15-744: Computer Networking

L-22 Security and DoS

Overview

- Security holes in IP stack
- Denial of service
- Capabilities
- Traceback
  - Slides Jun Zhao

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

5

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

6

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

7

Sequence Number Guessing Attack

Attacker → Victim: SYN(ISN_x), SRC=Trusted Host

Victim → Trusted Host: SYN(ISN_s), ACK(ISN_x)

Attacker → Victim: ACK($ISN_{\text{guess of s}}$), SRC=Trusted Host

Attacker → Victim: ACK($ISN_{\text{guess of s}}$), SRC=T, data = "rm -r /"

- Attacker must also make sure that Trusted Host does not respond to SYNACK
- Can repeat until guess is accurate

8

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 comprise of DNS servers can result in bogus responses
  - Solution – authenticated requests/responses

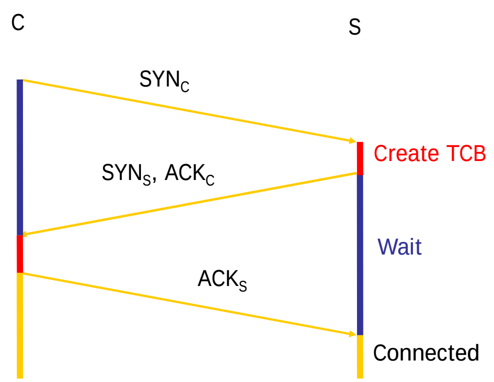
Overview

- 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



slide credit: Feamster

13

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

14

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

15

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.

16

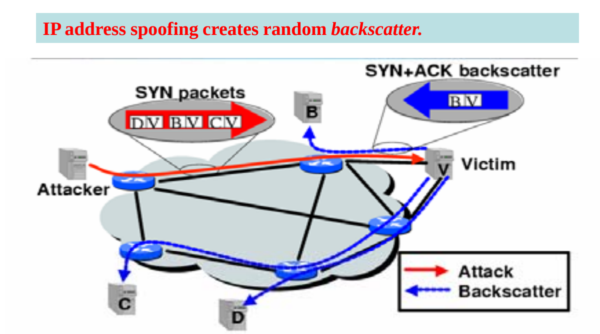
Bandwidth Floods

- 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 addrs (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”)

Inferring DoS Activity: Backscatter



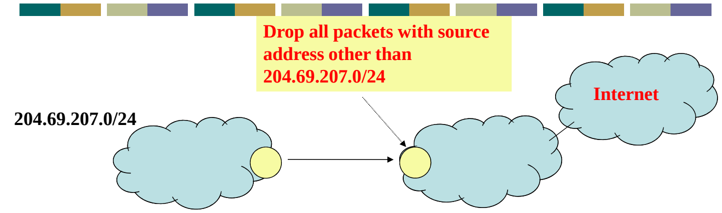
Backscatter Analysis

- Use a big block of addresses (N of them)?
  - People often use a /16 or /8
- Observe x backscatter packets/sec
  - How big is actual attack?
    - x * (2^32 / N)?
    - Assuming uniform distribution
- Sometimes called “network telescope”

Bandwidth DOS Attacks - Solutions

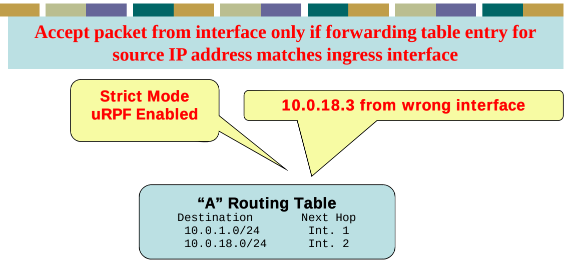
- Ingress filtering – examine packets to identify bogus source addresses
- Link testing – have routers either explicitly identify which hops are involved in attack or use controlled flooding and a network map to perturb attack traffic
- Logging – log packets at key routers and post-process to identify 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

Slide Credit: Feamster

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 capabilitiy to src
  - capability proves to routers that traffic is OK
- Good feature: stateless at routers

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

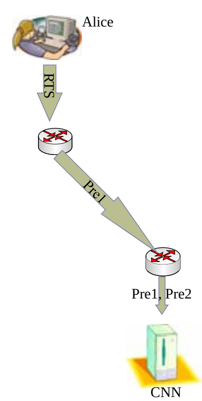
Pre-capability (routers)

timestamp (8 bits)	hash(src IP, dest IP, time, secret) (56 bits)

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

Capability (hosts)

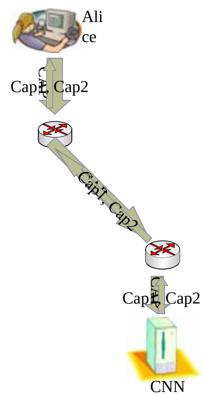
timestamp (8 bits)	hash(pre-capability, N, T) (56 bits)

TVA (Capability)

Capability =

timestamp || Hash (N, T, PreCap)

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



31

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

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

37

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.

38

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*

39

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)

40

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

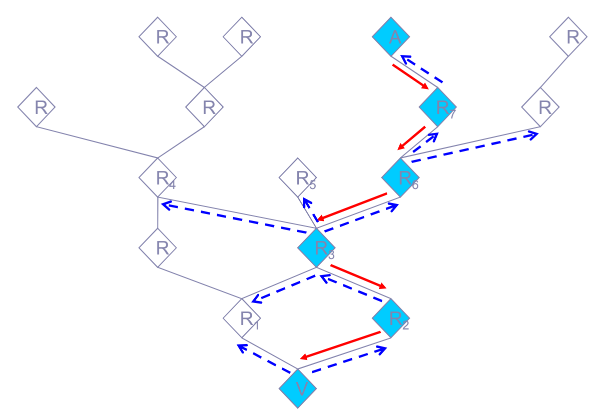
IP Traceback

- 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

Edge Sampling

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

Log-Based Traceback



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 ~10Mpkt/sec)
- Extensive packet logs are a privacy risk
  - Traffic repositories may aid eavesdroppers

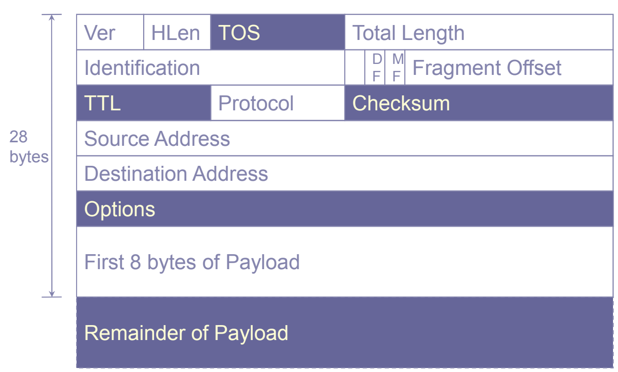
45

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

46

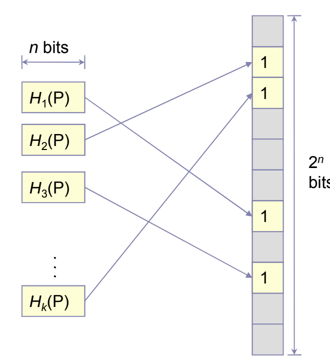
Invariant Content



47

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



48