20. Network Attacks II

Blase Ur and David Cash
(many slides borrowed from Ben Zhao, Christo Wilson, & others)
February 25th, 2022
CMSC 23200 / 33250

THE UNIVERSITY OF CHICAGO
DNS attacks
DNS (Uncached)

1st User

“What’s the IP for example.com?”

“192.0.0.16”

DNS server

“What’s the IP for example.com?”

“192.0.0.16”

Authoritative nameserver

example.com
IP address: 192.0.0.16

Images from https://www.cloudflare.com/learning/dns/dns-cache-poisoning/
DNS (Cached, Benign)

```
2nd User

"What's the IP for example.com?"

"192.0.0.16" (Cached)

example.com
IP address: 192.0.0.16

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

User

“What’s the IP for example.com?”
“192.0.0.17” (Cached)

DNS server

example.com
IP address: 192.0.0.16

Malicious website
IP address: 192.0.0.17

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

Defense: randomize 16-bit QID

Alice

Q: www.bank.com
QID: x

Local DNS resolver

Mallory

spoof src IP of ns.bank.com

A: 3.3.3.3
guess QID: x

Race

A: 2.2.2.2
QID: x

ns.bank.com
Kaminsky attack (2008)

Alice runs JavaScript from mallory.com

Q: a.bank.com
Q: b.bank.com
Q: c.bank.com

Local DNS resolver

Mallory wins if any $r_i = s_j$

Spoof entire *.bank.com zone by including “sibling” domain

See [http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html](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

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

I wanna knock those servers offline... but how?

66.66.0.11

Internet

Servers 128.91.0.*
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

PING Request
Src: 128.91.0.1
Dst: 10.7.0.255

• *.*.*.255 is a broadcast packet
• Forwarded to all hosts in the /24
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

\[
amp \text{ factor} = \frac{total \, response \, size}{request \, size}
\]
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

DNS Request
Src: 128.91.0.1
Dst: whatever
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

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

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
Old-School C&C: IRC Channels

- Problem: single point of failure
- Easy to locate and take down

snd spam: <subject> <msg>
snd spam: <subject> <msg>
snd spam: <subject> <msg>
Fast Flux DNS

HTTP Servers

12.34.56.78  6.4.2.0  31.64.7.22  245.9.1.43  98.102.8.1

www.my-botnet.com

But: ISPs can blacklist the rendezvous domain

Change DNS → IP mapping every 10 seconds
Domain Name Generation (DGA)

Bots generate many possible domains each day.

...But the Botmaster only needs to register a few.

Can be combined with fast flux.

HTTP Servers

www.sb39fwn.com  www.17-cjq0n.com  www.xx8h4d9n.com
Software Security in the Browser
Drive-by Exploits

• Browsers are extremely complex
  – Millions of lines of source code
  – Rely on equally complex plugins from 3rd party developers
    • e.g., Adobe Flash, Microsoft Silverlight, Java
• Must deal with untrusted, complex inputs
  – Network packets from arbitrary servers
  – HTML, JavaScript, stylesheets, images, video, audio, etc.
• Recipe for disaster
  – Attacker directs victim to website containing malicious content
  – Leverage exploits in browser to attack OS and gain persistence
Executing a Drive-by

• Host exploits on a *bulletproof host*
  – No need to distribute (expensive) exploit code to other websites
  – Resist law enforcement takedowns

• Victim acquisition
  – Spam containing links (email, SMS, messenger)
  – Compromise legitimate websites & add traps (*e.g.*, via XSS)
    • Hidden *iframes* that load exploit website
    • Force a redirect to the exploit website