Advanced Network Security

Denial of Service Attacks

Amir Mahdi Sadeghzadeh, Ph.D.

Most slides have been adapted from Steven Bellovin, COMS W4180, Columbia University, 2006, and Mehdi Kharrazi, CE40-817, Sharif University of Technology, 2015

Denial of Service (DoS) Attacks

- Attack availability
- Major problem on today's Internet

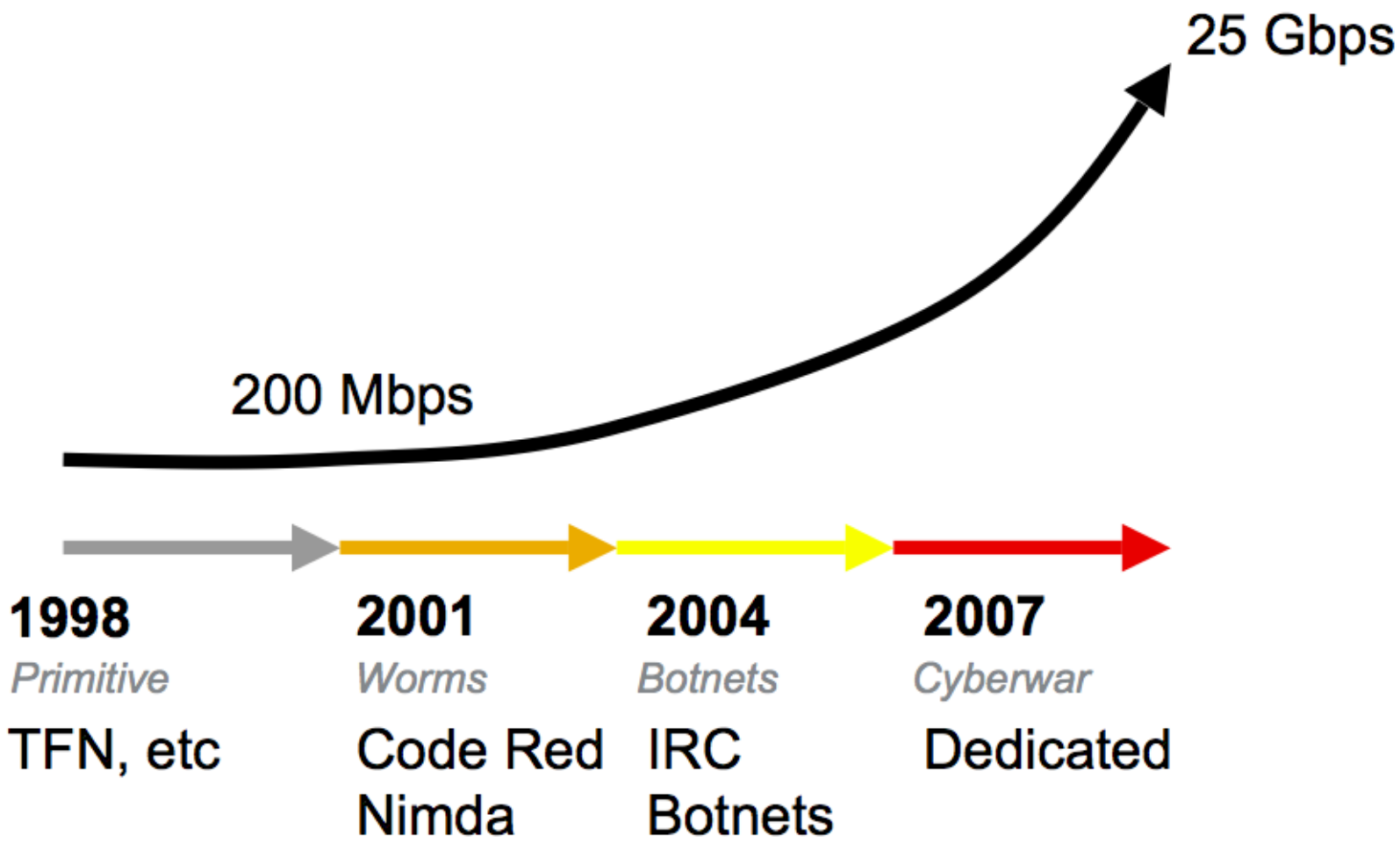
What Can be DoSed?

- 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

DDoS History

- Most viruses and worms simply perpetrate DoS attacks
- The phone system has experienced prank DoS attacks
- Must distinguish attacks from "flash crowds"
  - Why?

DDoS History



Use of DDoS

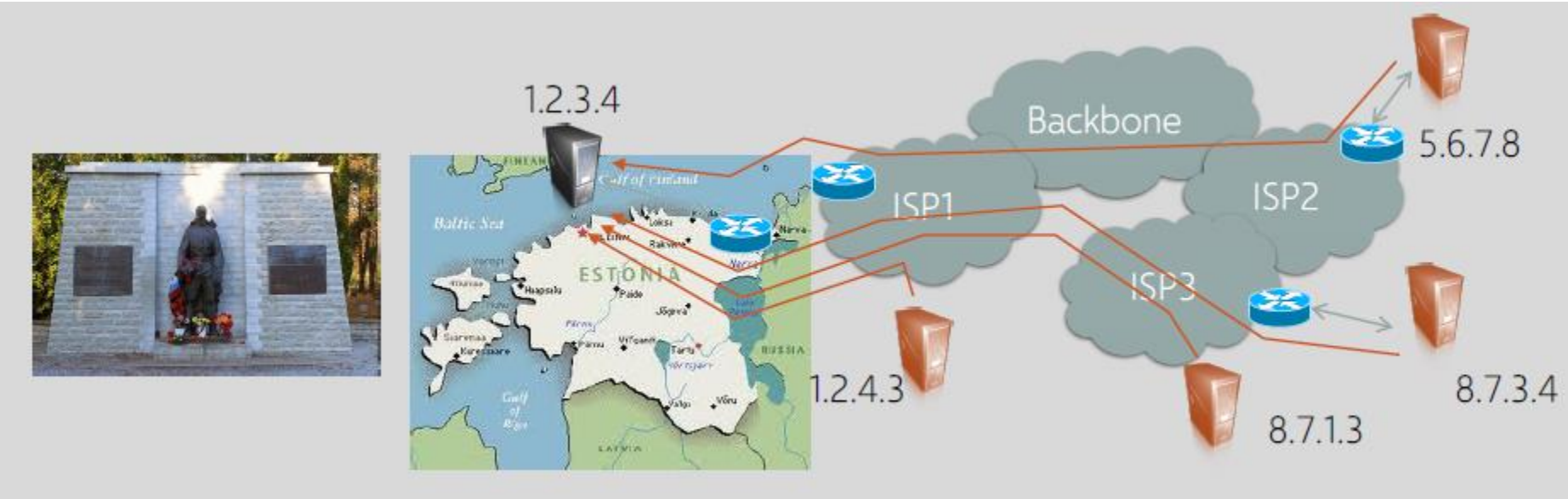
- DDoS for Profit
  - DDoS primarily used for extortion
    - sports-betting sites
      - They have a time-sensitive product and can't outwait the bad guys
- Occasional use: revenge against other (bad) guys
- DDoS can be used as part of other attacks
  - Example: divert people to "backup" bank site as part of phishing attack
- Cyber-warfare
  - State-level
    - Estonia
    - China
  - Company-level
    - to gain a competitive advantage

Estonian DDoS Attacks

- Estonia
  - A country of about 3 million people bordering Russia, has a well-developed network infrastructure
- After a dispute with Russia, it came under a crushing cyberattack in 2007

DDoS Attack on Estonia

- April 27, 2007
  - Continued for weeks, with varying levels of intensity
  - Government, banking, news, university websites
  - Government shut down international Internet connections



Attack Sources

- Botnets were most likely the source

Telegram

IP addresses coming mostly from China. Historically, all state actor-sized DDoS (200-400 Gb/s of junk) we experienced coincided in time with protests in Hong Kong (coordinated on @telegram). This case was not an exception.

— Pavel Durov (@durov) June 12, 2019

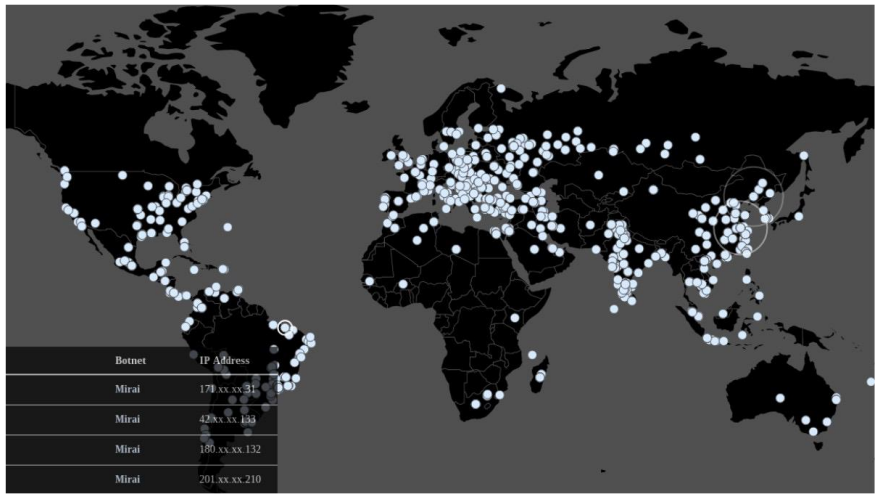
Telegram blames China for 'powerful DDoS attack' during Hong Kong protests

Telegram CEO says 'IP addresses coming mostly from China' were to blame

By Jon Porter | @JonPorty | Jun 13, 2019, 4:21am EDT

Mirai Botnet

- Scans big blocks of Internet address space for open telnet ports, logs in using default passwords
  - Assembled an army of 1 to 2.5 million IoT devices
- In 2016, used to stage massive DDoS attacks on DYN's DNS servers
  - Knocked out access to 1200 websites, including Twitter, Netflix, Paypal, Shopify, GitHub…



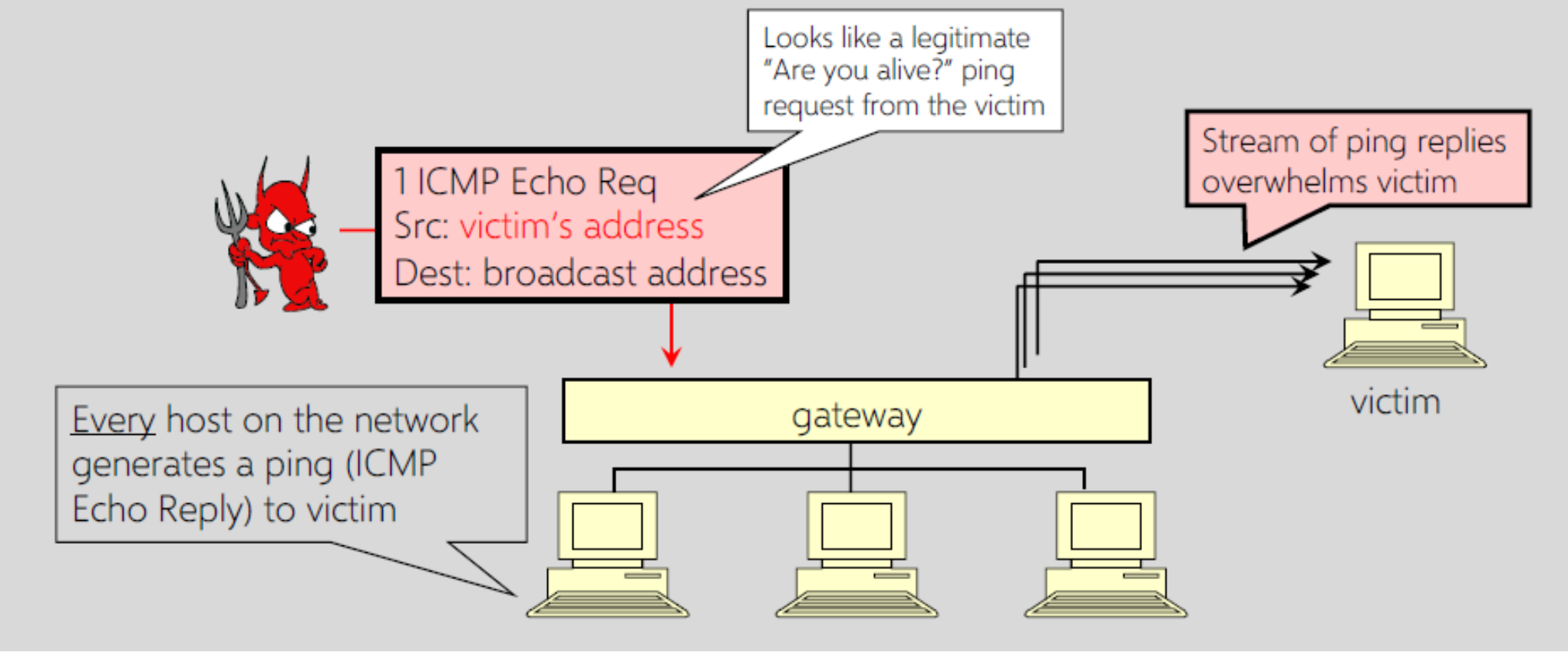
Mirai Exploits Default Passwords

Password	Device Type	Password	Device Type	Password	Device Type
123456	ACTi IP Camera	klv1234	HiSilicon IP Camera	1111	Xerox Printer
anko	ANKO Products DVR	jvbzd	HiSilicon IP Camera	Zte521	ZTE Router
pass	Axis IP Camera	admin	IPX-DDK Network Camera	1234	Unknown
888888	Dahua DVR	system	IQinVision Cameras	12345	Unknown
666666	Dahua DVR	meinsm	Mobotix Network Camera	admin1234	Unknown
vizxv	Dahua IP Camera	54321	Packet8 VOIP Phone	default	Unknown
7ujMko0vizxv	Dahua IP Camera	00000000	Panasonic Printer	fucker	Unknown
7ujMko0admin	Dahua IP Camera	realtek	RealTek Routers	guest	Unknown
666666	Dahua IP Camera	1111111	Samsung IP Camera	password	Unknown
dreambox	Dreambox TV Receiver	xmhdipc	Shenzhen Anran Camera	root	Unknown
juantech	Guangzhou Juan Optical	smcadmin	SMC Routers	service	Unknown
xc3511	H.264 Chinese DVR	ikwb	Toshiba Network Camera	support	Unknown
OxhlwSG8	HiSilicon IP Camera	ubnt	Ubiquiti AirOS Router	tech	Unknown
cat1029	HiSilicon IP Camera	supervisor	VideoIQ	user	Unknown
hi3518	HiSilicon IP Camera	<none>	Vivotek IP Camera	zlxx.	Unknown
klv123	HiSilicon IP Camera				

Amplification

- Key element of many powerful DoS attacks
- Achieves attacker resources >>> victim's
  - 1 request sent by attacker => N requests received by the victim
  - 1 byte sent by attacker => N bytes received by the victim (N can be 200+ in some attacks)
  - Force victim to expend computation, logical resources

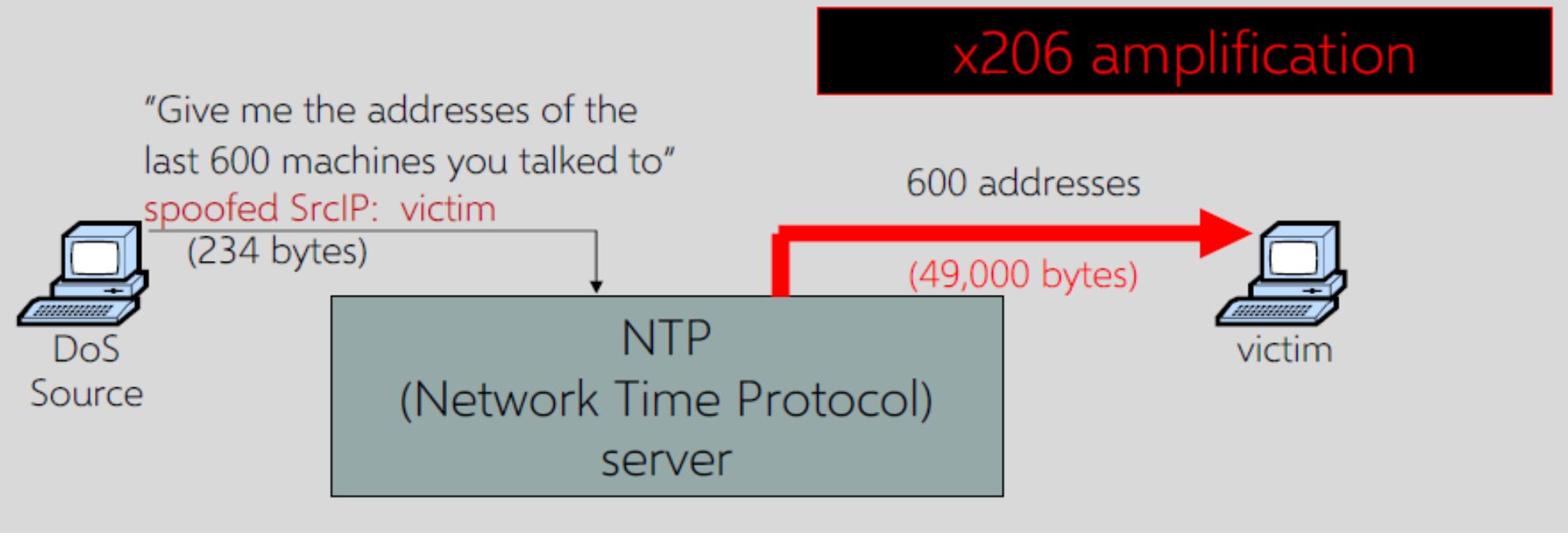
“Smurf” Reflector Attack



Solution: reject external packets to broadcast addresses

NTP Reflection + Amplification

- 2013 –2014: 400 Gbps DDoS attacks involving 4,529 NTP servers
  - NTP contains a command called monlist (or sometimes MON_GETLIST) for monitoring purposes.
  - It returns the addresses of up to the last 600 machines that the NTP server has interacted with.



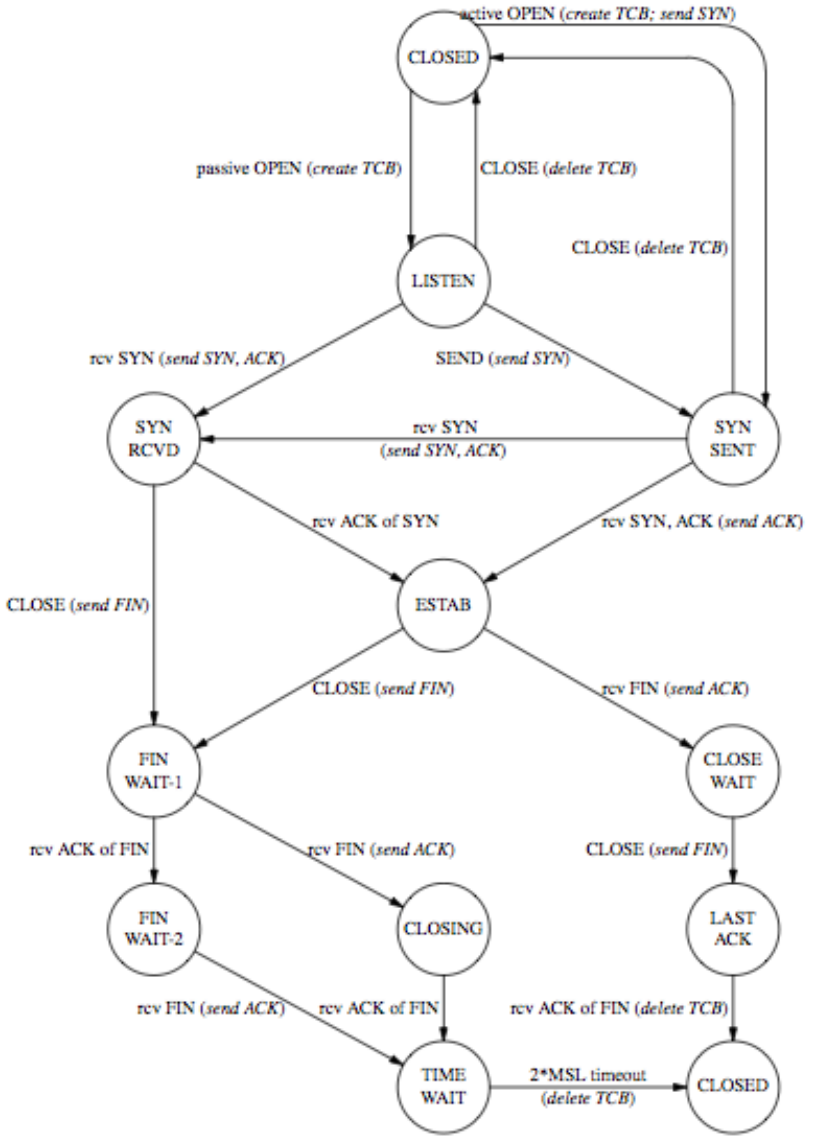
Largest European DDoS Attack on Record

- Victim: an Akamai customer in Eastern Europe
- Source: a highly-sophisticated, global botnet of compromised devices
  - 75 attacks over 30 days using UDP, UDP fragmentation, ICMP flood, RESET flood, SYN flood, TCP anomaly, TCP fragment, PSH ACK flood, FIN push flood, and PUSH flood. UDP most popular.
- Peak rates: 854 Gbps and 660M packets/sec
- Same customer attacked again on Sep 22:
  - 705M packets/sec

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
- Can still have local spoofing

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 200Mbps — not a lot for a large site

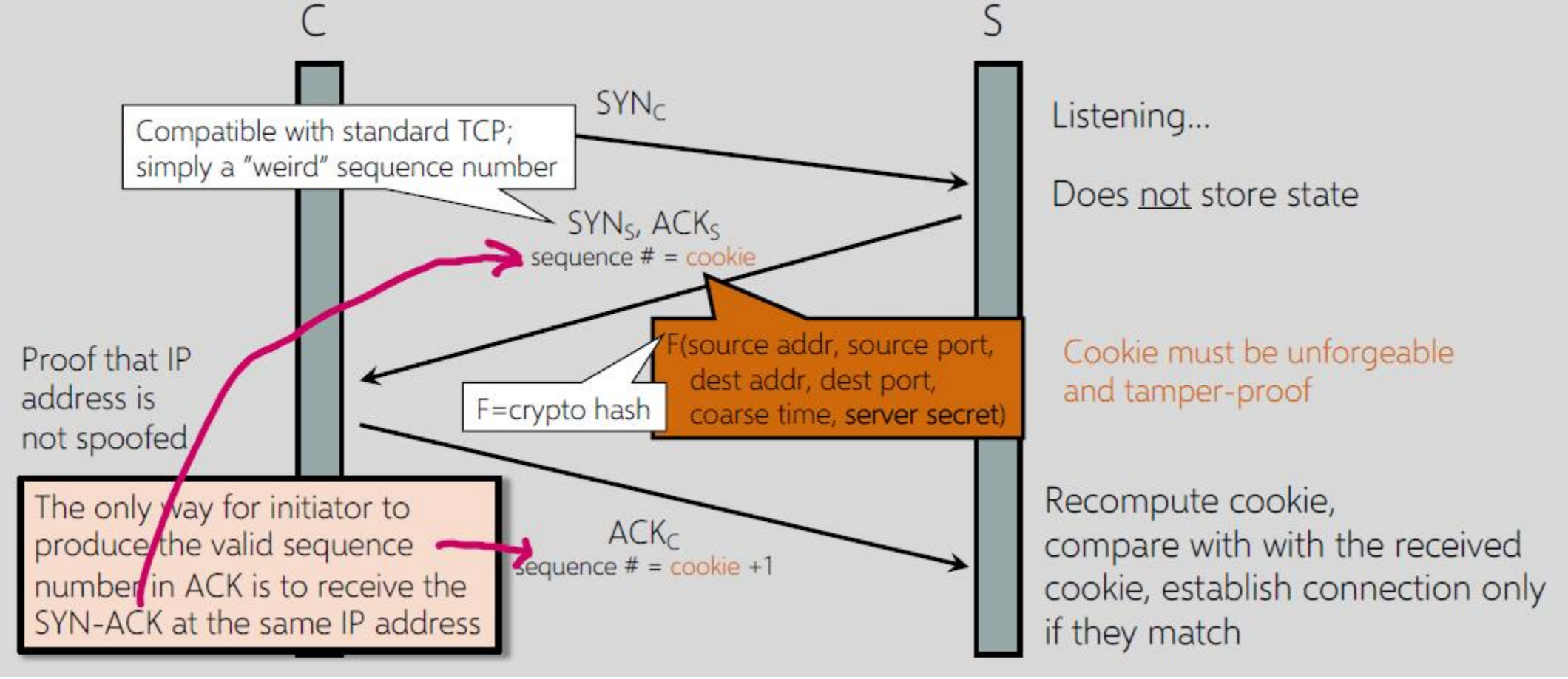
Preventing Denial of Service

- DoS is caused by asymmetric state allocation
  - 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 state into some value sent from the server to the client
    - The client returns the state in its third message
- Cookies ensure that the responder is stateless until initiator produced at least two messages
  - Responder’s state (IP addresses and ports of the connection) is stored in a cookie and sent to initiator
  - After initiator responds, cookie is regenerated and compared with the cookie returned by the initiator

SYN Cookies

- Basic idea: generate the server's ISN from
  - a time counter
  - the client's MSS
  - 24-bit cryptographic function of the time 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

SYN Cookies


C
S
Compatible with standard TCP; simply a "weird" sequence number
SYN$_C$
Listening...
Does not store state
SYN$_S$, ACK$_S$
sequence # = cookie
F(source addr, source port, dest addr, dest port, coarse time, server secret)
F=crypto hash
Cookie must be unforgeable and tamper-proof
Proof that IP address is not spoofed
The only way for initiator to produce the valid sequence number in ACK is to receive the SYN-ACK at the same IP address
ACK$_C$
sequence # = cookie +1
Recompute cookie, compare with with the received cookie, establish connection only if they match

SYN Cookies

SYN cookies are initial sequence numbers that are carefully constructed according to the following rules:

- let **t** be a slowly incrementing timestamp (typically `time()` logically right-shifted 6 positions, which gives a resolution of 64 seconds)
- let **m** be the maximum segment size (MSS) value that the server would have stored in the SYN queue entry
- let **s** be the result of a cryptographic hash function computed over the server IP address and port number, the client IP address and port number, and the value **t**. The returned value **s** must be a 24-bit value.

The initial TCP sequence number, i.e. the *SYN cookie*, is computed as follows:

- Top 5 bits: **t** mod 32
- Middle 3 bits: an encoded value representing **m**
- Bottom 24 bits: **s**

*(Note: since **m** must be encoded using 3 bits, the server is restricted to sending up to 8 unique values for **m** when SYN cookies are in use.)*

When a client sends back a TCP ACK packet to the server in response to the server's SYN+ACK packet, the client must (according to the TCP spec) use *n+1* in the packet's *Acknowledgement number*, where *n* is the initial sequence number sent by the server. The server then subtracts 1 from the acknowledgement number to reveal the SYN cookie sent to the client.

The server then performs the following operations.

- Checks the value **t** against the current time to see if the connection has expired.
- Recomputes **s** to determine whether this is, indeed, a valid SYN cookie.
- Decodes the value **m** from the 3-bit encoding in the SYN cookie, which it then can use to reconstruct the SYN queue entry.

From this point forward, the connection proceeds as normal.

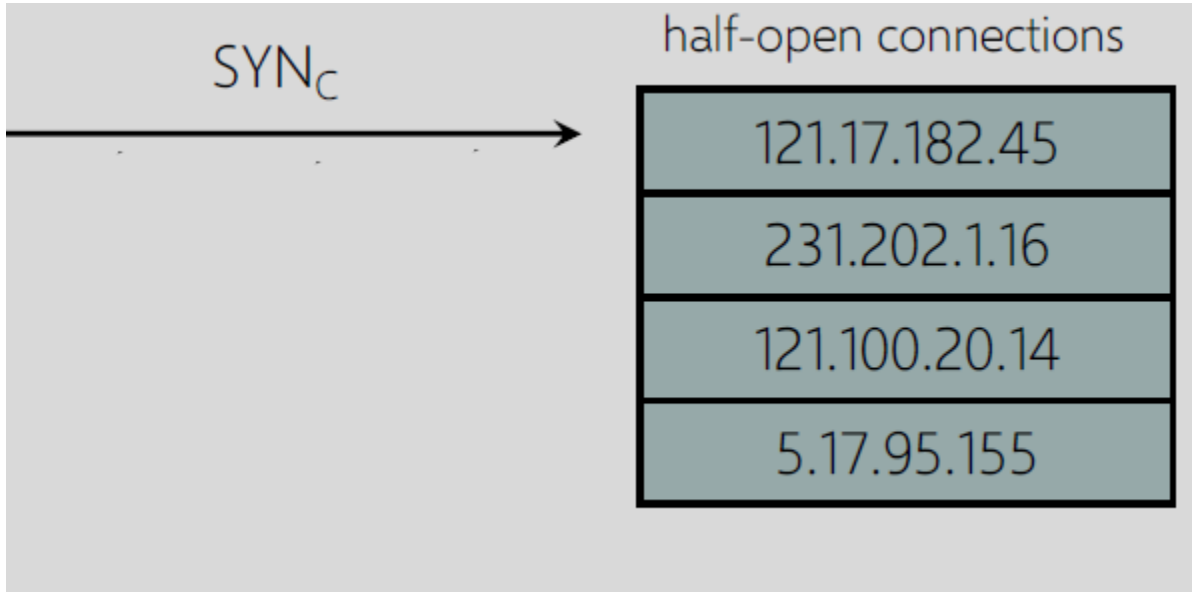
https://en.wikipedia.org/wiki/SYN_cookies

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

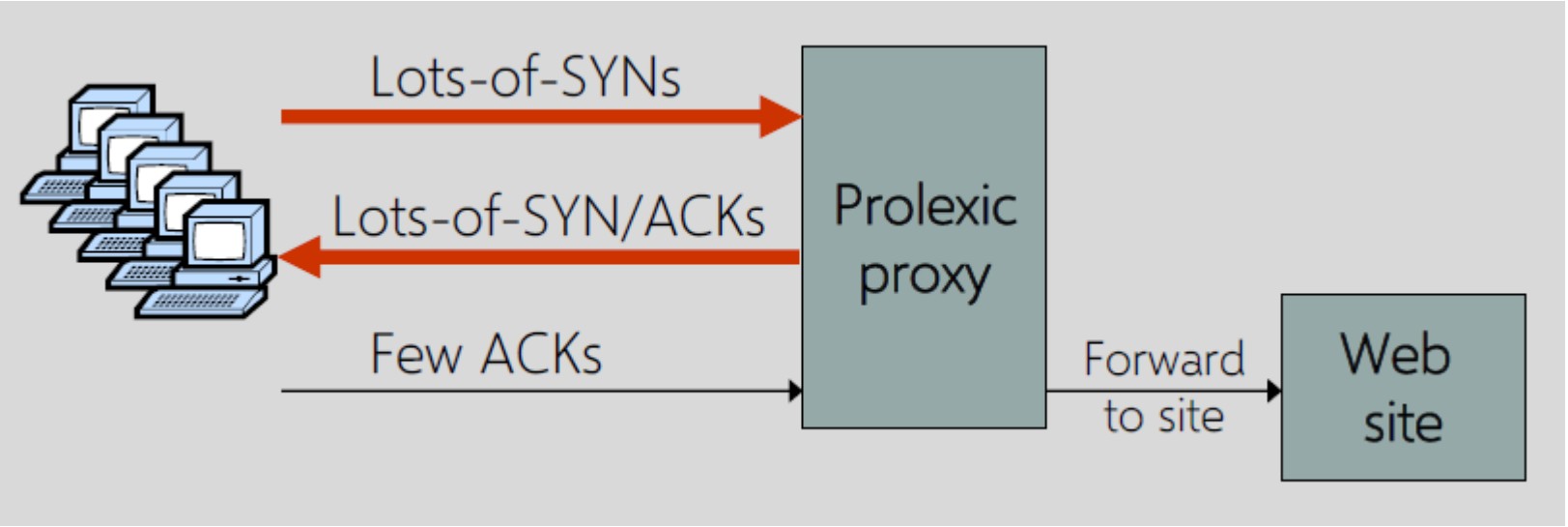
Another Defense: Random Deletion

- If SYN queue is full, delete random entry
  - Legitimate connections have a chance to complete
  - Fake addresses will be eventually deleted
- Easy to implement



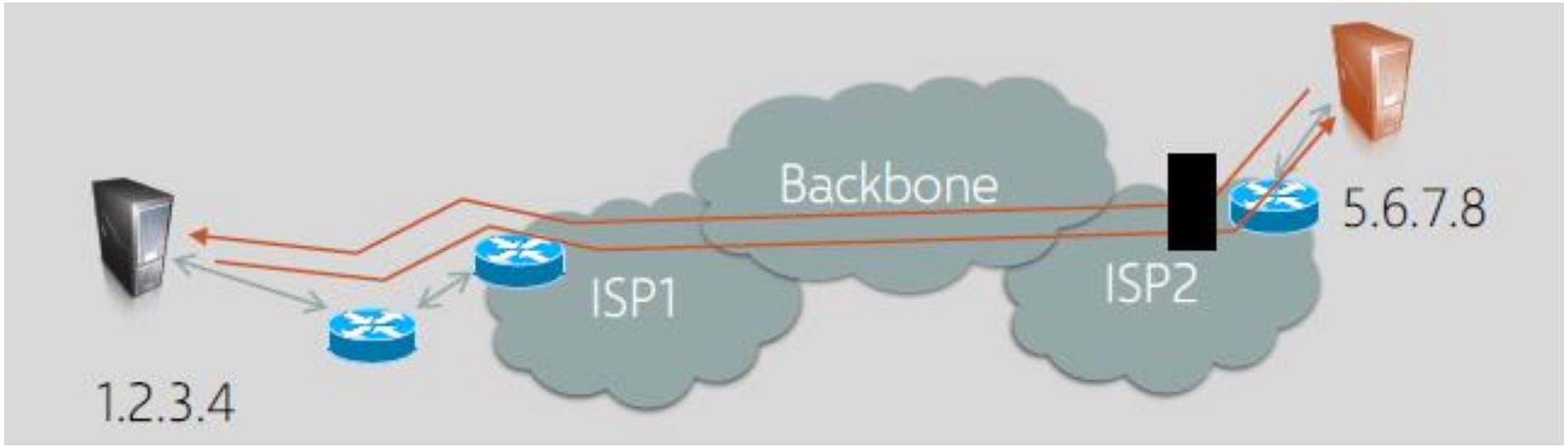
Prolexic Proxy

- Idea: only forward established TCP connections to site
  - Prolexic purchased by Akamai in 2014



Ingress Filtering

- Attacker's goal: prevent legitimate users from accessing victim (1.2.3.4)
  - ICMP ping flood
    - Attacker sends ICMP pings as fast as possible to victim
    - When will this work as a DoS? Attacker resources > victim's
    - How can this be prevented?
      - Ingress filtering of attacker IP addresses near victim once attack identified



Other Junk-Packet Attacks

- Proxy must keep floods of these away from website

Attack Packet	Victim Response	Rate: attk/day [ATLAS 2013]
TCP SYN to open port	TCP SYN/ACK	773
TCP SYN to closed port	TCP RST	
TCP ACK or TCP DATA	TCP RST	
TCP RST	No response	
TCP NULL	TCP RST	
ICMP ECHO Request	ICMP ECHO Response	50
UDP to closed port	ICMP Port unreachable	387

Stronger Attack: TCP Con Flood

- Command bot army to:
  - Complete TCP connection to web site
  - Send short HTTP HEAD request
  - Repeat
- Will bypass SYN flood protection proxy but …
  - Attacker can no longer use random source IPs
    - Reveals location of bot zombies
  - Proxy can now block or rate-limit bots

CPU Denial of Service

CPU Denial of Service

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

Puzzles

- General solution
  - Before doing any expensive work, challenge the client to solve the puzzle
  - Create a puzzle that's expensive to solve but cheap to verify

- Puzzle difficulty should be tunable, in response to server load
- Not a serious problem for legitimate clients; should pose a considerable burden for attackers

Hash Puzzle

- Generate
  - a difficulty metric n
  - a random value x
- Send the client ⟨n, h(x), x′ ⟩
  - where x′ is x with the low-order n bits set to zero
  - 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 ⟨n, h(x), x′ ⟩ 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 (Botnets)
  - 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)

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
- Newer ones use peer-to-peer meshes

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 victim to run the bot software
  - Use “come and get it” with infected “free” software
- 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 bothered
- 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!

Defenses

Solutions

- DoS attacks are a hard problem—by definition, they're exhausting some resource
- Maybe you can increase your resources—but it's probably cheaper for the attacker to increase their nastiness
- But there are approaches
  - Some heuristic defenses
  - Still an active research area

Heuristic Defenses

- Over provision
- Black-hole routing
- Filter anomalies
- Pushback

Over provisioning

- 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

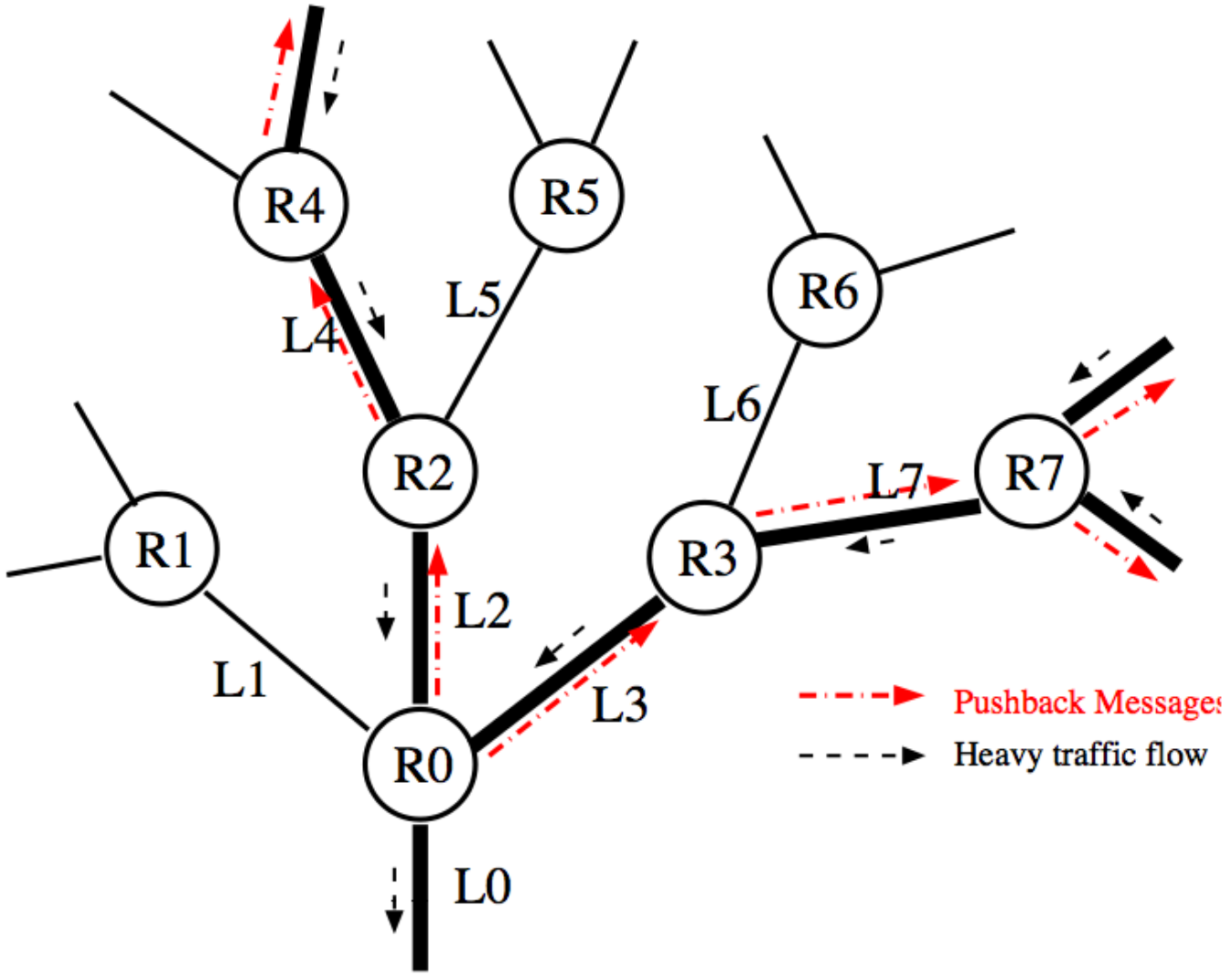
Anomaly Filtering

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

References

- Kurose, James F., and Keith W. Ross. "Computer networking: A top-down approach edition." Addision Wesley (2007), chapter 8.
- Steven Bellovin, COMS W4180, Columbia University, 2006
- Mehdi Kharrazi, CE40-817, Sharif University of Technology, 2015
- Vitaly Shmatikov, CS 361S, UT Austin, 2014
- Vitaly Shmatikov, CS 5435, Cornell University, 2022