Denial of Service Attacks

Denial of Service (DoS) Attacks

- Attack availability
- No direct benefit to the attacker, except for the victim's pain
- (But there are some exceptions)
- Major problem on today's Internet

History

- Most viruses and worms simply perpetrate DoS attacks
- The phone system has experienced prank DoS attacks
- Must distinguish attacks from "flash crowds", also known as the "Slashdot Effect"

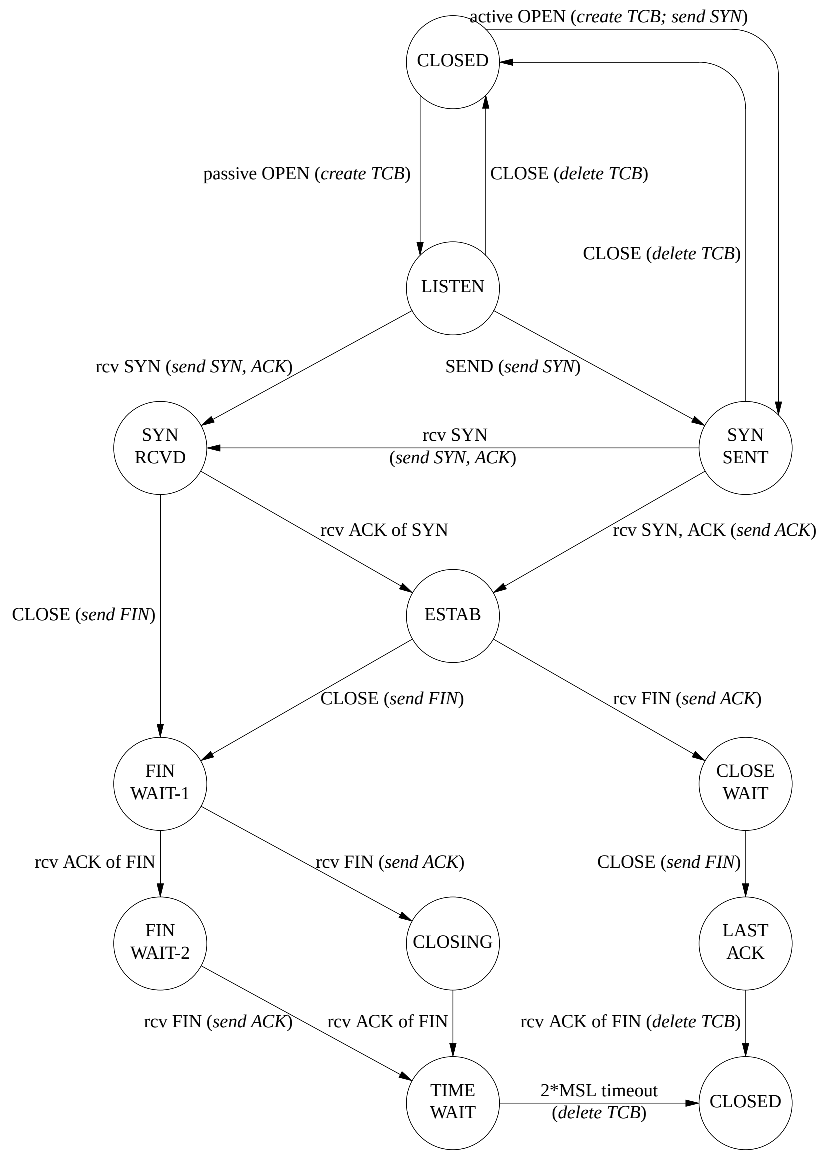
What Can be DoSed?

- Bandwidth — clog the link
- CPU time — make someone do expensive calculations
- Memory — tie up system state
- More generally, DoS can occur any time it costs less for an attacker to send a message than to process it

First Internet DoS Attack

- Attacker sends many SYN packets from a forged source address
- The SYN+ACK packets go nowhere
- No ACK to them ever arrives; the connection stays half-open
- Why is this a DoS?

The TCP State Diagram



SYN Flooding

- An arriving SYN sends the "connection" into `SYN-RCVD` state
- It can stay in this state for quite a while, awaiting the acknowledgment of the SYN+ACK packet, and tying up memory
- For this reason, the number of connections for a given port in `SYN-RCVD` state is limited
- Further SYN packets for that port are dropped
- The trick is the address forgery — if the attacker impersonates a non-existent host, neither the SYN+ACK nor a RST will ever arrive
- The port is thus blocked

Defenses

- Anti-spoofing
- Better data structures
- SYN cookies

Anti-Spoofing

- Conceptually simple, but requires wide-scale deployment
- Get most — all? — ISPs to filter outbound packets, to prevent spoofing
- Very hard — ISPs don't want to do that; it's expensive for some
- Can still have local spoofing
- But — can blacklist entire site if necessary

Better Data Structures

- No reason to allocate full protocol control block for just a SYN packet
- Allocate something *much* more compact, and raise the limit on half-open connections
- Can handle many more, but the attacker can still win

Attacking Compact Data Structures

- Bare minimum to store: 32-bit address, 16-bit port number, at least part of initial sequence number — call it 64 bits
- (Actually, must be higher)
- Allocate 256MB to connection table
- Assume each entry can persist for 10 seconds
- Attacker can keep it filled with bandwidth of about 200M bps — not a lot for a large site

Generic Solution

- Don't create state until necessary
- In particular, don't create connection state until you know that the far end is there
- General idea: encode (and cryptographically seal) state into some value sent from the server to the client
- The client returns the state in its third message
- The server unseals the state, makes sure it's authentic, and then creates the connection

SYN Cookies

- Generally credited to Dan Bernstein (though there's some evidence that others had the idea (but didn't publish widely) first
- Basic idea: generate the server's ISN from a time counter, the client's MSS, and a 24-bit cryptographic function of the timec counter and the connection four-tuple
- When the client's ACK message comes in, validate the connection data from the 24-bit function, and create the connection control block using the data in the ACK packet

It's Not Perfect

- Certain TCP features can't be handled, or are handled imperfectly
- Solution: fall back to this if and only if under attack
- It's better than no connection at all

CPU Denial of Service

CPU Denial of Service

- Using SYN cookies requires CPU time for a cryptographic calculation
- Suppose the attacker wants to exhaust CPU time
- Better yet, think of TLS — RSA calculations are very expensive
- Need a way to rate-limit requests from compromised clients

Puzzles

- General solution: create a puzzle that's expensive to solve but cheap to verify
- Puzzle difficulty should be tunable, in response to server load
- Before doing any expensive work, challenge the client to solve the puzzle
- Not a serious problem for legitimate clients; should pose a considerable burden for attackers

Hash Puzzle

- Generate n, a difficulty metric, and a random value x
- Send the client $\langle n, h(x), x' \rangle$, where x' is x with the low-order n bits set to zero and h is a cryptographic hash function
- Client must find x
- Client's guesses – and its answer — are validated by calculating $h(x)$ and seeing if it matches the server's value

Why it Works

- Since h is a cryptographic hash function (i.e., SHA-1), there is no faster way to find x from $\langle n, h(x), x' \rangle$ than brute force
- This takes 2^{n-1} operations on average
- A guess is easy to validate; it takes just 1 operation

Why it Doesn't Work

- Attackers have *lots* of machines
- It's easier for the attacker to throw more machines at the problem than it is for the defender
- (If the server increases n too much, it's difficult for legitimate clients)

History

- Attack not (yet?) seen in the wild
- Similar to anti-spam technique ("hash cash") proposed in 1992
- Merkle used puzzles in an early approach to public key-like key distribution
- Laurie and Clayton showed why it doesn't work against spam

Distributed Denial of Service Attacks (DDoS)

Distributed Denial of Service Attacks (DDoS)

- Most common form of DoS today
- Exhaust network bandwidth
- Uses large network of compromised "zombies" or "bots"
- "Command and control" node tells bots what to do
- IRC frequently used for control channels
- Newer ones use peer-to-peer meshes

History

- First seen in late 1999
- Comments in the code suggested that a massive attack was scheduled for December 31 — just in time to exacerbate possible Y2K troubles
- Fortunately, neither happened

Address-Spoofing

- Early versions used address-spoofing — make it harder to trace or filter bots
- As a result, early defense attempts focused on traceback
- Most newer attacks don't bother with address-spoofing — because traceback and filtering don't work

Too Many of Them!

- A defender can't do much with a list of 10,000 bots
- Tracing down the person responsible is time-consuming and sometimes futile
- Most routers can't handle a filter list with 10,000 entries

Building Botnets

- Get someone to run the bot software
- Use "come and get it" with infected "free" software
- Use web pages with nasty ActiveX controls (plus trickery to make users accept them)
- Use exploits to penetrate machines, possibly via worms
- Buy or rent them
- Steal them!

Bot-Jacking

- Bot-jacking — stealing botnets from other bad guys
- To prevent this, some bots patch other security holes on "their" machines
- One recent one includes current anti-virus software!

State of the Art

- Modern bots are fully updatable by the botherd
- Download new software to them for bug fixes or new functions: spam, DDoS, scanning, etc.
- Many bots use encrypted communications channels

Uses of Botnets

- Primary uses: DDoS and spamming
- (Spamming is a denial of service attack on mailers!)
- DDoS primarily used for extortion, especially against sports-betting sites
- They have a time-sensitive product and can't outwait the bad guys
- (Occasional use: revenge against other bad guys)

Combination Attacks

- DDoS can be used as part of other attacks
- Example: interrupt communication to SecurID servers
- Example: divert people to “backup” bank site as part of phishing attack

Defenses

Defending Against DDoS

It's Not Quite that Bad. . .

- No comprehensive defenses
- Some heuristic defenses
- Still an active research area

Heuristic Defenses

- Overprovision
- Black-hole routing
- Filter anomalies
- Replication

Overprovisioning

- Design DDoS-proof site with really big pipes
- Ideally, ride out multi-gigabit attack
- Of course, there are really big botnets, too

Black-Hole Routing

- Set up ISP routing to make it really easy to divert all traffic for the victim to a sinkhole
- The ISP takes the victim site off the air!
- But — it avoids collateral damage to other sites
- Most DDoS attacks have been relatively short-lived

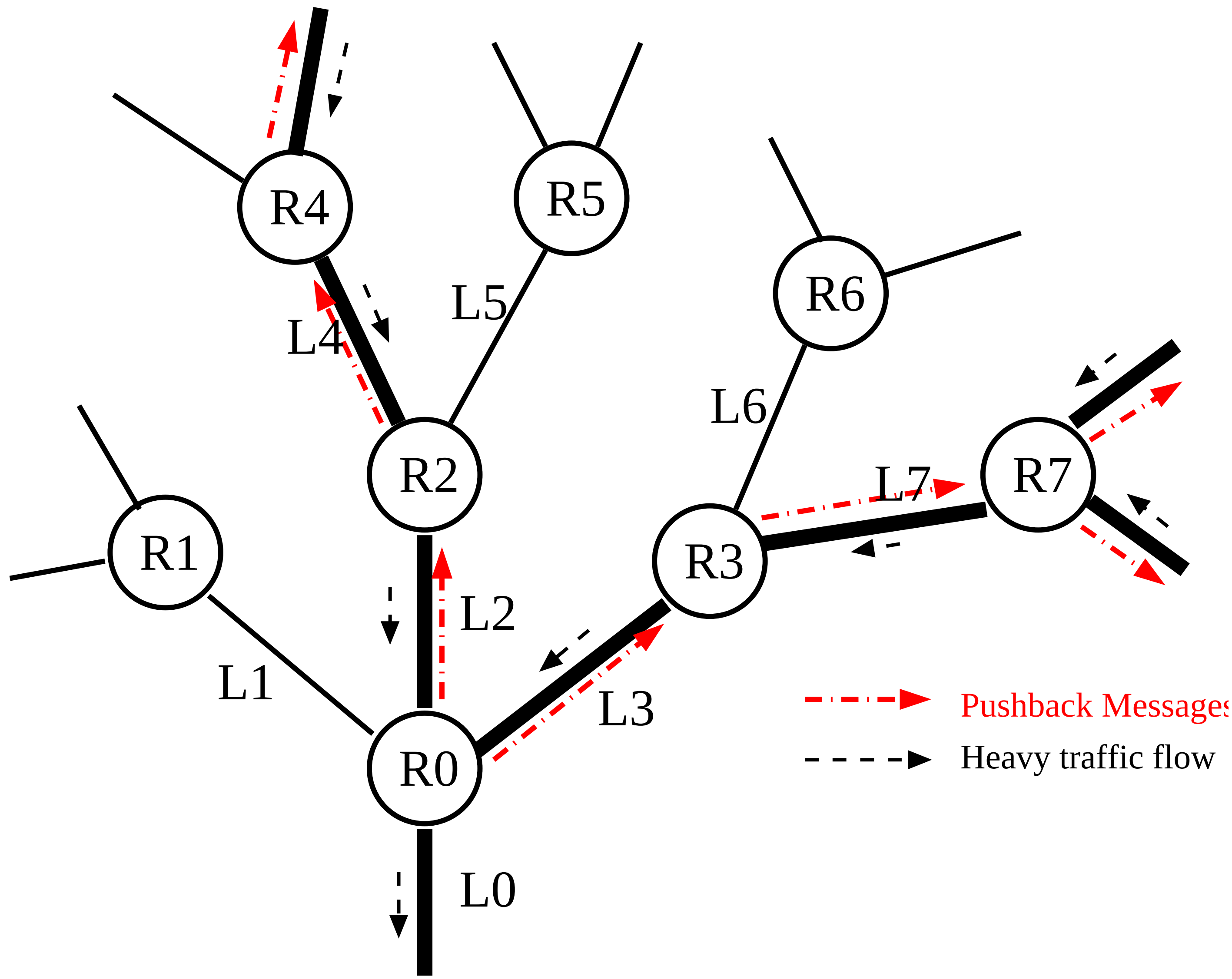
Anomaly Filtering

- DDoS traffic usually isn't perfectly "normal"
- TTLs, protocols, etc., are often unusual
- Route traffic through filtering boxes; filter based on these anomalies
- Imperfect, but frequently good enough

Pushback

- When a router output link is overloaded, see which input links the packets are coming from
- Tell the upstream nodes to rate-limit packets to this router
- Apply the algorithm recursively

Data Flow



R4
R5
R6
R7
R2
R1
R3
R0
L0
L1
L2
L3
L4
L5
L6
L7
Pushback Messages
Heavy traffic flow

Other DoS Attacks

Other DoS Attacks

- Bayesian filter
- Program availability
- Reflector attacks

Bayesian Filter

- Bayesian filters are used for anti-spam
- Spammers have sometimes sent email carefully crafted to consume most CPU cycles on Bayesian filters
- Result: sites turn off the filters to let email go through
- Consequence: spam gets through, too

Reflector Attacks

- Attacker sends a small packet with a forged source address to some service, especially the DNS
- The packet generates a much larger response
- This response is sent to the forged source address
- Attacker gets a *multiplier effect*, and hides, too

Program Availability

- Find bugs and exploit them, to crash some programs
- Persistent worry: is there a penetration exploit, too?
- If you see lots of core dumps on your system, worry. . .