CSE508  Network Security

9/20/2017  Denial of Service Attacks

Michalis Polychronakis
Stony Brook University
Denial of Service

Goal: harm availability

Strain software, hardware, or network links beyond their capacity
Shut down or degrade the quality of a service

Not always the result of an attack

Flash crowds, “Slashdot effect”

Motives

Protest/attention
Financial gain/damage
Revenge
Blackmail
Evasion/diversion
DoS Attack Characteristics

Attack source: single vs. many
   More than a single source: Distributed DoS (DDoS)

Overload vs. complete shutdown
   Degradation vs. completely disabling software or equipment
   Crash, restart, bricking, website defacement, …

Consumed resource
   Network bandwidth, CPU, memory, sockets, disk storage, …

Amplification factor
   Symmetric vs. asymmetric attacks
   Broadcast addresses, large protocol responses, propagation, …

Algorithmic complexity attacks
   Induce worst-case behavior by triggering corner cases

Spoofing
   Hide the true source(s) of the attack
Lower Layer DoS

Physical layer

Wirecutting, equipment manipulation, physical destruction
RF jamming, interference

Link Layer

MAC flooding: overload switch/network
ARP poisoning: send fake ARP replies to insert erroneous MAC-IP mappings in existing systems’ caches
DHCP starvation
WiFi Deauthentication
Spectrum Blames Vandals For Internet Outages In Brooklyn, Queens

By SCOTT HEINS IN NEWS ON SEP 15, 2017 11:11 AM

Damage to Spectrum's fiber optic cables led to widespread internet loss throughout Queens and Brooklyn this morning. (Getty Images)

K Rowling is Brooklyn or Queens and had a tough time refreshing your Instagram feed this morning.

The Best Bars In NYC Where You Can Read In Peace

These Are The Best Restaurants In Williamsburg Right Now

The Best Breakfast Eggs
Dynamic Host Configuration Protocol (DHCP)

Used by hosts to request IP configuration parameters
IP address, gateway, DNS server, domain name, time server, …
UDP, no authentication: no way to validate a DHCP server’s identity

DHCP exhaustion
Prevent other clients from receiving IP addresses by consuming all available addresses in the DHCP server’s pool
DHCP relies on a client’s MAC address: *spoof it!*
Tool: DHCPwn

Rogue DHCP server *(may come after DHCP exhaustion)*
Provide incorrect information to clients, causing disruption
Worse: MitM attack

Defenses
DHCP snooping: network switch blocks bogus DHCP offers
(real server is assigned a *trusted* switch port)
Dynamic ARP Inspection (DAI): prevents ARP spoofing by validating
IP-to-MAC address bindings (derived from DHCP snooping)
Deauth Attacks

Send a spoofed deauth frame to AP with victims’ address (no authentication!)

- Client is disassociated from access point
- Can also use the broadcast address to disassociate all clients
  
  *They may then connect to an “evil twin” access point…*

Deauthentication is also sometimes used as a protection mechanism

- Prevent the operation of rogue access points

Tools: aireplay-ng (aircrack-ng), deauth (metasploit)

Also possible: auth attacks

- Flood with spoofed random addresses to authenticate and associate to a target access point ➔ exhaust AP resources
MARRIOTT TO PAY $600,000 TO RESOLVE WIFI-BLOCKING INVESTIGATION

Hotel Operator Admits Employees Improperly Used Wi-Fi Monitoring System to Block Mobile Hotspots; Agrees to Three-Year Compliance Plan

Washington, D.C. – Marriott International, Inc. and its subsidiary, Marriott Hotel Services, Inc., will pay $600,000 to resolve a Federal Communications Commission investigation into whether Marriott intentionally interfered with and disabled Wi-Fi networks established by consumers in the conference facilities of the Gaylord Opryland Hotel and Convention Center in Nashville, Tennessee, in violation of Section 333 of the Communications Act. The FCC Enforcement Bureau’s investigation revealed that Marriott employees had used containment features of a Wi-Fi monitoring system at the Gaylord Opryland to prevent individuals from connecting to the Internet via their own personal Wi-Fi networks while at the same time charging consumers.
Network Layer DoS

Flooding: bombard target with network packets
  Saturate the available network bandwidth (aka “volumetric” attacks)
  Long ICMP packets, UDP/TCP packets with garbage data, …

IP spoofing: conceal the attack source
  Makes it more difficult to block the attack
  Ingress and egress filtering limit its applicability, but not universally deployed
  Applicable only when connection establishment is not needed: ICMP, UDP, TCP SYN, …

Broadcast Amplification
  One packet generates many more packets
  ICMP Smurf Attack (spoofed broadcast Echo request)

IP hijacking (covered in previous lecture)
  False BGP route advertisements to attract and drop traffic or cause connectivity instability
**Smurf Attack** (90’s)

Attacker sends spoofed ICMP Echo requests to the victim’s network broadcast address

Src IP == victim’s IP

Victim machine is flooded with responses from all internal hosts

Initial form of *amplification*

**Mitigation**

Configure hosts to not respond to broadcast ICMP requests

Configure routers to not forward packets destined to broadcast addresses
Transport Layer DoS

SYN flooding
  Server-side resource exhaustion
  Source IP address can be spoofed
  Can be combined with normal flooding to also saturate link

Connection termination
  RST injection
  Mostly used for blocking specific unwanted traffic
SYN Flooding

Flood server with spoofed connection initiation requests (SYN packets)

- Saturate server’s max number of concurrent open sockets: no more connections can be accepted
- Each half-open connection consumes memory resources
- Server sends SYN/ACKs back, but ACKs never return…

Mitigation

- Drop old half-open connections after reaching a certain threshold (in FIFO order or randomly)
- SYN cookies: eliminate the need to store state per half-open connection
**SYN Cookies**

Don’t ignore connections after SYN queue fills up

*Instead:*

Send SYN/ACK with special “cookie” seq

Secret function of the src/dst IP, src/dst port, coarse timestamp

**Stateless!** SYN queue entry is rebuilt based on the returned cookie value in the ACK
TCP Connection Termination

FIN: this side is done sending, but can still receive
   “Half-closed” state
   Should be sent by each side and acknowledged by the other

RST: this side is done sending and receiving
   No more data will be sent from this source on this connection
   Program closed, abort established connection, …

A MotS attacker can easily send spoofed RST packets
   5-tuple (src/dst IP/port and protocol) must match
   Sequence number should be in window
   More strict stacks will only accept RSTs in sequence \(\rightarrow\) Prevent blind TCP RST injection

Legitimate and not so legitimate uses
   Censorship, blocking of non-standard port traffic (e.g., P2P protocols),
   termination of malicious connections, …
Comcast settles P2P throttling class-action for $16 million

Comcast got itself in hot water when it decided to use reset packets to slow...

by Jacqui Cheng - Dec 22, 2009 4:22pm EST

Comcast has agreed to settle a class-action lawsuit over the throttling of P2P connections that had users up in arms in late 2007 and 2008. The company still stands behind its controversial methods for "managing" network traffic, but claims that it wants to "avoid a potentially lengthy and distracting legal dispute that would serve no useful purpose."

It was more than two years ago when Comcast subscribers began finding evidence that the broadband provider was blocking packets—particularly those being sent through BitTorrent. When the complaints mounted, the Associated Press went ahead with its own investigation and came to the same conclusion: downloads through BitTorrent were either being blocked altogether or being slowed down significantly.

At that time, Comcast vehemently denied that it had anything to do with these mysterious slowdowns. This was despite the fact that numerous customers reported that their Comcast connections were sending reset packets out to the rest of the Internet—the AP discovered that nearly half of the reset packets being received by cable competitor Time Warner were coming from Comcast. Eventually, Comcast acknowledged that it had engaged in "traffic management" techniques in order to keep its network speedy, which eventually resulted in an FCC investigation and a subsequent abandoning of its P2P-hating ways in favor of a more neutral congestion management system.
Application Layer DoS

Connection flooding

Reflection

Software vulnerabilities

Algorithmic complexity attacks

Trigger worst-case processing (e.g., hashtable collisions, regular expression backtracking)

Exhaustion of server resources

Example: fill up FTP server with junk files

Spam can be considered as a DoS attack on our time…

And server resources
Connection Flooding

Saturate the server with many established connections
   Can’t use spoofing: just use bots…

For forking servers, the whole system might freeze (process exhaustion)

Slowloris attack: slowly send a few bytes at a time to keep many concurrent connections open
   Keep the server busy with “infinite-size” HTTP requests by periodically sending more and more bogus HTTP headers
   Alternatives: read response slowly, POST data slowly, …
   Requires minimal bandwidth
Amplification/Reflection Attacks

Like the ICMP Smurf attack

Abuse network services that reply to certain requests with much larger responses

  Attacker sends a small packet with a forged source IP address
  Server sends a large response to the victim (forged IP address)

UDP: connectionless protocol ➔ easy to spoof

Used by many services:
  NTP, DNS, SSDP, SNMP, NetBIOS, QOTD, CharGen, …
Technical Details Behind a 400Gbps NTP Amplification DDoS Attack

13 Feb 2014 by Matthew Prince.

On Monday we mitigated a large DDoS that targeted one of our customers. The attack peaked just shy of 400Gbps. We’ve seen a handful of other attacks at this scale, but this is the largest attack we’ve seen that uses NTP amplification. This style of attacks has grown dramatically over the last six months and poses a significant new threat to the web.
Amplification Factor

<table>
<thead>
<tr>
<th>Protocol</th>
<th>BAF all</th>
<th>BAF 50%</th>
<th>BAF 10%</th>
<th>PAF all</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP v2</td>
<td>6.3</td>
<td>8.6</td>
<td>11.3</td>
<td>1.00</td>
<td>GetBulk request</td>
</tr>
<tr>
<td>NTP</td>
<td>556.9</td>
<td>1083.2</td>
<td>4670.0</td>
<td>3.84</td>
<td>Request client statistics</td>
</tr>
<tr>
<td>DNS-NS</td>
<td>54.6</td>
<td>76.7</td>
<td>98.3</td>
<td>2.08</td>
<td>ANY lookup at author. NS</td>
</tr>
<tr>
<td>DNS-OR</td>
<td>28.7</td>
<td>41.2</td>
<td>64.1</td>
<td>1.32</td>
<td>ANY lookup at open resolv.</td>
</tr>
<tr>
<td>NetBIOS</td>
<td>3.8</td>
<td>4.5</td>
<td>4.9</td>
<td>1.00</td>
<td>Name resolution</td>
</tr>
<tr>
<td>SSDP</td>
<td>30.8</td>
<td>40.4</td>
<td>75.9</td>
<td>9.92</td>
<td>SEARCH request</td>
</tr>
<tr>
<td>CharGen</td>
<td>358.8</td>
<td>n/a</td>
<td>n/a</td>
<td>1.00</td>
<td>Character generation request</td>
</tr>
<tr>
<td>QOTD</td>
<td>140.3</td>
<td>n/a</td>
<td>n/a</td>
<td>1.00</td>
<td>Quote request</td>
</tr>
<tr>
<td>BitTorrent</td>
<td>3.8</td>
<td>5.3</td>
<td>10.3</td>
<td>1.58</td>
<td>File search</td>
</tr>
<tr>
<td>Kad</td>
<td>16.3</td>
<td>21.5</td>
<td>22.7</td>
<td>1.00</td>
<td>Peer list exchange</td>
</tr>
<tr>
<td>Quake 3</td>
<td>63.9</td>
<td>74.9</td>
<td>82.8</td>
<td>1.01</td>
<td>Server info exchange</td>
</tr>
<tr>
<td>Steam</td>
<td>5.5</td>
<td>6.9</td>
<td>14.7</td>
<td>1.12</td>
<td>Server info exchange</td>
</tr>
<tr>
<td>ZAv2</td>
<td>36.0</td>
<td>36.6</td>
<td>41.1</td>
<td>1.02</td>
<td>Peer list and cmd exchange</td>
</tr>
<tr>
<td>Sality</td>
<td>37.3</td>
<td>37.9</td>
<td>38.4</td>
<td>1.00</td>
<td>URL list exchange</td>
</tr>
<tr>
<td>Gameover</td>
<td>45.4</td>
<td>45.9</td>
<td>46.2</td>
<td>5.39</td>
<td>Peer and proxy exchange</td>
</tr>
</tbody>
</table>

**TABLE III:** Bandwidth amplifier factors per protocols. *all* shows the average BAF of all amplifiers, 50% and 10% show the average BAF when using the worst 50% or 10% of the amplifiers, respectively.
Evil Packets

Trigger a server-side bug to crash a processor even the kernel (system restart)
  Typically just a single packet/request

Ping of death (1996)
  Typical ICMP Echo request (ping) packet size: 84 bytes
  Max IPv4 packet size: 65,535 bytes
  Oversized ICMP ping packets would trigger a buffer overflow

LAND (1997)
  Spoofed TCP SYN with target IP == source IP
  TCP stack gets confused and eventually crashes

Teardrop (1997)
  Specially crafted overlapping IP fragments would trigger IP
defragmentation bug
FreeBSD has patched a denial-of-service vulnerability affecting versions configured to support SCTP and IPv6, the default configurations on later version of the open source OS.

Researchers at Positive Technologies in the U.K. said versions 9.3, 10.1 and 10.2 are affected and can be exploited by a specially crafted ICMPv6 packet, which will cause a kernel panic; kernel panics are the UNIX equivalent of a Windows Blue Screen of Death.
Evil Packets/Requests/Inputs

WinNuke (1997)
  String of out of band (OOB) data to NetBIOS service (port 139)
  Blue screen of death on Windows NT/95

Internet worms would often crash infected hosts
  Besides the internet-wide network flood due to their rapid propagation
  and occasional DDoS activity
  *Morris worm* (1988): internet was partitioned for several days…
  www.whitehouse.gov
  *Blaster* (2003): DoS against
  windowsupdate.com, system instability
  causing endless reboots
  *Witty* (2004): Single UDP packet, slow
  disk corruption leading to crash

Malware can even brick the system
  Erroneous firmware update, BIOS flashing,
  driver malfunction, data corruption, …
Distributed Denial of Service (DDoS)

Any DoS attack that originates from multiple sources
   Early internet worms were the first instances of DDoS

These days usually launched by botnets
   Networks of compromised systems ("bots") awaiting commands by an attacker ("botmaster")
   Not only PCs/servers: mobile and IoT devices equally useful
   Can be rented through online marketplaces ("booter" or "stresser" services)
DDoS attack on Dyn came from 100,000 infected devices

DNS service provider Dyn says Mirai-powered botnets were the primary source for Friday's disruption

By Michael Kan

U.S. Correspondent, IDG News Service | OCT 26, 2016 2:21 PM PT
Puppetnets: Browser-based Bots

Browsers can be indirectly misused to attack others
  JS code running in the browsers of unsuspecting visitors

Continuously fetch images or other large files from the victim’s server
  Can masquerade as “good” bots (e.g., Googlebot, Baiduspider, other legitimate spiders) using a spoofed User-Agent

Many injection ways
  Compromised websites
  Ad networks
  MitM/MotS attacks
The Pirate Bay is Pirating Your Processor for Bitcoin Mining

By Darryn Pollock

SEP 18, 2017

For updates and exclusive offers, enter your e-mail below.

Email Address

SUBSCRIBE
Chinese attackers used the Great Firewall’s offensive sister-system, named the Great Cannon, to launch a recent series of distributed denial of service attacks targeting the anti-censorship site, GreatFire.org, and the code repository, Github, which was hosting content from the former.

The first set of DDoS attacks hit GreatFire.org on March 16. On March 26, Github, which was hosting content from the former.
March 16 and 26, 2015

DoS target: GreatFire.org and two related GitHub pages (anti-censorship project)

DoS attack script injected into 1.75% of the requests to Baidu’s analytics/ad scripts (probabilistic injection)
Energy DoS

Strain the power source of mobile, IoT, sensor devices

Battery exhaustion

Consume battery by performing power-hungry operations in the background
Computation, radio activity, …

Denial of sleep

Specific to energy-constrained embedded systems that wake up periodically for data transmission
An attack can force radios to remain constantly active
Can reduce battery life by orders of magnitude
DoS Defenses

No absolute solution

Asymmetry: little effort for the attacker, big impact for the victim
Any public service can be abused by the public
Prank phone calls, road blockades, …

General strategies

Filter out bad packets
Improve processing of incoming data
Hunt down and shut down attacking hosts
Increase hardware and network capacity
DoS Defenses

Ingress/egress filtering
   Ensure that incoming/outgoing packets actually come from the networks they claim to originate from ➔ drop spoofed packets

Content delivery networks (CDNs) and replication
   Distribute load across many servers

Client challenges
   Present a CAPTCHA whenever the system is under stress

Other (mostly academic) approaches
   IP Traceback: each router “marks” with its own IP the forwarded packets to facilitate determining the actual origin of packets
   Pushback filtering: iteratively block attacking network segments by notifying upstream routers
   Overlay-based systems: proactive defense based on secure overlay tunneling, hash-based routing, and filtering
To continue, please type the characters below:

![Image of characters to type]

Submit

---

About this page

Our systems have detected unusual traffic from your computer network. This page checks to see if it's really you sending the requests, and not a robot. [Why did this happen?](#)