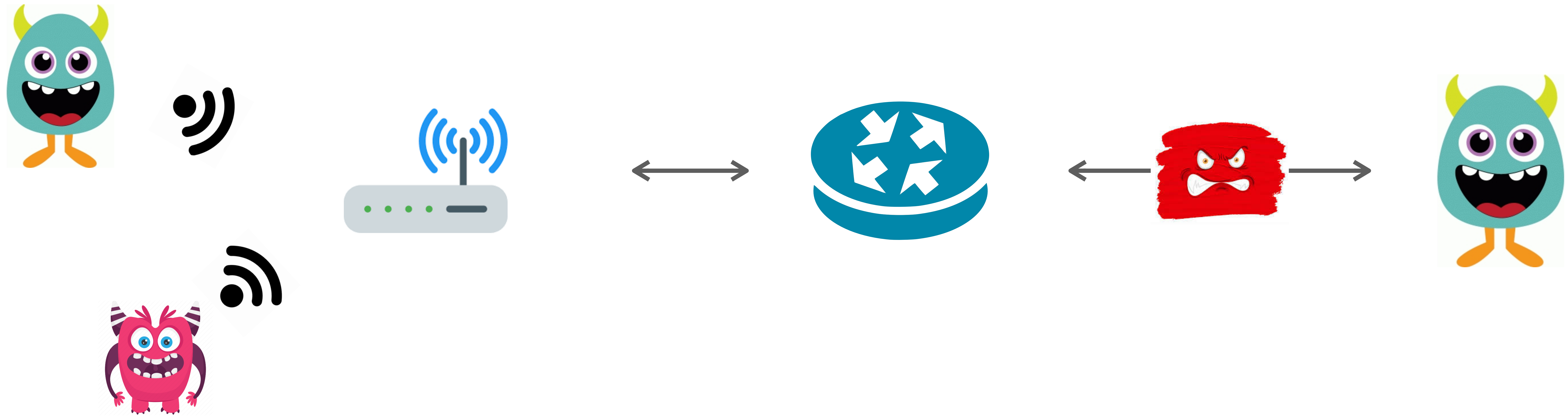
DoS Attacks and Network Defenses

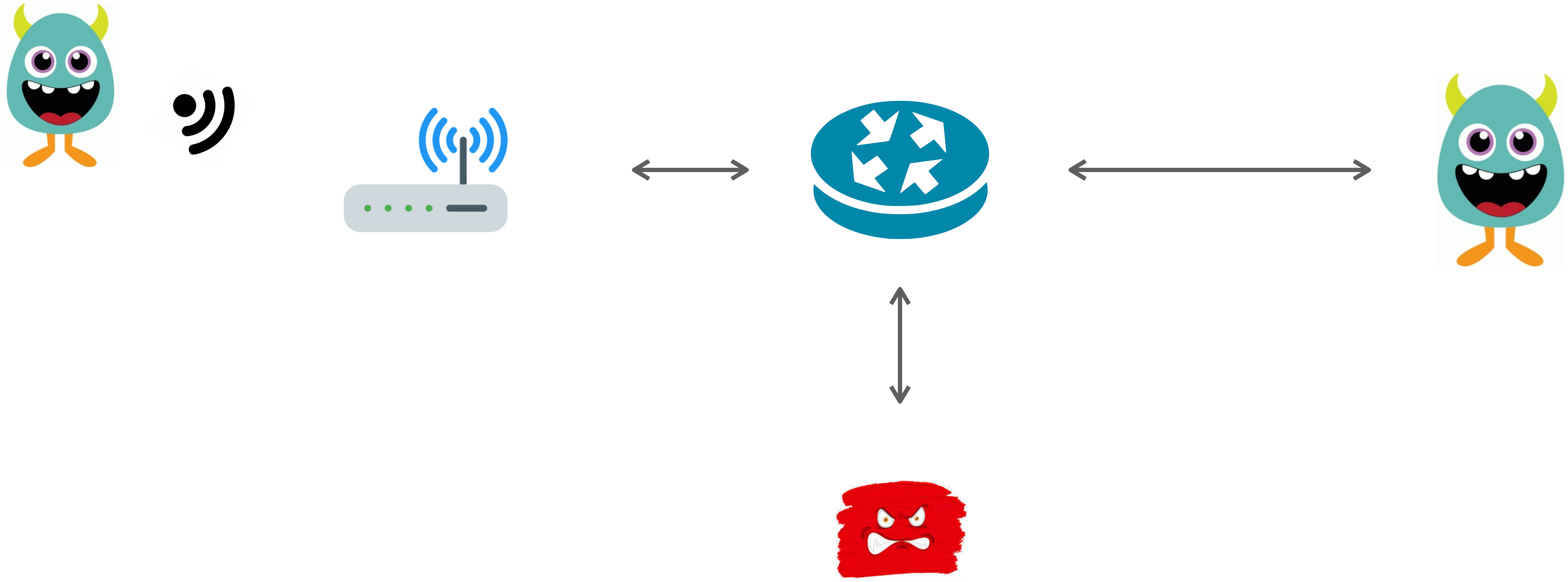
CS155 Computer and Network Security

Stanford University

Review: On Path Attacker



Review: Off Path Attacker



No security guarantees

Confidentiality — Ethernet, IP, UDP, and TCP do not provide any confidentiality. All traffic is in cleartext.

On-path attacker can do anything. ARP and BGP attacks allow an off-path attacker to become on-path and MITM connections.

Integrity — No guarantees that attacker hasn't modified traffic. Ethernet, IP and UDP have no protection against spoofed packets. TCP provides weak guarantee of source authentication.

Availability — Attackers can attempt to inject RST packets. More today.

Assume network is malicious

Always Assume: The network is out to get you.

Solution: Always use TLS if you want any protection against large-scale eavesdropping (e.g., intelligence agencies), or guarantee that data hasn't been modified or corrupted by an on-path attacker

Note! HTTPS and TLS aren't just for sensitive material! There have been attacks where malicious Javascript or malware is injected into websites.

Building a network protocol

Don’t build network proto from scratch

- Never not roll your own crypto
- Many opportunities to mess up parsing network packets

gRPC: http2 + TLS 1.3 RPC framework

- Safe parsing in 11 languages
- Exceptionally efficient
- Streaming/Sync/Async
- TLS-based authentication

```
syntax = "proto3";

package calc;

message AddRequest {
  int32 n1 = 1;
  int32 n2 = 2;
}

message AddReply{
  int64 res = 1;
}

service Calculator {
  rpc Add(AddRequest)       returns (AddReply) {}
  rpc Substract(SubRequest) returns (SubReply) {}
  rpc Multiply(MultRequest) returns (MultReply) {}
  rpc Divide(DivideRequest) returns (DivideReply) {}
}
```

Denial of Service Attacks

Goal: take large service/network/org offline by overwhelming it with network traffic such that they can't process real requests

How: find mechanism where attacker doesn't spend a lot of effort, but requests are difficult/expensive for victim to process

Types of Attacks

DoS Bug: design flaw that allows one machine to disrupt a service. Generally a protocol asymmetry, e.g., easy to send request, difficult to create response. Or requires server state.

DoS Flood: control a large number of requests from a botnet or other machines you control

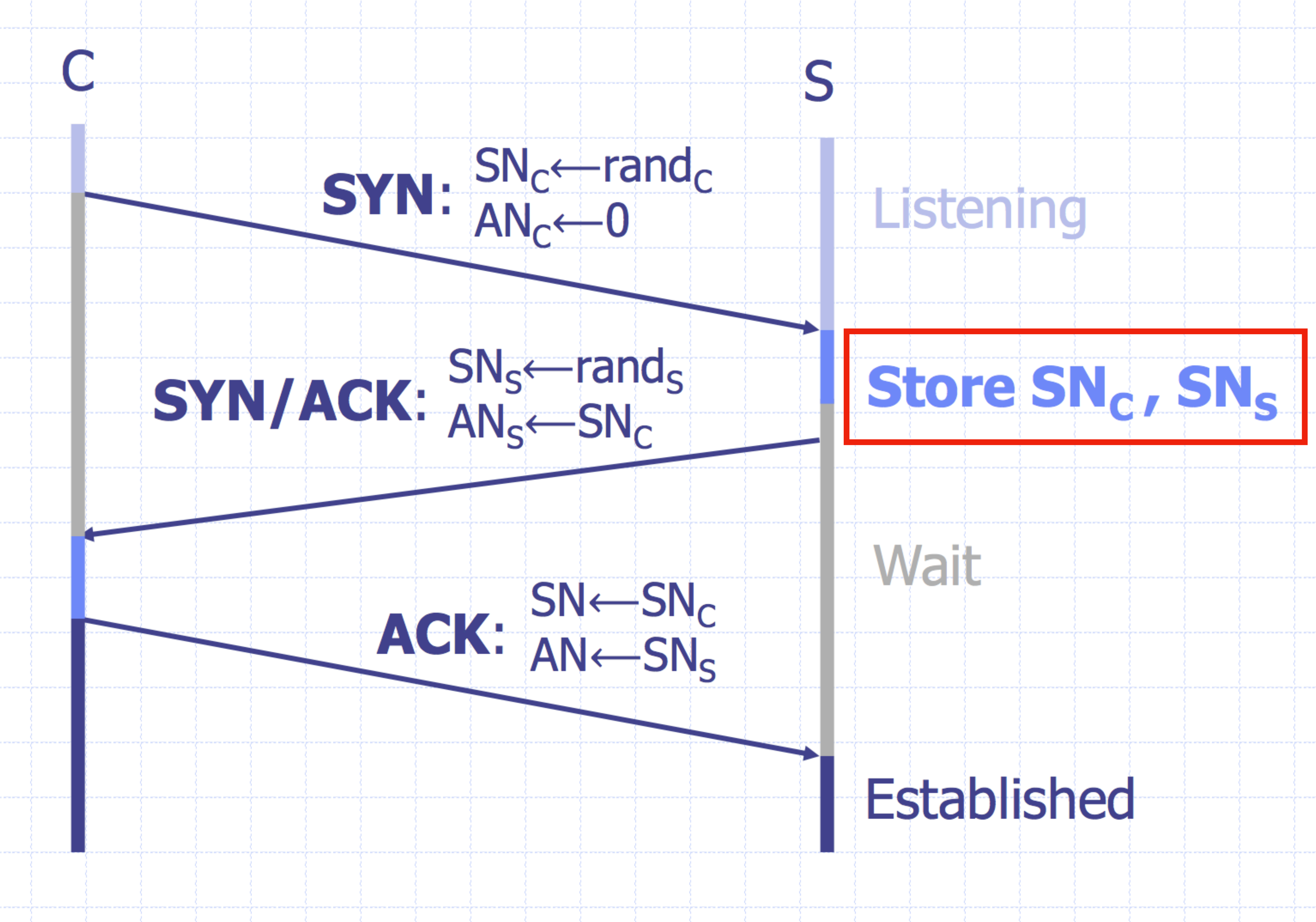
DoS at Every Layer

Link Layer: send too much traffic for switches/routers to handle

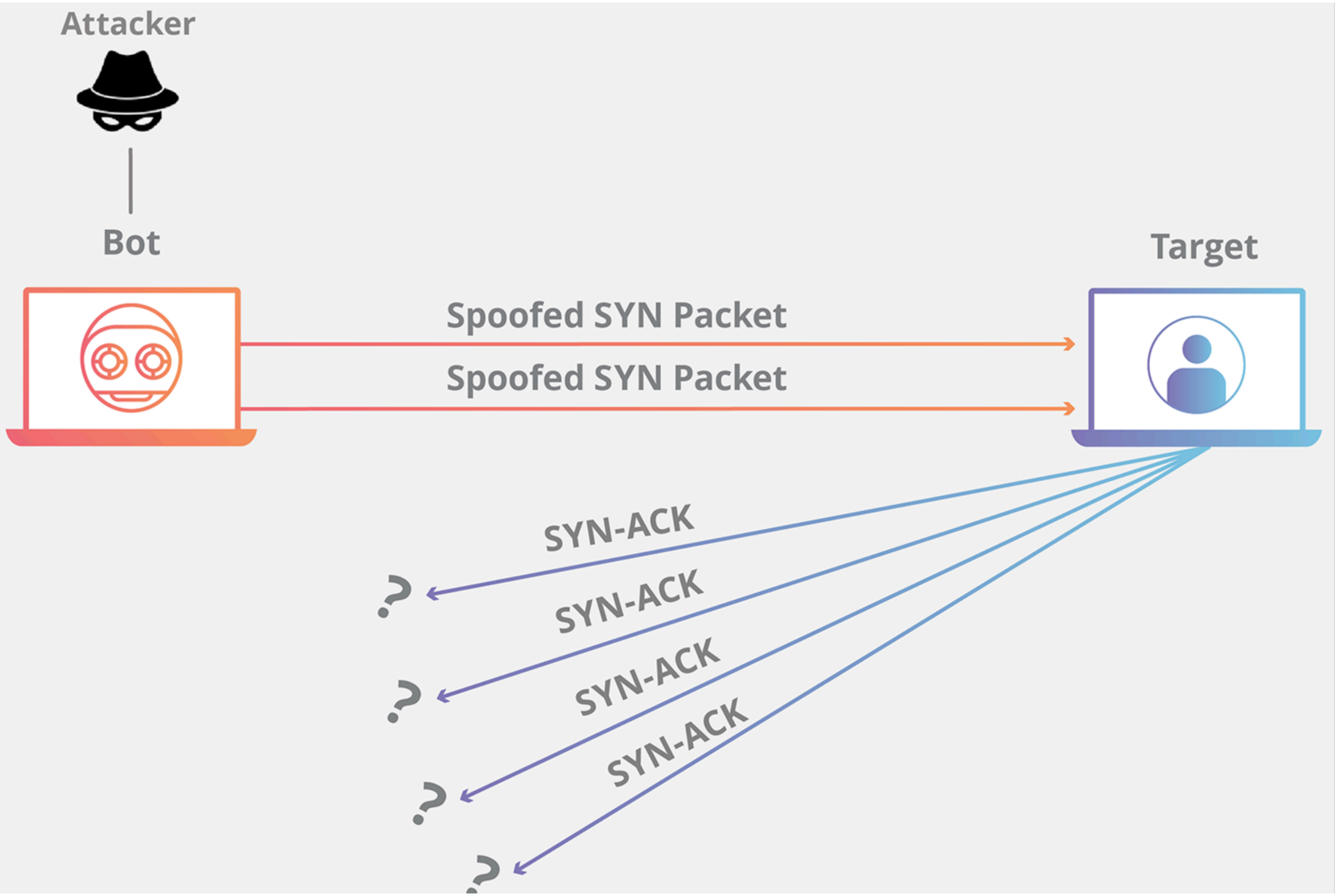
TCP/UDP: require servers to maintain large number of concurrent connections or state

Application Layer: require servers to perform expensive queries or cryptographic operations

TCP Handshake



SYN Floods


Attacker
Bot
Target
Spoofed SYN Packet
Spoofed SYN Packet
SYN-ACK
SYN-ACK
SYN-ACK
SYN-ACK
?
?
?
?

Core Problem

Problem: server commits resources (memory) before confirming identify of the client (when client responds)

Bad Solution:

- Increase backlog queue size
- Decrease timeout

Real Solution: Avoid state until 3-way handshake completes

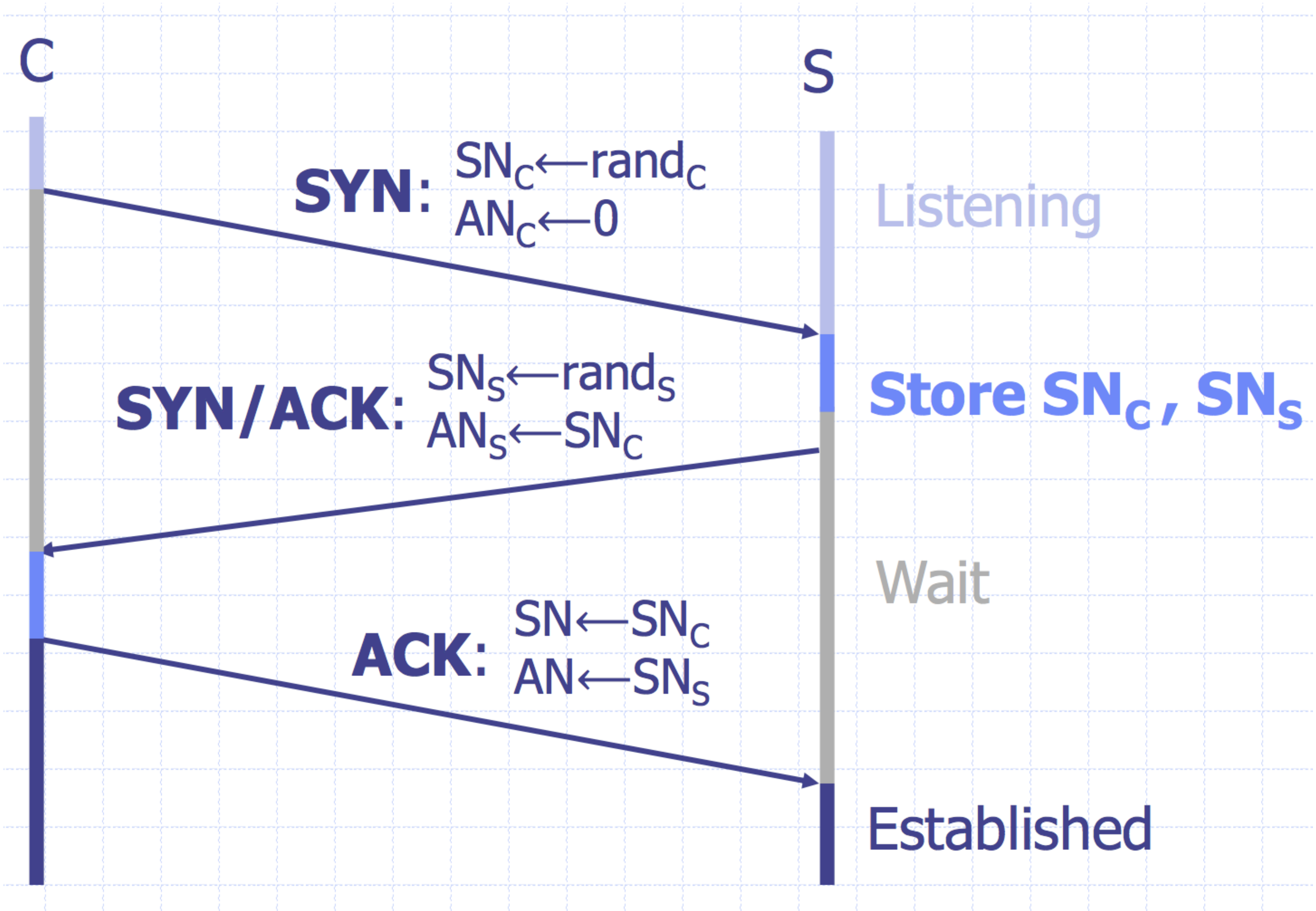
SYN Cookies

Idea: Instead of storing SN_c and SN_s…
send a cookie back to the client.

$L = MAC_{key}$ (SAddr, SPort, DAddr, DPort, SN_C, T)
key: picked at random during boot

T = 5-bit counter incremented every 64 secs.
$SN_s = (T \| mss \| L)$

Honest client sends ACK ($AN=SN_s$, $SN=SN_C+1$)
Server allocates space for socket only if valid SN_s



C
S
SYN: $SN_C \leftarrow rand_C$ $AN_C \leftarrow 0$
Listening
SYN/ACK: $SN_S \leftarrow rand_S$ $AN_S \leftarrow SN_C$
Store SN_C , SN_S
Wait
ACK: $SN \leftarrow SN_C$ $AN \leftarrow SN_S$
Established

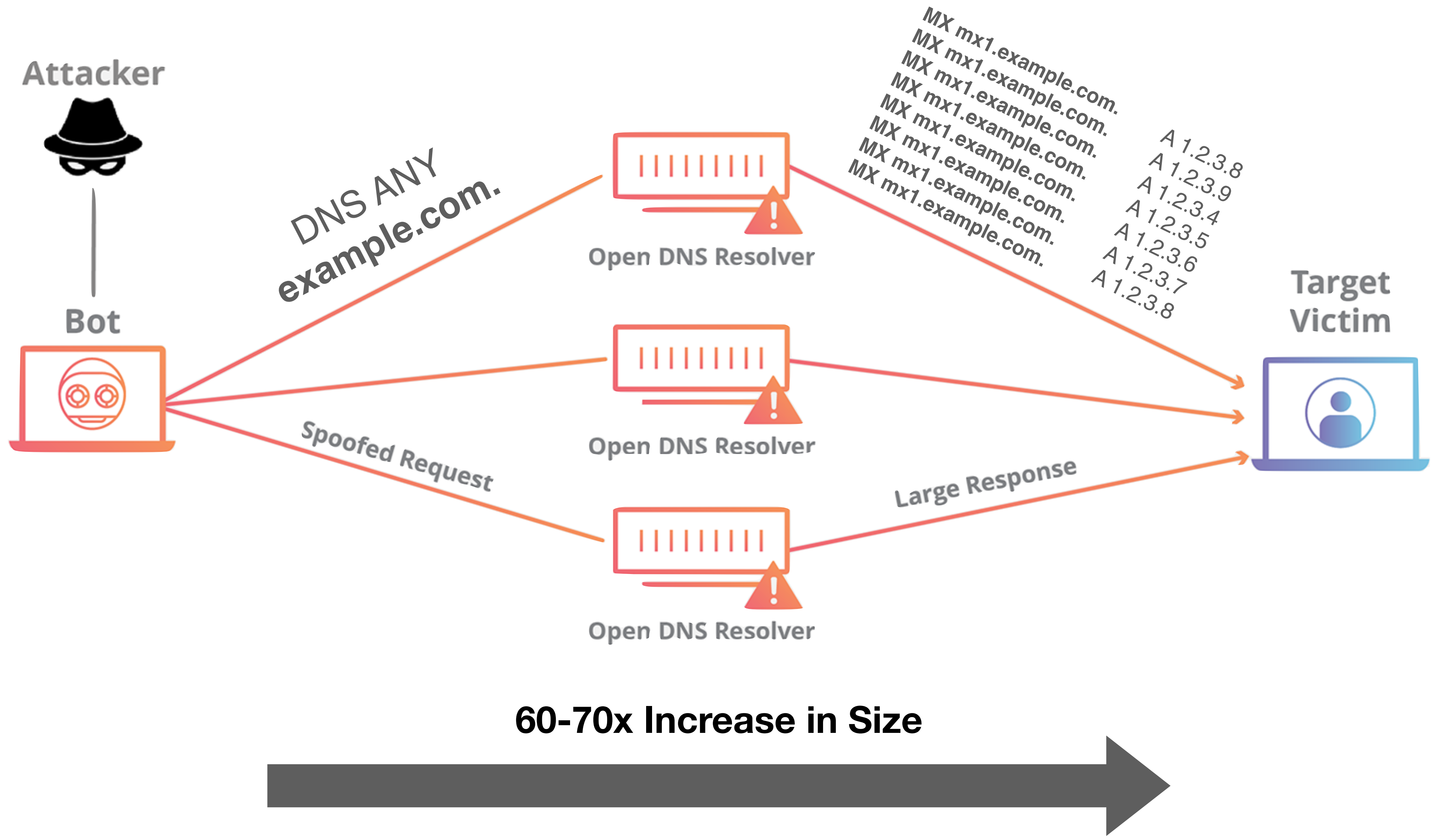
Server does not save state
(loses TCP options)

Amplification Attacks

Services that respond to a single (small) UDP packet with a large UDP packet can be used to amplify DOS attacks

Attacker forges packet and sets source IP to victim's IP address. When service responds, it sends large amount of data to the spoofed victim

The attacker needs a large number of these services to amplify packets. Otherwise, the victim could just drop the packets from the small number of hosts



Common UDP Amplifiers

DNS: ANY query returns *all* records server has about a domain

NTP: MONLIST returns list of last 600 clients who asked for the time recently

DNS: Do not have recursive resolvers on the public Internet.

NTP: Do not respond to commands like MONLIST

Both are considered misconfigurations today, but often 100Ks of misconfigured hosts on the public Internet

Amplification Attacks

2013: DDoS attack generated 300 Gbps (DNS)

- 31,000 misconfigured open DNS resolvers, each at 10 Mbps
- Source: 3 networks that allowed IP spoofing

2014: 400 Gbps DDoS attacked used 4,500 NTP servers

THE WALL STREET JOURNAL.

October 21, 2016

Cyberattack Knocks Out Access to Websites

Popular sites such as Twitter, Netflix and PayPal were unreachable for part of the day

United States
of America

twitter

amazon web services™

PayPal

NETFLIX

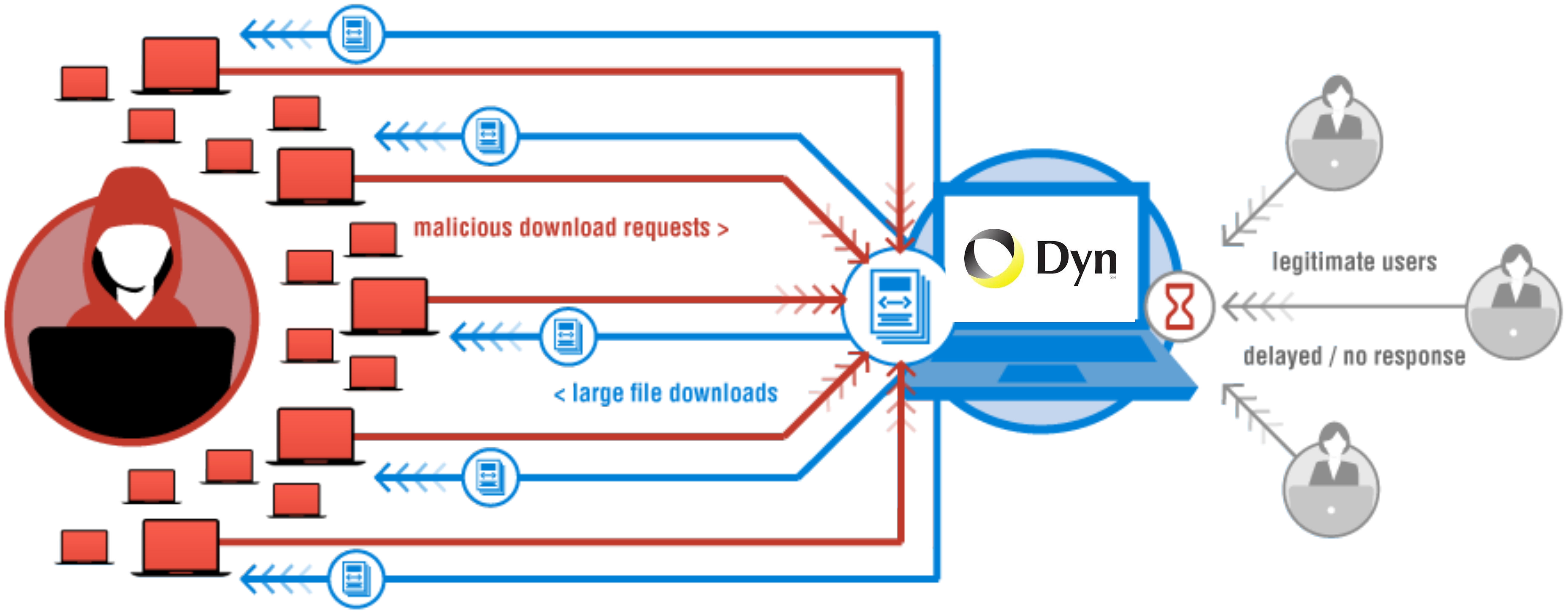
SOUNDCLOUD

Spotify

GitHub

reddit

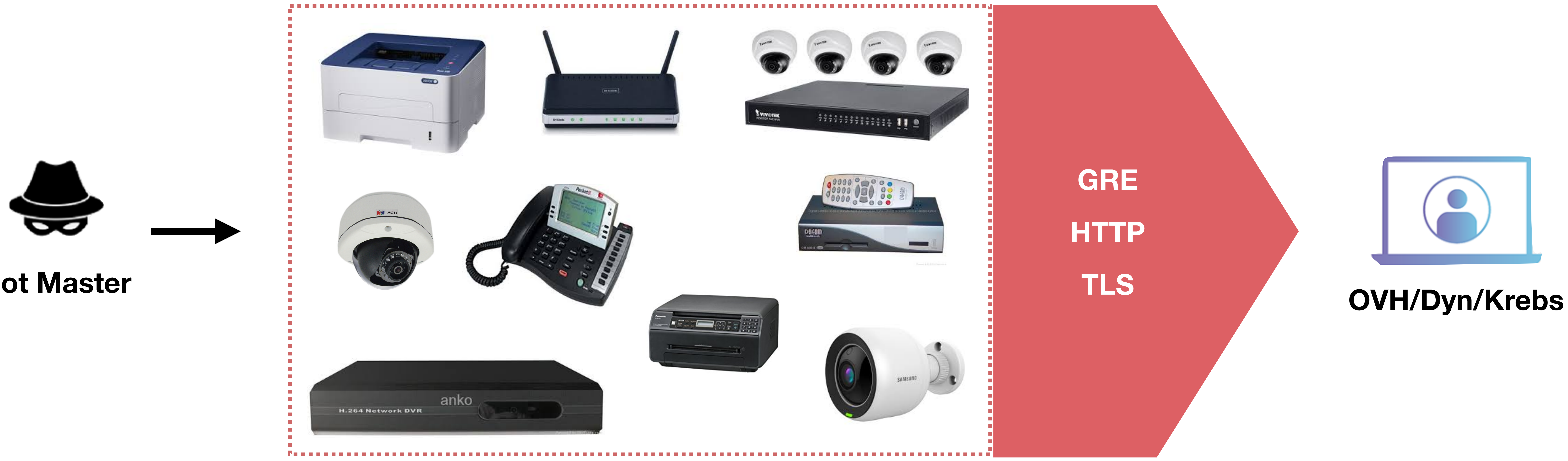
New York Times



> “We are still working on analyzing the data but the estimate at the time of this report is up to 100,000 malicious endpoints. […] There have been some reports of a magnitude in the 1.2 Tbps range; at this time we are unable to verify that claim.”

Image: Verisign

A Botnet of IoT Devices



Bot Master

GRE

HTTP

TLS

OVH/Dyn/Krebs

200K IoT devices

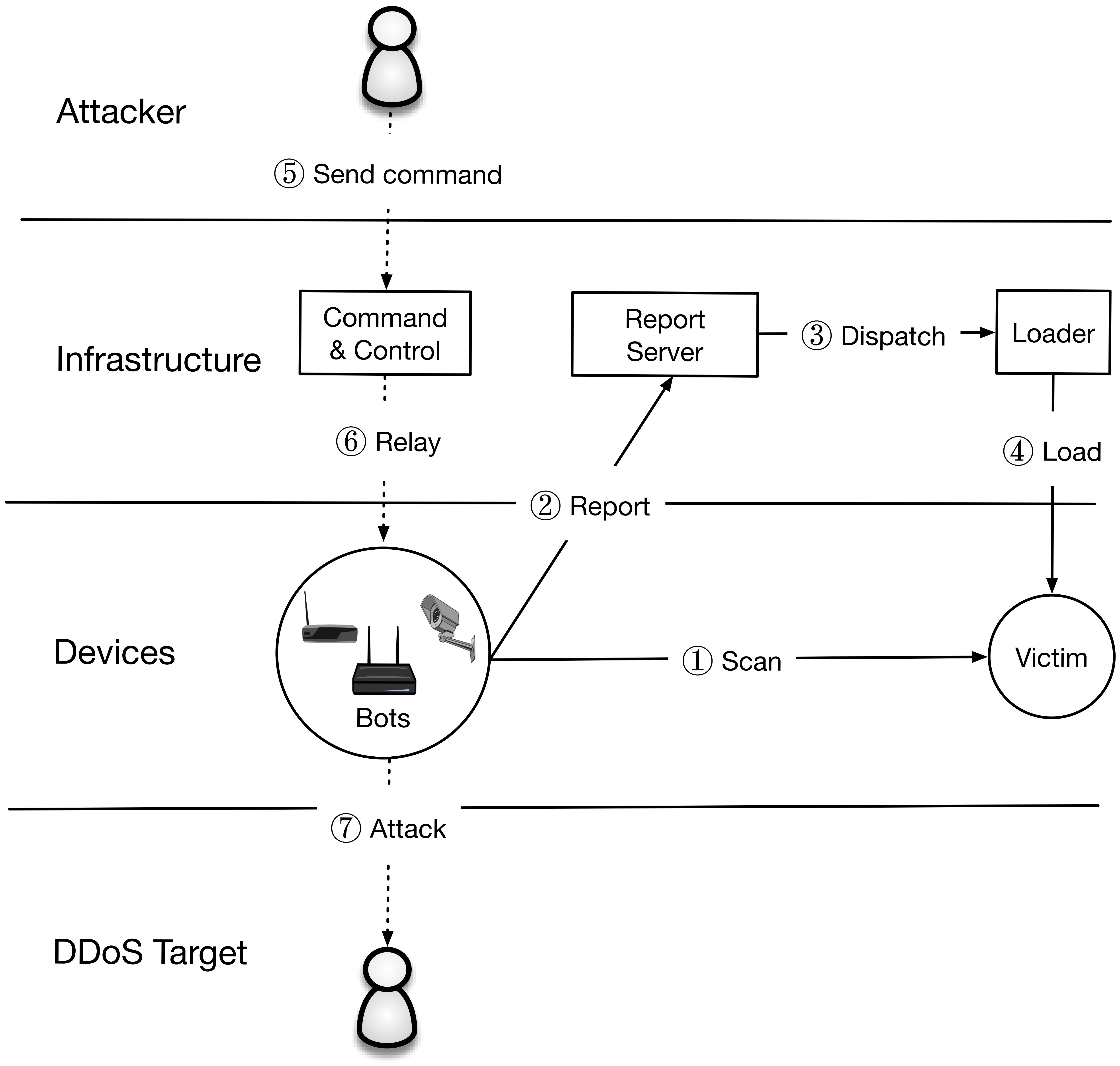
Not Amplification.
Flood with SYN, ACK, UDP, and GRE packets

The Mirai Malware

Bot master will issue commands to scan or start an attack

Attack Command:

- Action (e.g., START, STOP)
- Target IP(s)
- Attack Type (e.g., GRE, DNS, TCP)
- Attack Duration

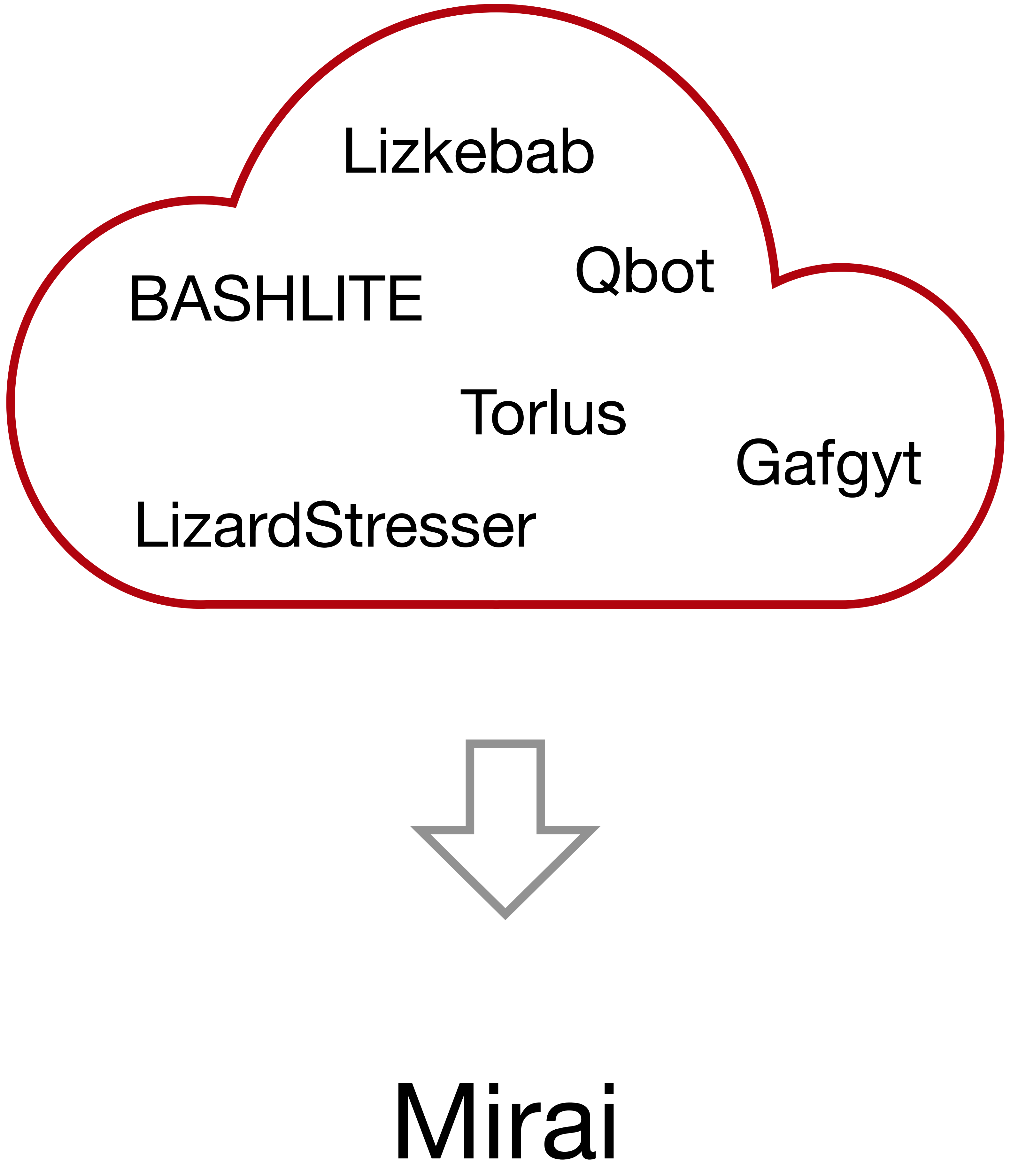


What made Mirai Successful?

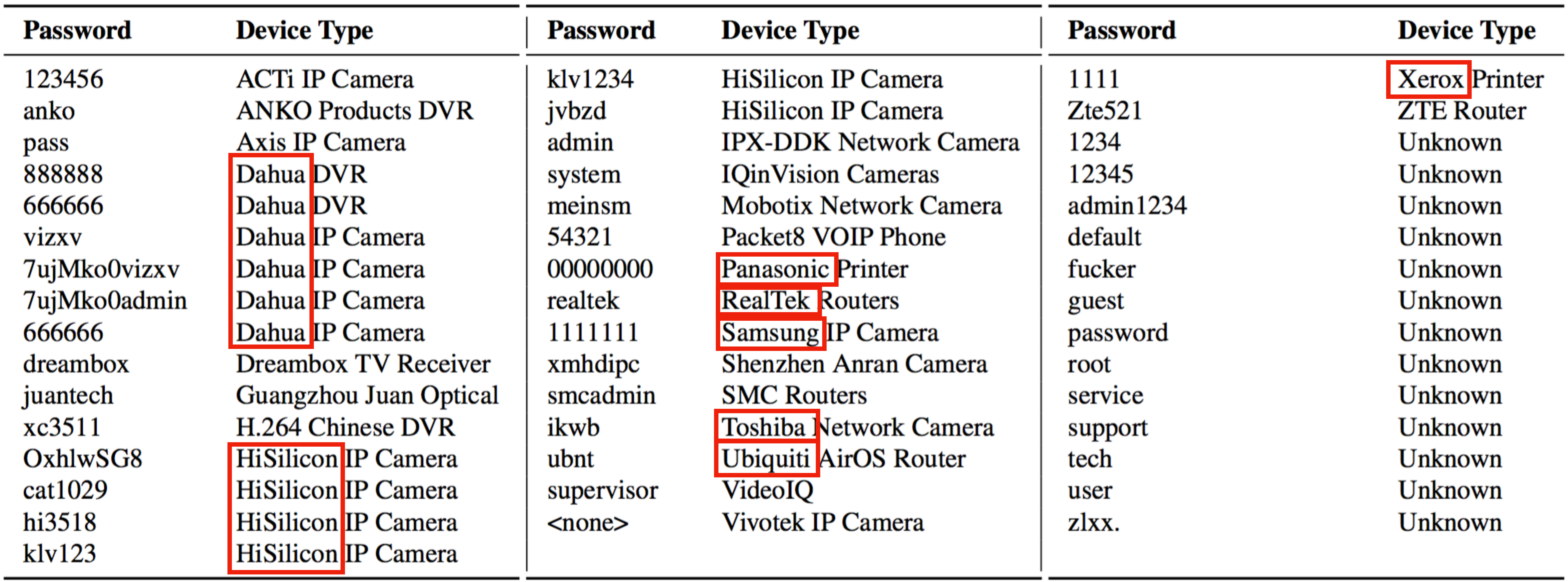
The Mirai malware is (astoundingly) badly written. It uses no new or complex techniques.

Mirai was successful because:

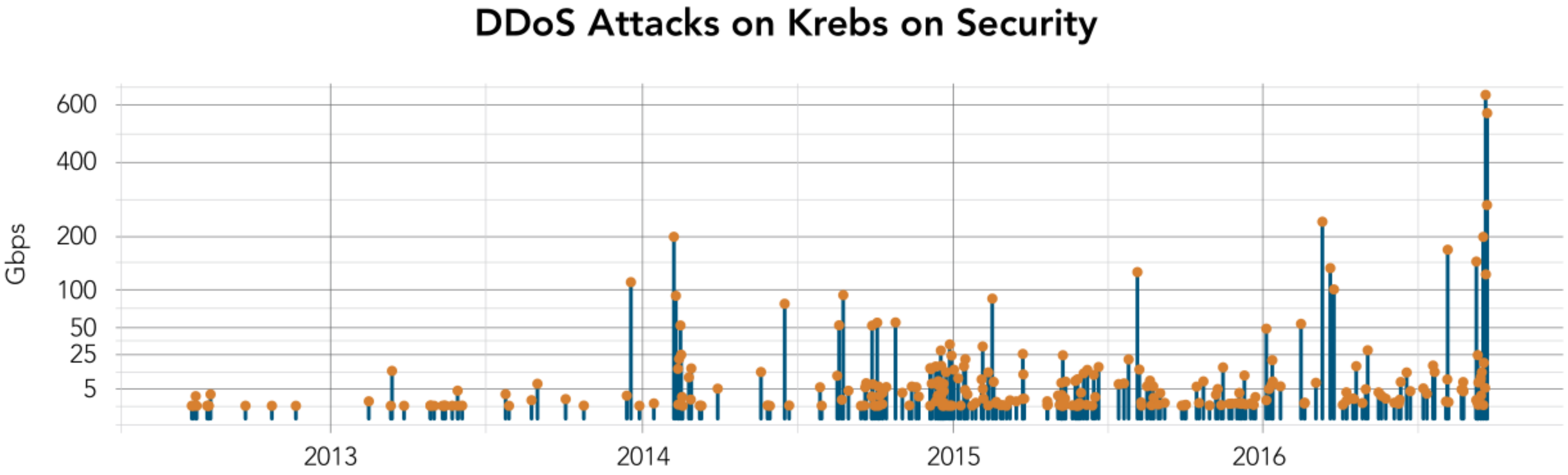
1. IoT security bar is very low
2. Attack simplicity enabled the malware to compromise heterogeneous hardware
3. Stateless scanning was an improvement over prior versions



Password Guessing



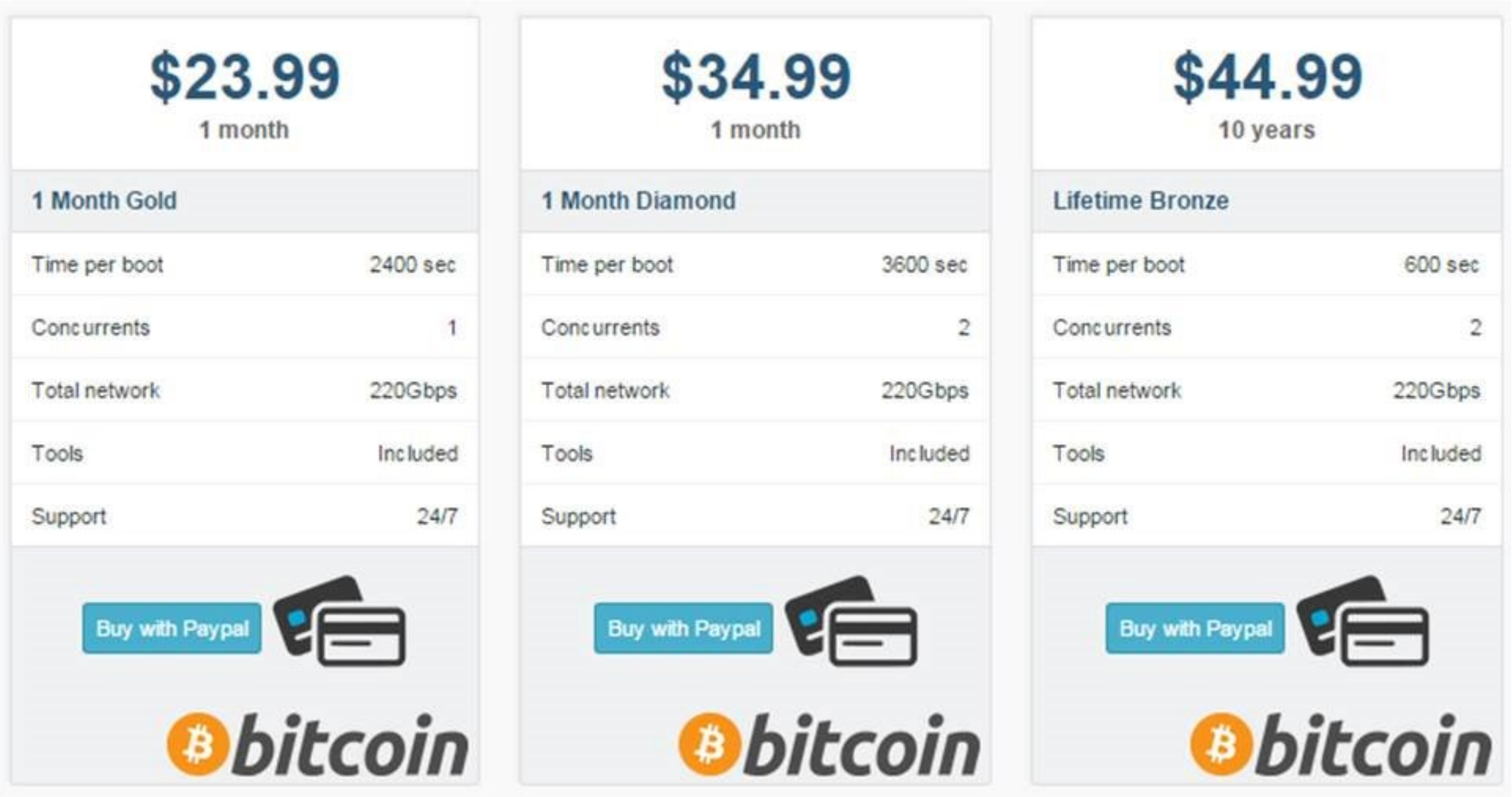
Password	Device Type
123456	ACTi IP Camera
anko	ANKO Products DVR
pass	Axis IP Camera
888888	Dahua DVR
666666	Dahua DVR
vizxv	Dahua IP Camera
7ujMko0vizxv	Dahua IP Camera
7ujMko0admin	Dahua IP Camera
666666	Dahua IP Camera
dreambox	Dreambox TV Receiver
juantech	Guangzhou Juan Optical
xc3511	H.264 Chinese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



> “The magnitude of the attacks seen during the final week were significantly larger than the majority of attacks Akamai sees on a regular basis. […] In fact, while the attack on September 20 was the largest attack ever mitigated by Akamai, the attack on September 22 would have qualified for the record at any other time, peaking at 555 Gbps.”

Source: 2017 Akamai State of the Internet

Booter Services



$23.99
1 month

1 Month Gold	
Time per boot	2400 sec
Concurrents	1
Total network	220Gbps
Tools	Included
Support	24/7

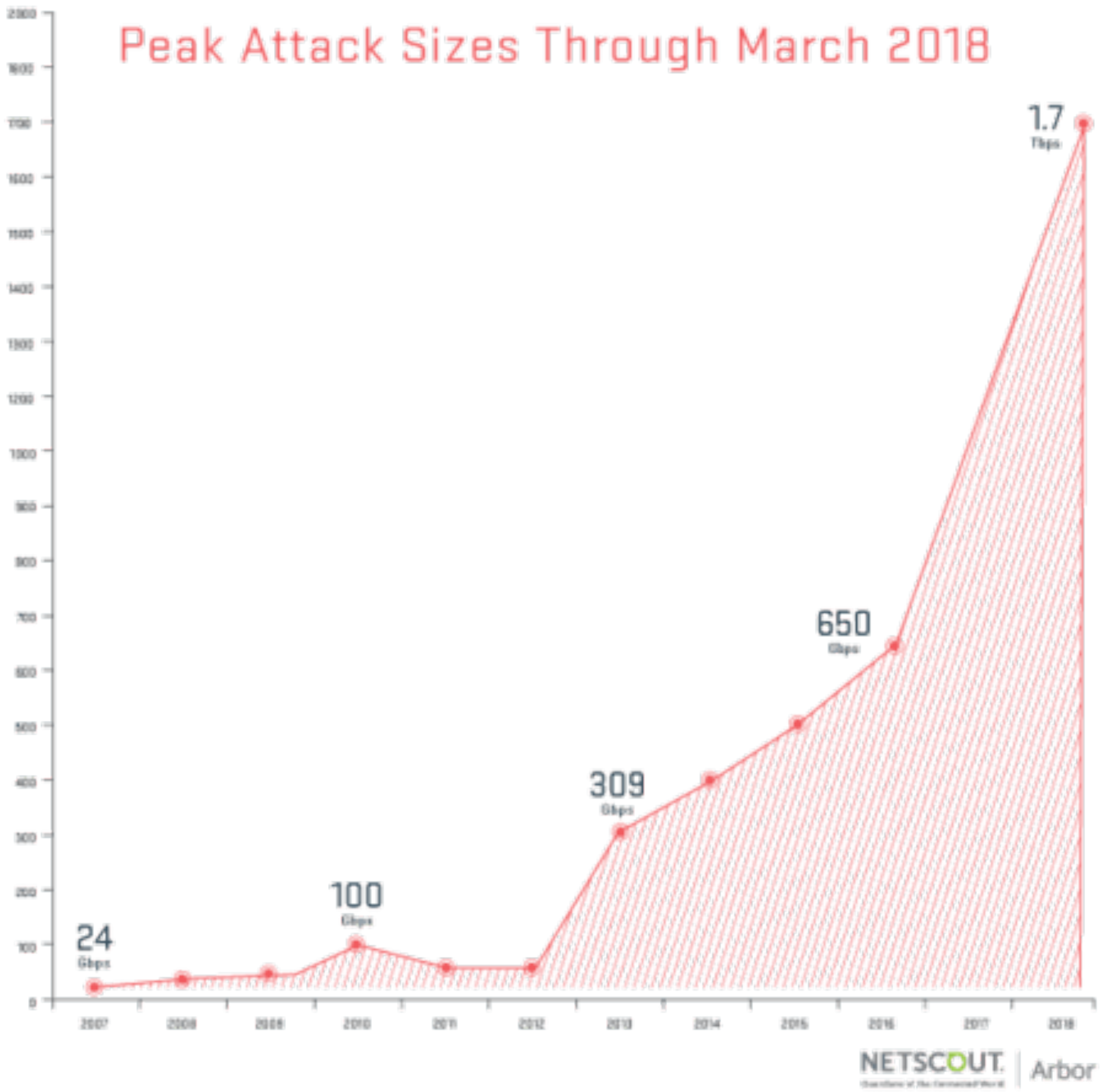
$34.99
1 month

1 Month Diamond	
Time per boot	3600 sec
Concurrents	2
Total network	220Gbps
Tools	Included
Support	24/7

$44.99
10 years

Lifetime Bronze	
Time per boot	600 sec
Concurrents	2
Total network	220Gbps
Tools	Included
Support	24/7

Memcache



Memcache: retrieve large record

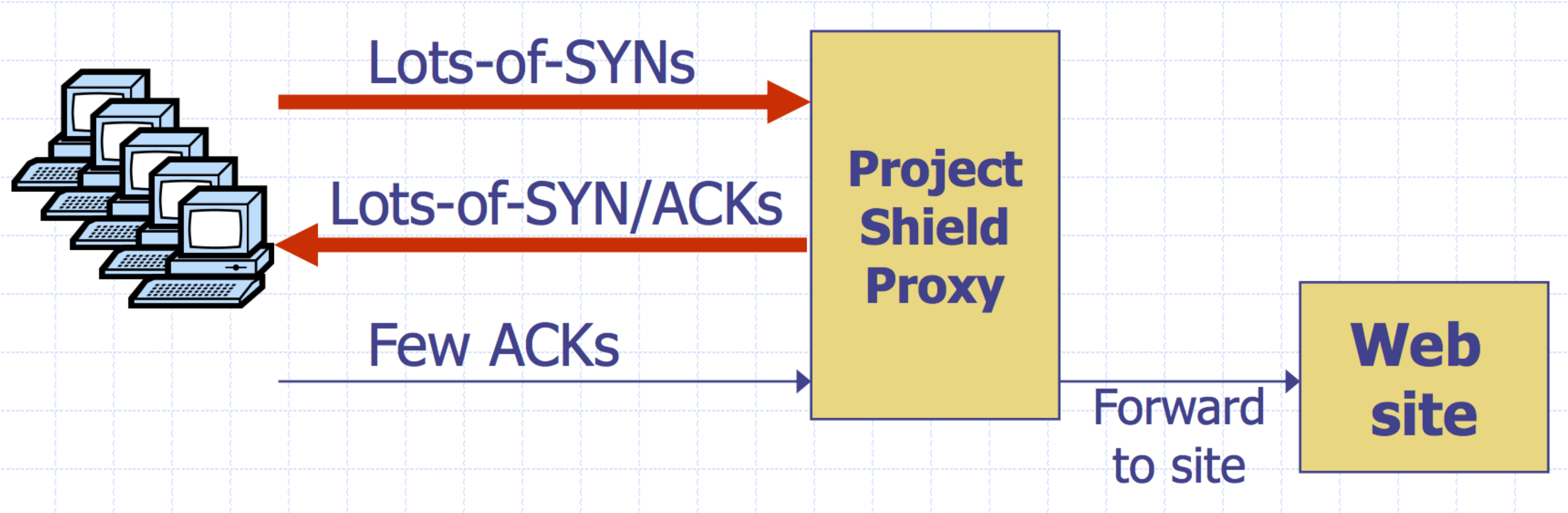
The server responds by firing back as much as 50,000 times the data it received.

Exist both a UDP and TCP version. Only works for UDP! TCP would require a three-way handshake and server would realize IP had been spoofed.

Google Project Shield

DDoS Attacks are often used to censor content. In the case of Mirai, Brian Kreb’s blog was under attack.

Google Project shield uses Google bandwidth to shield vulnerable websites (e.g., news, blogs, human rights orgs)



Moving Up Stack: GET Floods

Command bot army to:

* Complete real TCP connection
* Complete TLS Handshake
* GET large image or other content

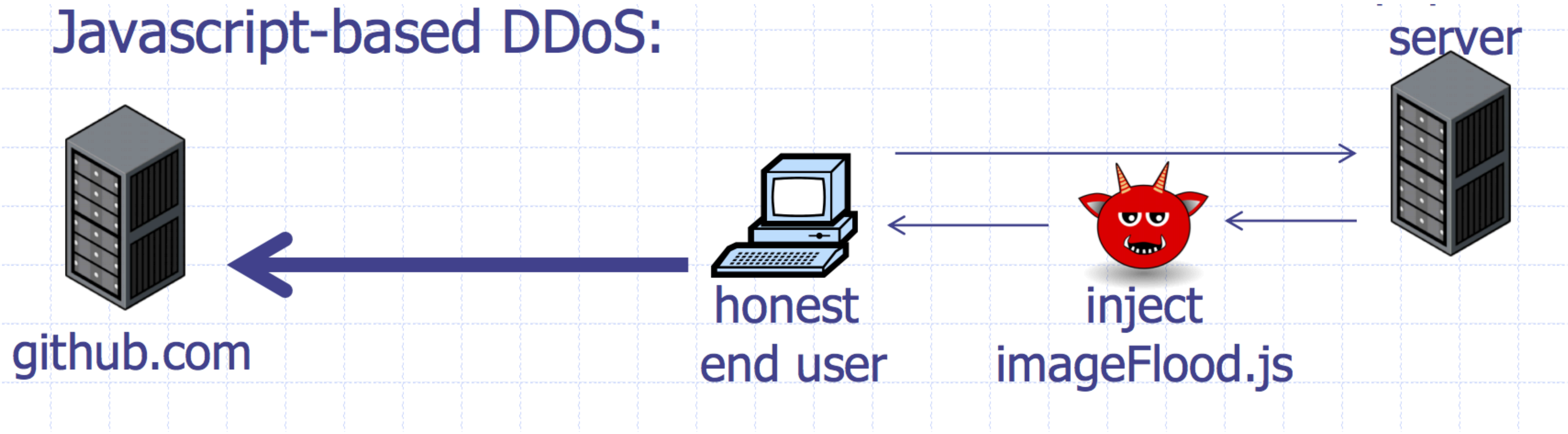
Will bypass flood protections…. but attacker can no longer use random source IPs

Victim site can block or rate limit bots

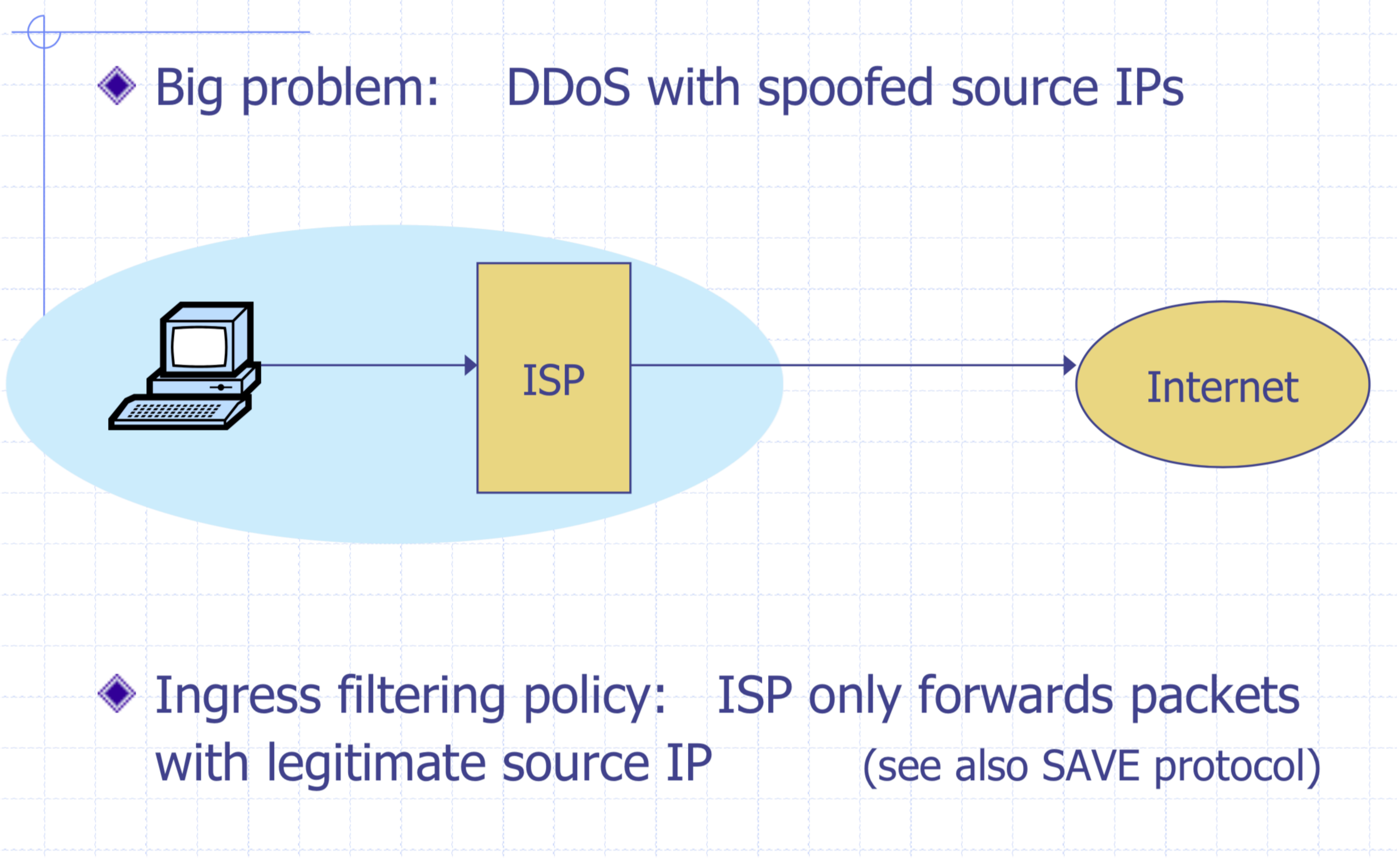
Github Attacks

1.35 Tbps attack against Github caused by javascript injected into HTTP web requests

The Chinese government was widely suspected to be behind the attack



Ingress Filtering



- Big problem: DDoS with spoofed source IPs

- Ingress filtering policy: ISP only forwards packets with legitimate source IP (see also SAVE protocol)

Ingress Filtering

All ISPs need to do this — requires global coordination

- If 10% of networks don't implement, there's no defense
- No incentive for an ISP to implement — doesn't affect them

As of 2017 (from CAIDA):

- 33% of autonomous systems allow spoofing
- 23% of announced IP address space allow spoofing

2013 300 Gbps attack sent attack traffic from only 3 networks

Client Puzzles

Idea: What if we force every client to do moderate amount of work for every connection they make?

Example:

1) Server Sends: C

2) Client: find X s.t. $LSB_n(SHA\text{-}1(C||X)) = 0^n$

Assumption:

Puzzle takes 2^n for the client to compute (0.3 s on 1Ghz core)

Solution is trivial for server to check (single SHA-1)

Client Puzzles

Not frequently used in the real world

Benefits:

* Can change *n* based on amount of attack traffic

Limitations:

* Requires changes to both protocols, clients, and servers

* Hurts low power legitimate clients during attack (e.g., phones)

Network Defenses

Local Services

Review: Popular TCP and UDP services live on standardized ports. HTTPS servers listen on TCP/443. SSH on TCP/22.

Some services you don’t want listening on the public Internet.

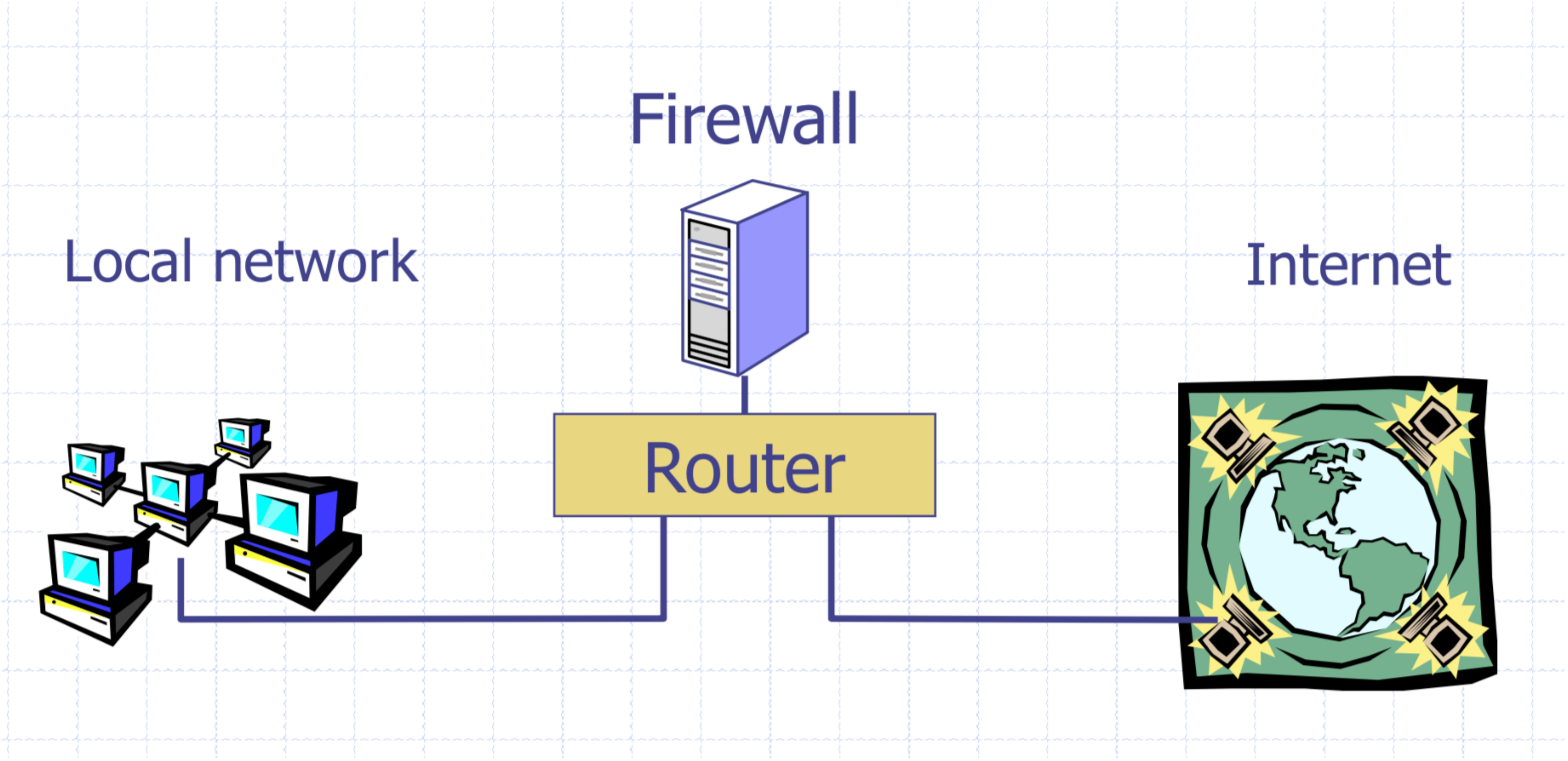
Recursive DNS Resolvers: allows attackers to mount DDoS attacks

Windows File Sharing: historically full of vulnerabilities. What if a local machine doesn’t have a secure password on it?

Firewalls

Separate local area network (LAN) from the Internet. Only allow some traffic to transit.

Sometimes rules on a router. Sometimes a standalone device.



Basic Packet Filtering

Uses transport and IP layer information only
- IP Source Address, Destination Address
- Protocol (TCP, UDP, ICMP, etc.)
- TCP and UDP source and destination ports

Examples:

- “Do not allow external hosts to connect to Windows File Sharing”
  -> DROP ALL INBOUND PACKETS TO TCP PORT 445

What’s the rule?

What if you have a network with lots of servers but only want outsiders to be able to access a web server?

DROP ALL INBOUND PACKETS IF DEST PORT != 80

All outbound connections also have a source port! Their responses will blocked!

IANA Port Numbering

System or Well-Known Ports [1,1023]:

Common services, e.g., HTTP -> 80, SSH -> 22

User or registered ports [1024, 49151]

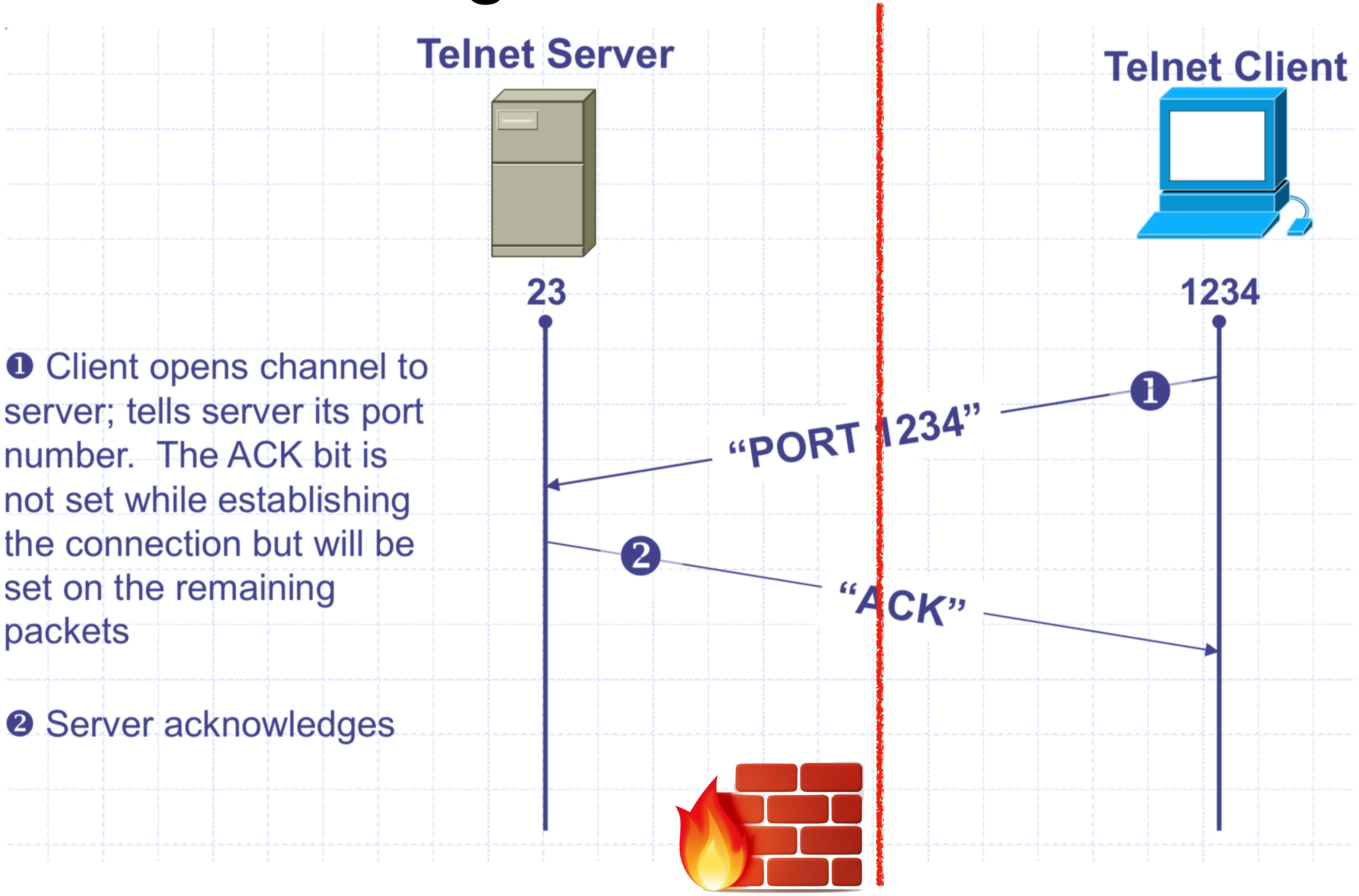
Less well-known services

Ephemeral/Dynamic/Private Ports [49152, 65535]

Short lived connections

Stateful Filtering

Firewall tracks outgoing connections and allows associated inbound traffic back through

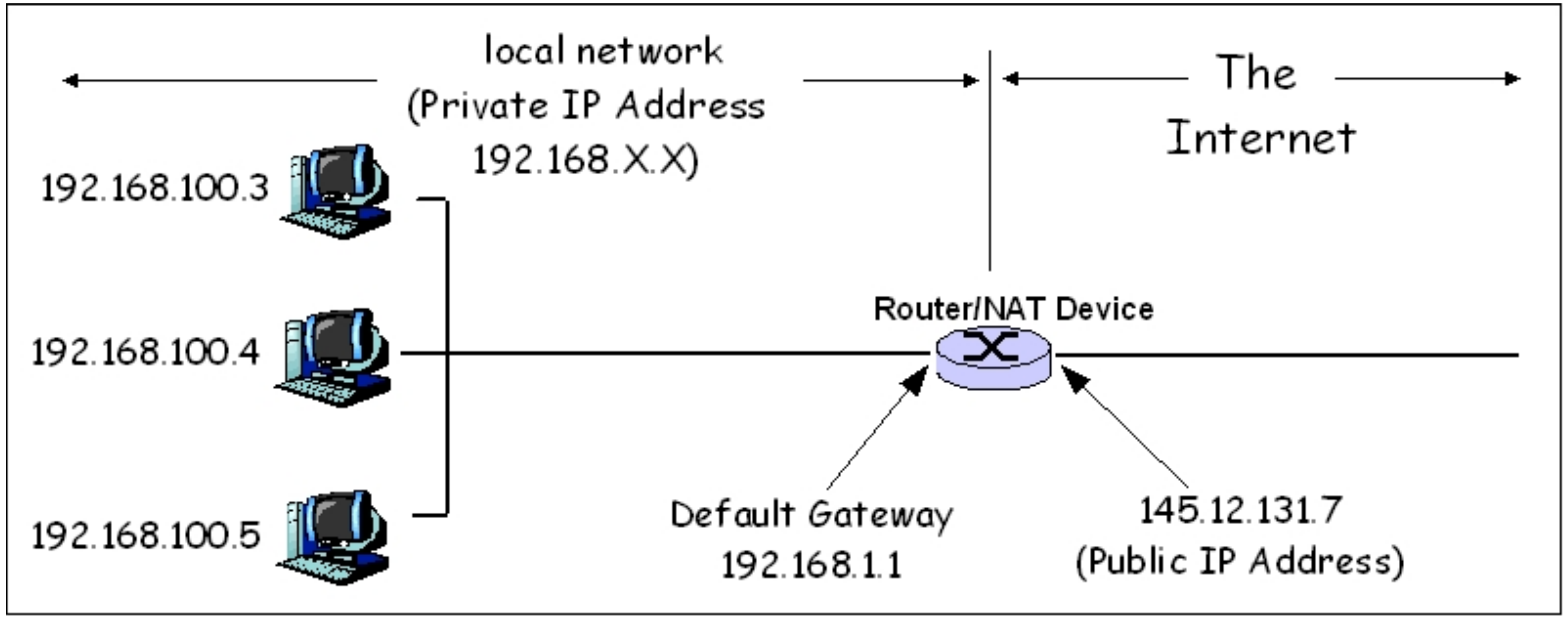


❶ Client opens channel to server; tells server its port number. The ACK bit is not set while establishing the connection but will be set on the remaining packets

❷ Server acknowledges

Network Address Translation (NAT)

NATs map between two different address spaces. Most home routers are NATs and firewalls.



Private Subnets

10.0.0.0 – 10.255.255.255

172.16.0.0 – 172.31.255.255

192.168.0.0 – 192.168.255.255

Local vs. Network Firewall

Firewalls we’ve discussed so far have all been network firewalls. Most have lived at the edge of the organization.

Firewalls also run on individual hosts. Linux servers use **iptables**.

Typically have a combination of network and host firewalls

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
```

Application Layer Filtering

Enforce protocol-specific policies:

- Virus scanning for SMTP
    - Need to understand protocol, MIME encoding, ZIP files, etc
- Look for SQL injection attacks in HTTP POSTs

Outbound Too!

Organizations will often inspect outbound traffic as well

- Block access to sites with known malicious behavior
- Prevent exfiltrating data
- Block services like bit torrent

Be careful on enterprise networks! Sometimes companies will even install their own root certificates on employee workstations to monitor TLS traffic.

Intrusion Detection Systems (IDS)

Software/device to monitor network traffic for attacks or policy violations

Violations are reported to a central security information and event management (SIEM) system where analysts can later investigate

Signature Detection: maintains long list of traffic patterns (rules) associated with attacks

Anomaly Detection: attempts to learn normal behavior and report deviations

Open Source IDS

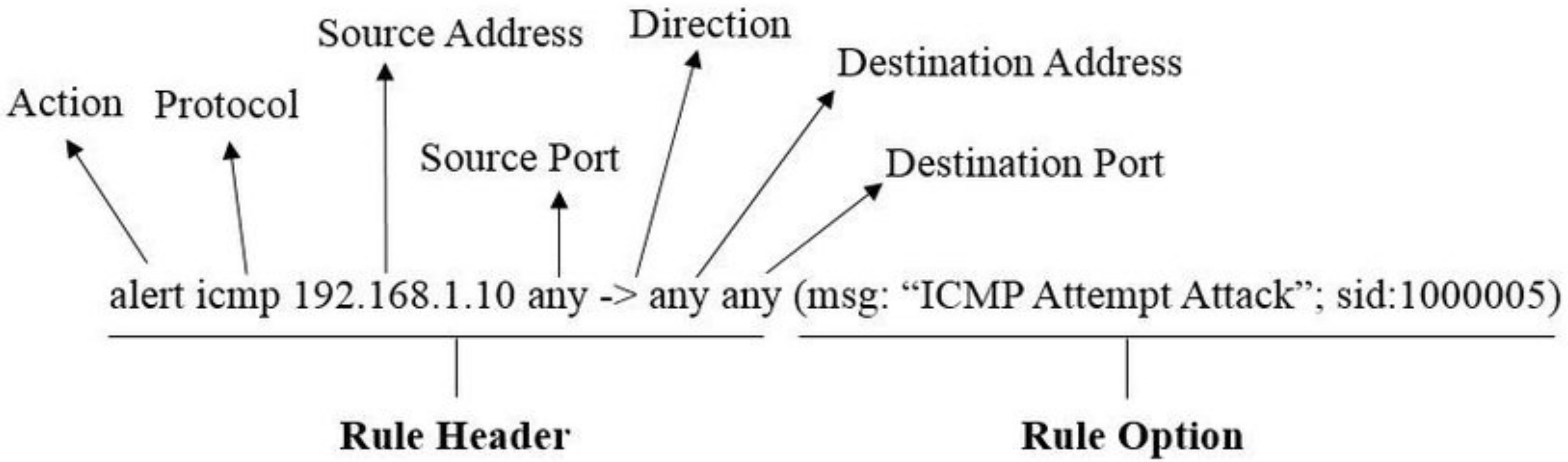
Three Major Open Source IDS (and a *tremendous* number of commercial products)

Snort

~~Bro~~ Zeek

Suricata

Example Snort Rule



Action

Protocol

Source Address

Source Port

Direction

Destination Address

Destination Port

alert icmp 192.168.1.10 any -> any any (msg: “ICMP Attempt Attack”; sid:1000005)

Rule Header

Rule Option

Remote Access

Virtual Private Networks (VPNs)

Problem: How do you provide secure communication for non-TLS protocols across the public Internet?

VPNs create a fake shared network on which traffic is encrypted

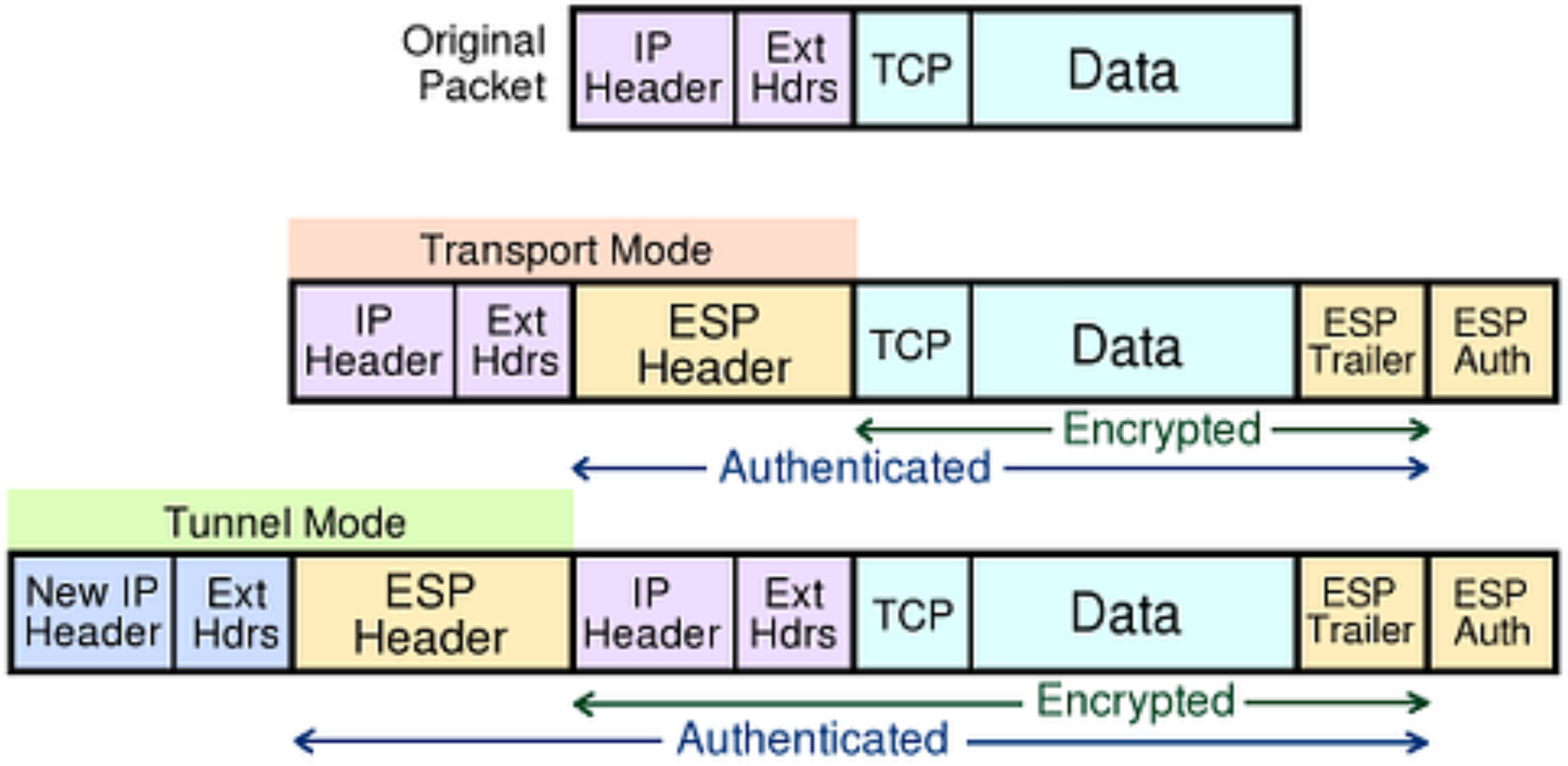
Two Broad Types:

- Remote client (e.g., traveler with laptop) to corporate network
- Connect two remote networks across Internet

IPSec

Several VPN protocols exist (PPTP, L2TP, IPsec, OpenVPN)

Most popular is IPsec. OpenVPN is open source.



Cisco AnyConnect

Stanford and many other organizations use Cisco AnyConnect

Encapsulates traffic in TLS! Initial handshake uses normal TCP-based TLS for initial handshake and then DTLS (UDP-based TLS) to transport data

Gooey Middle

VPNs support the idea of having a secure internal network and untrusted public Internet. Unfortunately, attacker has a ton of access once the network perimeter is breached.

Unfortunately, internal networks aren’t *that* secure. Computers are compromised all the time and attackers have free rein.

Zero Trust Security (BeyondCorp)

Google: assume internal network is *also* out to get you. Remove privileged intranet and put all corporate applications on the Internet.

Access depends solely on device and user credentials, regardless of a user’s network location

Protect applications, not the network