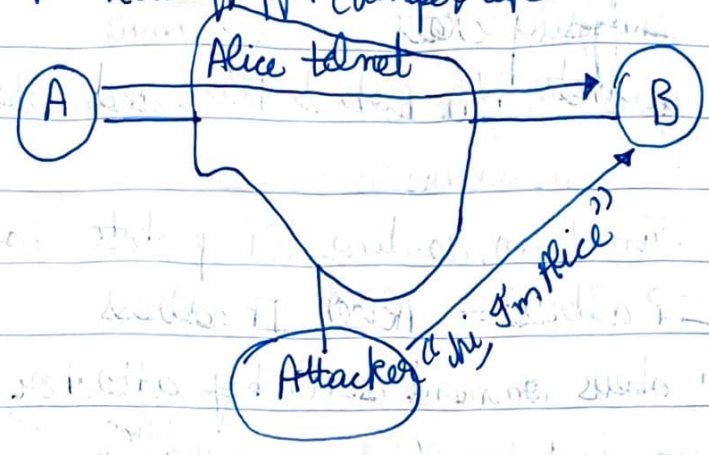
Ingress filtering : Summary

* Effectiveness depends on widespread deployment at ISP's
* Deployment in upstream ISP helps, but does not eliminate IP spoofing
  → Filtering can impact router forwarding performance. (cost associated with it)
* Even if universally deployed at access, hacker can still spoof another address in its access n/w. 12.1a/24
* see RFC 2827: Network Ingress filtering : Defeating DDoS"

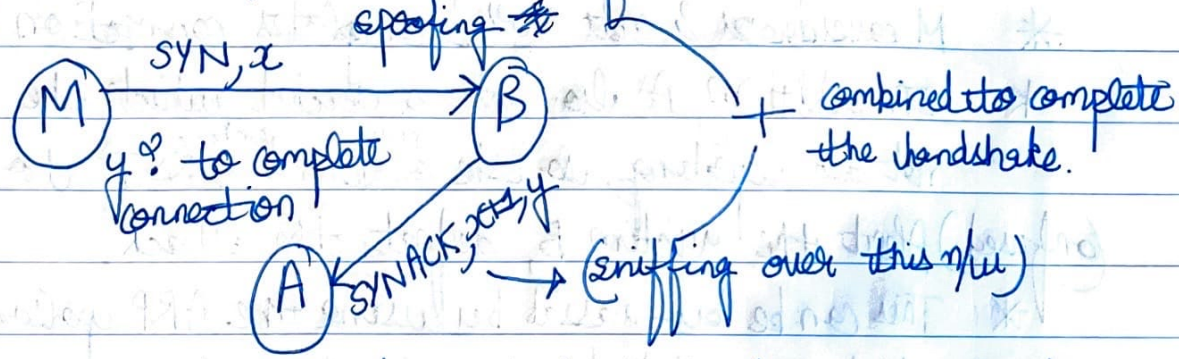
→ Use cryptography (Signature / Mac) to completely prevent spoofing / impersonification problem.

Lecture 3: Session Hijacking

* Take control of one side of TCP connection.
* Marriage of Sniffing & Spoofing
  (eavesdrop) + (temporal communication) → Send new packets / modify existing packets



* Man-in-the-middle attack
* Impersonate as Alice in front of Bob or impersonate as Bob in front of Alice. [works in both-direction]
* Just doing IP spoofing is not sufficient to compromise a TCP protocol. Study this in context of TCP.



* If attacker M can somehow get the "y" value, he can complete the connection. If M somehow eavesdrop the "y" value
* Attacker not only eavesdrop the packets, but also modifies the packets.

IMP

The details

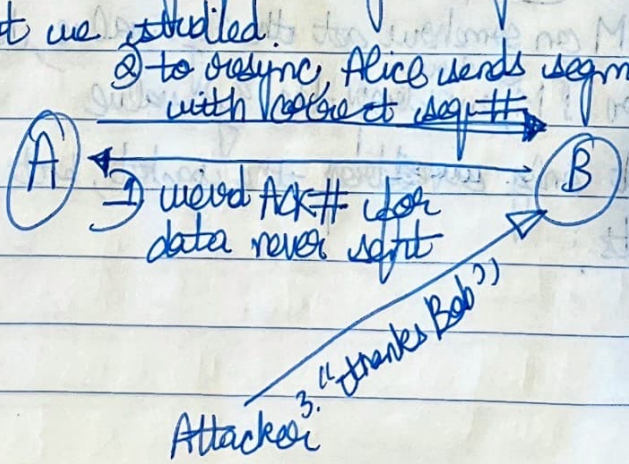
* Attacker is on segment where traffic passes from Alice to Bob.
  - → Attacker sniffs packets
  - → Sees TCP packets b/w Bob & Alice and their sequence#
* Attacker Jumps in, sending TCP packets to Bob; source IP address = Alice's IP address
  - → Bob now obeys command sent by attacker thinking they were sent by Alice.
* Principal defence : encryption + MAC (sniffing, spoofing)
  - → Attacker does not have key to encrypt/authenticate and insert meaningful traffic.

Limitations

* M eavesdrops & gets "y" & completes connection
* In addition A also gets a packet which she was not expecting, so she may send back "RST" to B.

(confused) Alert the victim B, defeat the attack.

* This can be by-passed by using the ARP poisoning that we studied.



* Bob is getting segments from attacker and Alice. Source IP address same but seq# different. Bob likely drops connection.

* Using ARP poisoning, M completely blocks the connection b/w A & B. ~~A sends a package~~, eg → A tries to reach B, packet does not reach Bob & vice versa.

* ~~Insert fake MAC~~ IP - address corresponding to ~~MAC~~ addresses of Alice & Bob

↳ Attacker's Solution

- Send unsolicited ARP replies to Alice & Bob with non-existent MAC addresses.
- overwrite IP-to-MAC ARP table.
- Alice's segment will not reach Bob and vice-versa.
- But attacker continues to hear Bob's segment, communicates with Bob.

* Sniffing + Spoofing + ARP-poisoning
* complete compromise of session b/w A & B
* A & B think they are talking to eachother, but everything is passing through M.

Session-Hijacking tools

* Hunt
⇒ ~~*~~ Provides ARP poisoning
* Netcat
⇒ General purpose widget
⇒ very popular

Lecture 4: DoS and DDoS

* Denial of service / Distributed " " "
  (bring down a service so legitimate users can't use it)

⟹ Prevent ~~illegitimate~~ access by legitimate users or stop critical system processes.

1) Implementation vulnerability Attack:

* Send a few crafted messages to target app that has vulnerability. (this lecture!)
* Malicious messages called the "exploit".
* Remotely stopping or crashing services.

→ Bug in service to bring it down

2) Connection flooding attack: (next lecture)

* Overwhelming connection queue with SYN flood

3) Bandwidth flooding attack:

* overwhelming communication links with packets.
* Strength in flooding attacker lies in volume rather than content.

DoS and DDoS:

1) DOS:
   - source of attack small # of nodes (10/20 machines to launch attack on web service/ service on web)
   - source IP typically spoofed
2) DDoS: (1 million)
   - from 1000's of nodes (Compromised machine bots)
   - IP addresses often not spoofed. (machines of day to day users so not spoofed)
3) Good book → Internet Denial of service by J. Mirkovic, D. Dittrich, P. Reiher, 2005

(principle same! bring down service & be as malicious as possible so that admin is not able to track you back)

DOS: eg's of vulnerability attacks

1) Land → sends spoofed packet with source and dest address/port the same.
2) Ping of death → sends oversized ping packet.
3) Jolt dl → sends a stream of fragments, none of which have offset of 0. Rebuilding consumes all processor capacity.
4) Teardrop, Newtear, Bonk, syndrop → tools send overlapping segments, that is, fragment offsets incorrect.

* Patches fix the problem, but malformed packets attack continue to be discovered.

LAND :

* Local area n/w denial
* spoofed SYN packet with source & destination both being the victim
* on receipt, victims machine keeps on responding to itself in a loop → causes the victim to crash
* Many OS are vulnerable eg
→ Windows 95, NT, XP SP2.
→ Mac OS MacTCP

Ping of Death

* ICMP Echo Request (Ping) is 56 bytes.
* If a ping msg is more than 65536 bytes (max of IP packet) this can cause some machines to crash.
* Older windows system.

Solution → Patch OS, filter out ICMP packets

"Teardrop" "Bonk" and kins → classical implementation based denial of service attack

- TCP/IP fragments contain offset field.
- Attacker sets offset field to:
  - overlapping values
  - ... or to very large value
- Bad/old implementation of TCP/IP stack crashes when attempting to re-assemble the fragments. (based on offset values)
  - Target system crashes [try to reassemble over & over, exhausts & crashes]

Bug → vulnerability easy to fix

Software developer were not careful when writing code & attackers can exploit

Solution → use up-to-date TCP/IP implementation

lecture: Connection and Bandwidth flooding

Denial of service attack

(I) Connection flooding: Overwhelming connection queue w/ SYN flood.

Handshaking

* Recall client sends SYN packet with initial seq. number when initiating a connection.
* TCP on server machine allocates memory on its connection queue to track the status of the new half-open connection.
* For each half-open connection, server waits for ACK segment using a timeout that is often > 1 min.
* Attack: Send many SYN packets, filling connection queue with half-open connections.
  → can spoof source IP address
* When connection queue is exhausted, no new connection can be initiated by legit users. (crashed)

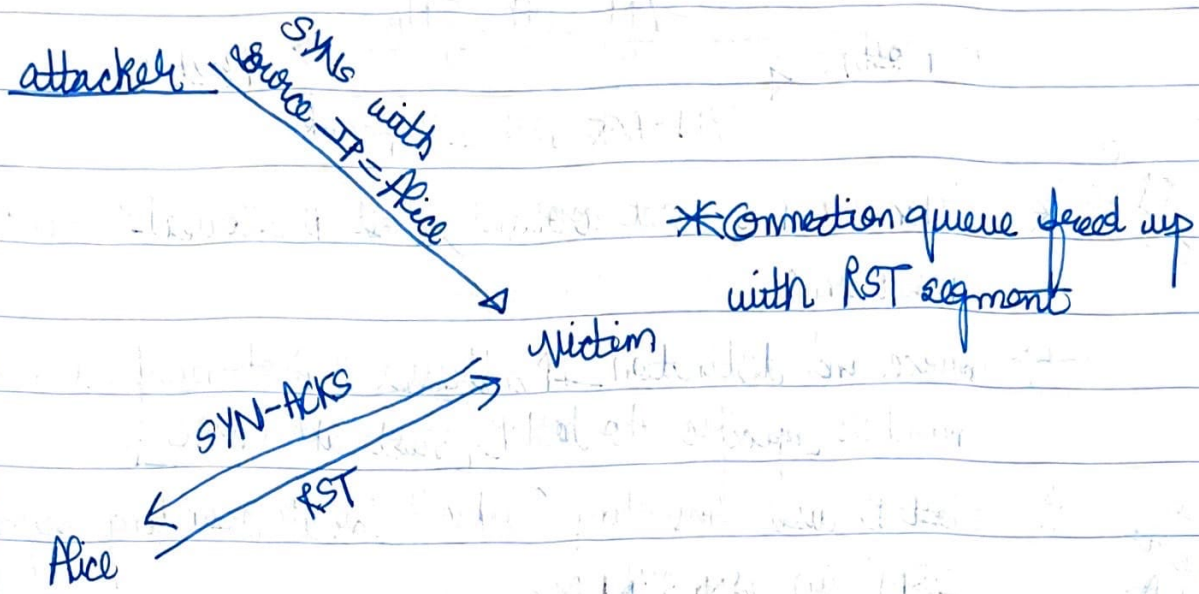
→ Need to know of open ports on victims machine: Port scanning.

* exploits asymmetric nature of TCP where server allocates both resources when half-open connection while client doesn't have any such notion of allocating resources. (can open up many connections & bring down the server)

SYN flooding Explained

* Attacker sends many connection requests (SYN) with spoofed source addresses.
* Victim allocates resources for each request
  → New thread, connection state maintained until timeout
  → Fixed bound on half-open connections.
* Once resources exhausted, requests from legitimate clients are denied.
* This is a classical denial of service attack
  → Common pattern: it costs nothing to TCP client to send a connection-request, but TCP server must spawn a thread for each request - asymmetry!
* Wants another eg of this behavior?
  ↳ 1) encripted email using PGP
  sender, creates encripted key, sends to recipient, it decrypts it.
  2) RSA → asymmetric Encryption → fast
  Decryption → much slower
  sender sends many emails to decrypt & potentially bring down recipient device.
  3) SSL → similar vulnerability (client ↔ server) (work work)
* vulnerability in terms of computation that client & server have to perform.

* Amateur attack



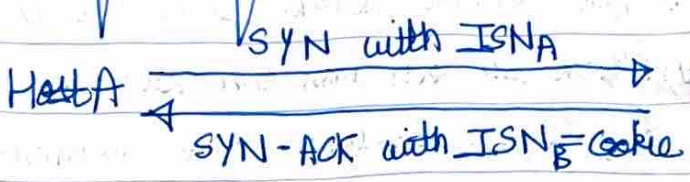
* Connection queue fixed up with RST segment

* Expert attack : use multiple source IP addresses each from unresponsive address.

Preventing Denial of service (SYN Flood)

* DoS is caused by asymmetric state allocation.
→ If server opens new state for each connection attempt, attacker can initiate many connections from bogus or forged IP addresses.
* Cookies allow server to remain stateless until client produces :
→ server state (IP addresses and ports) stored in a cookie and originally sent to client.
* when client responds, cookie is verified.
→ Do not have server allocate resources rather have the server make the connection on the fly by the use of the cookies.

SYN flood defense : SYN cookies (1)



IMP

* When SYN segment arrives, host B calculates function (hash) based on:
  - → Source and destination IP addresses & port-numbers and a secret number (specific to host B) [Just like HMAC]
* Host B uses resulting "cookie" for its resulting seq-# (ISN) in SYN-ACK
* Host B does not allocate anything to half-open connection:
  - → does not remember its ISN
  - → does not remember cookie.

Host B (server)

→ generates cookie on the fly, validates it & then only allocate resources.

sends cookie to Host A. when Host A sends back cookie in ACK to Host B, it compares it with its own generated cookie. match, then accept connection as valid & allocate resource

* coarse time → you don't want cookie from one session to be replayed to be the same as in another session ∴ use coarse level time (for hourly/minute basis) (unpredictable!!)
* Attacker should not be able to predict the cookie he can do IP-address spoofing, session IP hi-jacking etc cookie should be un-tempered

tamper-proof → by looking at many cookies, you should not be able to predict the cookie for a new session. will allow attacker to perform TCP level attacks

* cookie uses a key → server secret → HMAC/MAC computed on the addresses & ports of source/destination. ∴ cannot be forged. CBC-MAC, collision resistant MAC system.

(II) Overwhelming link bandwidth with packets

* Attack traffic can be made similar to legitimate traffic, hindering detection.
* flow of traffic must consume target's BW resources.
→ Attacker needs to engage more than one machine ⇒ DDOS
* May be easier to fill targets' get target to fill-up its upstream BW: async access
→ Eg: attacking BitTorrent seeds.

* Attacker tries to flood target machine with many many packets bombards service with too much traffic/BW, it crashes.
* level of information sent to device ↑, compromised machines (bots) millions, sent packets to this machine, it crashes.

DDoS: Reflection Attack

* DNS reflection attack.

(No Replaying Involved)

Spoofing ✓
Sniffing ✓
Stopping replies ✓

* sent many many DNS requests to many many DNS servers, the IP address source = victims machine. All responses are sent to victims device as source IP = victims device. victims device will get overwhelmed, so many responses from millions / trillions DNS servers, it gets overwhelmed & crashes.
* reflecting all traffic, route all traffic to victim & bring down the victim.

"Smurf Attack"

* exploit ICMP echo request/pinging mechanism.
* send ICMP echo request to a broadcast address in a given n/w with a spoofed IP-address of the victim
* so all the machines will reply with ICMP response to the victim, as
* sent to broadcast address, may have many many IP-address
* victim will get overwhelmed with responses & crash-down.

1. ICMP Echo Req Src: victim address Dest: broadcast address

Looks like legitimate "Are you alive" ping request from victim

3) Stream of ping replies overwhelms victim

2) Every host on the n/w generates a ping reply (ICMP echo reply) to victim

Solution → Reject external packets to broadcast addresses

DDoS Defenses:

1) Don't let your systems become bots (not get compromised)
   → keep system patched-up (Anti-malware software)
   → Employ egress anti-spoof filtering on external router.

2) Filter dangerous packets
   → Vulnerability attacks
   → Intrusion prevention system
      ↳ Differentiate b/w legitimate access & DoS attacks based on patterns.

3) Signature and anomaly detection & filtering [on a normal-day web service may not be receiving millions of request on a specific set of data say 2 mins.) while when attack happens set of traffic (lots of packets at a high rate, not a benign pattern)

4) Rate limiting
   → limit # of packets sent from source to dest. (limit them)

5) CAPTCHAs
   → could be useful against application level attacks (e.g. against web-servers)

* brute-force attacks against password dictionary if bot trying to force a attack, cannot solve the captcha & attack on victim.

* can be used at different layers, captcha's at application layer & other defense at web networking layer to provide chain defense against denial of service attack.

USE ALL TOGETHER

Lecture 6: DNS Attacks

→ Attacks against DNS Infrastructure.

1) Reflector attack: already discussed
   - Leverage DNS for attacks on arbitrary targets.
   - → exploiting DNS server for external attack. Now, see how DNS service can itself be exploited.
2) Denying DNS service
   - * Stop DNS root servers.
   - * Stop top-level-domain servers (eg. .com domain)
   - * Stop local (default name servers) (authoritative name servers)

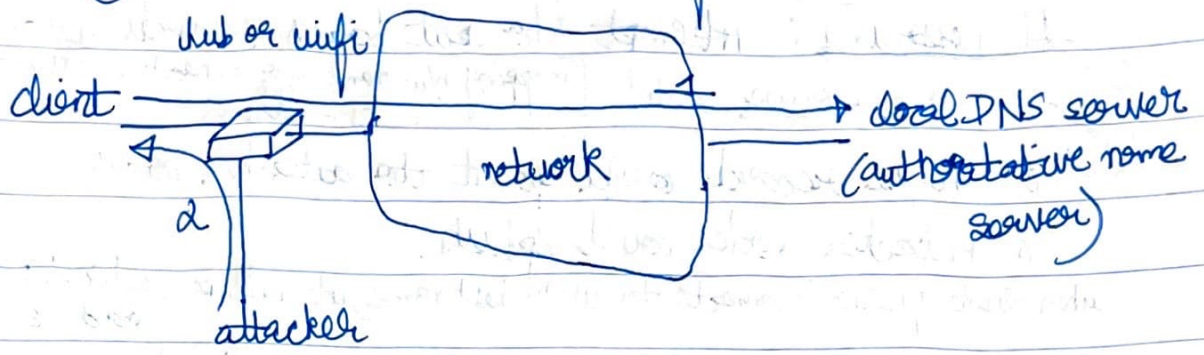
robust to these attacks due to distributed/hierarchical nature of DNS servers structure

(I) 3) Use fake DNS replies to redirect user name corresponding to host to IP address (attacker's machine)

(II) 4) Poisoning DNS:
   - Insert false resource records into various DNS cache.
   - False records contains IP-address operated by attackers
   - * mapping b/w name of host corresponds to IP address of attackers machine

poison such that →

(I) DNS attack: redirecting



1) client sends DNS query to its local DNS server; sniffed by attacker.

2) Attacker responds with bogus DNS reply. (which points the name for a given host to the IP address of attacker's machine)

Client think that name-to-address mapping is what is provided by attacker, then client connects to the IP address of attacker's machine (easy)

Issues: Must spoof IP address: set that local DNS server (easy)

- Must match reply ID with request ID (easy)
- May need to stop reply from the local DNS server (harder)
- Must sniff on queries made by the client.

(client will get confused as getting 2 replies.)

(raise suspicion & used to detect the attack)

* Can be done by using DoS attack against name servers.