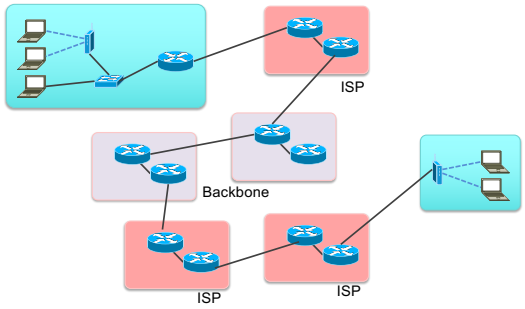
Computer Security

11. Network Security

Paul Krzyzanowski
Rutgers University
Spring 2018

The Internet

Packet switching: store-and-forward routing across multiple physical networks
... across multiple organizations

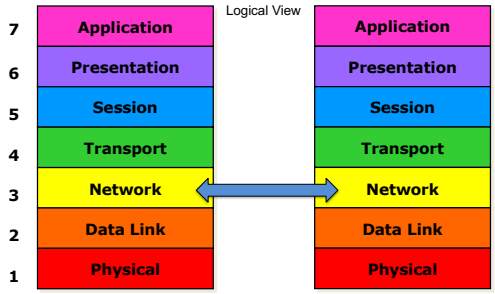


The Internet: Key Design Principles

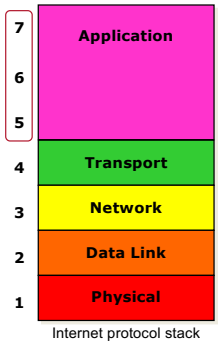
1. Support interconnection of networks
   - No changes needed to the underlying physical network
   - IP is a *logical network*
2. Assume unreliable communication
   - If a packet does not get to the destination, software on the receiver will have to detect it and the sender will have to retransmit it
3. Routers connect networks
   - Store & forward delivery
4. No global (centralized) control of the network

Protocol layers

Protocol layers communicate with their counterparts
Low-level attacks can affect higher levels



IP Protocol Stack



SMTP, IMAP, HTTP, FTP, …

BGP, DNS, NTP

TCP, UDP

IP

Ethernet MAC, 802.11, ARP

Internet protocol stack

Data Link Layer

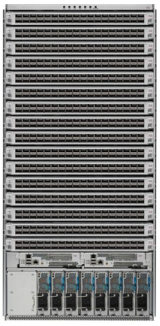
Data Link Layer (Layer 2)

Layer 2 generally has weak security

- MAC Attacks – CAM overflow
- VLAN Hopping
- ARP cache poisoning
- DHCP spoofing

Link Layer: CAM overflow

Layer 2: Ethernet Switches



Cisco Nexus 9516 Switch
- 1/10/40 GbE
- 21-rack-unit chassis
- Up to 576 1/10 Gb ports

TP-Link Switch
- 8 1-GbE ports

Ethernet MAC addresses

- Ethernet frames are delivered based on their 46-bit MAC address
  - Top 24 bits: manufacturer code assigned by IEEE
  - Bottom 24 bits: assigned by manufacturer
  - `ff:ff:ff:ff:ff:ff` = broadcast address

Ethernet MAC address ≠ IP address

How does an Ethernet switch work?

- A switch contains a switch table (MAC address table)
  - Contains entries for known MAC addresses & their interface
- Forwarding & filtering: a frame arrives for some destination address *D*
  - Look up *D* in the switch table to find the interface
  - If found & the interface is the same as the one the frame arrived on
    - Discard the frame (**filter**)
  - If found & *D* is on a different interface
    - **Forward** the frame to that interface: queue if necessary
  - If not found
    - **Forward** to ALL interfaces

The switch table

A switch is self-learning

- Switch table (MAC address → interface): initially empty
- Whenever a frame is received, associate the interface with the source MAC address in the frame
- Delete switch table entries if they have not been used for some time

Switches have to be fast: can't waste time doing lookups
- They use CAM – Content Addressable Memory
- Fixed size table

CAM overflow attack

Exploit size limit of CAM-based switch table

- Send bogus Ethernet frames with random source MAC addresses
  - Each new address will displace an entry in the switch table
  - *macof* tool: ~100 lines of perl
- With the CAM table full, legitimate traffic will be broadcast to all links
  - A host on any port can now see all traffic
  - CAM overflow attack turns a switch into a hub
- Countermeasures: port security
  - Some managed switches let you limit # of addresses per switch port

dsniff: collection of tools for network auditing and penetration testing
`https://monkey.org/~dugsong/dsniff/`

Link Layer: VLANs & VLAN hopping

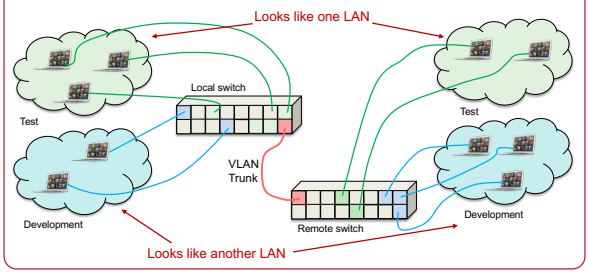
VLANs

- A switch + cables creates a local area network (LAN)
- We use LANs to
  - Isolate broadcast traffic from other groups of systems
  - Isolate users into groups
  - What if users move? What if switches are inefficiently used?
- Virtual Local Area Networks (VLANs)
  - Create multiple virtual LANs over one physical switch infrastructure
  - Network manager can assign a switch's ports to a specific VLAN
  - Each VLAN is a separate broadcast domain

VLAN Trunking

VLANs across multiple locations/switches

- VLAN Trunking: a single connection between two VLAN-enabled switches carries all traffic for all VLANs

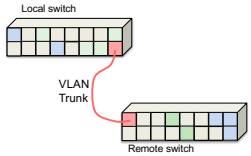


VLAN Hopping Attack

- VLAN trunk carries traffic for *all* VLANs
- Extended Ethernet frame format
  - 802.1Q for frames on an Ethernet trunk = Ethernet frame + VLAN tag
  - Sending switch adds VLAN tag for traffic on the trunk
  - Receiving switch removes VLAN tag and sends traffic to appropriate VLAN ports based on VLAN ID

Attack: **switch spoofing**

Devices can spoof themselves to look like a switch with a trunk connection and become a member of all VLANs



Avoiding VLAN Hopping

- Disable unused ports & assign them to an unused VLAN
- Disable auto-trunking
- Explicitly configure trunking on switch ports that are used for trunks

ARP Cache Poisoning (ARP Spoofing)

Find MAC address given an IP address

- We need to send a datagram to an IP address
- It is encapsulated in an Ethernet frame and a MAC address

- How do we know what MAC address to use?

Address Resolution Protocol (ARP)

- ARP table
  - Kernel table mapping IP addresses & corresponding MAC addresses
  - OS uses this to fill in the MAC header given an IP destination address
  - *What if the IP address we want is not in the cache?*
- ARP Messages
  - A host creates an ARP query packet & broadcasts it on the LAN
    - Ethernet broadcast MAC address: `ff:ff:ff:ff:ff:ff`
  - All adapters receive it
    - If an adapter's IP address matches the address in the query, it responds
    - Response is sent to the MAC address of the sender

ARP packet structure

see the **arp** command on Linux/BSD/Windows/OS X

ARP Cache Poisoning

- Network hosts cache any ARP replies they see … even if they did not originate them … on the chance that they might have to use that IP address
- Any client is allowed to send an *unsolicited* ARP reply
  - Called a **gratuitous ARP**
- ARP replies will overwrite older entries in the ARP table … even if they did not expire
- **An attacker can create fake ARP replies**
  - Containing the attacker's MAC address and the target's IP address
  - This will direct any traffic meant for the target to the attacker
  - Enables man-in-the-middle or denial of service attacks

See *Ettercap* – a multipurpose sniffer/interceptor/logger https://github.com/Ettercap/ettercap

Defenses against ARP cache poisoning

- Ignore replies that are not associated with requests
  - But you have to hope that the reply you get is a legitimate one
- Use static ARP entries
  - But can be an administrative nightmare
- Enable Dynamic ARP Inspection
  - Validates ARP packets against DHCP Snooping database information or static ARP entries

DHCP Server Spoofing

DHCP

- Computer joins a network – needs to be configured
  - *Broadcasts* a *DHCP Discover* message
- A DHCP server picks up this requests and sends back a response
  - IP address
  - Subnet mask
  - Default router (gateway)
  - DNS servers
  - Lease time
- Spoof responses that would be sent by a valid DHCP server

DHCP Spoofing

- Anybody can pretend to be a DHCP server
  - Spoof responses that would be sent by a valid DHCP server
  - Provide:
    - False gateway address
    - False DNS server address
- Attacker can now direct traffic from the client to go anywhere
- The real server may reply too
  - If the attacker responds first, he wins
  - Can delay or disable the real server: denial of service attack

Defenses

- Some switches (Cisco, Juniper) support **DHCP snooping**
  - Switch ports can be configured as "trusted" or "untrusted"
  - Only specific machines are allowed to send DHCP responses
  - The switch will use DHCP data to track client behavior
    - Ensure hosts use only the IP address assigned to them
    - Ensure hosts do not fake ARP responses

Network Layer (IP)

Network Layer: IP

Responsible for end-to-end delivery of packets

- No guarantees on message ordering or delivery
- Key functions
  - **Routing**
    - Each host knows the address of one or more connected routers (gateways)
    - The router knows how to route to other networks
  - **Fragmentation & reassembly**
    - An IP fragment may be split if the MTU size on a network is too small
    - Reassembled at its final destination
  - **Error reporting**
    - ICMP messages sent back to the sender (e.g., if packet is dropped)
  - **Time-to-live**
    - Hop count avoids infinite loops; packet dropped when TTL = 0

Source IP address

No source IP address authentication

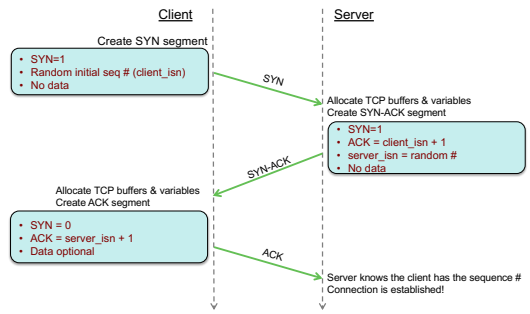
- Clients are *supposed* to use their own source IP address
  - Can override with raw sockets
  - Error responses will be sent to the forged source IP address
- Enables
  - Anonymous DoS attacks
  - DDoS attacks
    - Sent lots of packets from many places that will cause routers to generate ICMP responses
    - All responses go to the forged source address

Transport Layer (UDP, TCP)

TCP & UDP

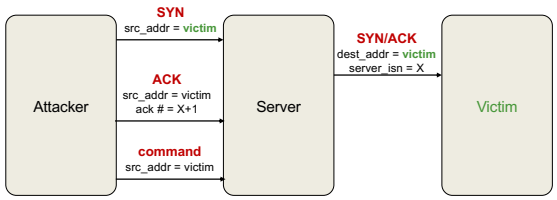
- UDP: User Datagram Protocol
  - Stateless, connectionless & unreliable
  - Anyone can send forged UDP messages
- TCP
  - Stateful, connection-oriented & reliable
  - Every packet contains a sequence number (byte offset)
    - Receiver assembles packets into correct order
    - Sends acknowledgements
    - Missing packets are retransmitted

TCP connection setup: three-way handshake



Why random initial sequence numbers?

If predictable, an attacker can create a TCP session on behalf of a forged source IP address



Random numbers make this attack harder – especially if the attacker cannot sniff the network

Denial of service: SYN Flooding

An OS will allocate only a finite # of TCP buffers

- SYN Flooding attack
  - Send lots of SYN segments but never complete the handshake
  - The OS will not be able to accept connections until those time out
- SYN Cookies: Dealing with SYN flooding attacks
  - Do not allocate buffers & state when a SYN segment is received
  - Create initial sequence # =
    hash(src_addr, dest_addr, src_port, dest_port, SECRET)
  - When an ACK comes back, validate the ACK #
    Compute the hash as before & add 1
  - If valid, then allocate resources necessary for the connection & socket

Denial of service: Reset

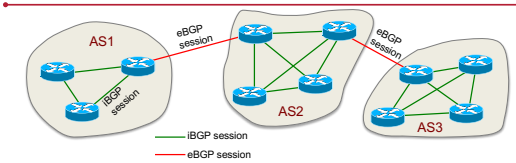
- Attacker can send a **RESET** (RST) packet to an open socket
- If the server sequence number is correct then the connection will close
- Sequence numbers are 32 bits
  - Chance of success is $1/2^{32} \approx 1$ in 4 billion
  - But many systems allow for a large range of sequence numbers
  - Attacker can send a flood of RST packets until the connection is broken

Routing Protocols

Routing protocols

- **OSPF: Open Shortest Path First**
  - Interior Gateway Protocol (IGP) within an autonomous system (AS)
  - Uses a **link state routing algorithm** (Dijkstra's shortest path)
- **BGP: Border Gateway Protocol**
  - Exterior Gateway Protocol (EGP) between autonomous systems (AS)
  - Exchanges routing and reachability information
  - **Distance vector routing protocol**

BGP sessions maintained via TCP links

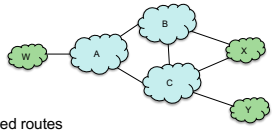


Pairs of routers exchange information via semi-permanent TCP connections

- One connection for each link between gateway routers
  - External BGP (eBGP) session
- Also BGP TCP connections between routers *inside* an AS
  - Internal BGP (iBGP) session

Route selection

- A, B, C: transit ASes – ISPs & backbone
- W, X, Y: stub ASes – customers



BGP route selection

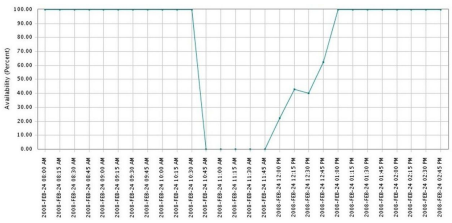
- Policies allow selection of preferred routes
- Otherwise, pick the route with the shortest path
- If there's a tie, choose the shortest path with the closest router

Security problems with BGP

- **Route advertisements are not authenticated**
  - Anyone can inject advertisements for arbitrary routes
  - Information will propagate throughout the Internet
  - Can be used for DoS or eavesdropping
- (Partial) Solutions — See RFC 6480
  - **RPKI** (Resource Public Key Infrastructure) framework
    - Each AS obtains an X.509 certificate from the Regional Internet Registry (RIR)
    - AS admin creates a Route Origin Authorization (ROA)
      - Associates the set of prefixes managed by that AS
    - ROA is signed by the AS's private key
    - Advertisements without a valid, signed ROA are ignored
  - **BGPsec** — Still a draft standard
    - Integral part of BGP protocol
    - Each hop in the AS path is protected with a signature

Pakistan's attack on YouTube in 2008

- YouTube service was cut off the global web for over an hour
- Pakistan Telecom received a censorship order from the telecommunications ministry to block YouTube
  - The company sent spoofed BGP messages claiming to be the best route for YouTube's range of IP addresses



Pakistan's attack on YouTube in 2008

- Pakistan Telecom sent BGP advertisements that it was the correct route for 256 addresses in YouTube's 208.65.153.0 network
  - Advertise a /24 network
- That is a more specific destination than YouTube's broadcast, which covered 1024 addresses
  - YouTube advertised a /22 network
- Within minutes, all YouTube traffic started to flow to Pakistan
- YouTube immediately tried countermeasures
  - Narrowed its broadcast to 256 addresses … but too late
  - Then tried an even more specific group: 64 addresses
    Advertise a /26 network ⇒ priority over /24 routes
    - Routes for more specific addresses overrule more general ones
  - Route updates finally fixed after 2 hours

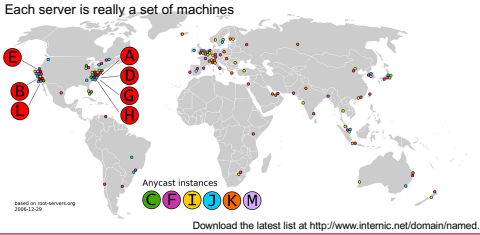
Domain Name System

Domain Name System

- Hierarchical service to map domain names to IP addresses
- How do you find the DNS Server for rutgers.edu?
  - That's what the domain registry keeps track of
  - When you register a domain,
    - You supply the addresses of at least two DNS servers that can answer queries for your zone
    - You give this to the domain registrar, who updates the database at the domain registry
- So how do you find the right DNS server?
  - Start at the root

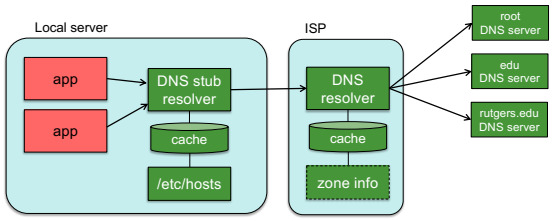
Root name servers

- The root name servers provide lists of authoritative name servers for top-level domains
- 13 root name servers
  - `A.ROOT-SERVERS.NET`, `B.ROOT-SERVERS.NET`, …
  - Each has redundancy (via *anycast* routing or load balancing)
    - Each server is really a set of machines



Download the latest list at http://www.internic.net/domain/named.root

DNS Resolvers in action



Local stub resolver:
- check local cache
- check local hosts file
- send request to external resolver

E.g., on Linux: resolver is configured via the /etc/resolv.conf file

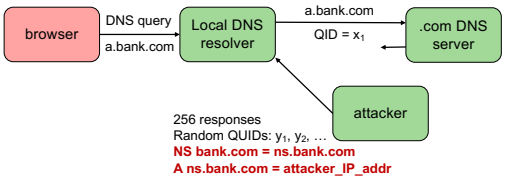
External resolver
- Running at ISP, Google Public DNS, OpenDNS, etc.

DNS Vulnerabilities

- **We trust the host-address mapping**
  - This is the basis for some security policies
    - Browser same-origin policy, URL address bar
- Each DNS query contains a Query ID (QID)
  - Response must have a matching QID
- Responses can be intercepted & modified
  - Malicious responses can direct messages to different hosts
- Solution: DNSsec
  - Secure extension to DNS that provide authenticated requests & responses
  - Few use it

DNS Cache Poisoning

JavaScript on a website may launch a DNS attacker



If there is some j such that $x_1 = y_j$ then the response is cached

All future DNS queries for anything at bank.com will go to attacker_IP_addr

If it doesn't work … try again with b.bank.com

Defenses against DNS cache poisoning

- Randomize source port # – *where DNS queries originate*
  - Attack will take several hours instead of a few minutes
- Issue double DNS queries
  - Attacker will have to guess the Query ID twice (32 bits)

DNS Rebinding

- **Same-origin policy**
  - Web application security model
  - Client web browser scripts can only access data from other web pages **only** if they have the same **origin**
  - Origin { URI, host name, port number }
- The policy relies on comparing domain names
- If we can change the underlying address:
  - We can access private machines in the user's local area network
  - Send results to attacker

DNS Rebinding

- **Attacker**
  - Registers a domain (attacker.com)
  - Sets up a DNS server
  - DNS server responds with very short TTL values – response won't be cached
- Client (browser)
  - Script on page causes access to malicious domain
  - Attacker's DNS server responds with IP address of a server hosting malicious client-side code
  - Malicious client-side code makes additional references to the domain
    - Permitted under **same-origin policy**
      - A browser permits scripts in one page to access data in another only if both pages have the same origin & protocol
    - The script causes the browser to issue a new DNS request
    - Attacker replies with a new IP address (e.g., a target somewhere else on the Internet)

Defending against DNS rebinding

- Force minimum TTL values
  - This may affect some legitimate dynamic DNS services
- DNS pinning: refuse to switch the IP address for a domain name
  - This is similar to forcing minimum TTL values
- Make sure DNS responses don't contain private IP addresses
- Server-side defense
  - Reject HTTP requests with unrecognized Host headers
  - Authenticate users

The end