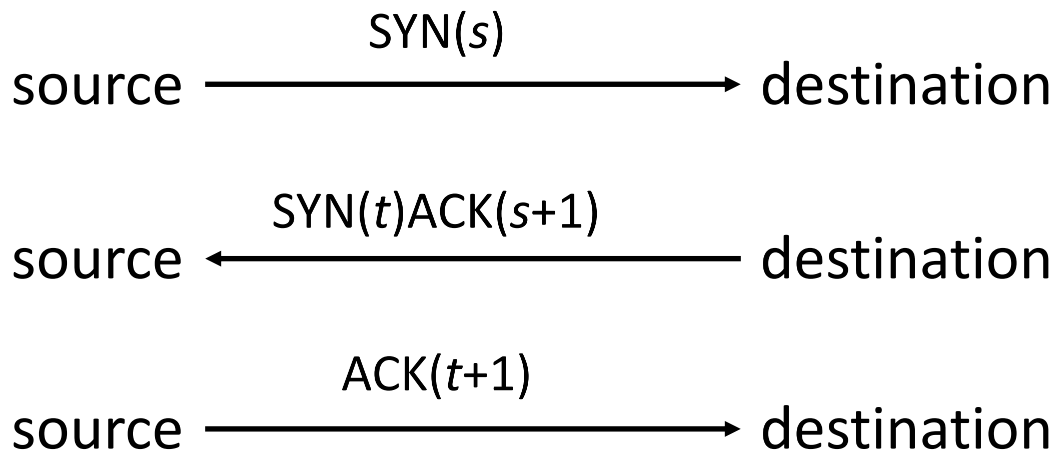
ECS 235B Module 27
Network Flooding

Availability and Network Flooding

- Access over Internet must be unimpeded
  - Context: flooding attacks, in which attackers try to overwhelm system resources
- If many sources flood a target, it's a *distributed denial of service attack*

TCP 3-Way Handshake and Availability



- Normal three-way handshake to initiate connection
- Suppose source never sends third message (the last ACK)
  - Destination holds information about pending connection for a period of time before the space is released

Analysis

- Consumption of bandwidth
  - If flooding overwhelms capacity of physical network medium, SYNs from legitimate handshake attempts may not be able to reach the target
- Absorption of resources on destination host
  - Flooding fills up memory space for pending connections, causing SYNs from legitimate handshake attempts to be discarded
- In terms of the models:
  - Waiting time is the time that destination waits for ACK from source
  - Fairness policy must assure host waiting for ACK (resource) will receive (acquire) it

Analysis in Terms of Model

- Waiting time is the time that destination waits for ACK from source
- Fairness policy must assure host waiting for ACK (resource) will receive (acquire) it
  - But goal of attack is to make sure it never arrives
- Yu-Gligor model: finite wait time does not hold
  - So model says denial of service can occur
- Millen model: T_p(ACK) > *max*(ACK)
  - *max*(ACK) is the time-out period for pending connections
  - So model says denial of service can occur

Countermeasures

- Focus on ensuring resources needed for legitimate handshakes to complete are available
  - So every legitimate client gets access to server
- First approach: manipulate opening of connection at end point
  - If focus is to ensure connection attempts will succeed at some time, focus is really on waiting time
  - Otherwise, focus is on user agreement
- Second approach: control which packets, or rate at which packets, sent to destination
  - Focus is on implicit user agreements

Intermediate Systems

- Approach is to reduce consumption of resources on destination by diverting or eliminating illegitimate traffic so only legitimate traffic reaches destination
  - Done at infrastructure level
- Example: Cisco routers try to establish connection with source (TCP intercept mode)
  - On success, router does same with intended destination, merges the two
  - On failure, short time-out protects router resources and target never sees flood

Track Connection Status

- Use network monitor to track status of handshake
- Example: *synkill* monitors traffic on network
    - Classifies IP addresses as not flooding (good), flooding (bad), unknown (new)
    - Checks IP address of SYN
        - If good, packet ignored
        - If bad, send RST to destination; ends handshake, releasing resources
        - If new, look for ACK or RST from same source; if seen, change to good; if not seen, change to bad
    - Periodically discard stale good addresses

Intermediate Systems near Sources

- D-WARD relies on routers close to the sources to block attack
  - Reduces congestion in network without interfering with legitimate traffic
- Placed at gateways of possible sources to examine packets leaving (internal) network and going to Internet
- Deployed on systems in research lab for 4 months
  - First month: large number of false alerts
  - Tuning D-WARD parameters reduced this number

D-WARD: Observation Component

- Has set of legitimate internal addresses
- Gathers statistics on packets leaving network, discarding packets without legitimate addresses
- Tracks number of simultaneous connections to each remote destination
  - Unusually large number may indicate attack from this network
- Examines connections with large amount of outgoing traffic but little incoming (response) traffic
  - May indicate destination host is overwhelmed

D-WARD: Observation Component

- Also aggregates traffic statistics to each remote address
- Classifies flows as *attack*, *suspicious*, *normal*
    - *Normal*: statistics match legitimate traffic model
    - *Attack*: if not
- Once traffic classified as attack begins to match legitimate traffic model, indicates attack has ended, so flow reclassified as *suspicious*
    - If it stays suspicious for predetermined time, reclassified as *normal*

D-WARD: Rate-Limiting Component

- When attack detected, this component limits amount of packets that can be sent
- This reduces volume of traffic going from this network to destination
- How it limits rate is based on D-WARD’s best guess of amount of traffic destination can handle
    - When flow reclassified as normal, D-WARD raises rate limit until sending rate is as before

D-WARD: Traffic-Policing Component

- Component obtains information from other 2 components
- Based on this, decides whether to drop packets
  - Packets for normal connections always forwarded
  - Packets for other flows may be forwarded provided doing so does not exceed rate limit associated with flow

Endpoint Protection

- Control how TCP state is stored
  - When SYN received, entry in queue of pending connections created
    - Remains until an ACK received or time-out
    - In first case, entry moved to different queue
    - In second case, entry made available for next SYN
  - In SYN flood, queue is always full
    - So, assure legitimate connections space in queue to some level of probability
    - Two approaches: SYN cookies or adaptive time-outs

SYN Cache

- Space allocated for each pending connection
  - But much less than for a full connection
- How it works on FreeBSD
  - On initialization, hash table (*syncache*) created
  - When SYN packet arrives, system generates hash from header and uses that to determine which bucket to store enough information to be able to send SYN/ACK on the pending connection (and does so)
    - If bucket full, oldest element dropped
  - If peer returns ACK, entry removed and connection created
  - If peer returns RST, entry removed
  - If no response, repeat fixed number of times; if no responses, remove entry

SYN Cookies

- Source keeps state
- How it works
  - When SYN arrives, generate number (*syncookie*) from header data and random data; use as ACK sequence number in SYN/ACK packet
    - Random data changes periodically
  - When reply ACK arrives, recompute syncookie from information in header
- FreeBSD uses this technique when pending connection cannot be inserted into syncache

Adaptive Time-Out

- Change time-out time as space available for pending connections decreases
- Example: modified SunOS kernel
  - Time-out period shortened from 75 to 15 sec
  - Formula for queueing pending connections changed:
    - Process allows up to *b* pending connections on port
    - *a* number of completed connections but awaiting process
    - *p* total number of pending connections
    - *c* tunable parameter
    - Whenever $a + p > cb$, drop current SYN message

Other Flooding Attacks

- These use *reflectors* (typically, infrastructure systems) to augment traffic, creating flooding
  - Attacker need only send small amount of traffic; reflectors create the rest
  - Called *amplification attack*
- Hides origin of attack, which appears to come from reflectors

Smurf Attack

- Relies on router forwarding ICMP packets to all hosts on network
- Attacker sends ICMP packet to router with destination address set to broadcast address of network
- Router sends copy of packet to each host on network
  - If attacker sends steady stream of packets, has the effect of sending that stream to all hosts on network
- Example of an *amplification attack*

DNS Amplification Attack

- Uses DNS resolvers that are configured to accept queries from any host rather than only hosts on their own network
- Attacker sends packet with source address set to that of target
    - Packet has query that causes DNS resolver to send large amount of information to target
    - Example: zone transfer query is a small query, but typically sends large amount of data to target, typically in multiple packets, each larger than a query packet

Pulse Denial of Service Attack

- Like flooding, but packets sent in pulses
  - May only degrade target's performance, but that may be enough of a denial of service
- Induces 3 anomalies in traffic to target
  - Ratio of incoming TCP packets to outgoing ACKs increases dramatically
    - Rate of incoming packets much higher than system can send ACKs
  - When attacker reduces number of packets to target, number of ACKS drop
  - Distribution of incoming packet interarrival time will be anomalous
- Vanguard detection scheme uses these 3 anomalies to detect pulse denial-of-service attack