

08. Network Attacks & Defenses

Blase Ur and Grant Ho

(some slides borrowed from Ben Zhao, Christo Wilson, & others)

January 31st, 2024

CMSC 23200



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

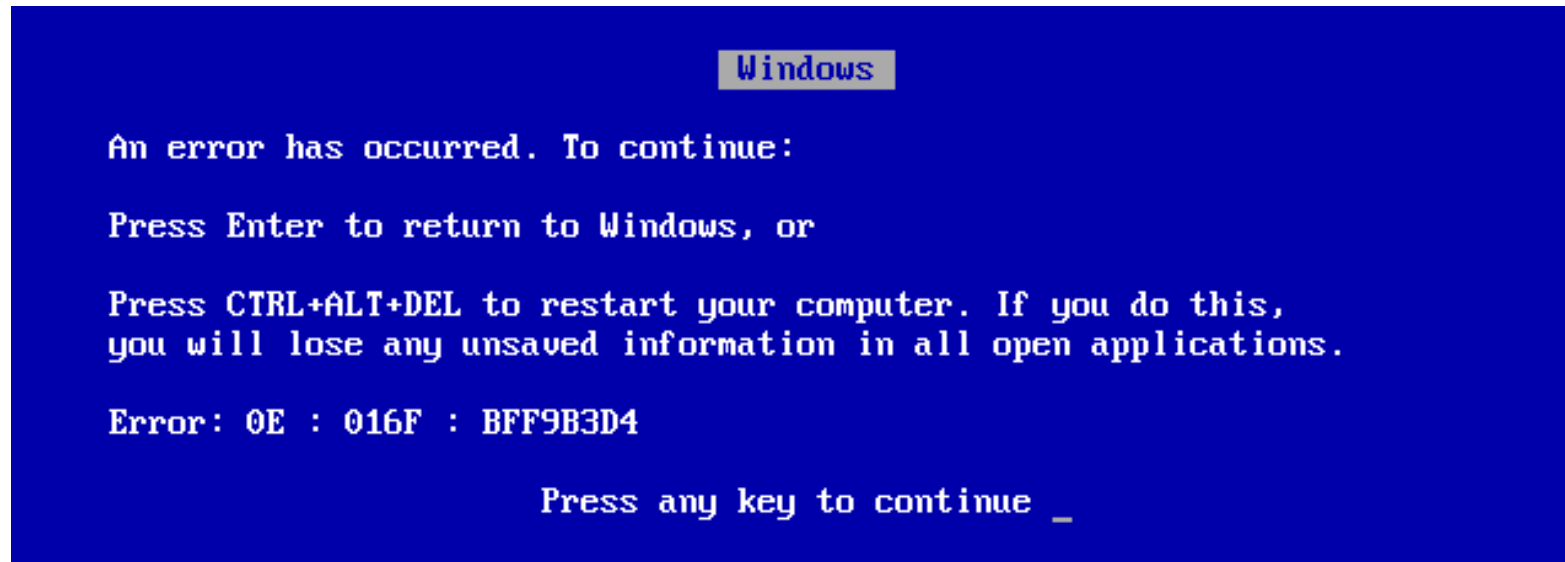
- Essential, low-level network utility
- Sends a “ping” ICMP message to a host on the internet

```
$ 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 (size is stored in a 16-bit field)
- What if you send a ping that is >65535 bytes long?

Attack: Ping of Death

- A ping longer than the max could cause a buffer overflow
 - Attack first identified in 1997
 - IPv6 versions identified in 2013 (CVE-2013-3181) and 2020 (CVE-2020-16898)



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  vllrouter (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)[~]:->
```

Attack-Adjacent: 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!!

Attack-Adjacent: SYN scan

Only send SYN

Responses:

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

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

The image shows a Wireshark capture of network traffic. The main pane displays a list of packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The selected packet (No. 5) is a TCP SYN packet from 24.4.97.251 to 80.231.19.118 on port 80. The packet details pane shows the structure of the packet: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	24.4.97.251	68.87.76.178	DNS	76	Standard query 0x6bc0 A www.symantec.com
2	0.011505000	68.87.76.178	24.4.97.251	DNS	262	Standard query response 0x6bc0 CNAME www.symantec.d4p.net CNAME s
3	0.275559000	24.4.97.251	68.87.76.178	DNS	93	Standard query 0xcdc6 A liveupdate.symantecliveupdate.com
4	0.291867000	68.87.76.178	24.4.97.251	DNS	286	Standard query response 0xcdc6 CNAME liveupdate.symantec.d4p.net
5	0.336805000	24.4.97.251	80.231.19.118	TCP	62	trim > http [SYN] Seq=0 win=65535 Len=0 MSS=1460 SACK_PERM=1
6	0.508336000	80.231.19.118	24.4.97.251	TCP	62	http > trim [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PE
7	0.508459000	24.4.97.251	80.231.19.118	TCP	54	trim > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
8	0.508953000	24.4.97.251	80.231.19.118	HTTP	307	GET /minitri.flg HTTP/1.1
9	0.686341000	80.231.19.118	24.4.97.251	TCP	60	http > trim [ACK] Seq=1 Ack=254 Win=6432 Len=0
10	0.686838000	80.231.19.118	24.4.97.251	HTTP	288	HTTP/1.1 304 Not Modified
11	0.843702000	24.4.97.251	80.231.19.118	TCP	54	trim > http [ACK] Seq=254 Ack=235 Win=65301 Len=0
12	1.635308000	24.4.97.251	80.231.19.118	HTTP	298	GET /automatic\$20liveupdate_3.0.0.171_english_livetri.zip HTTP/1.1
13	1.808631000	80.231.19.118	24.4.97.251	HTTP	536	HTTP/1.1 404 Not Found (text/html)

Frame 5: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0

Ethernet II, Src: AsustekC_e0:d3:f7 (00:17:31:e0:d3:f7), Dst: Cadant_22:a5:82 (00:01:5c:22:a5:82)

Internet Protocol Version 4, Src: 24.4.97.251 (24.4.97.251), Dst: 80.231.19.118 (80.231.19.118)

Transmission Control Protocol, Src Port: trim (1137), Dst Port: http (80), Seq: 0, Len: 0

Source port: trim (1137)

Destination port: http (80)

Stream index: 01

```
0000  00 01 5c 22 a5 82 00 17 31 e0 d3 f7 08 00 45 00  ..\".... 1....E.
0010  00 30 0a 33 40 00 80 06 12 39 18 04 61 fb 50 e7    .0.3@... .9..a.P.
0020  13 76 04 71 00 50 fc be 21 3b 00 00 00 00 70 02  .v.q.P.. !;....p.
0030  ff ff 82 08 00 00 02 04 05 b4 01 01 04 02      ..... .....
```

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



Side Channels

- 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

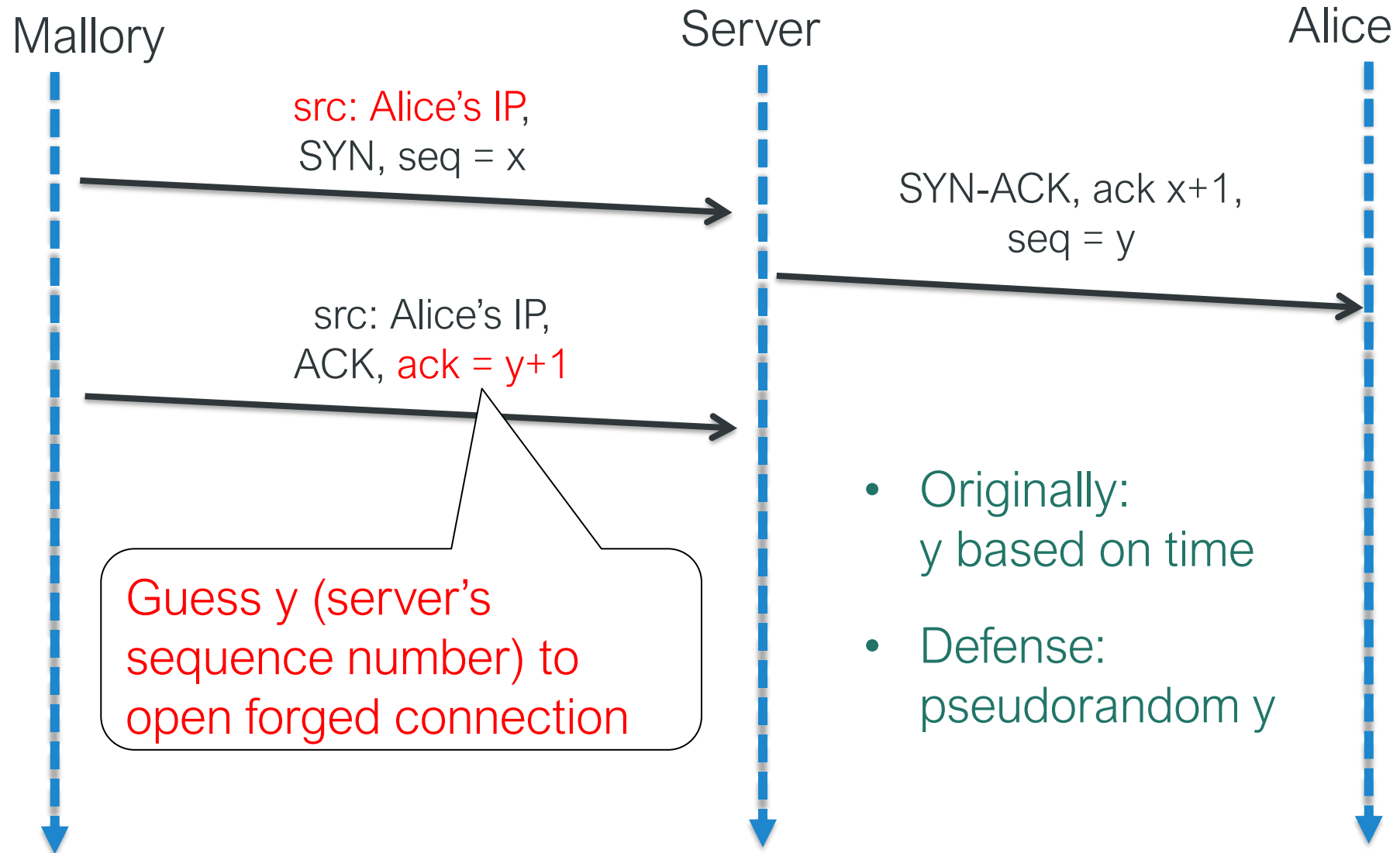
Timing attacks are usually used to attack weak computing devices such as smartcards. We show that timing attacks apply to general software systems. Specifically, we devise a timing attack against OpenSSL. Our experiments show that we can extract private keys from an OpenSSL-based web server running on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

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.

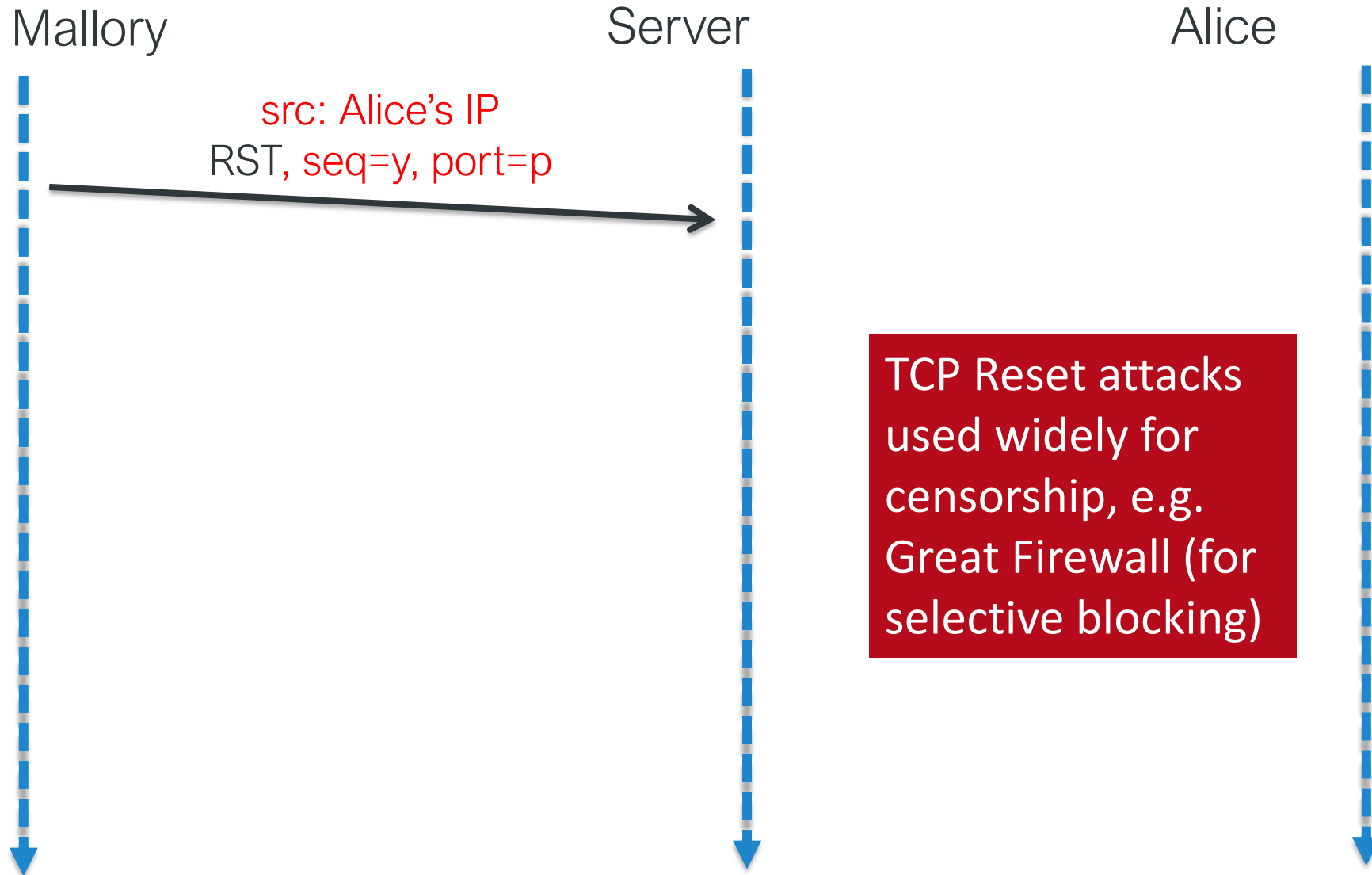
Interprocess. We successfully mounted the attack between two processes running on the same machine. A hosting center that hosts two domains on the same machine might give management access to the admins of each domain. Since both domains are hosted on the same machine, one admin could use

Protocol Attacks

Blind Spoofing



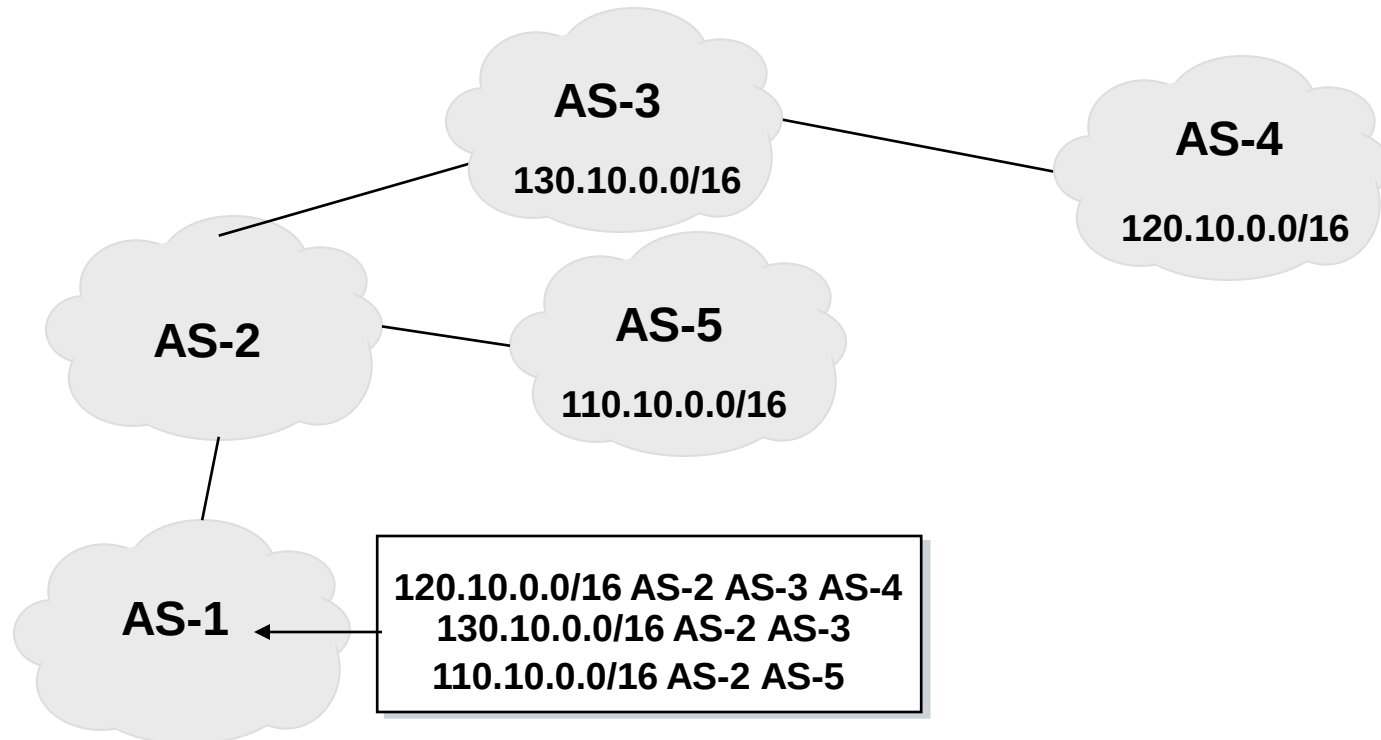
RST Hijacking



Inter-Domain Routing (BGP) Attacks and Large-Scale Observation

Recall BGP

- 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](#)



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.

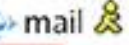
To understand what happened, we need to cover the basics of how Internet routing works. When I type, for example, HypotheticalDomain.com into my browser and hit enter, my computer creates a web request and sends it to HypotheticalDomain.com servers. These servers likely reside in a different state or 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>



Hotmail®

YAHOO!



(TS//SI//NF) **FAA702 Operations**
Two Types of Collection



Upstream

- Collection of communications on fiber cables and infrastructure as data flows past.

(FAIRVIEW, STORMBREW, BLARNEY, OAKSTAR)

**You
Should
Use Both**

PRISM

- Collection directly from the servers of these U.S. Service Providers: Microsoft, Yahoo, Google, Facebook, PalTalk, AOL, Skype, YouTube, Apple.

From Snowden archives, dated April 2013

TOP SECRET//SI//ORCON//NOFORN



Gmail

facebook



Hotmail

YAHOO!



palTalk.com

YouTube

AOL mail

(TS//SI//NF)

PRISM Collection Details



Current Providers

What Will You Receive in Collection
(Surveillance and Stored Comms)?

It varies by provider. In general:

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



- 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

TOP SECRET//SI//ORCON//NOFORN

TOP SECRET//SI//ORCON//NOFORN



Gmail

facebook



Hotmail

YAHOO!

Google



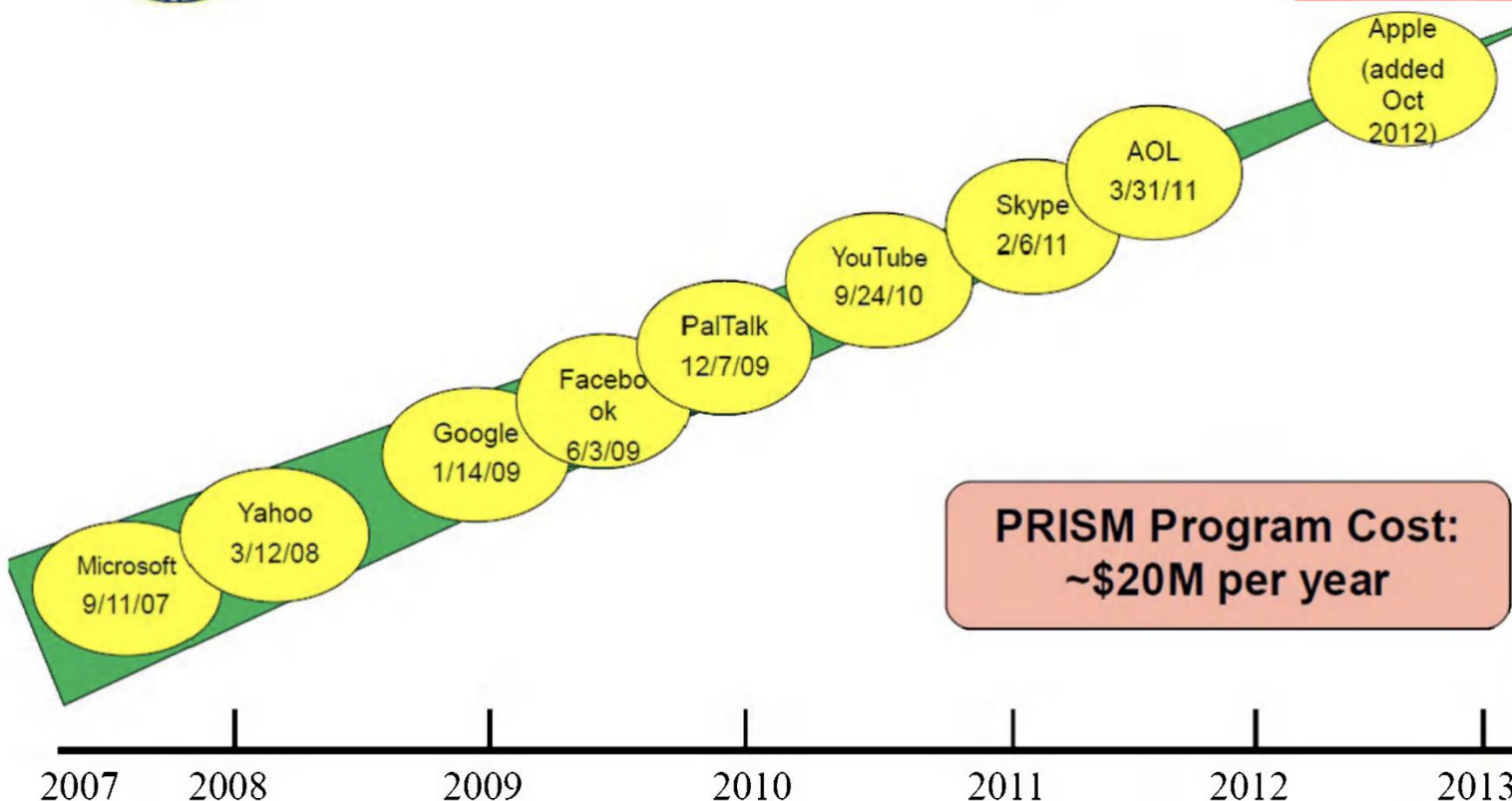
skype

paltalk.com

YouTube

AOL mail

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



PRISM Program Cost:
~\$20M per year

TOP SECRET//SI//ORCON//NOFORN

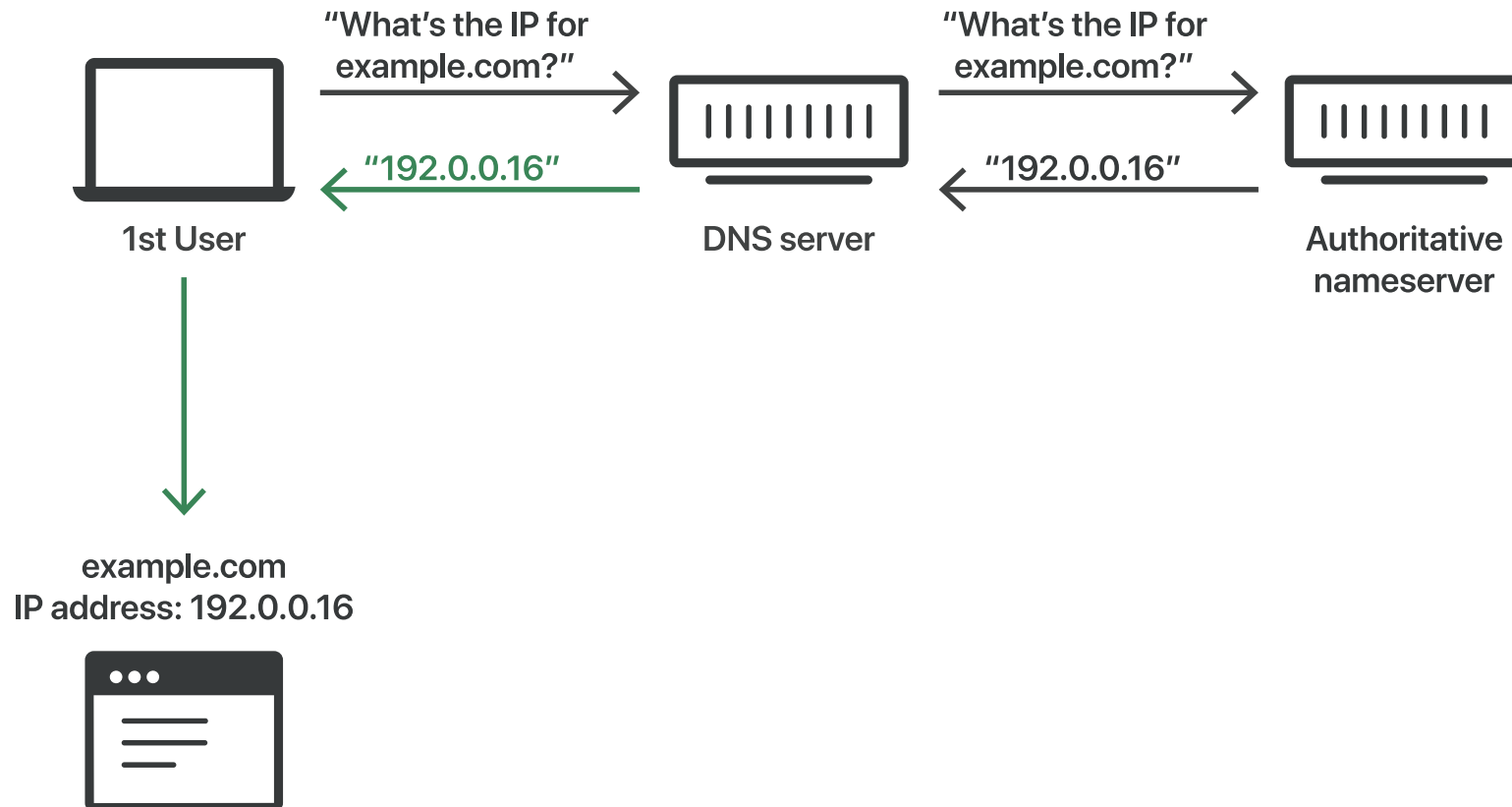
S-BGP / BGPsec

- 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, ...

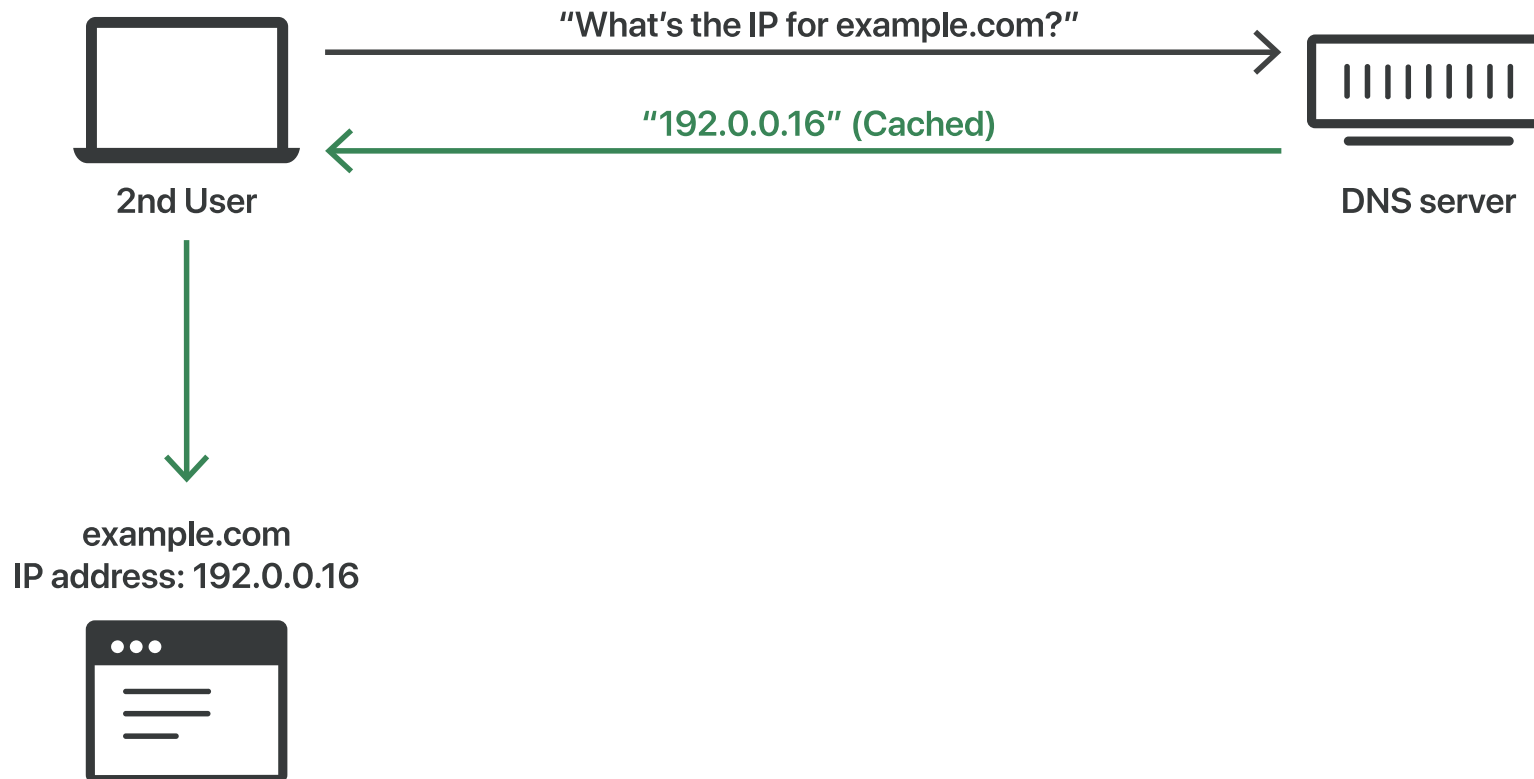
Problem?
Costly and slow adoption

DNS Attacks

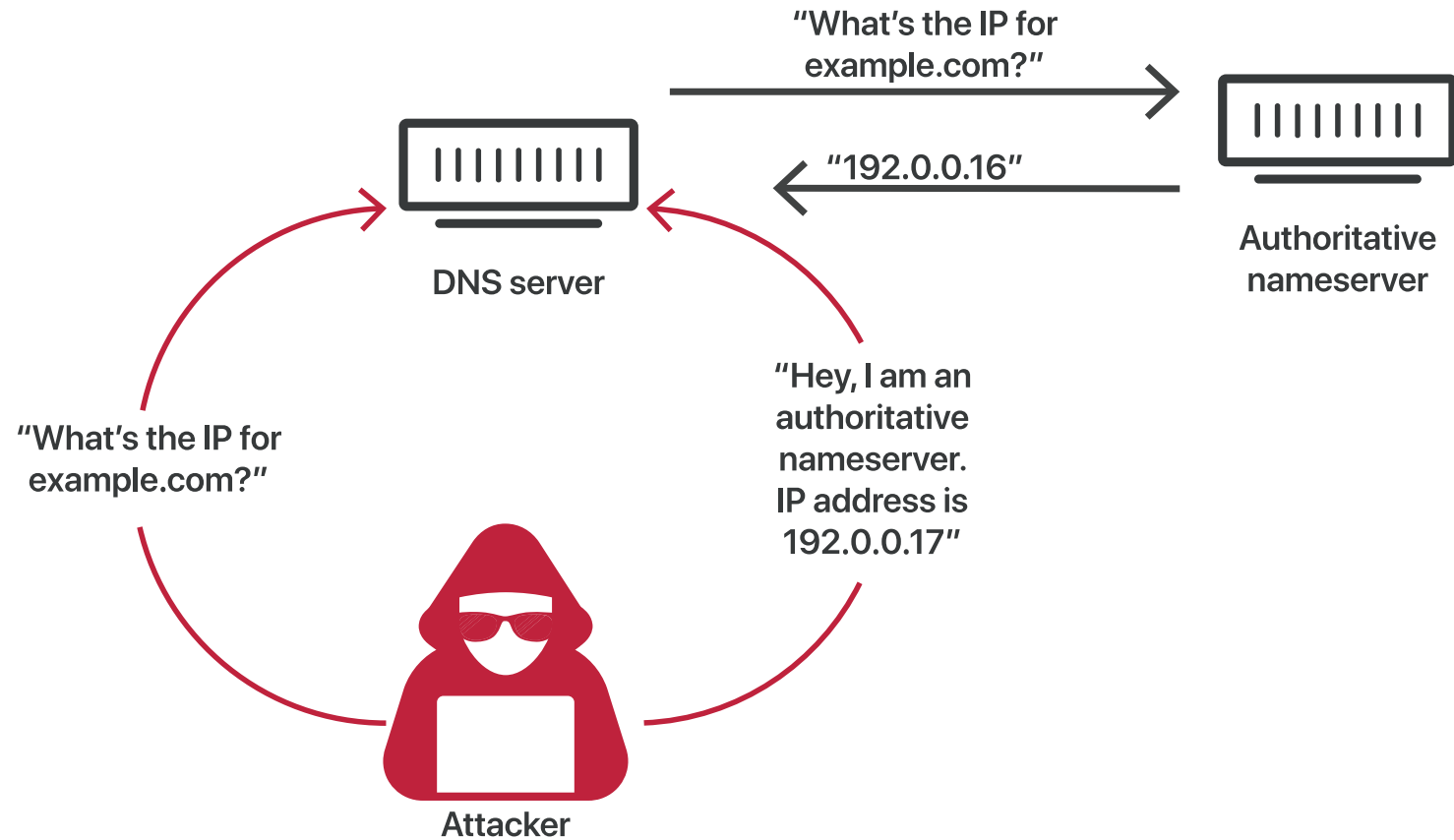
DNS (Uncached)



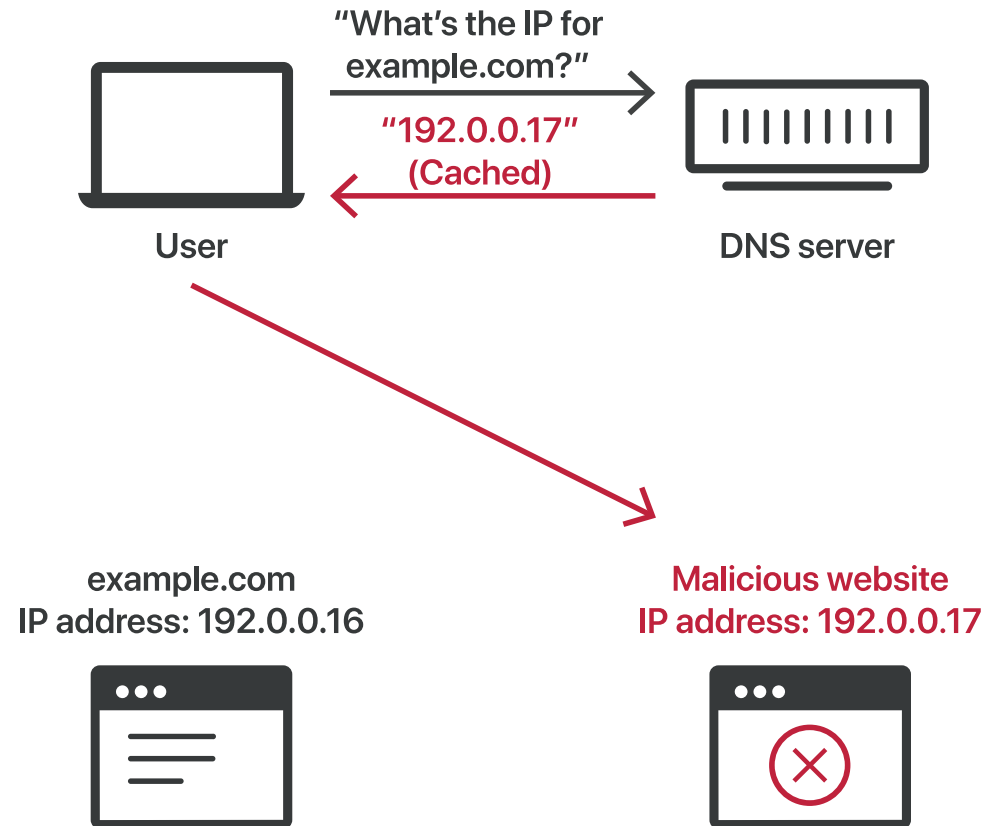
DNS (Cached, Benign)



DNS Cache Poisoning Attack



DNS Cache Poisoning Result



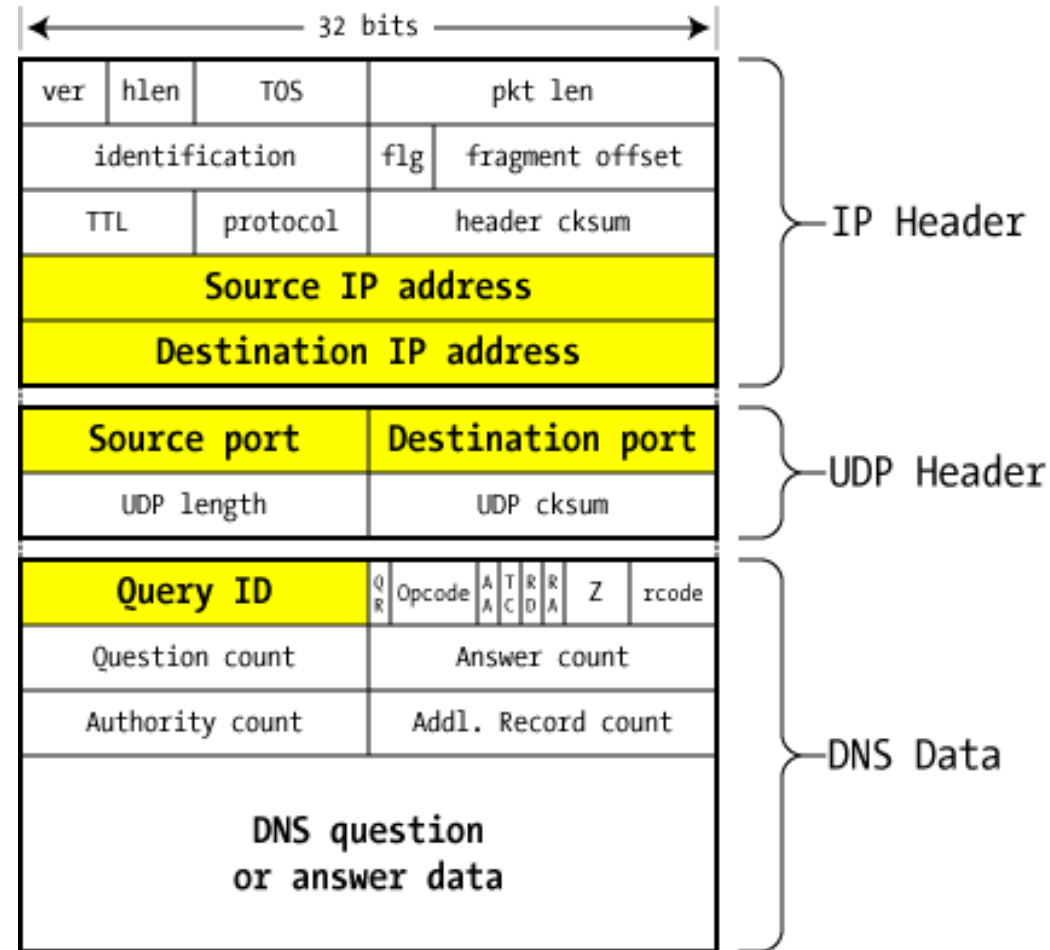
DNS Cache Poisoning Result

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

DNS Packet Structure

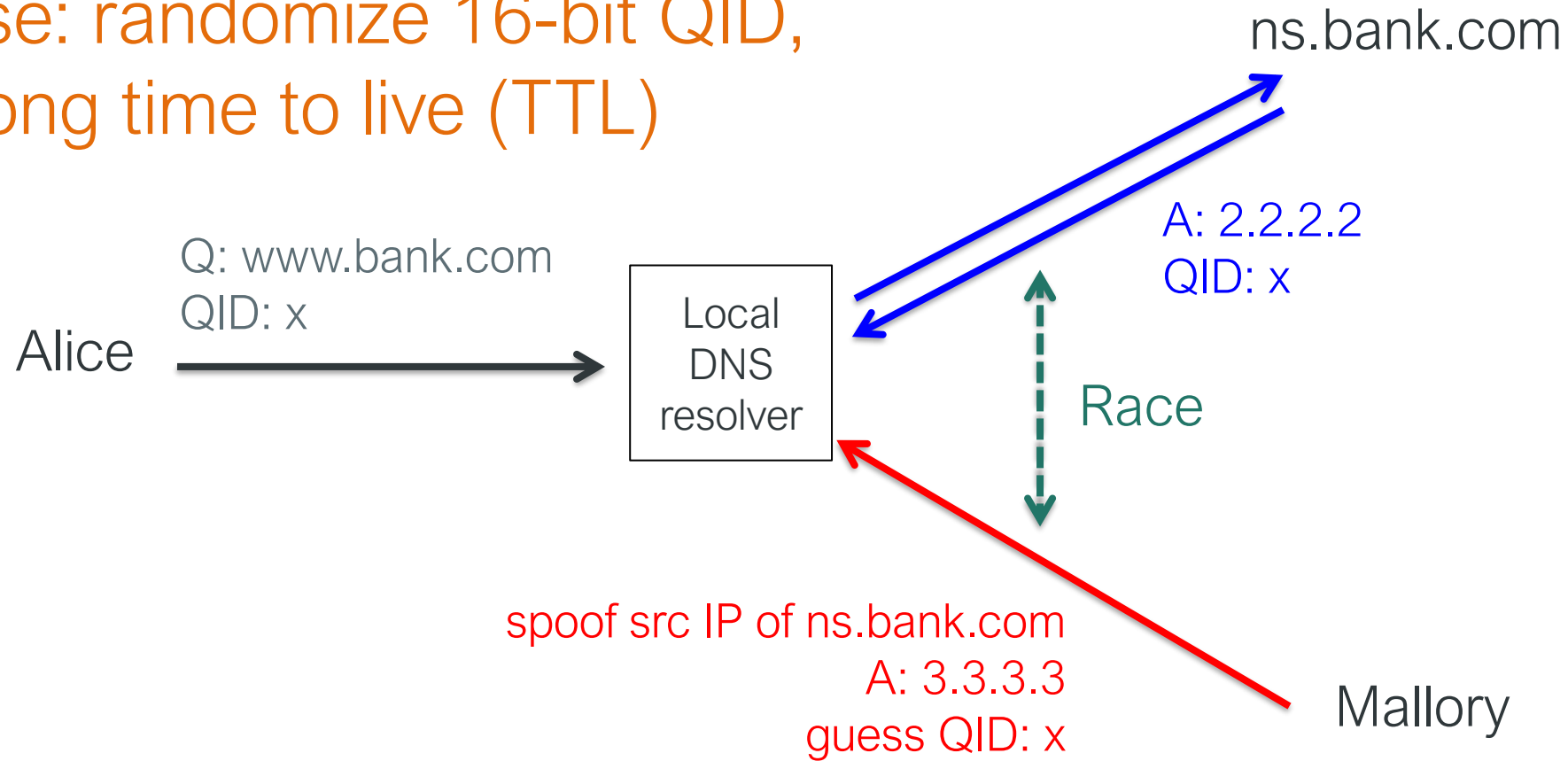


DNS packet on the wire

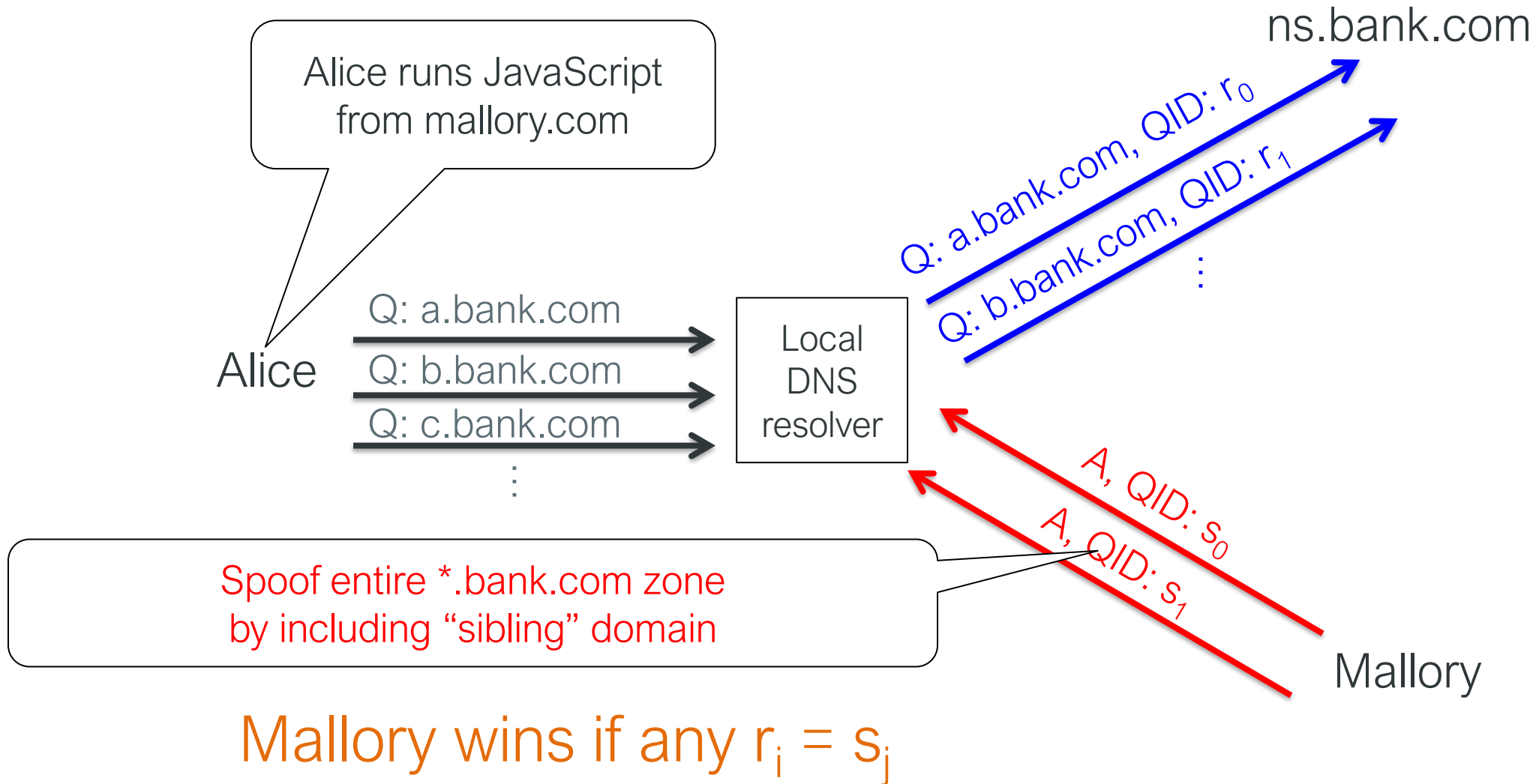
Image from <http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html>

DNS Cache Poisoning

Defense: randomize 16-bit QID,
set a long time to live (TTL)



Kaminsky Attack (2008)



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

Kaminsky Attack (2008)

Bait and Switch

- Is it possible for a bad guy, who has won the race for 83.foo.com, to end up stealing www.foo.com as well?
 - He has three possible replies that can be associated with correctly guessed TXID
 - 1) “Here’s your answer for 83.foo.com – it’s 6.6.6.6”
 - 2) “I don’t know the answer for 83.foo.com.”
 - 3) “83.foo.com? I don’t know, go ask the www.foo.com server, it’s at 6.6.6.6”
 - This has to work – it’s just another delegation
 - 13 root servers: “83.foo.com? I don’t know, go ask the com server, it’s at 1.2.3.4”
 - Com server: “83.foo.com? I don’t know, go ask the foo.com server, it’s at 2.3.4.5”
 - Foo.com server: “83.foo.com? I don’t know, go ask the www.foo.com server, it’s at 6.6.6.6”

Kaminsky Attack (2008)

Enter The DNSRake

- Named after a common method for lockpicking
- 1) Send a query to a nameserver, for \$RANDOM.foo.com
 - The bad guy has the starter pistol
- 2) Send 200 fake replies to that nameserver, with TXID 0-200
 - The bad guy can reply multiple times
- 3) Send replies containing nameserver redirections to www.foo.com
 - \$RANDOMwww.foo.com IN NS www.foo.com
 - www.foo.com IN A 6.6.6.6
 - If this works, it works
 - If it fails, return to step 1

Kaminsky Attack (2008)

- Key insight: The attacker's fraudulent responses for fake subdomains (e.g., z123123.bank.com) return a response saying that the authoritative nameserver for all of bank.com is a host controlled by the attacker
- Immediate response: Randomize both the query ID and source port

DNSSEC

- DNS responses signed
- Higher levels vouch for lower levels
 - e.g., root vouches for .edu, .edu vouches for .uchicago, ...
- Root public key published

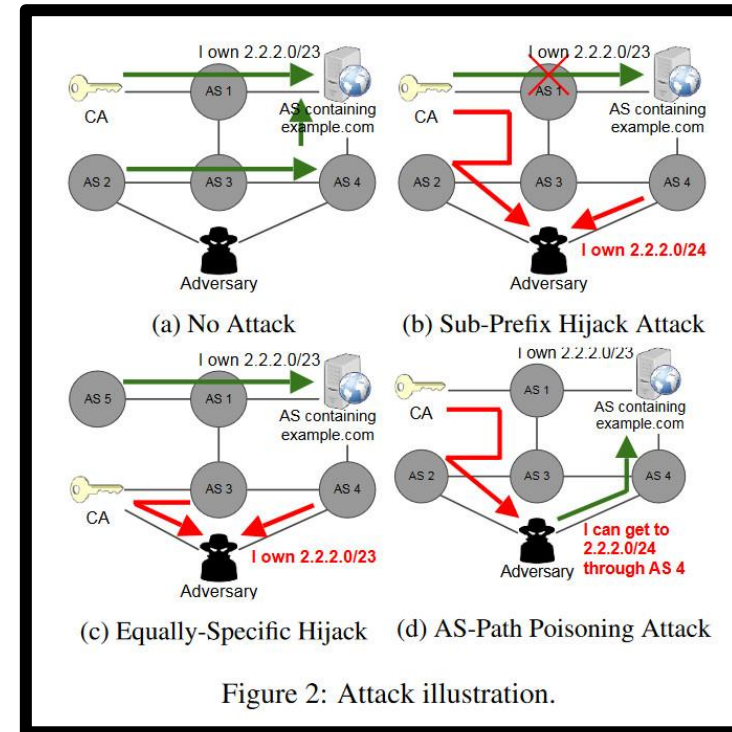
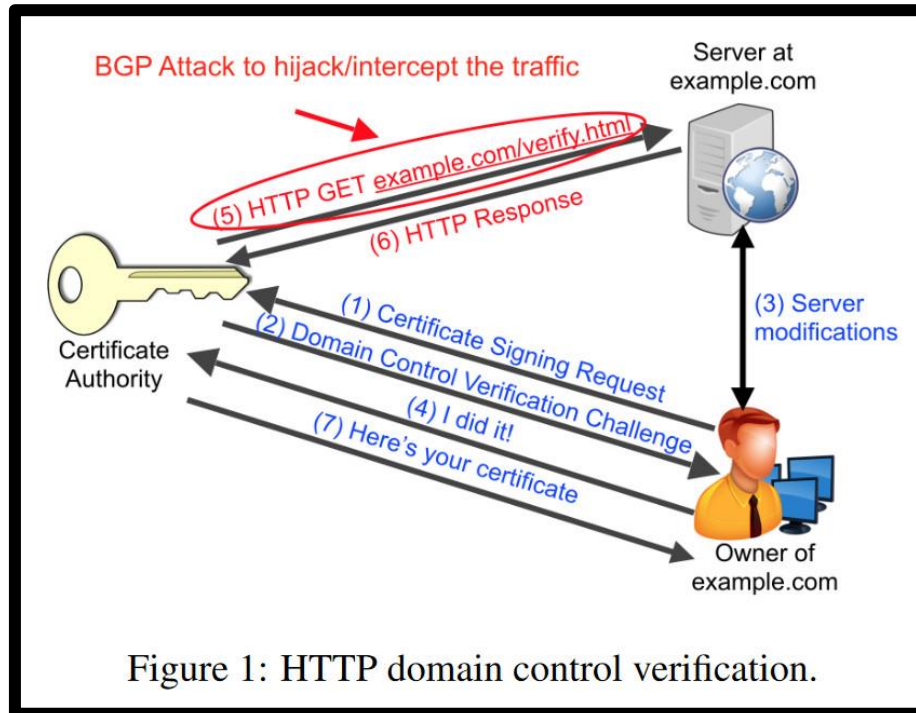
Problem?
Costly and slow adoption

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, ...
- **Certificates to the rescue!**

The Subtleties That Make Security So Challenging

Security Subtleties: DNS to Trick CAs



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.

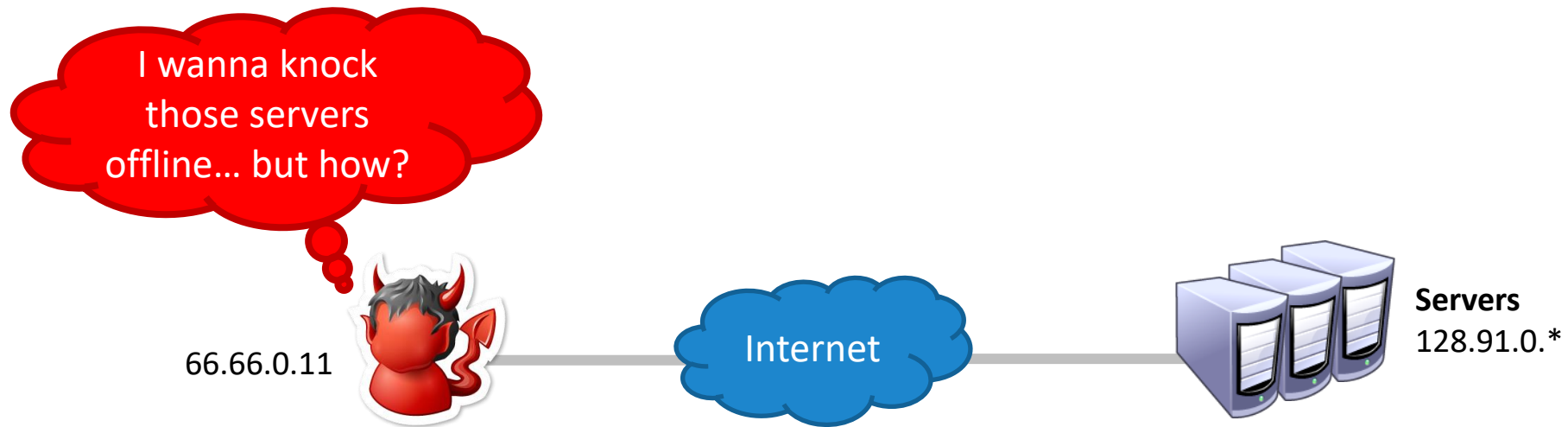
Denial of Service (Attacks on Availability)

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

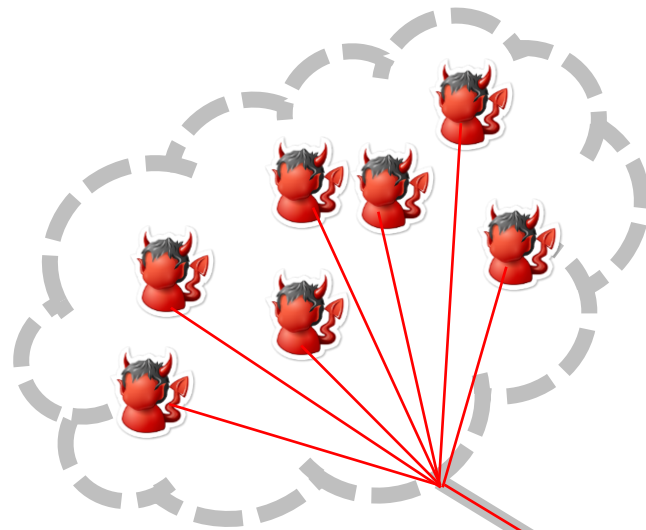
- Active attacker who may send arbitrary packets
- Goal is to reduce the availability of the victim



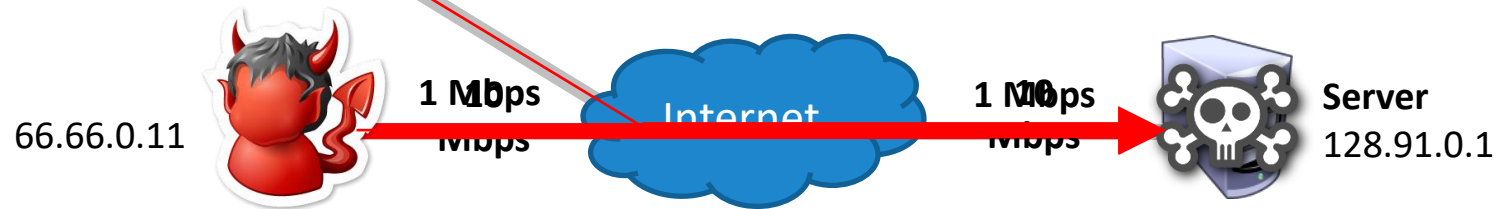
DoS Attack Parameters

- 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

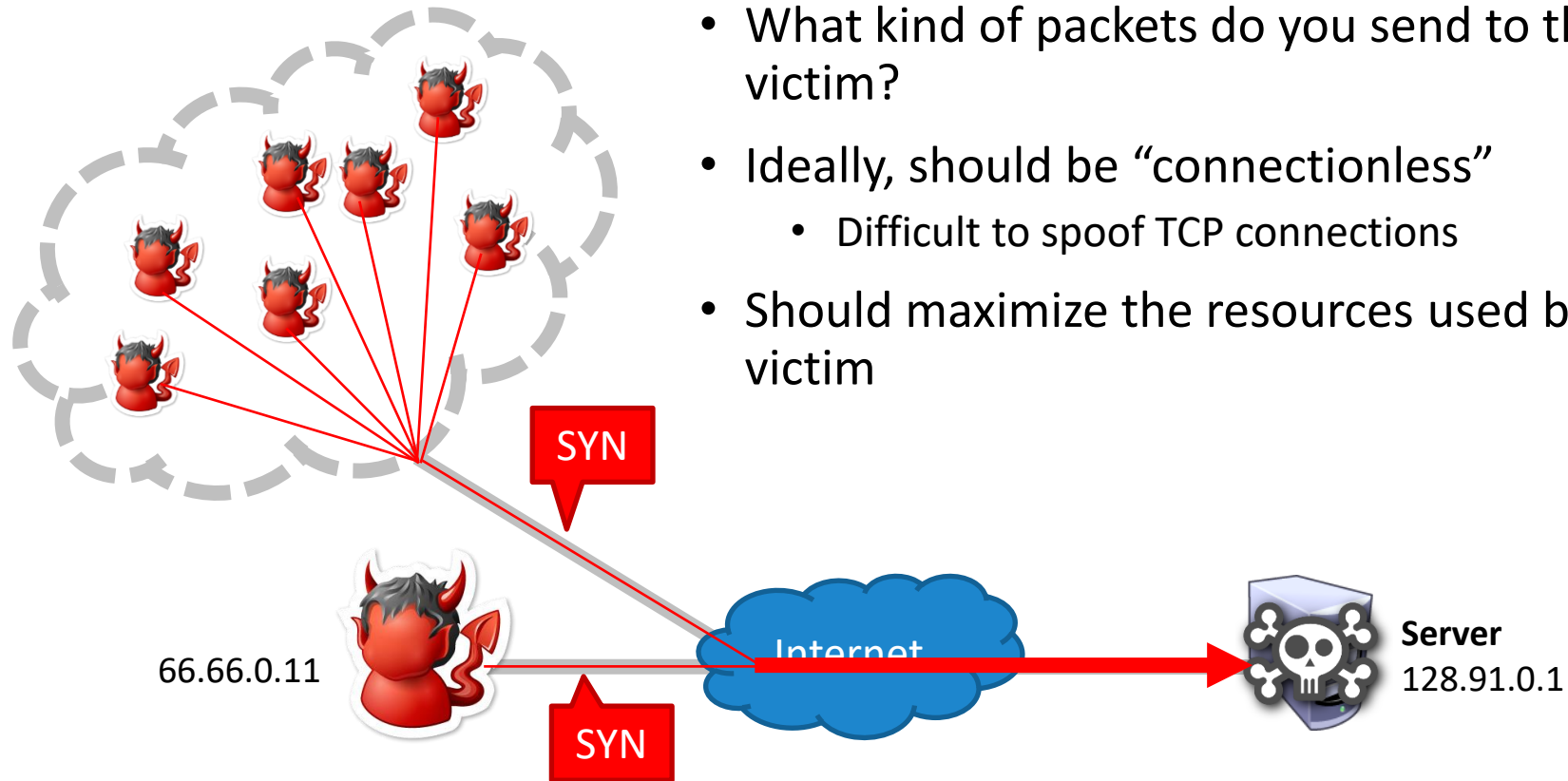
Distributed Denial of Service (DDoS)



- 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

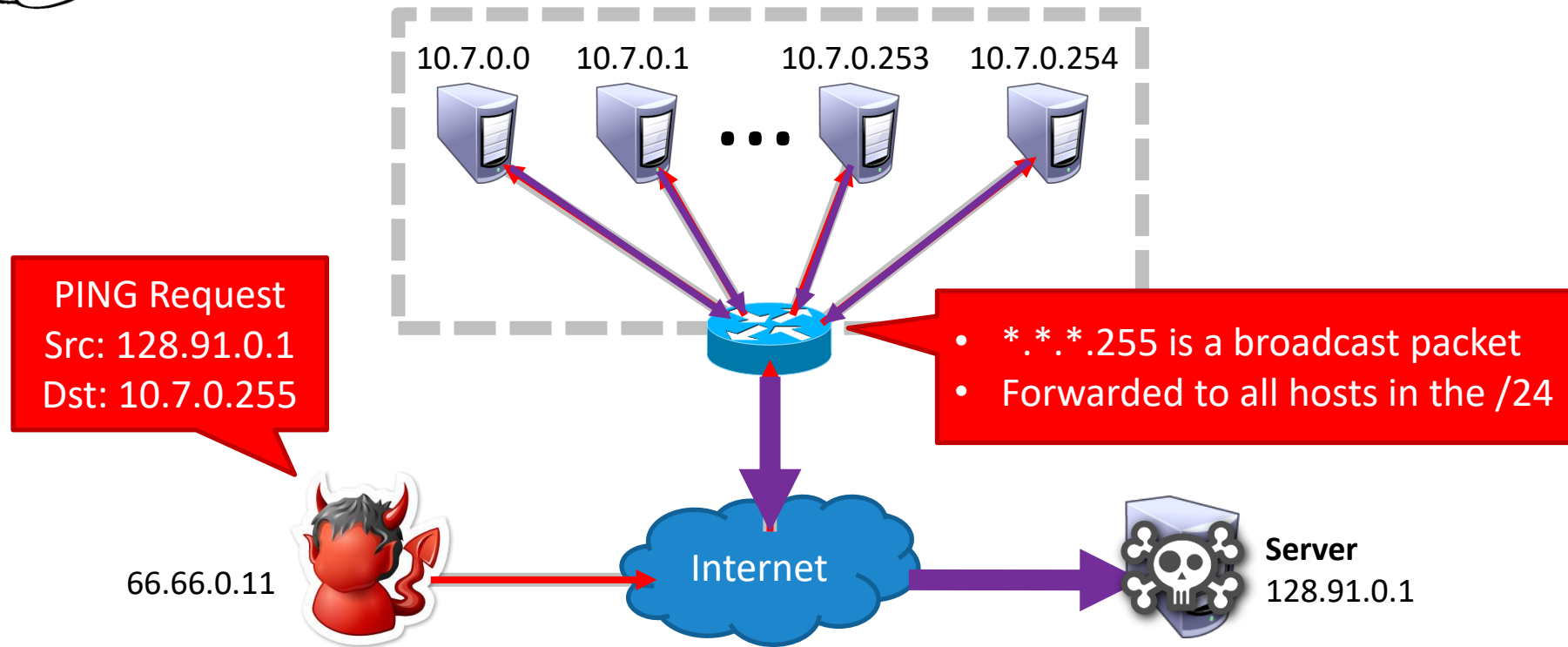


- What kind of packets do you send to the victim?
- Ideally, should be “connectionless”
 - Difficult to spoof TCP connections
- Should maximize the resources used by the victim

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**

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

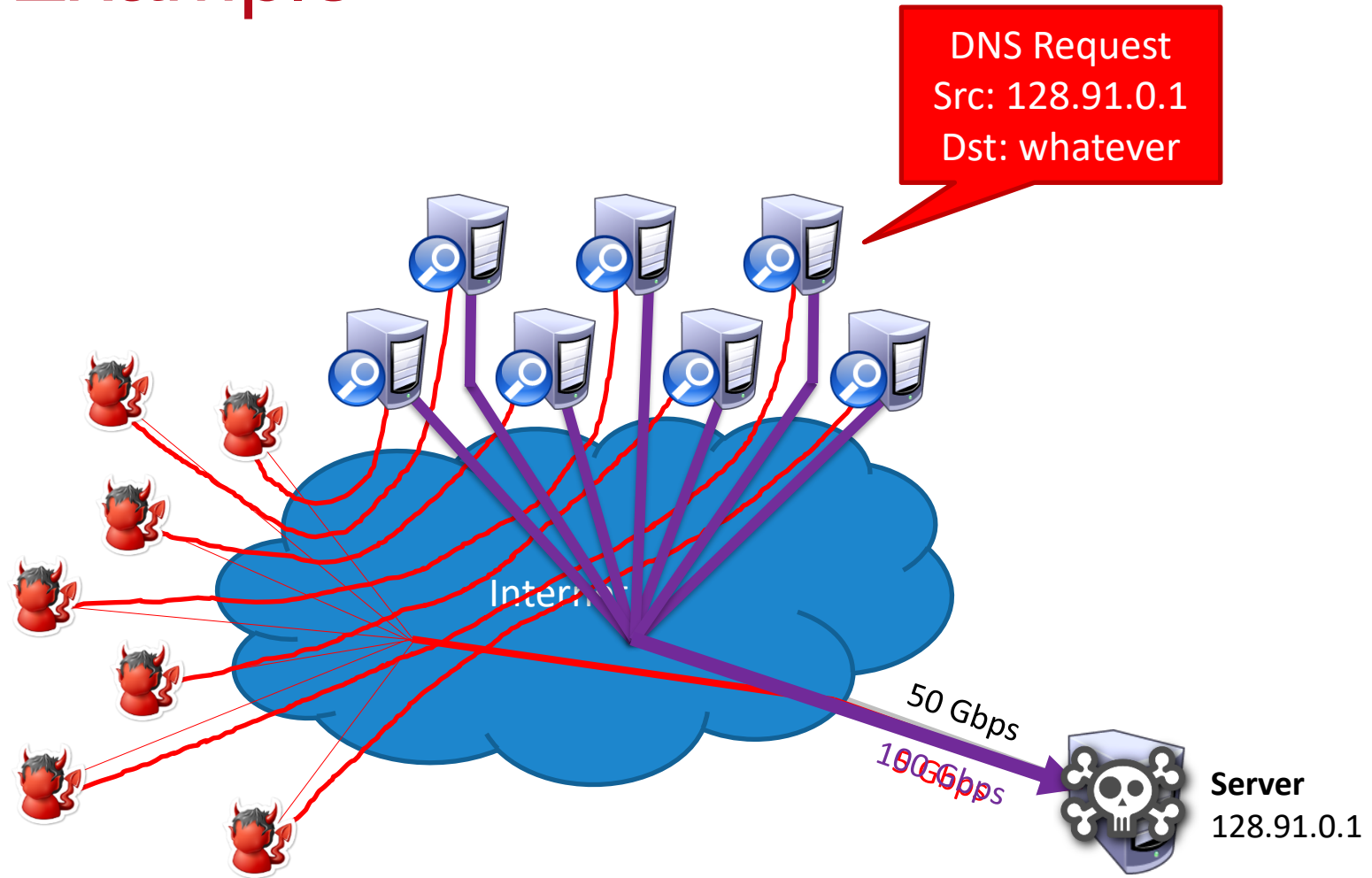
Reflection/Amplification Attacks

- 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

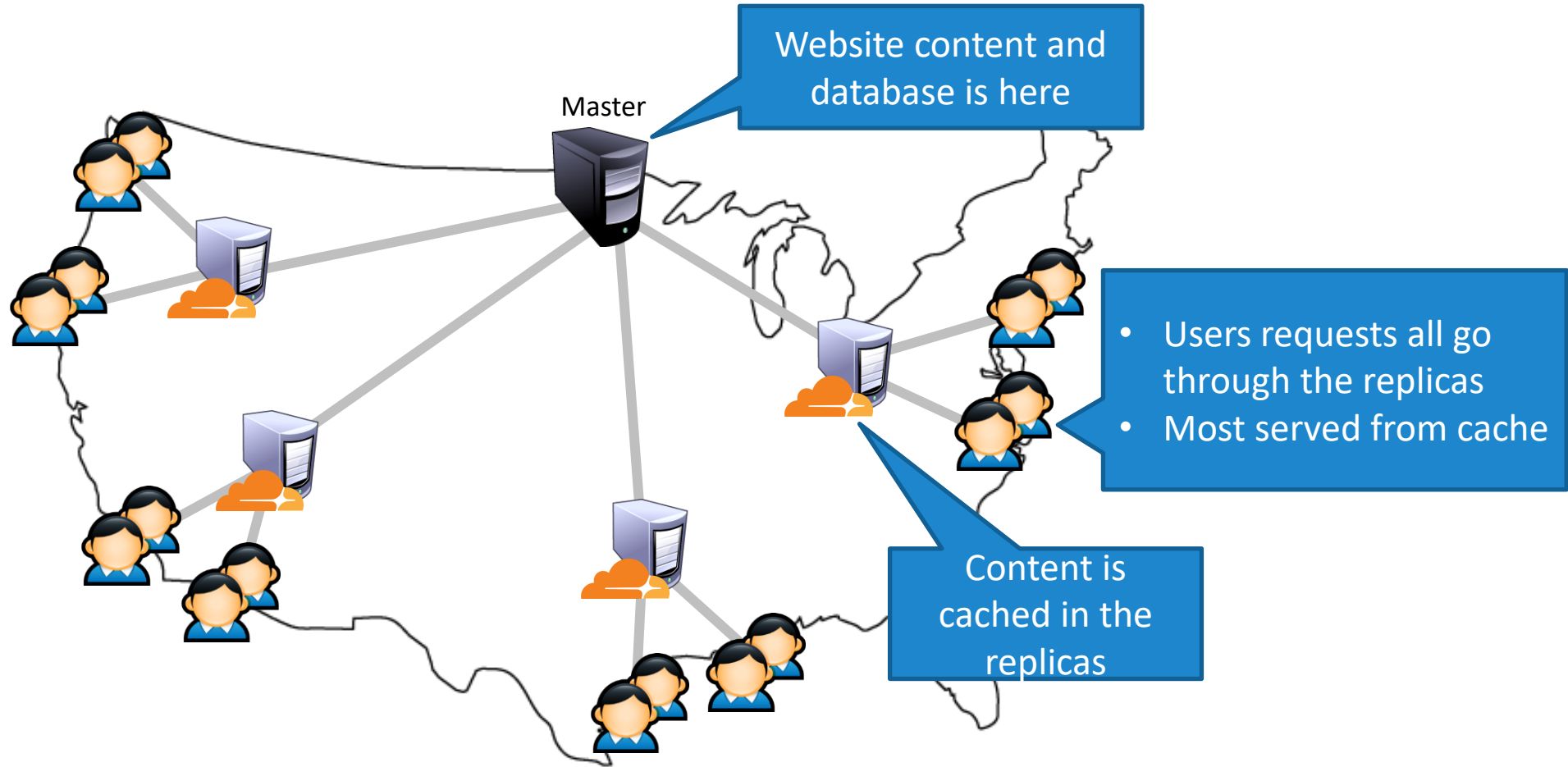
When	Against Who	Size	How
March 2013	Spamhaus	120 Gbps	Botnet + DNS reflection
February 2014	Cloudflare	400 Gbps	Botnet + NTP reflection
September 2016	Krebs	620 Gbps	Mirai
October 2016	Dyn (major DNS provider)	1.2 Tbps	Mirai
March 2018	Github	1.35 Tbps	Botnet + memcached reflection

Content Delivery Networks (CDNs)

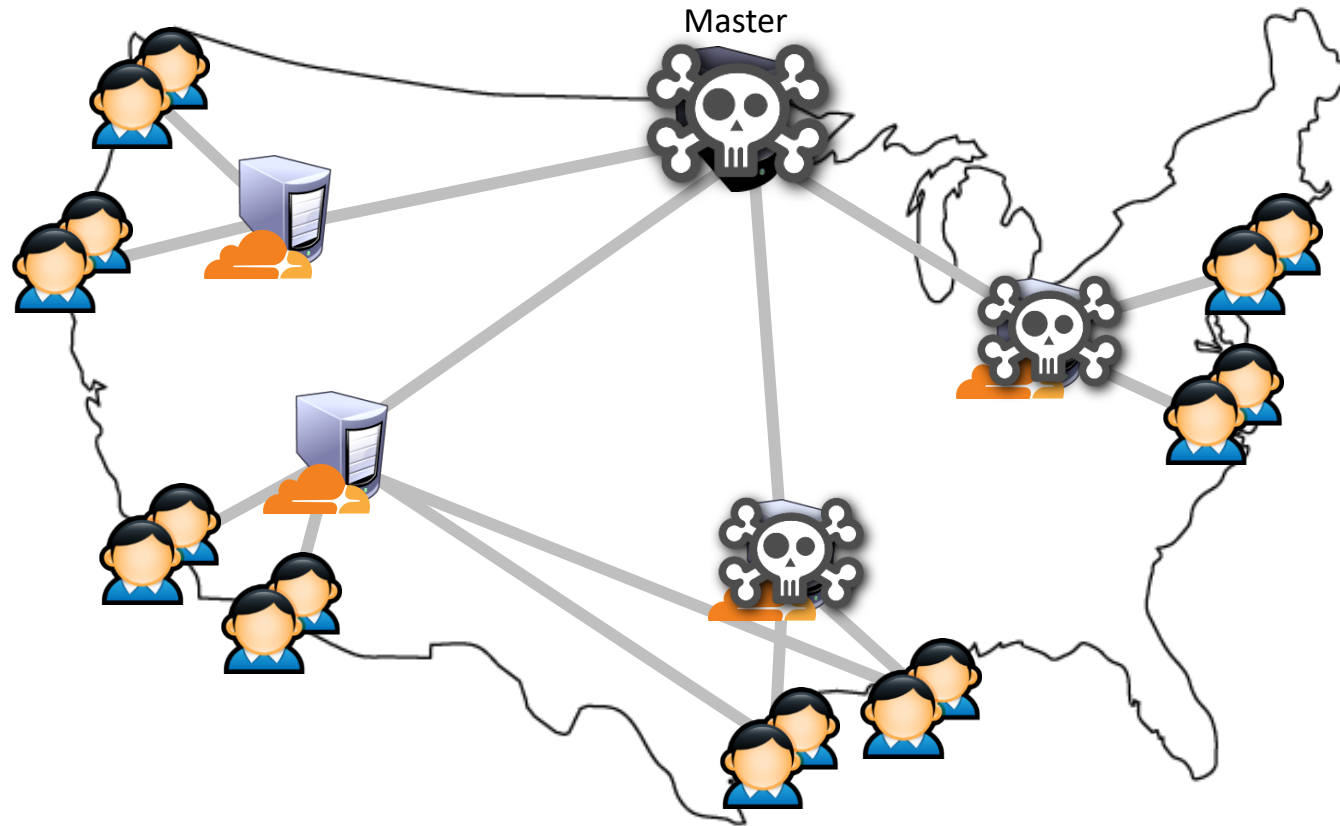
- CDNs help companies scale-up their websites
 - Cache customer content on many replica servers
 - Users access the website via the replicas
- Examples: Akamai, Cloudflare, Rackspace, Amazon Cloudfront, etc.
- Side-benefit: DDoS protection
 - CDNs have many servers, and a huge amount of bandwidth
 - 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



Content Delivery Networks (CDNs)



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