Overview

- Security holes in IP stack
- Denial of service
- Capabilities
- Traceback

Basic IP

- End hosts create IP packets and routers process them purely based on destination address alone (not quite in reality)
- Problem – End host may lie about other fields and not affect delivery
  - Source address – host may trick destination into believing that packet is from trusted source
    - Many applications use IP address as a simple authentication method
    - Solution – reverse path forwarding checks, better authentication
  - Fragmentation – can consume memory resources or otherwise trick destination/firewalls
    - Solution – disallow fragments

Routing

- Source routing
  - Destinations are expected to reverse source route for replies
  - Problem – Can force packets to be routed through convenient monitoring point
    - Solution – Disallow source routing – doesn’t work well anyway!
Routing

- Routing protocol
  - Malicious hosts may advertise routes into network
  - Problem – Bogus routes may enable host to monitor traffic or deny service to others
- Solutions
  - Use policy mechanisms to only accept routes from or to certain networks/entities
  - In link state routing, can use something like source routing to force packets onto valid route
  - Routing registries and certificates

ICMP

- Reports errors and other conditions from network to end hosts
- End hosts take actions to respond to error
- Problem
  - An entity can easily forge a variety of ICMP error messages
    - Redirect – informs end-hosts that it should be using different first hop route
    - Fragmentation – can confuse path MTU discovery
    - Destination unreachable – can cause transport connections to be dropped

TCP

- Each TCP connection has an agreed upon/negotiated set of associated state
  - Starting sequence numbers, port numbers
  - Knowing these parameters is sometimes used to provide some sense of security
- Problem
  - Easy to guess these values
    - Listening ports #’s are well known and connecting port #’s are typically allocated sequentially
    - Starting sequence number are chosen in predictable way
  - Solution – make sequence number selection more random

Sequence Number Guessing Attack

Attacker → Victim: SYN(ISN_x), SRC=Trusted Host
Victim → Trusted Host: SYN(ISN_y), ACK(ISN_x)
Attacker → Victim: ACK(ISN_guess of y), SRC=Trusted Host
Attacker → Victim: ACK(ISN_guess of y), SRC=T, data = “rm -r /

- Attacker must also make sure that Trusted Host does not respond to SYNACK
- Can repeat until guess is accurate
TCP
- TCP senders assume that receivers behave in certain ways (e.g. when they send acks, etc.)
  - Congestion control is typically done on a “packet” basis while the rest of TCP is based on bytes
- Problem – misbehaving receiver can trick sender into ignoring congestion control
  - Ack every byte in packet!
  - Send extra duplicate acks
  - Ack before the data is received (needs some application level retransmission – e.g. HTTP 1.1 range requests)
- Solutions
  - Make congestion control byte oriented
  - Add nonces to packets – acks return nonce to truly indicate reception

DNS
- Users/hosts typically trust the host-address mapping provided by DNS
- Problems
  - Zone transfers can provide useful list of target hosts
  - Interception of requests or compromise of DNS servers can result in bogus responses
- Solution – authenticated requests/responses

Overview
- Security holes in IP stack
- Denial of service
- Capabilities
- Traceback

Denial of Service: What is it?
- Crash victim (exploit software flaws)
- Attempt to exhaust victim's resources
  - Network: Bandwidth
  - Host
    - Kernel: TCP connection state tables, etc.
    - Application: CPU, memory, etc.
- Often high-rate attacks, but not always
TCP Reminder: 3-Way Handshake

- C SYN → S SYN
- C SYN, ACK → S
- S ACK → C
- Connected
- Create TCB
- Wait
- Slide credit: Feamster

Example DoS: TCP SYN Floods

- Each arriving SYN stores state at the server
  - TCP Control Block (TCB)
    - ~ 280 bytes
      - FlowID, timer info, Sequence number, flow control status, out-of-band data, MSS, other options
  - Attack:
    - Send TCP SYN packets with bogus src addr
    - Half-open TCB entries exist until timeout
    - Kernel limits on # of TCBs
  - Resources exhausted ⇒ requests rejected

Preventing SYN floods

- Principle 1: Minimize state before auth
  - (3 way handshake == auth)?
- Compressed TCP state
  - Very tiny state representation for half-open conns
    - Don't create the full TCB
  - A few bytes per connection == can store 100,000s of half-open connections

SYN Cookies

- Idea: Keep no state until auth.
  - In response to SYN send back self-validating token to source that source must attach to ACK
  - SYN → SYN/ACK+token → ACK+token
    - Validates that the receiver's IP is valid
  - How to do in SYN? sequence #s!
    - top 5 bits: time counter
    - next 3: Encode the MSS
    - bottom 24: F(client IP, port, server IP, port, t)?
  - Downside to this encoding: Loses options.
Bandwidth Floods
- 1990s: Brute force from a few machines
  - Pretty easy to stop: Filter the sources
  - Until they spoof their src addr!
- Late 90s, early 00s: Traffic Amplifiers
  - Spoofed source addr (next)?
- Modern era: Botnets
  - Use a worm to compromise 1000s+ of machines
  - Often don't need to bother with spoofing

Reflector Attacks
- Spoof source address
- Send query to service
- Response goes to victim
- If response >> query, “amplifies” attack
- Hides real attack source from victim
- Amplifiers:
  - DNS responses (50 byte query → 400 byte resp)?
  - ICMP to broadcast addr (1 pkt → 50 pkts) (“smurf”)
Bandwidth DOS Attacks - Solutions

- Ingress filtering – examine packets to identify bogus source addresses
- Link testing – have routers either explicitly identify which hops are involved in attack or use controlled flooding and a network map to perturb attack traffic
- Logging – log packets at key routers and post-process to identify attacker’s path
- ICMP traceback – sample occasional packets and copy path info into special ICMP messages

Capabilities
- IP traceback + filtering

Spoofing 1: Ingress/Egress Filtering

- RFC 2827: Routers install filters to drop packets from networks that are not downstream
- Feasible at edges; harder at “core”

Spoofing 2: RPF Checks

- Unicast Reverse Path Forwarding
  - Cisco: “ip verify unicast reverse-path”
  - Requires symmetric routing

Secure Overlay Services

- Authenticate client communication
- Longer/slower route
- Closed network
Overview

- Security holes in IP stack
- Denial of service
- Capabilities
- Traceback

Capabilities

- Filters: prevent the bad stuff
- Capabilities: must have permission to talk
- Sender must first ask dst for permission
  - If OK, dst gives capability to src
  - capability proves to routers that traffic is OK
- Good feature: stateless at routers

Pi (Packet marking)

- Marking Scheme
  - Each router marks n bits into IP Identification field
- Marking Function
  - Last n bits of hash (eg. MD5) of router IP address
- Marking Aggregation
  - Router pushes marking into IP Identification field

Unforgeable Capabilities

- It is required that a set of capabilities be not easily forgeable or usable if stolen from another party
- Each router computes a cryptographic hash when it forwards a request packet
- The destination receives a list of pre-capabilities with fixed source and destination IP, hence preventing spoofed attacks
**TVA (Capability)**

PreCapability (Pi) = hash(srcIP, destIP, time, secret)

- RTS rate limited
  - 1-5% of bandwidth
- Pi Queue at Router
  - Most recent Pi

**Fine-Grained Capabilities**

- False authorizations even in small number can cause a denial of service until the capability expires
- An improved mechanism would be for the destination to decide the amount of data (N) and also the time (T) along with the list of pre-capabilities

<table>
<thead>
<tr>
<th>Capability (ncols)</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp (6 bits)</td>
</tr>
</tbody>
</table>

**TVA (Capability)**

Capability = timestamp || Hash (N, T, PreCap)

- N bytes, T seconds
- Stateless receiver
  - Does not store N, T

**Bounded Router State**

- The router state could be exhausted as it would be counting the number of bytes sent
- Router state is only maintained for flows that send faster than N/T
  - When new packets arrive, new state is created and a byte counter is initialized along with a time-to-live field that is decremented/incremented
**TVA**

- Routers put pre-capability in src→dst request
  - Timestamp | Hash(src, dst, time, router secret)?
  - secret changes slowly
  - dst sees these pre-capabilities and can echo them back to src if it wants to.
- Routers can verify pre-capability w/out state
- Limited time & b/w:
  - Timestamp | H(pre-caps, N bytes, Time T)?
  - dst gives src more N,T as appropriate

**Efficient Capabilities**

- In order to efficiently use the bandwidth, only a single set of capabilities are computed for the entire flow
- It is also required that for a secured set of capabilities, a longer set is used
- To further reduce the load on the network, only a random nonce is sent with the subsequent packets and the router caches the previous nonces and compares them

**Balancing Authorized Traffic**

- It is quite possible for a compromised insider to allow packet floods from outside
- A fair-queuing policy is implemented and the bandwidth is decreased as the network becomes busier
- To limit the number of queues, a bounded policy is used which only queues those flows that send faster than N/T
- Other senders are limited by FIFO service

**Short, Slow or Asymmetric Flows**

- Even for short or slow connections, since most byte belong to long flows the aggregate efficiency is not affected
- No added latency are involved in exchanging handshakes
- All connections between a pair of hosts can use single capability
- TVA experiences reduced efficiency only when all the flows near the host are short; this can be countered by increasing the bandwidth
Overview

- Security holes in IP stack
- Denial of service
- Capabilities
- Traceback

Filters & Pushback

- Assumption: Can identify anomalous traffic?
  - Add "filters" that drop this traffic
  - Access control lists in routers
    - e.g. deny ip from dave.cmu.edu to victim.com tcp port 80
- Pushback: Push filters further into network towards the source
  - Need to know where to push the filters (traceback)?
  - Need authentication of filters...
  - Tough problems. Filters usually deployed near victim.

The Need for Traceback

- Internet hosts are vulnerable
  - Many attacks consist of very few packets
  - Fraggle, Teardrop, ping-of-death, etc.
- Internet Protocol permits anonymity
  - Attackers can “spoof” source address
  - IP forwarding maintains no audit trails
- Need a separate traceback facility
  - For a given packet, find the path to source

Approaches to Traceback

- Path data can be noted in several places
  - In the packet itself [Savage et al.],
  - At the destination [I-Trace], or
  - In the network infrastructure
- Logging: a naïve in-network approach
  - Record each packet forwarding event
  - Can trace a single packet to a source router, ingress point, or subverted router(s)
IP Traceback

- Node append (record route) – high computation and space overhead
- Node sampling – each router marks its IP address with some probability $p$
  - $P$(receiving mark from router $d$ hops away) = $p(1 - p)^{d-1}$
  - $p > 0.5$ prevents any attacker from inserting false router
  - Must infer distance by marking rate → relatively slow
  - Doesn’t work well with multiple routers at same distance → i.e. multiple attackers

Edge Sampling

- Major problem – need to add about 72 bits (2 address + hop count) of info into packets
- Solution
  - Encode edge as xor of nodes → reduce 64 bits to 32 bits
  - Ship only 8 bits at a time and 3 bits to indicate offset → 32 bits to 11 bits
  - Use only 5 bit for distance → 8 bits to 5 bits
  - Use IP fragment field to store 16 bits
  - Some backward compatibility issues
  - Fragmentation is rare so not a big problem

Log-Based Traceback

- Edge sampling
  - Solve node sampling problems by encoding edges & distance from victim in messages
  - Start router sets “start” field with probability $p$ and sets distance to 0
  - If distance is 0, router sets “end” field
  - All routers increment distance
  - As before, $P$(receiving mark from router $d$ hops away) = $p(1 - p)^{d-1}$
  - Multiple attackers can be identified since edge identifies splits in reverse path
Challenges to Logging

• Attack path reconstruction is difficult
  • Packet may be transformed as it moves through the network
• Full packet storage is problematic
  • Memory requirements are prohibitive at high line speeds (OC-192 is ~10M pkt/sec)
• Extensive packet logs are a privacy risk
  • Traffic repositories may aid eavesdroppers

Solution: Packet Digesting

• Record only invariant packet content
  • Mask dynamic fields (TTL, checksum, etc.)
• Store information required to invert packet transformations at performing router
• Compute packet digests instead
  • Use hash function to compute small digest
  • Store probabilistically in Bloom filters
• Impossible to retrieve stored packets

Invariant Content

<table>
<thead>
<tr>
<th>Ver</th>
<th>HLen</th>
<th>TOS</th>
<th>Total Length</th>
<th>Identification</th>
<th>Fragment Offset</th>
<th>TTL</th>
<th>Protocol</th>
<th>Checksum</th>
<th>Source Address</th>
<th>Destination Address</th>
<th>Options</th>
<th>First 8 bytes of Payload</th>
<th>Remainder of Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>2n</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bloom Filters

• Fixed structure size
  • Uses 2n bit array
  • Initialized to zeros
• Insertion is easy
  • Use n-bit digest as indices into bit array
  • Mitigate collisions by using multiple digests
• Variable capacity
  • Easy to adjust
  • Page when full
Mistake Propagation is Limited

- Bloom filters may be mistaken
  - Mistake frequency can be controlled
  - Depends on capacity of full filters
- Neighboring routers won’t be fooled
  - Vary hash functions used in Bloom filters
  - Each router select hashes independently
- Long chains of mistakes highly unlikely
  - Probability drops exponentially with length

Adjusting Graph Accuracy

- False positives rate depends on:
  - Length of the attack path
  - Complexity of network topology
  - Capacity of Bloom filters
- Bloom filter capacity is easy to adjust
  - Required filter capacity varies with router speed and number of neighbors
  - Appropriate capacity settings achieve linear error growth with path length

Simulation Results

How long can digests last?

- Filters require 0.5% of link capacity
  - Four OC-3s require 47MB per minute
  - A single drive can store a whole day
- Access times are equally important
  - Current drives can write >3GB per minute
  - OC-192 needs SRAM access times
- Still viable tomorrow
  - 128 OC-192 links need <100GB per minute
Next Lecture

- Trust and Reputation
- Required reading:
  - SybilGuard: Defending Against Sybil Attacks via Social Networks