14. Network Attacks & Defenses

Blase Ur and David Cash
(some slides borrowed from Ben Zhao, Christo Wilson, & others)

February 8th, 2023

CMSC 23200 / 33250

THE UNIVERSITY OF CHICAGO

Network threat model

- Network scanning
- Attacks on confidentiality
  (e.g., eavesdropping, side channel information)
- Attacks on integrity
  (e.g., spoofing, packet injection)
- Attacks on availability
  (e.g., denial of service, or **DoS**)

Scanning and observing networks

Network Scanning: Ping

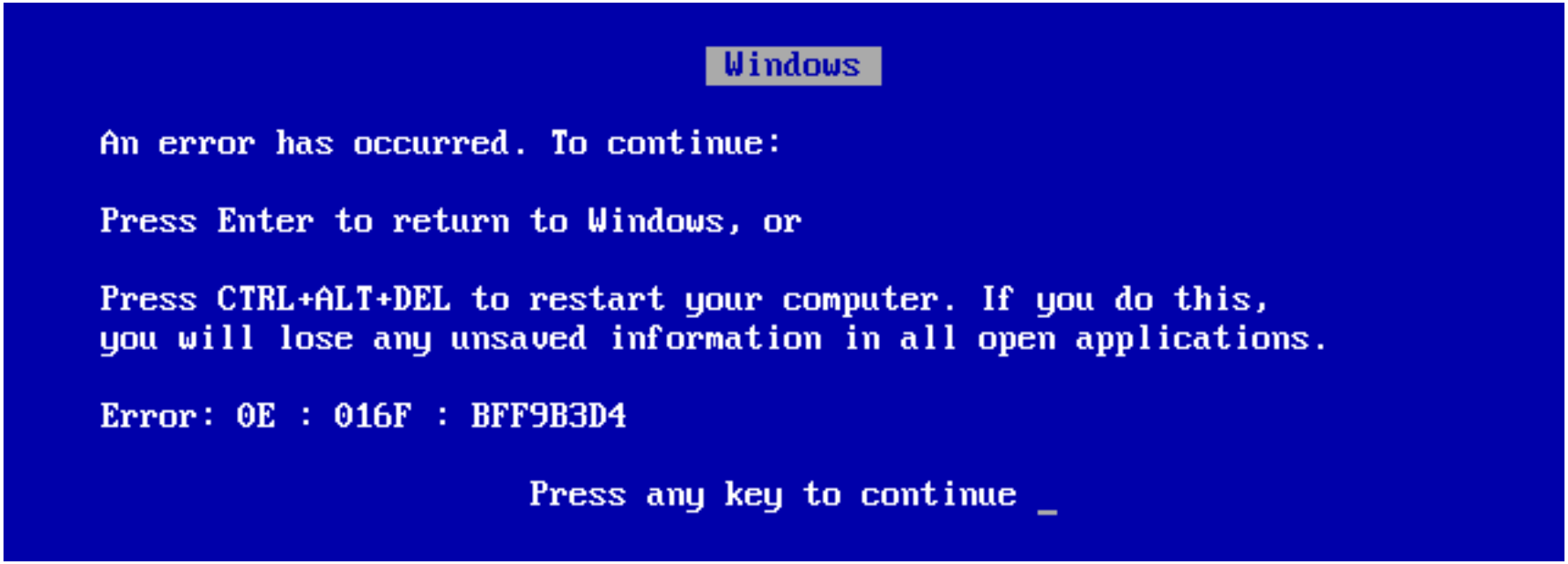
- Essential, low-level network utility
- Sends a “ping” ICMP message to a host on the internet

```
$ ping 66.66.0.255
PING 66.66.0.255 (66.66.0.255) 56(84) bytes of data.
64 bytes from 66.66.0.255: icmp_seq=1 ttl=58 time=41.2 ms
```

- Destination host is supposed to respond with a “pong”
  - Indicating that it can receive packets
- By default, ping messages are 56 bytes long (+ some header bytes)
  - Maximum size 65535 bytes
- What if you send a ping that is >65535 bytes long?

Ping of Death

- $ ping –s 65535 66.66.0.255
  - Attack identified in 1997
  - IPv6 version identified/fixed in 2013



Network Scanning: Traceroute

- traceroute — hops between me and host
  - Sends repeated ICMP reqs w/ increasing TTL

```
thor Wed Oct 24(12:51am)[~]:-> traceroute www.slack.com
traceroute to www.slack.com (52.85.115.213), 64 hops max, 52 byte packets
 1  v11router (128.135.11.1)  1.265 ms  0.788 ms  0.778 ms
 2  a06-021-100-to-d19-07-200.p2p.uchicago.net (10.5.1.186)  1.292 ms  0.749 ms  0.833 ms
 3  d19-07-200-to-h01-391-300.p2p.uchicago.net (10.5.1.46)  2.124 ms  2.435 ms  2.072 ms
 4  192.170.192.34 (192.170.192.34)  0.755 ms
    192.170.192.32 (192.170.192.32)  0.810 ms  0.701 ms
 5  192.170.192.36 (192.170.192.36)  0.887 ms  0.918 ms  0.877 ms
 6  r-equinix-isp-ae2-2213.wiscnet.net (216.56.50.45)  1.625 ms  1.803 ms  1.866 ms
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  178.236.3.103 (178.236.3.103)  4.516 ms  4.326 ms  4.320 ms
12  * * *
13  * * *
14  * * *
15  server-52-85-115-213.ind6.r.cloudfront.net (52.85.115.213)  4.554 ms  4.398 ms  4.757 ms
thor Wed Oct 24(12:52am)[~]:->
```

Port Scanning

- What services are running on a server? Nmap

```
linux3 Wed Oct 24(12:54am)[~]:-> nmap www.cs.uchicago.edu

Starting Nmap 7.01 ( https://nmap.org ) at 2018-10-24 00:55 CDT
Nmap scan report for www.cs.uchicago.edu (34.203.108.171)
Host is up (0.019s latency).
Other addresses for www.cs.uchicago.edu (not scanned): 54.164.17.80 54.85.61.218
rDNS record for 34.203.108.171: ec2-34-203-108-171.compute-1.amazonaws.com
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 4.99 seconds
linux3 Wed Oct 24(12:55am)[~]:->
```

- 5 seconds to scan a single machine!!

SYN scan

Only send SYN

Responses:

- SYN-ACK — port open
- RST — port closed
- Nothing — filtered (e.g., firewall)

Port Scanning on Steroids

- How do you speed up scans for all IPv4?
  - Don't wait for responses; pipeline
  - Parallelize: divide & conquer IPv4 ranges
  - Randomize permutations w/o collisions
- Result: the zmap tool
  - Scan all of IPv4 in 45mins (gigabit connection)
  - IPv4 in 5 mins (10 gigabit connection)

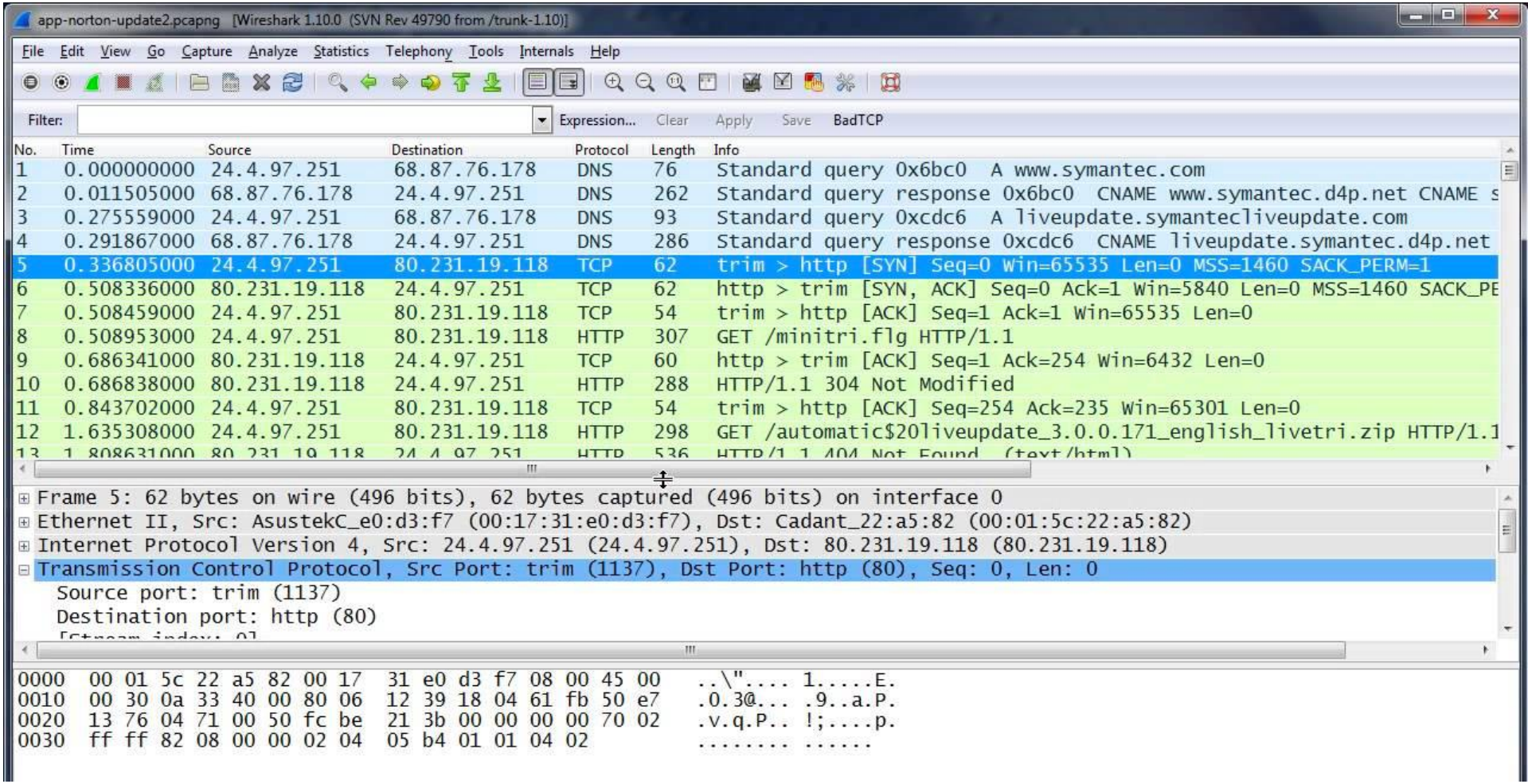
Eavesdropping

Tools: Wireshark, tcpdump, Zeek (Bro), …

Steps:

1. Parse data link layer frames
2. Identify network flows
3. Reconstruct IP packet fragments
4. Reconstruct TCP connections
5. Parse app protocol messages

Wireshark, Detailed Protocol Analyzer



Side channels

Overview

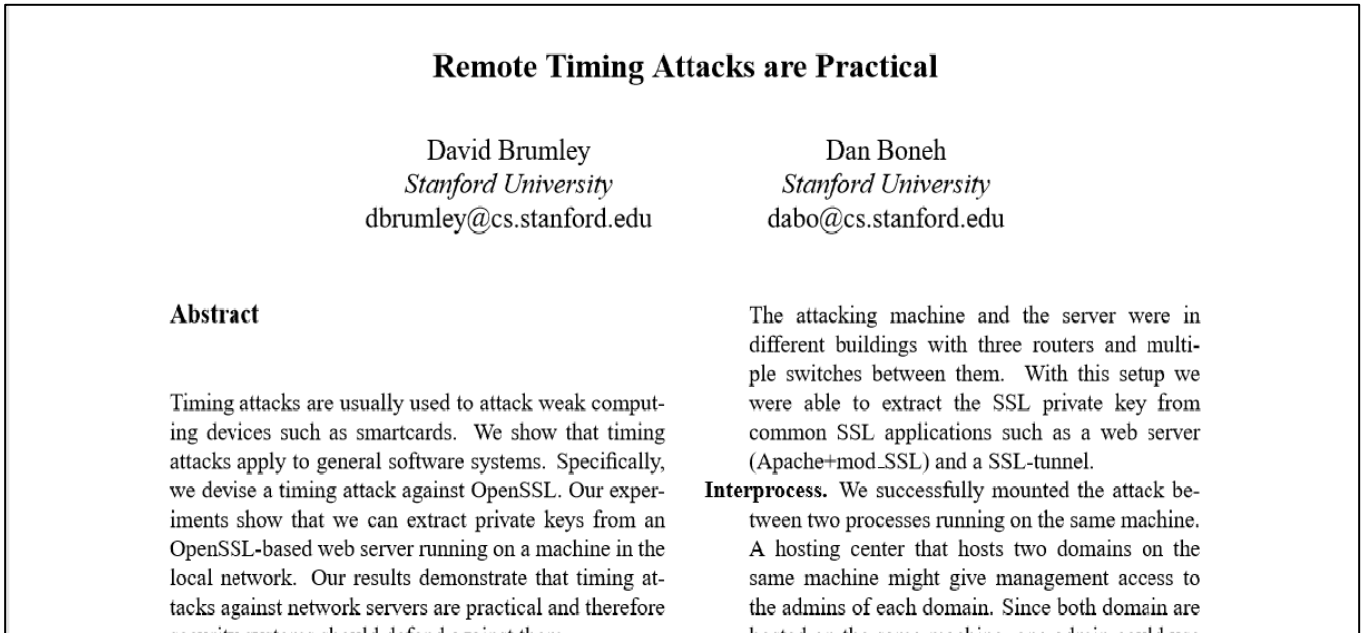
- Transport Layer Security (TLS) enables secure communication
- Frequently encountered with web browsing (HTTPS) and more behind the scenes in app, VOIP, etc.

Side Channels

- Using metadata or outside observations to make inferences about the data

Web Side Channels Include:

- Size of packets
  - How can this reveal what pages you are visiting?
- Timing



Remote Timing Attacks are Practical

David Brumley
Stanford University
dbrumley@cs.stanford.edu

Dan Boneh
Stanford University
dabo@cs.stanford.edu

Abstract

Timing attacks are usually used to attack weak computing devices such as smartcards. We show that timing attacks apply to general software systems. Specifically, we devise a timing attack against OpenSSL. Our experiments show that we can extract private keys from an OpenSSL-based web server running on a machine in the local network. Our results demonstrate that timing attacks against network servers are practical and therefore

The attacking machine and the server were in different buildings with three routers and multiple switches between them. With this setup we were able to extract the SSL private key from common SSL applications such as a web server (Apache+mod_SSL) and a SSL-tunnel.

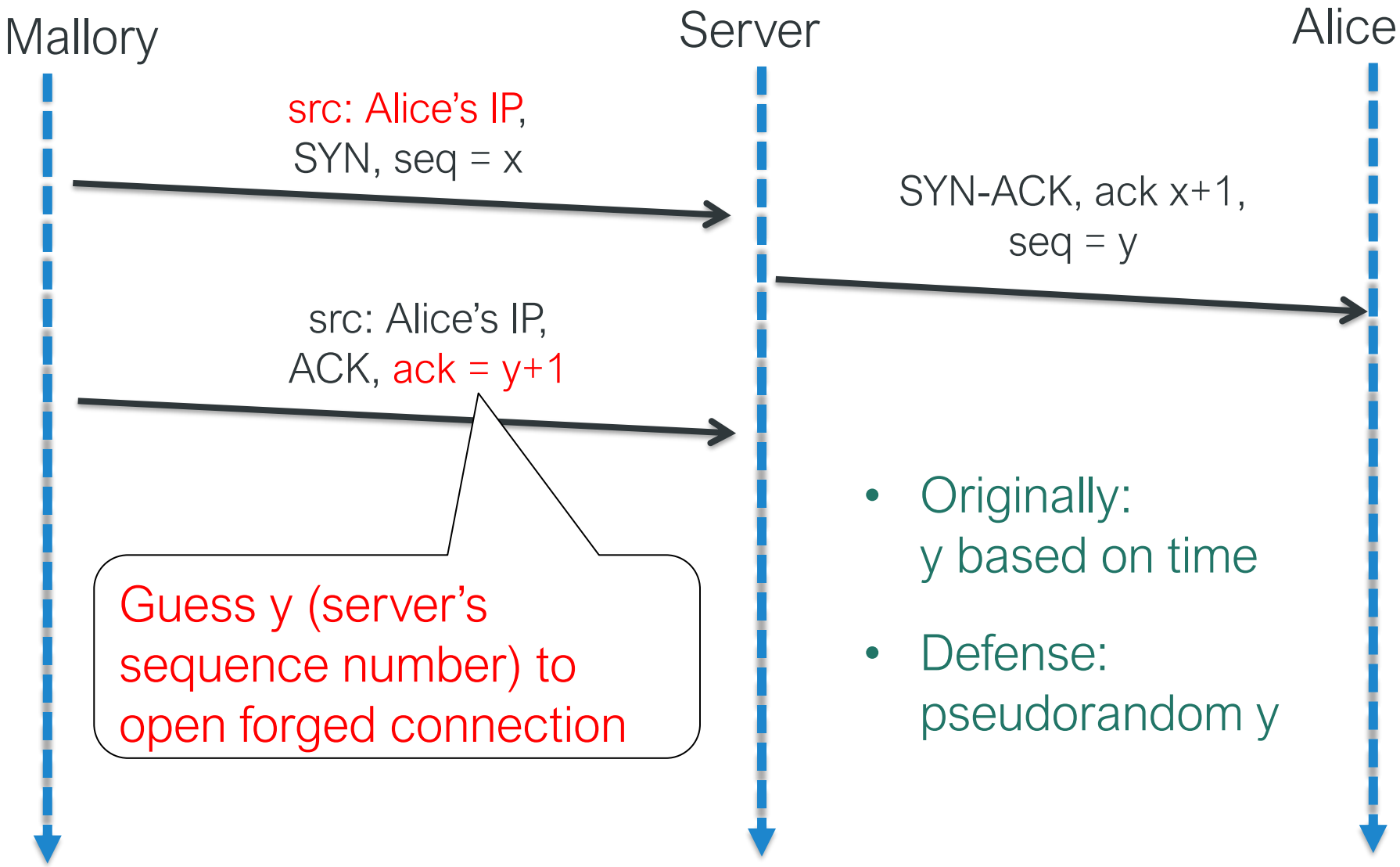
Interprocess. We successfully mounted the attack between two processes running on the same machine. A hosting center that hosts two domains on the same machine might give management access to the admins of each domain. Since both domain are

Web Side Channels Include:

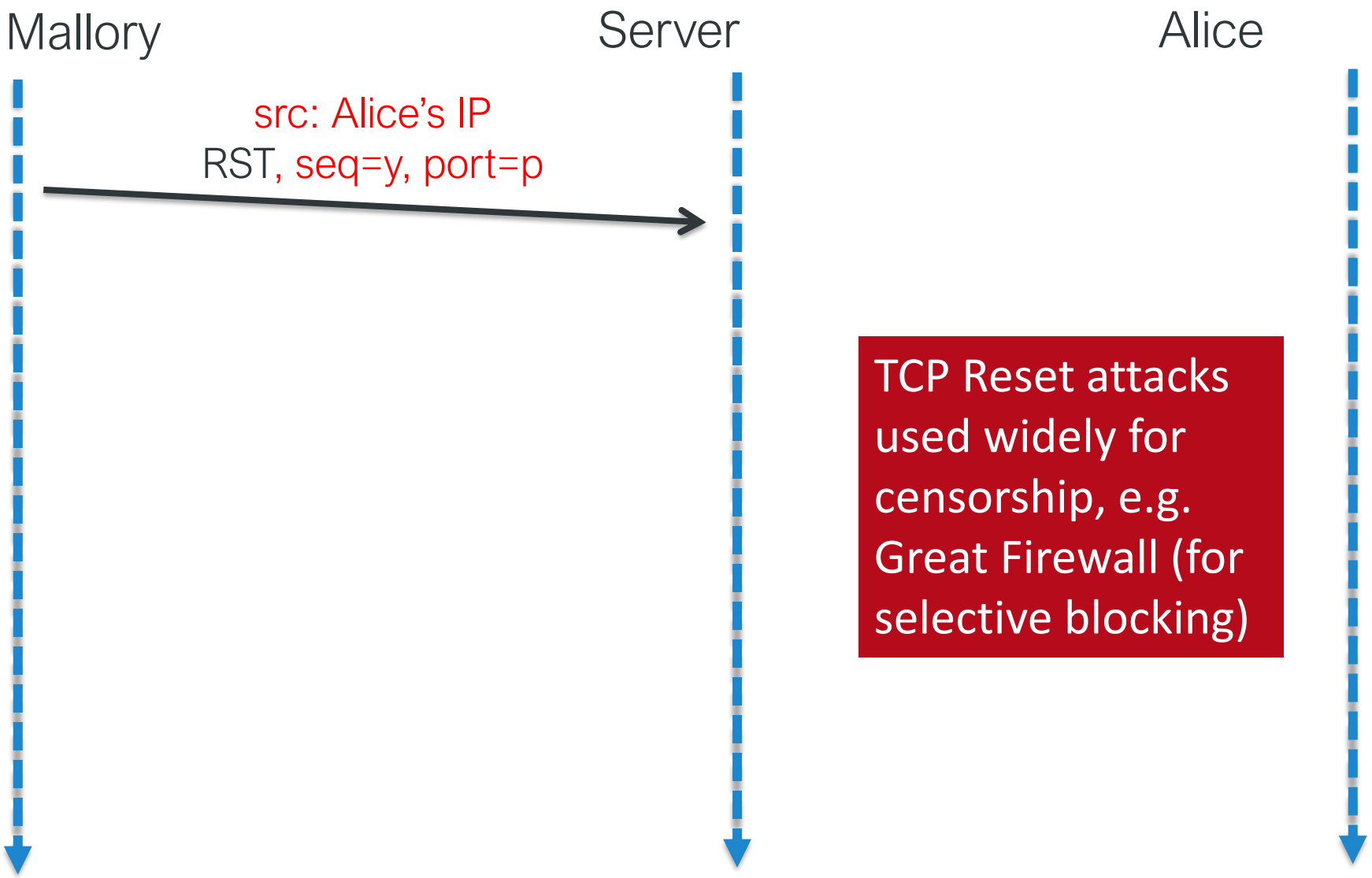
- Color
  - link one
  - second link
  - link three (visited)
  - fourth link

Protocol attacks

Active Attacks: Blind Spoofing



RST Hijacking



TCP Reset attacks used widely for censorship, e.g. Great Firewall (for selective blocking)

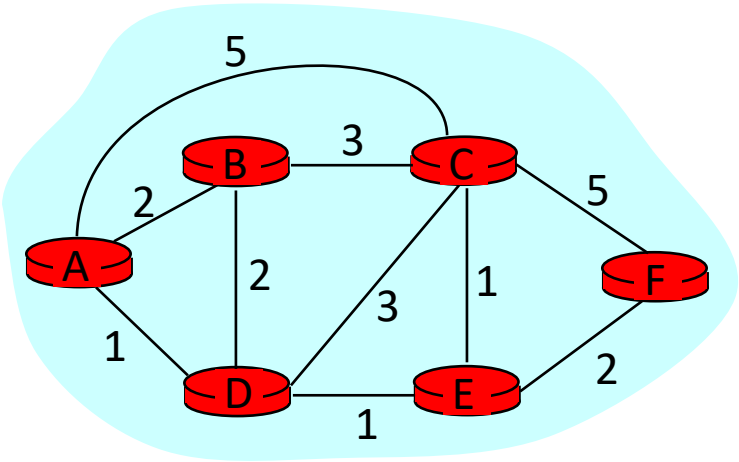
Inter-domain routing (BGP) attacks and large-scale observation

(Necessary Detour: Routing)

Routing

- Goal: determine “good” path through network from source to destination

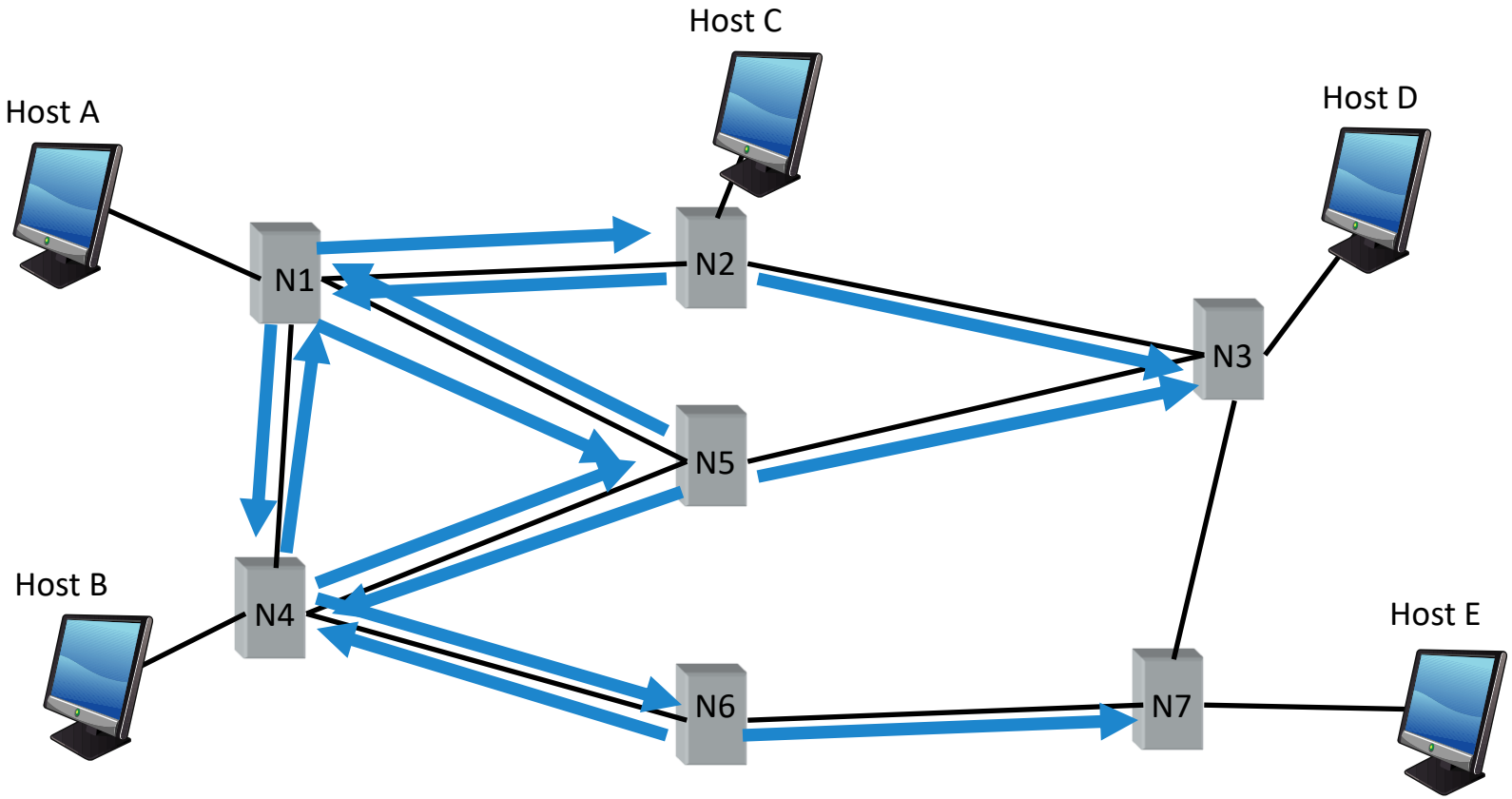
- Network modeled as a graph
  - Routers → nodes, Link →edges
    - Edge cost: delay, congestion level,.
  - A node knows only its neighbors and the cost to reach them



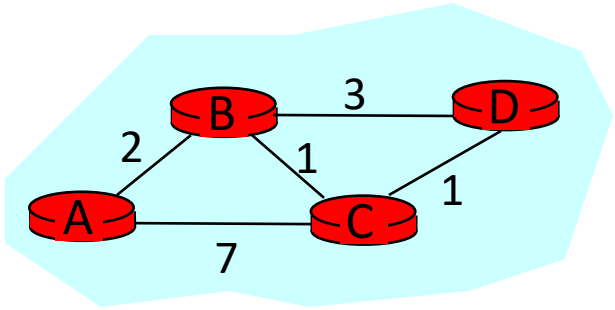
- How does each node learns how to reach every other node along the shortest path?

Distance Vector: Control Traffic

- When the routing table of a node changes, it sends table to neighbors
  - A node updates its table with information received from neighbors



Example: Distance Vector Algorithm



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

1 ***Initialization:***
2 **for all** neighbors *V* **do**
3 **if** *V* adjacent to *A*
4 D(*A, V*) = c(*A,V*);
5 **else**
6 D(*A, V*) = ∞;
…

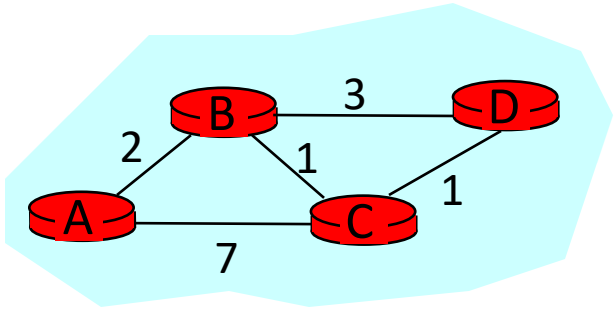
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: 1st Iteration (C → A)



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	∞	-

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

(D(C,A), D(C,B), D(C,D))

Node C

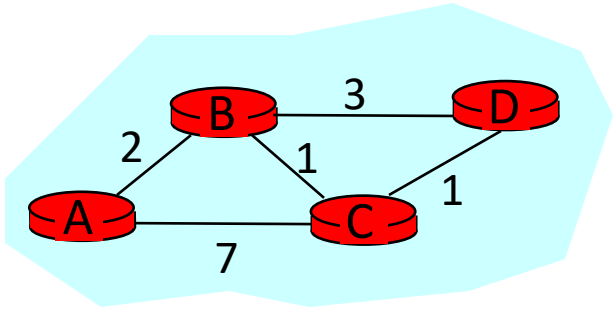
Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

```
...
7 loop:
...
12  else if (update D(V, Y) received from V)
13    for all destinations Y do
14      if (destination Y through V)
15        D(A,Y) = D(A,V) + D(V, Y);
16      else
17        D(A, Y) = min(D(A, Y),
                        D(A, V) + D(V, Y));
18  if (there is a new minimum for dest. Y)
19    send D(A, Y) to all neighbors
20  forever
```

Example: 1st Iteration (C → A)




```
...
7 loop:
...
12  else if (update D(V, Y) received from V)
13    for all destinations Y do
14      if (destination Y through V)
15        D(A,Y) = D(A,V) + D(V, Y);
16      else
17        D(A, Y) = min(D(A, Y),
                        D(A, V) + D(V, Y));
18  if (there is a new minimum for dest. Y)
19    send D(A, Y) to all neighbors
20  forever
```

Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	**8**	**C**

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A,D)= min(D(A, D),D(A,C)+D(C,D)
= min(∞ , 7 + 1) = 8

(D(C,A), D(C,B), D(C,D))

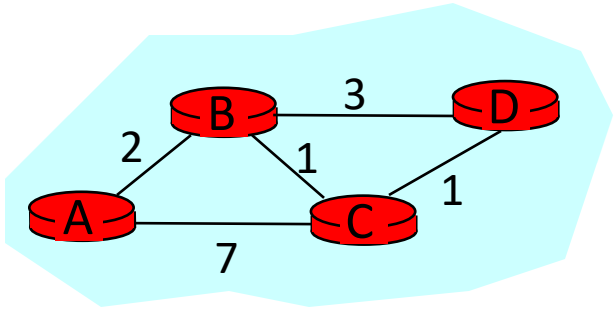
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: 1st Iteration (C → A)



Node A

Dest.	Cost	NextHop
B	2	B
C	7	C
D	8	C

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

…
7 ***loop:***
…

```
12  else if (update D(V, Y) received from V)
13    for all destinations Y do
14      if (destination Y through V)
15        D(A,Y) = D(A,V) + D(V, Y);
16      else
17        D(A, Y) = min(D(A, Y),
                        D(A, V) + D(V, Y));
18  if (there is a new minimum for dest. Y)
19    send D(A, Y) to all neighbors
20  forever
```

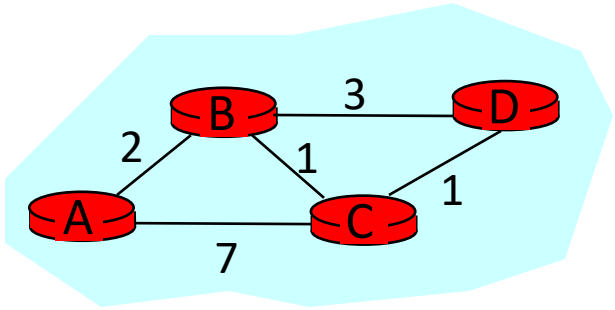
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: 1st Iteration (B→A, C→A)



Node A

Dest.	Cost	NextHop
B	2	B
C	**3**	**B**
D	**5**	**B**

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	3	D

D(A,D) = min(D(A,D), D(A,B) + D(B,D))
= min(8, 2 + 3) = 5

D(A,C) = min(D(A,C), D(A,B) + D(B,C))
= min(7, 2 + 1) = 3

…
7 ***loop:***
…

```
12  else if (update D(V, Y) received from V)
13    for all destinations Y do
14      if (destination Y through V)
15        D(A,Y) = D(A,V) + D(V, Y);
16      else
17        D(A, Y) = min(D(A, Y),
                        D(A, V) + D(V, Y));
18  if (there is a new minimum for dest. Y)
19    send D(A, Y) to all neighbors
20  forever
```

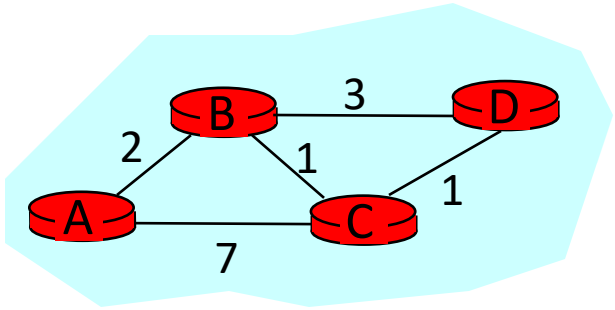
Node C

Dest.	Cost	NextHop
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	∞	-
B	3	B
C	1	C

Example: End of 1st Iteration



...
7 ***loop:***
...
12 **else if** (update D(*V, Y*) received from *V*)
13 **for all** destinations Y **do**
14 **if** (destination *Y* through *V*)
15 D(*A,Y*) = D(*A,V*) + D(*V, Y*);
16 **else**
17 D(A, Y) = min(D(*A, Y*),
D(*A, V*) + D(*V, Y*));
18 **if** (there is a new minimum for dest. *Y*)
19 **send** D(*A, Y*) to all neighbors
20 **forever**

Node A

Dest.	Cost	NextHop
B	2	B
C	**3**	**B**
D	**5**	**B**

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	**2**	**C**

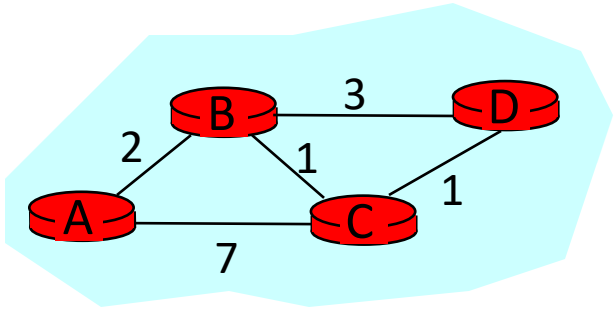
Node C

Dest.	Cost	NextHop
A	**3**	**B**
B	1	B
D	1	D

Node D

Dest.	Cost	NextHop
A	**4**	**B**
B	3	B
C	1	C

Example: End of 3nd Iteration



...
7 ***loop:***
...
12 **else if** (update D(*V, Y*) received from *V*)
13 **for all** destinations Y **do**
14 **if** (destination *Y* through *V*)
15 D(*A,Y*) = D(*A,V*) + D(*V, Y*);
16 **else**
17 D(A, Y) = min(D(*A, Y*),
D(*A, V*) + D(*V, Y*));
18 **if** (there is a new minimum for dest. *Y*)
19 **send** D(*A, Y*) to all neighbors
20 **forever**

Node A

Dest.	Cost	NextHop
B	2	B
C	3	B
D	4	B

Node B

Dest.	Cost	NextHop
A	2	A
C	1	C
D	2	C

Node C

Dest.	Cost	NextHop
A	3	B
B	1	B
D	1	D

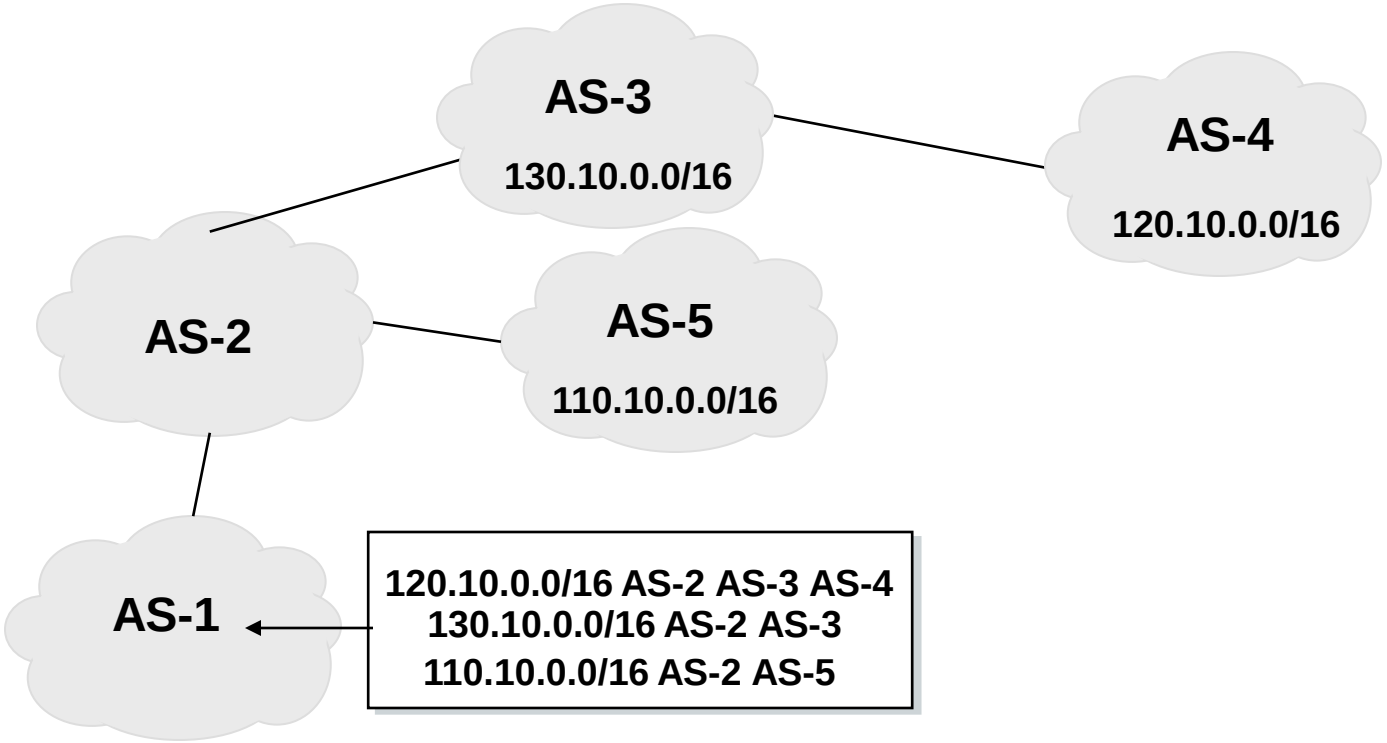
Node D

Dest.	Cost	NextHop
A	4	C
B	2	C
C	1	C

Nothing changes → algorithm terminates

BGP: a Path-Vector Protocol

- An AS-path: sequence of AS's a route traverses
- Used for loop detection and to apply policy
- *Possible* default choice: route with fewest # of AS's



BGP Prefix Hijacking

- Advertise a more desirable route even if the route isn't actually more desirable, or even real

- Goal 1: Route traffic through networks you control so that you can observe the traffic
- Goal 2: Send lots of traffic to someone you don't like (denial of service)

Corrigendum- Most Urgent

GOVERNMENT OF PAKISTAN
PAKISTAN TELECOMMUNICATION AUTHORITY
ZONAL OFFICE PESHAWAR
Plot-11, Sector A-3, Phase-V, Hayatabad, Peshawar.
Ph: 091-9217279- 5829177 Fax: 091-9217254
www.pta.gov.pk

NWFP-33-16 (BW)/06/PTA February ,2008

Subject: **Blocking of Offensive Website**

Reference: This office letter of even number dated 22.02.2008.

I am directed to request all ISPs to immediately block access to the following website

URL: http://www.youtube.com/watch?v=o3s8jtvvg00

IPs: 208.65.153.238, 208.65.153.253, 208.65.153.251

Compliance report should reach this office through return fax or at email peshawar@pta.gov.pk today please.

Deputy Director
(Enforcement)

To:

1. M/s Comsats, Peshawar.
2. M/s GOL Internet Services, Peshawar.
3. M/s Cyber Internet, Peshawar.
4. M/s Cybersoft Technologies, Islamabad.
5. M/s Paknet, Limited, Islamabad
6. M/s Dancom, Peshawar.
7. M/s Supernet, Peshawar.

BGP Prefix Hijacking

4/25/2019
02:30 PM

Marc Laliberte
Commentary

Connect Directly

0 COMMENTS
COMMENT NOW

Login

100% 0%

How a Nigerian ISP Accidentally Hijacked the Internet

For 74 minutes, traffic destined for Google and Cloudflare services was routed through Russia and into the largest system of censorship in the world, China's Great Firewall.

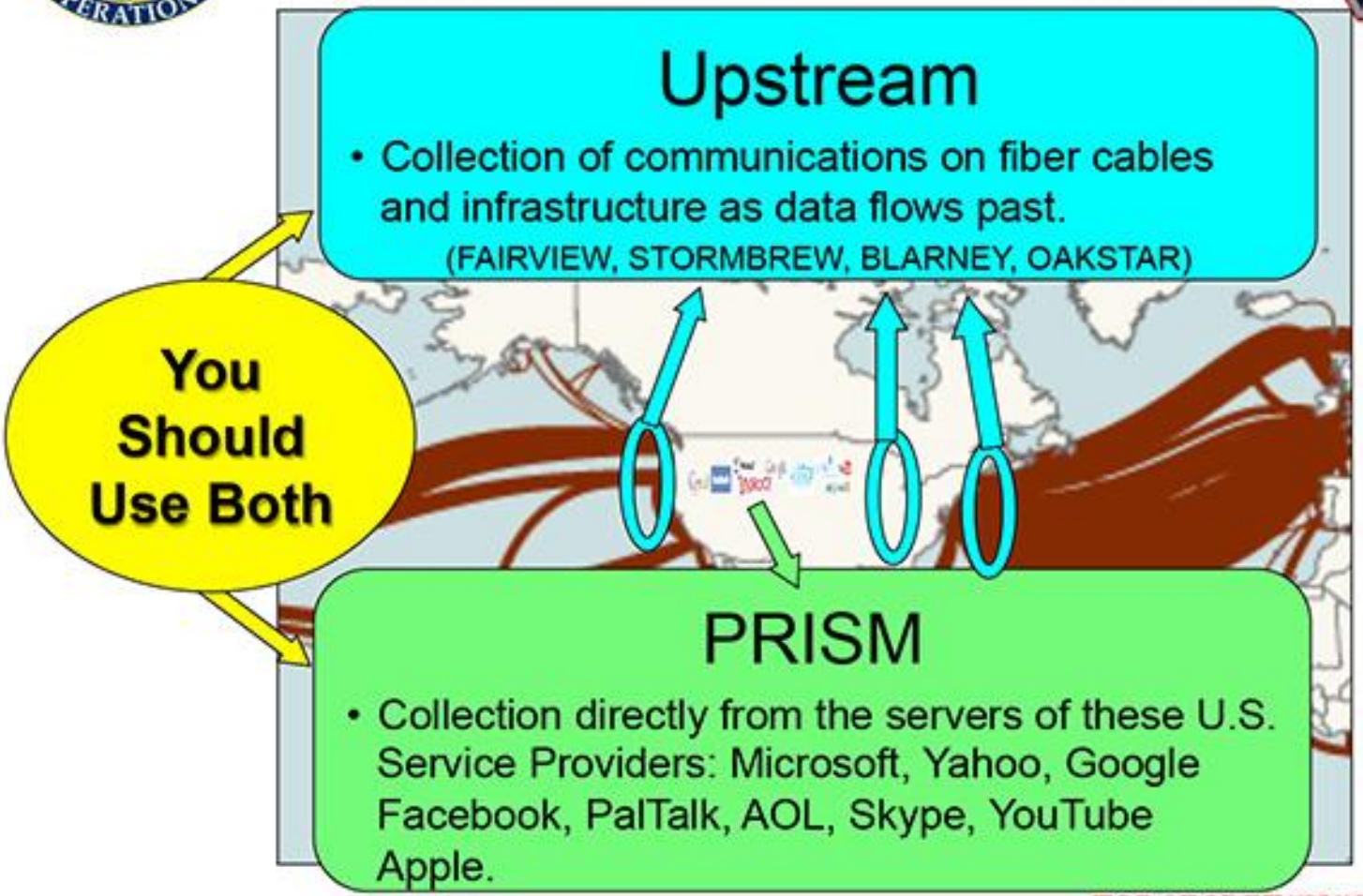
On November 12, 2018, a small ISP in Nigeria made a mistake while updating its network infrastructure that highlights a critical flaw in the fabric of the Internet. The mistake effectively brought down Google — one of the largest tech companies in the world — for 74 minutes.

To understand what happened, we need to cover the basics of how Internet routing works. When I type, for example, HypotheticalDomain.com into my browser and hit enter, my computer creates a web request and sends it to Hypothtetical.Domain.com servers. These servers likely reside in a different state or country than I do. Therefore, my Internet service provider (ISP) must determine how to route my web browser's request to the server across the Internet. To maintain their routing tables, ISPs and Internet backbone companies use a protocol called Border Gateway Protocol (BGP).

https://www.darkreading.com/cloud/how-a-nigerian-isp-accidentally-hijacked-the-internet/a/d-id/1334482

(TS//SI//NF) FAA702 Operations

Two Types of Collection



From Snowden archives, dated April 2013

(TS//SI//NF) PRISM Collection Details

Current Providers

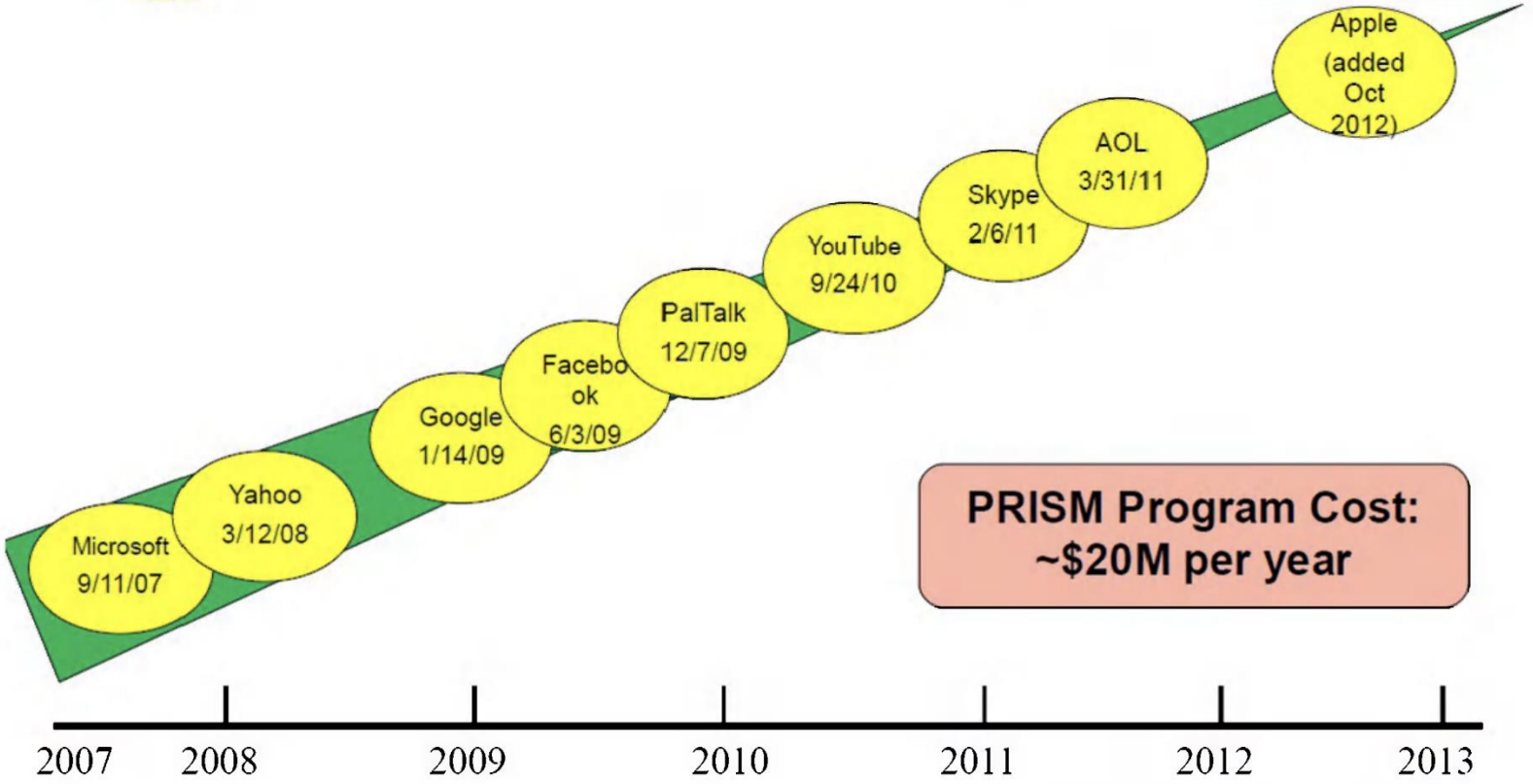
- Microsoft (Hotmail, etc.)
- Google
- Yahoo!
- Facebook
- PalTalk
- YouTube
- Skype
- AOL
- Apple

What Will You Receive in Collection (Surveillance and Stored Comms)?
It varies by provider. In general:

- E-mail
- Chat – video, voice
- Videos
- Photos
- Stored data
- VoIP
- File transfers
- Video Conferencing
- Notifications of target activity – logins, etc.
- Online Social Networking details
- **Special Requests**

Complete list and details on PRISM web page:
Go PRISMFAA

(TS//SI//NF) Dates When PRISM Collection Began For Each Provider



PRISM Program Cost: ~$20M per year

S-BGP / BGPsec

IP prefix announcements signed

Routes signed
— previous hop authorizes next hop

Higher levels vouch for lower levels
— e.g., ICANN vouches for ARIN, ARIN vouches for AT&T, …

Problem?
Costly and slow adoption

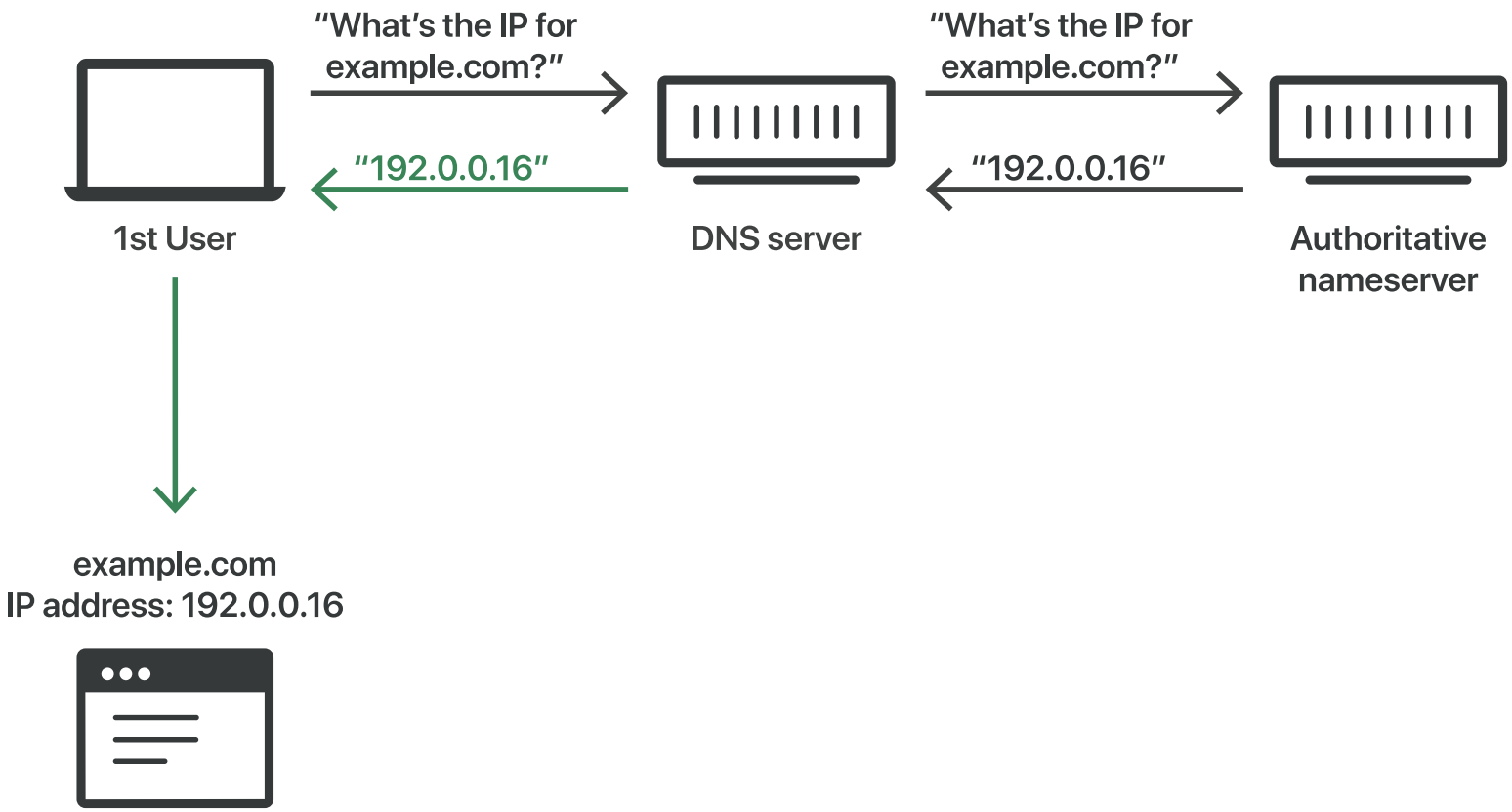
HTTP Session Hijacking

Firesheep (now discontinued)

- On shared networks (e.g., wifi), the Firesheep browser extension would sniff session cookies sent unencrypted (over HTTP)

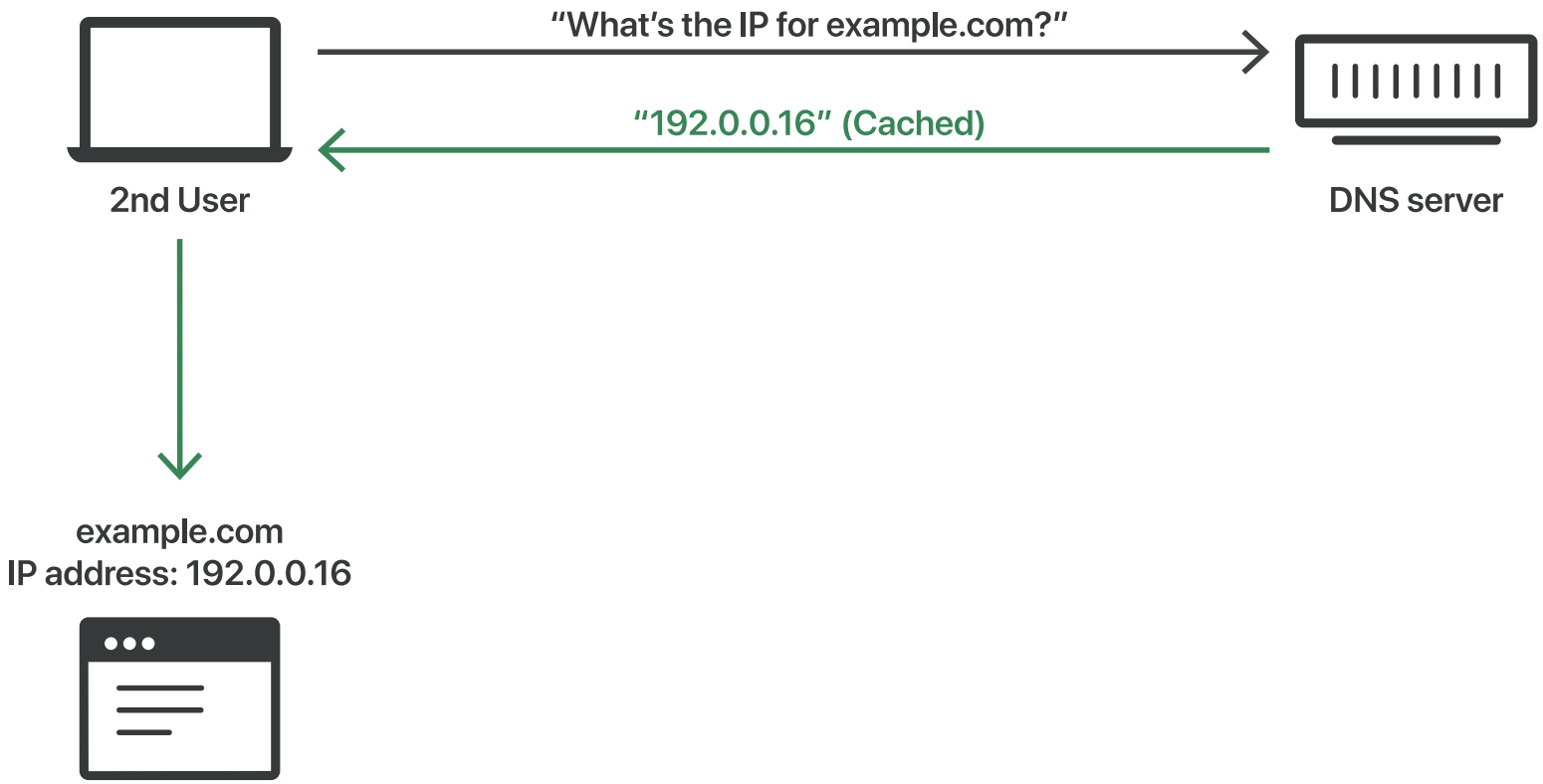
DNS attacks

DNS (Uncached)



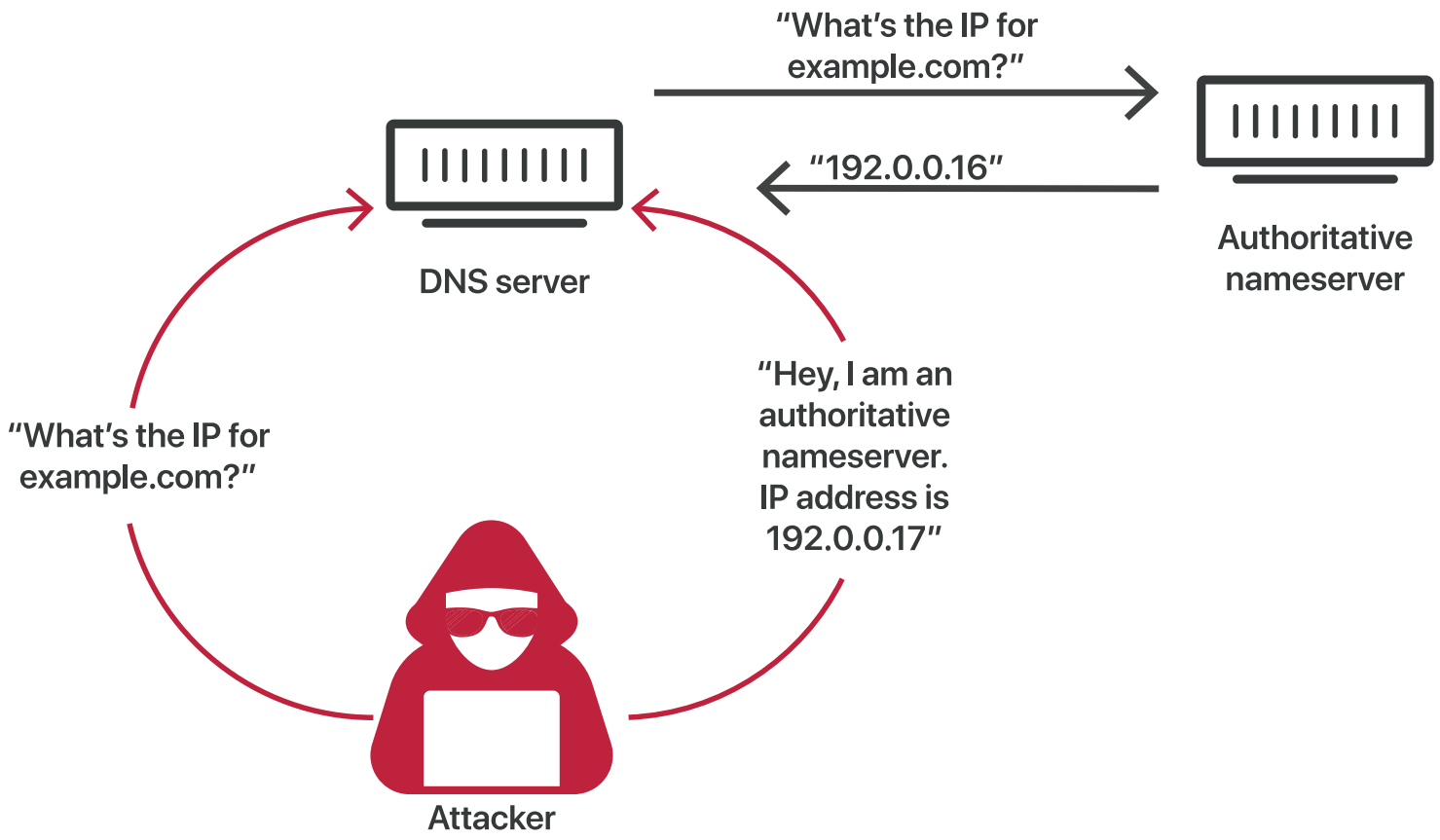
Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

DNS (Cached, Benign)



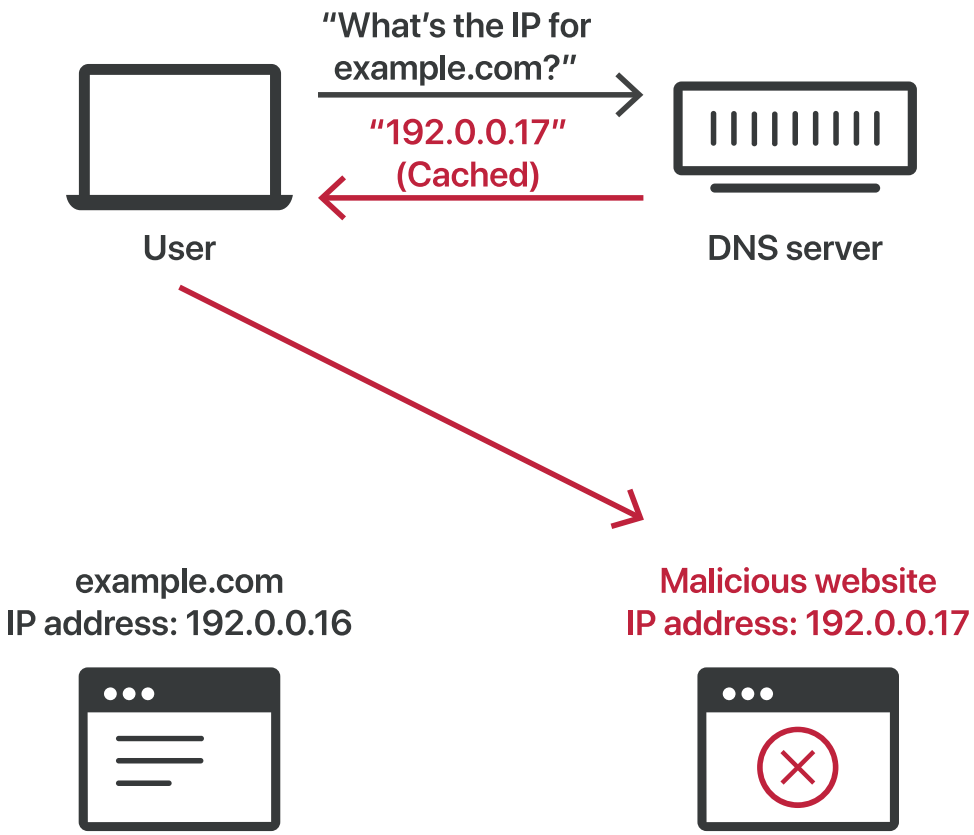
Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

DNS Cache Poisoning Attack



Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

DNS Cache Poisoning Result



Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

DNS Cache Poisoning Result

Despite these major points of vulnerability in the DNS caching process, DNS poisoning attacks are not easy. Because the DNS resolver does actually query the authoritative nameserver, attackers have only a few milliseconds to send the fake reply before the real reply from the authoritative nameserver arrives.

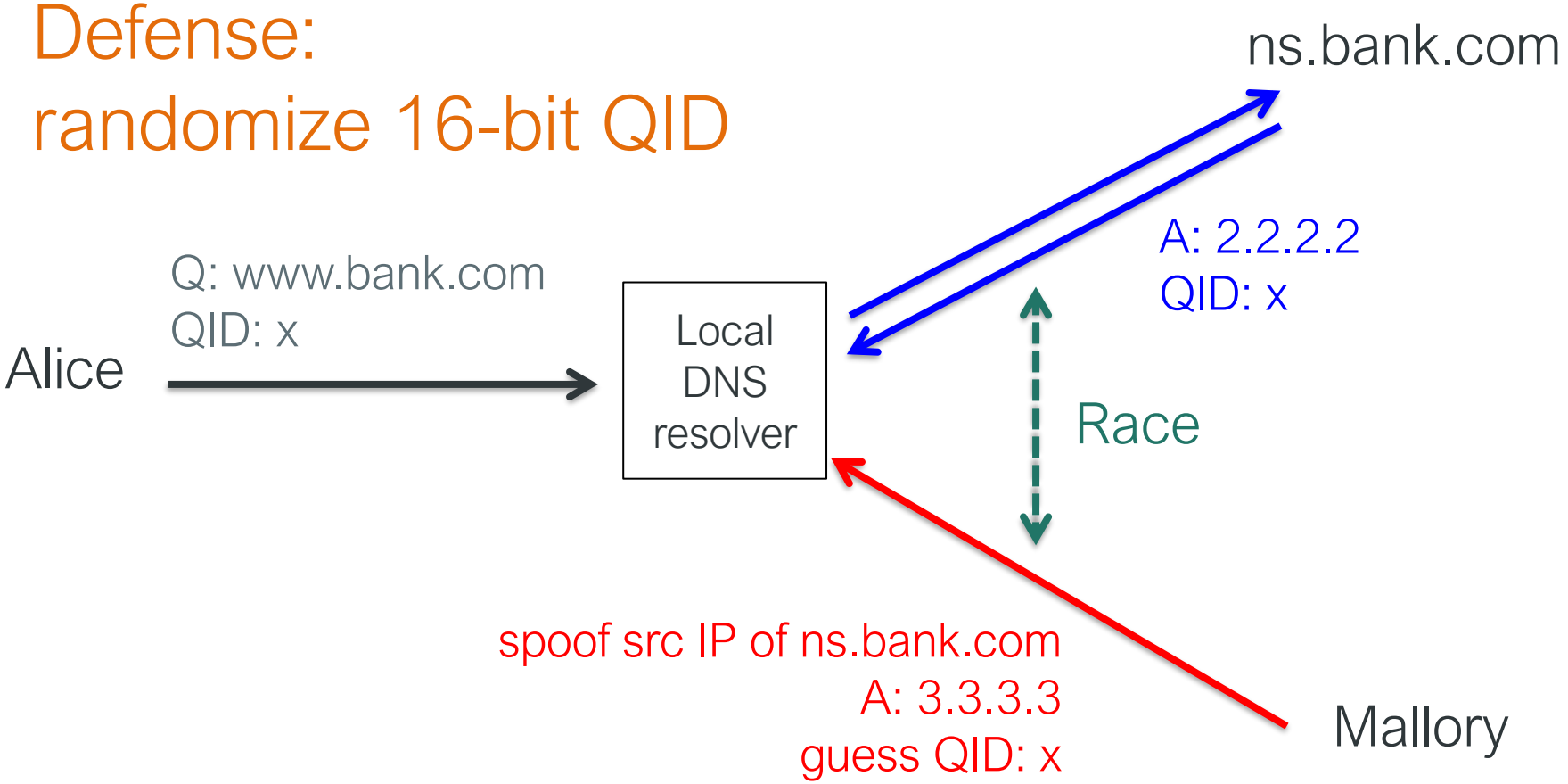
Attackers also have to either know or guess a number of factors to carry out DNS spoofing attacks:

- Which DNS queries are not cached by the targeted DNS resolver, so that the resolver will query the authoritative nameserver
- What port* the DNS resolver is using – they used to use the same port for every query, but now they use a different, random port each time
- The request ID number
- Which authoritative nameserver the query will go to

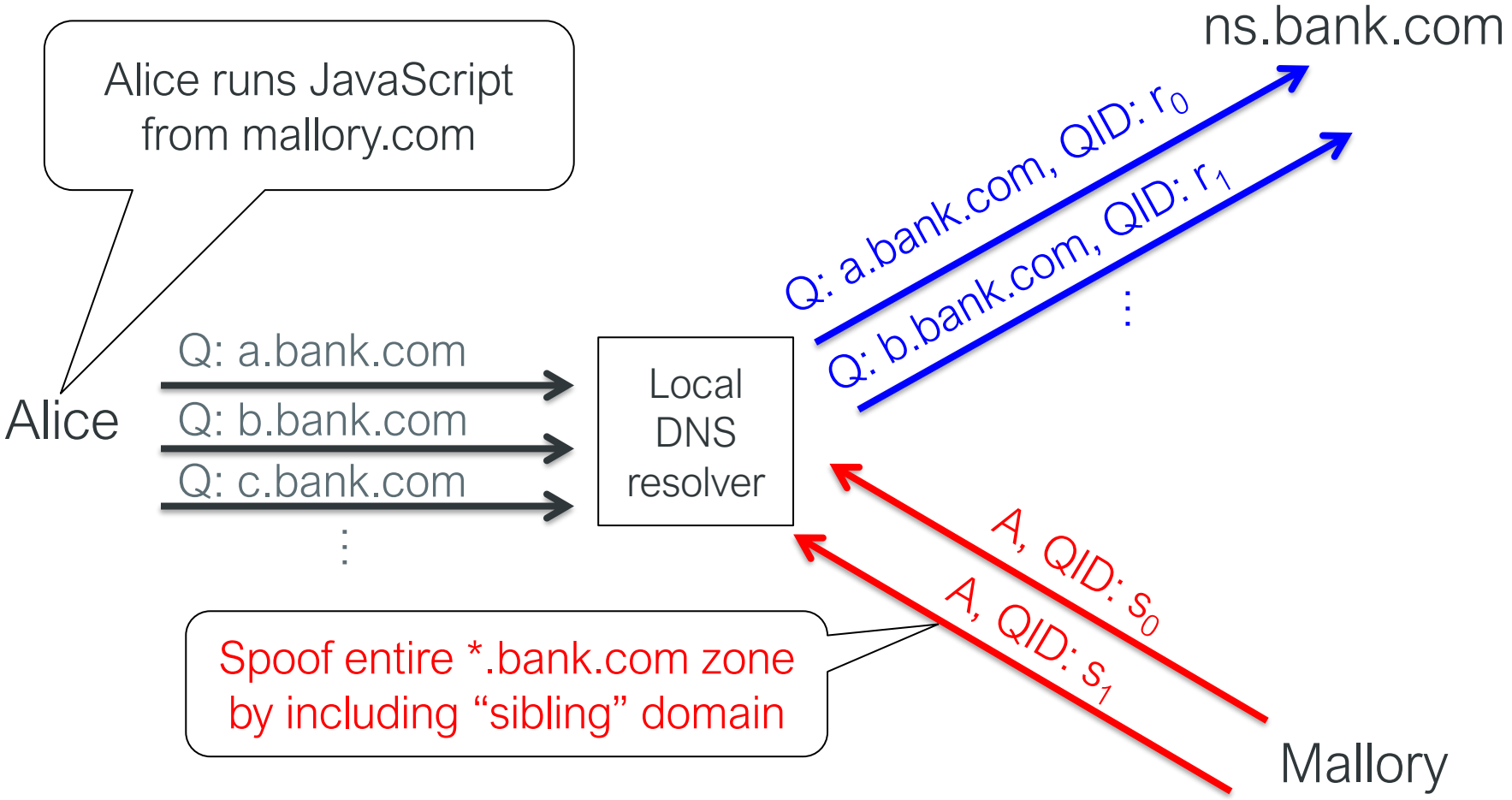
Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/

DNS Cache Poisoning

Defense:
randomize 16-bit QID



Kaminsky attack (2008)



Mallory wins if any $r_i = s_j$

See http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html for details

DNSSEC

DNS responses signed

Higher levels vouch for lower levels
— e.g., root vouches for .edu, .edu vouches for .uchicago, …

Root public key published

Problem?
Costly and slow adoption

The Coffeeshop Attack Scenario

- DNS servers bootstrapped by wireless AP
  - (default setting for WiFi)
- Attacker hosts AP w/ ID (O'Hare Free WiFi)
  - You connect w/ your laptop
  - Your DNS requests go through attacker DNS
  - www.bofa.com → evil bofa.com
  - Password sniffing, malware installs, …

- TLS certificates to the rescue!

The Subtleties That Make Security So Challenging

Security Subtleties: DNS to Trick CAs

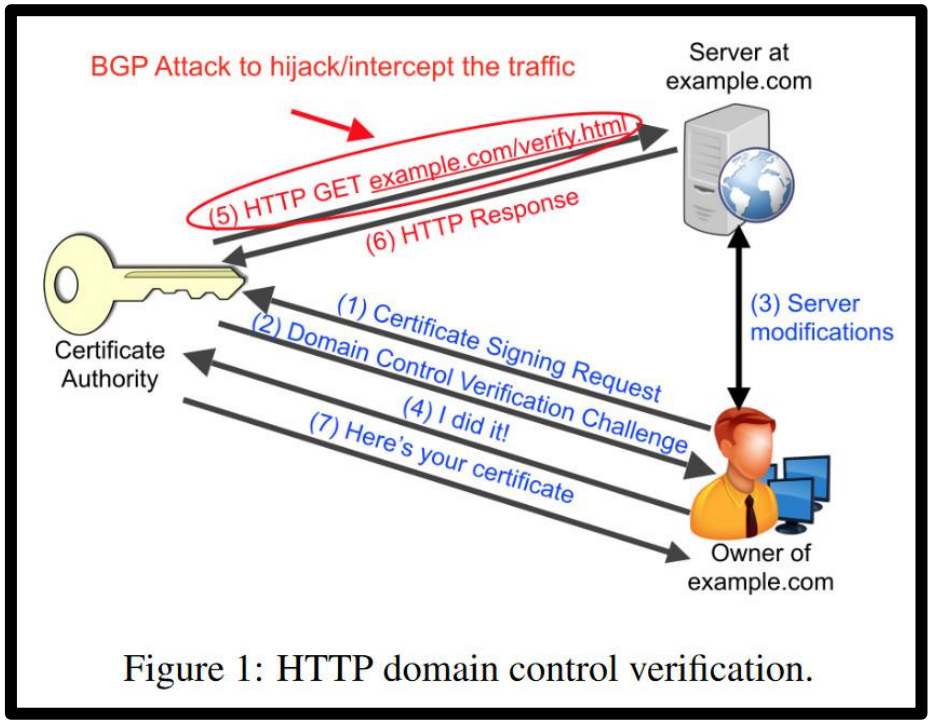


Figure 1: HTTP domain control verification.

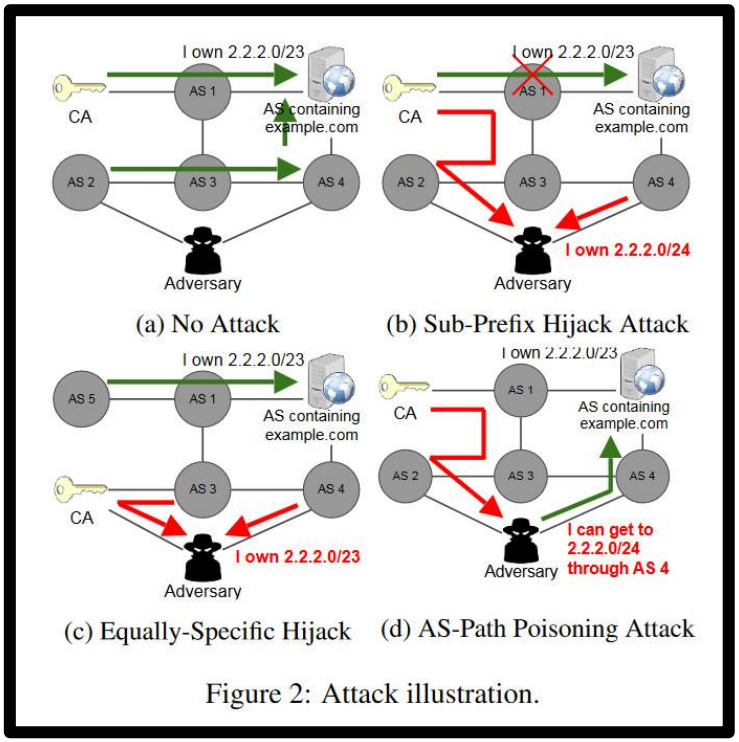


Figure 2: Attack illustration.

From Birge-Lee et al. "Bamboozling Certificate Authorities with BGP," in *Proc. USENIX Security*, 2018. See also Birge-Lee et al. "Experiences Deploying {Multi-Vantage-Point} Domain Validation at Let's Encrypt," in *Proc. USENIX Security*, 2021.

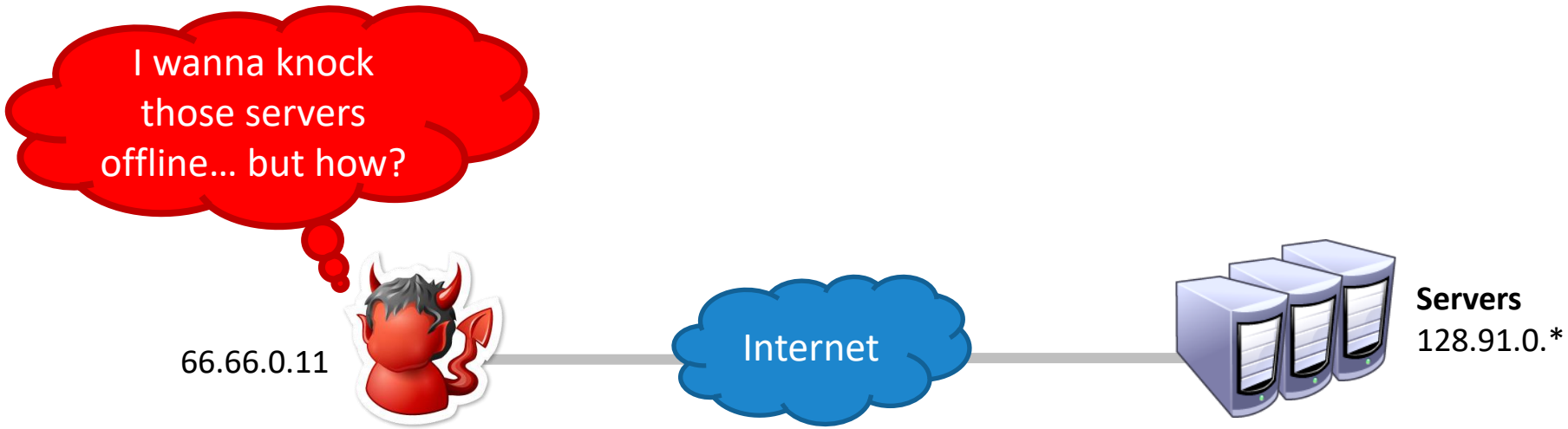
Denial of Service (Attacks on Availability)

Denial of Service (DoS)

- Prevent users from being able to access a specific computer, service, or piece of data
- In essence, an attack on availability
- Possible vectors:
  - Exploit bugs that lead to crashes
  - Exhaust the resources of a target
- Often very easy to perform…
- … and fiendishly difficult to mitigate

DoS Attack Goals & Threat Model

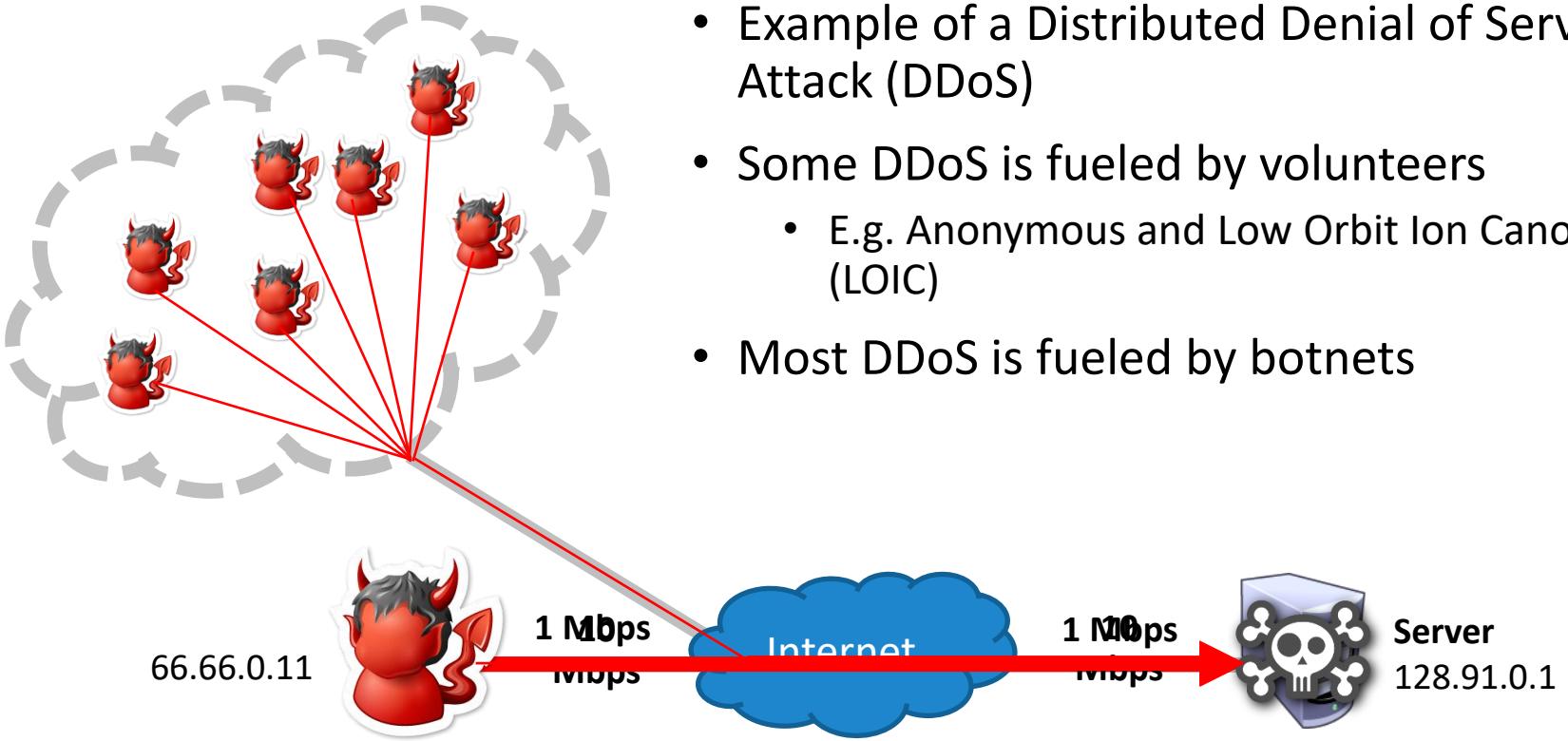
- Active attacker who may send arbitrary packets
- Goal is to reduce the availability of the victim



DoS Attack Parameters

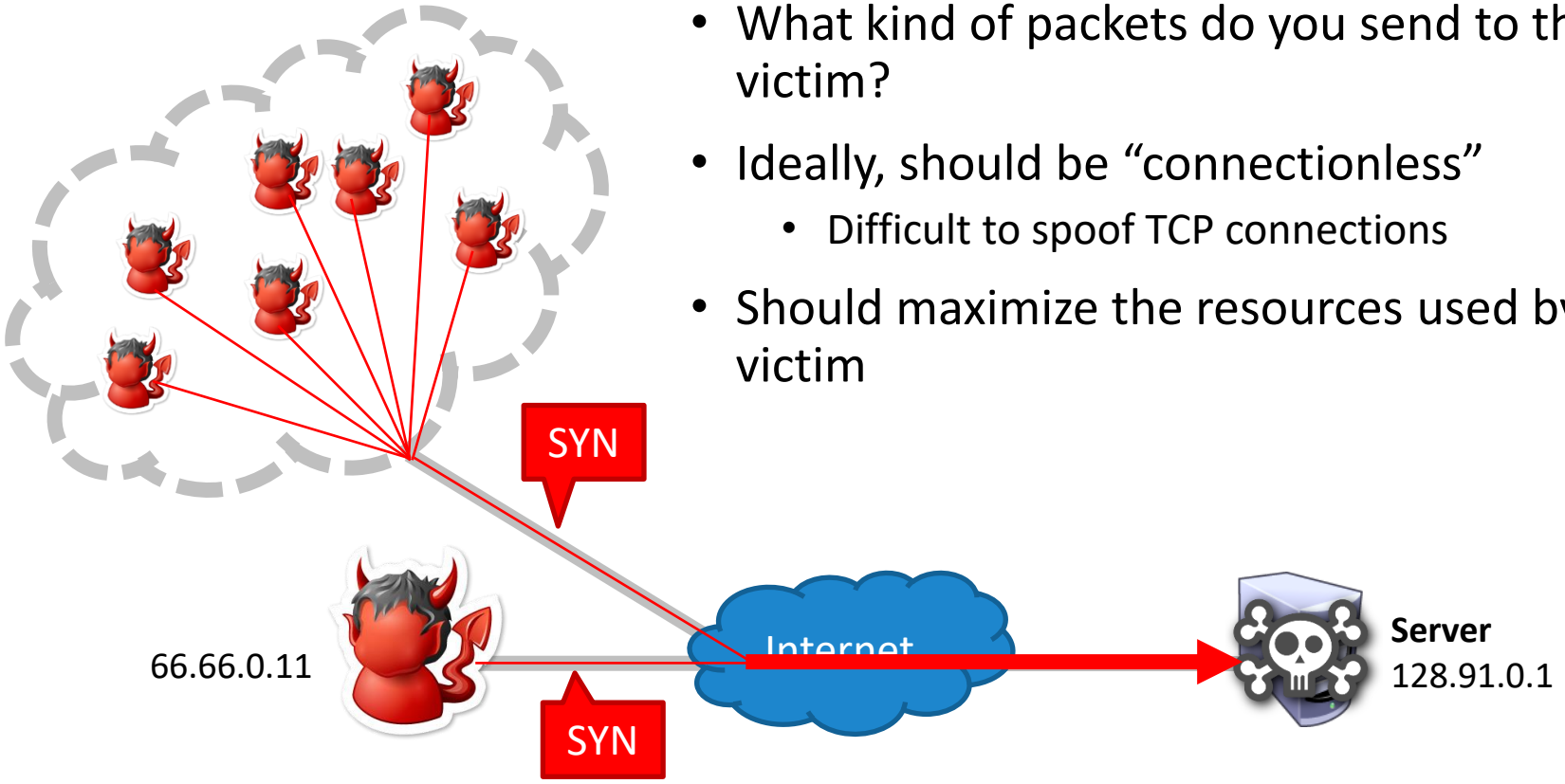
- How much bandwidth is available to the attacker?
  - Can be increased by controlling more resources…
  - Or tricking others into participating in the attack
- What kind of packets do you send to victim?
  - Minimize effort and risk of detection for attacker…
  - While also maximizing damage to the victim

Exploiting Asymmetry: DDoS



- Example of a Distributed Denial of Service Attack (DDoS)
- Some DDoS is fueled by volunteers
  - E.g. Anonymous and Low Orbit Ion Canon (LOIC)
- Most DDoS is fueled by botnets

SYN Flood

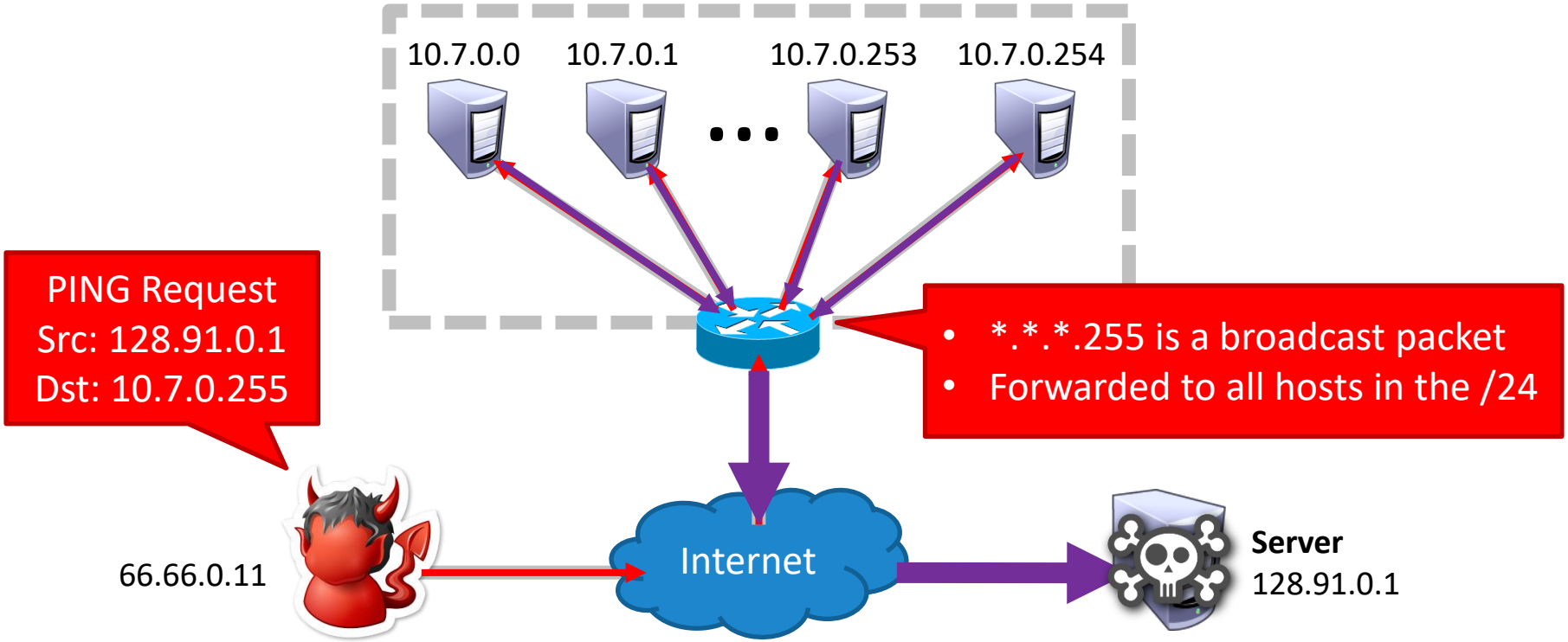


- What kind of packets do you send to the victim?
- Ideally, should be “connectionless”
  - Difficult to spoof TCP connections
- Should maximize the resources used by the victim

TCP SYN Flood

- TCP stack keeps track of connection state in data structures called Transmission Control Blocks (TCBs)
  - New TCB allocated by the kernel whenever a listen socket receives a SYN
  - TCB must persist for at least one RTO
- Attack: flood the victim with SYN packets
  - Exhaust available memory for TCBs, prevent legitimate clients from connecting
  - Crash the server OS by overflowing kernel memory
- Advantages for the attacker
  - No connection – each SYN can be spoofed, no need to hear responses
  - Asymmetry – attacker does not need to allocate TCBs

The Smurf Attack



Why Does Smurfing Work?

1. Internet Control Message Protocol (ICMP) does not include authentication
   - No connections
   - Receivers accept messages without verifying the source
   - Enables attackers to spoof the source of messages
2. Attacker benefits from an amplification factor

$$amp\ factor = \frac{total\ response\ size}{request\ size}$$

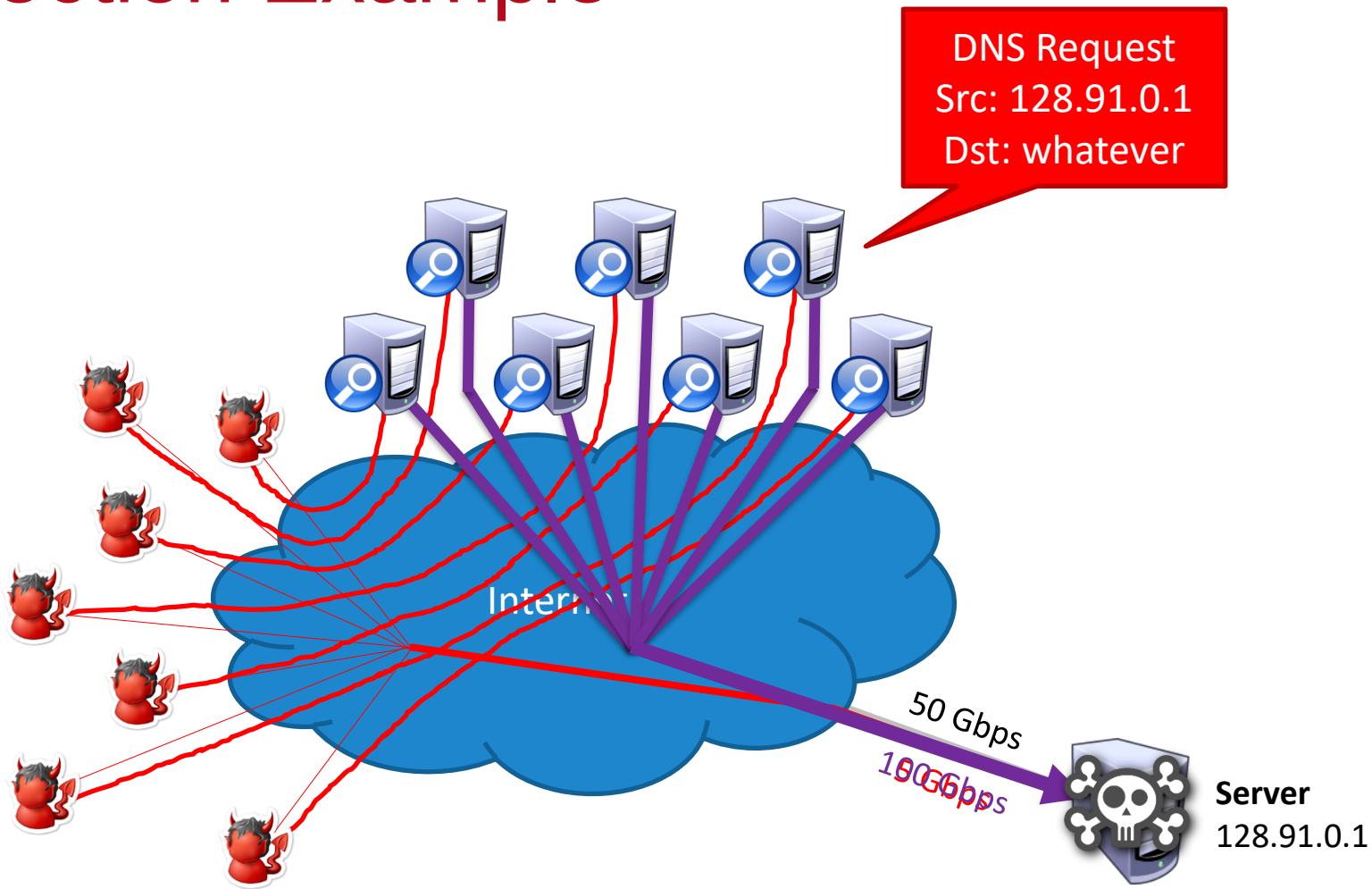
Reflection/Amplification Attacks

- Smurfing is an example of a reflection or amplification DDoS attack
- Fraggle attack similarly uses broadcasts for amplification
  - Send spoofed UDP packets to IP broadcast addresses on port 7 (*echo*) and 13 (*chargen*)
    - *echo* – 1500 bytes/pkt requests, equal size responses
    - *chargen* -- 28 bytes/pkt request, 10K-100K bytes of ASCII in response
  - Amp factor
    - *echo – [number of hosts responding to the broadcast]:1*
    - *chargen – [number of hosts responding to the broadcast]*360:1*

DNS Reflection Attack

- Spoof DNS requests to many **open** DNS resolvers
  - DNS is a UDP-based protocol, no authentication of requests
  - Open resolvers accept requests from any client
    - E.g. 8.8.8.8, 8.8.4.4, 1.1.1.1, 1.0.0.1
  - February 2014 – 25 million open DNS resolvers on the internet
- 64 byte DNS queries generate large responses
  - Old-school "A" record query → maximum 512 byte response
  - EDNS0 extension "ANY" record query → 1000-6000 byte response
    - E.g. $ dig ANY isc.org
  - Amp factor – *180:1*
- Attackers have been known to register their own domains and install very large records just to enable reflection attacks!

Reflection Example



NTP Reflection Attack

- Spoof requests to open Network Time Protocol (NTP) servers
  - NTP is a UDP-based protocol, no authentication of requests
  - May 2014 – 2.2 million open NTP servers on the internet
- 234 byte queries generate large responses
  - *monlist* query: server returns a list of all recent connections
  - Other queries are possible, i.e. *version* and *showpeers*
  - Amp factor – from *10:1* to *560:1*

memcached Reflection Attack

- Spoof requests to open memcached servers
  - Popular <key:value> server used to cache web objects
  - memcached uses a UDP-based protocol, no authentication of requests
  - February 2018 – 50k open memcached servers on the internet
- 1460 byte queries generate large responses
  - A single query can request multiple 1MB <key:value> pairs from the database
  - Amp factor – up to *50000:1*

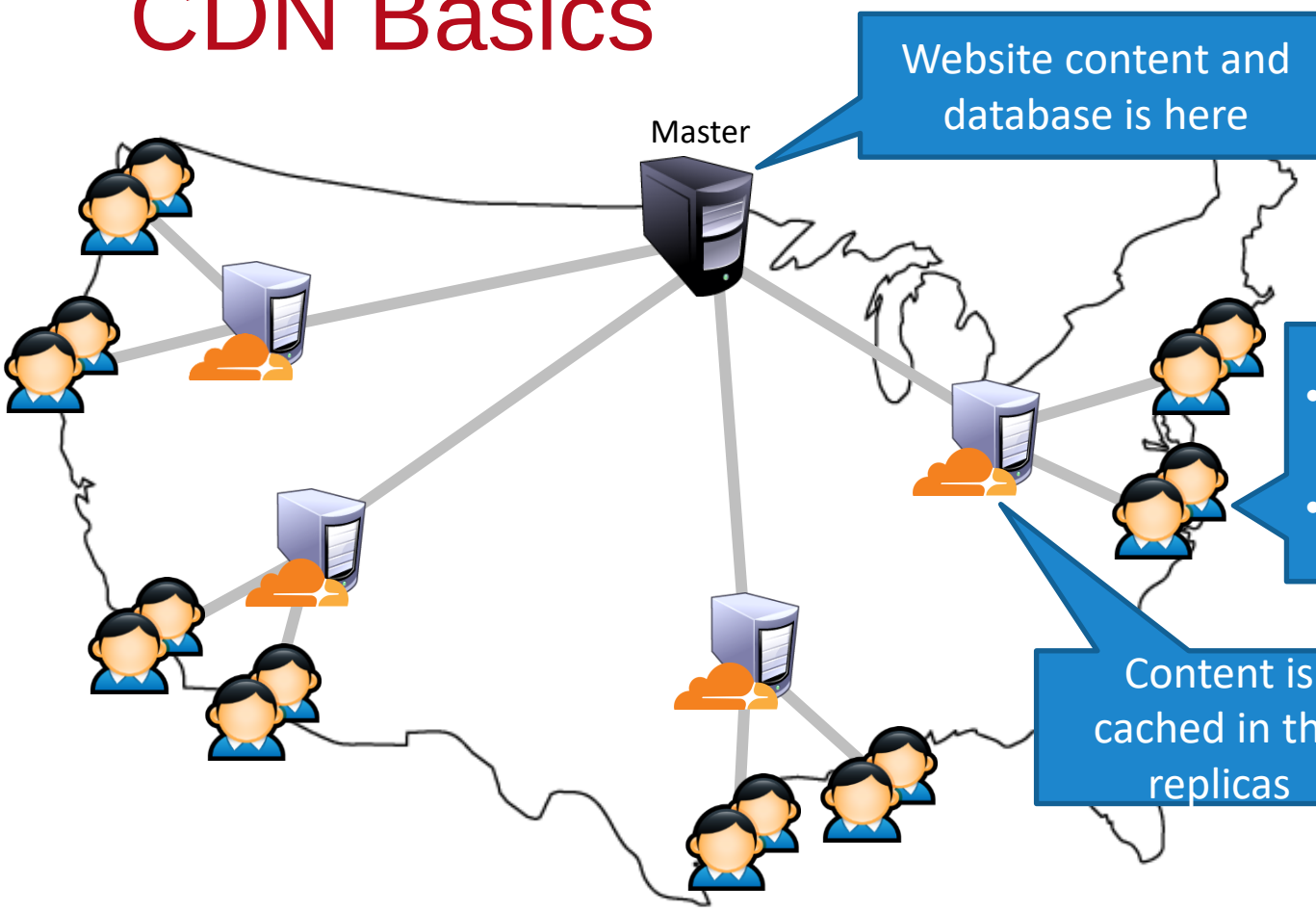
Infamous DDoS Attacks

When	Against Who	Size	How
March 2013	Spamhaus	120 Gbps	Botnet + DNS reflection
February 2014	Cloudflare	400 Gbps	Botnet + NTP reflection
September 2016	Krebs	620 Gbps	Mirai
October 2016	Dyn (major DNS provider)	1.2 Tbps	Mirai
March 2018	Github	1.35 Tbps	Botnet + memcached reflection

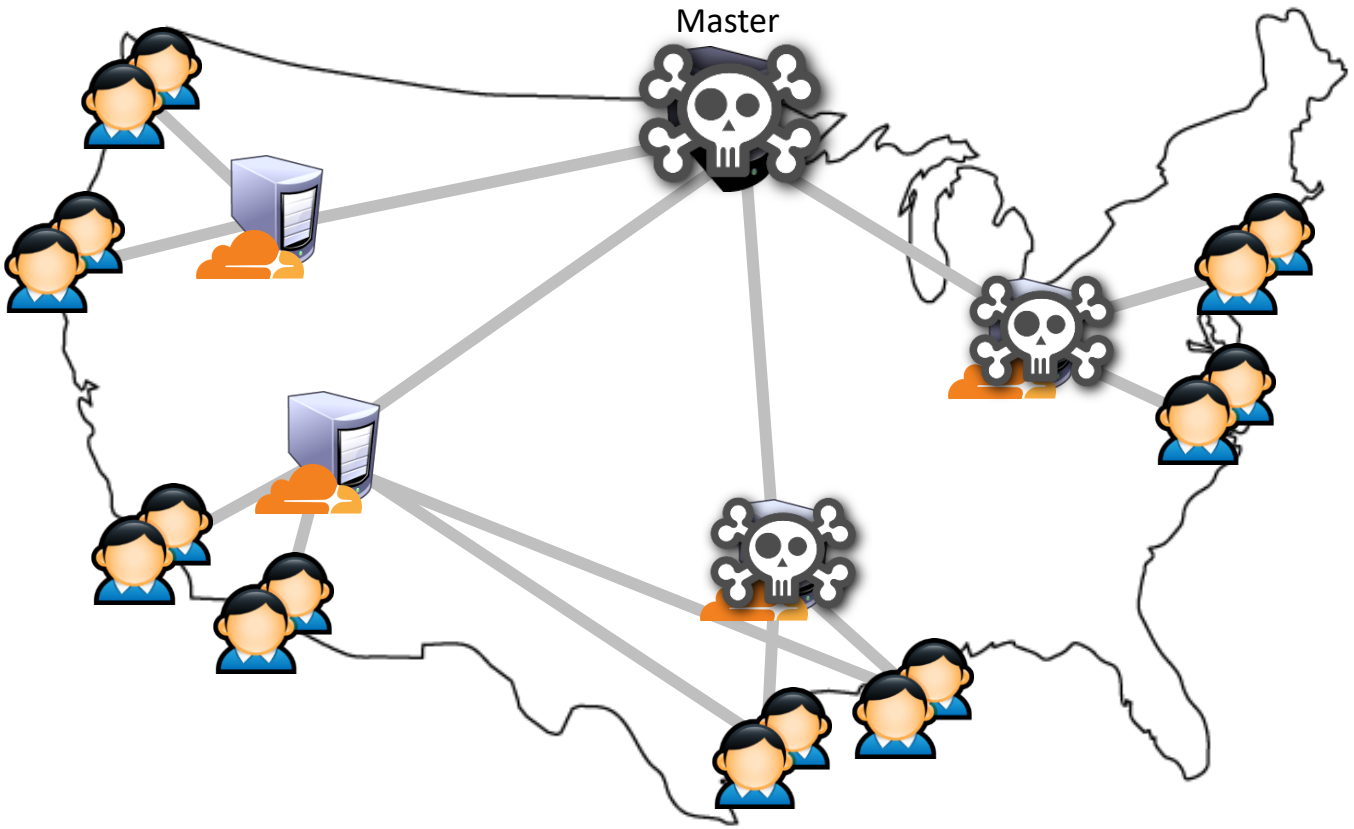
Content Delivery Networks (CDNs)

- CDNs help companies scale-up their websites
  - Cache customer content on many replica servers
  - Users access the website via the replicas
- Examples: Akamai, Cloudflare, Rackspace, Amazon Cloudfront, etc.
- Side-benefit: DDoS protection
  - CDNs have many servers, and a huge amount of bandwidth
  - Difficult to knock all the replicas offline
  - Difficult to saturate all available bandwidth
  - No direct access to the master server
- Cloudflare: 15 Tbps of bandwidth over 149 data centers

CDN Basics


Master
Website content and database is here
Users requests all go through the replicas
Most served from cache
Content is cached in the replicas

DDoS Defense via CDNs



- What if you DDoS the master replica?
  - Cached copies in the CDN still available
  - Easy to do ingress filtering at the master
- What if you DDoS the replicas?
  - Difficult to kill them all
  - Dynamic DNS can redirect users to live replicas

BOTNETS

Botnets

- Infected machines are a fundamentally valuable resource
  - Unique IP addresses for spamming
  - Bandwidth for DDoS
  - CPU cycles for bitcoin mining
  - Credentials
- Early malware monetized these resources directly
  - Infection and monetization were tightly coupled
- Botnets allow criminals to rent access to infected hosts
  - Infrastructure as a service, i.e. the cloud for criminals
  - Command and Control (C&C) infrastructure for controlling bots
  - Enables huge-scale criminal campaigns