Details of selected DDoS Attacks

How the attacks work from a technical perspective

Klaus Möller
WP8-T1 - EaPConnect

Webinar, November 2021

Public

www.geant.org

What we will cover today - and what not

- Layer 3 & 4 Attacks
  - ICMP
  - UDP
  - TCP
- Layer 5+ Attacks
  - HTTP
  - SSL
  - Regular expressions
  - Hash functions

- Local DoS Attacks
  - Even those that have network coordination, downloads, payment, etc.
  - Encrypting Ransomware
- Physical Attacks (Layer 1)
- Data-Link Attacks (Layer 2)

DDoS Attack Characterisation by OSI Layer

- Layer 1 (Physical)
  - Cutting cables, electronic jamming, breaking line-of-sight, …
- Layer 2 (Link Layer)
  - ARP spoofing/poisoning, CAM table floods, WiFi (de)authentication attacks, …
- Layer 3 (Network)
  - ICMP/UDP flood, too large packets (Ping of Death), overlapping fragments (Teardrop), …
- Layer 4 (Transport)
  - TCP <Flag> Flood, TCP Connect, Window size 0, …
- Layer 5+ (Session, Presentation, Application)
  - Slow GET/POST (HTML) Re(gular Expression) DoS, SSL DoS, XML (Billion Laughs), …

DDoS Attack Characterisation by Effect on Target

- Crashes (***vulnerability***)
  - Most severe: permanent hardware/firmware damage
  - Less severe: Kernel panics (Blue screens), Reboots, Lock-ups
  - Limited: Application Crashes/Core dumps or Lock-ups
- Exhaustion of (limited) resources (***volumetric***)
  - Line Bandwidth
  - Packet switching capacity
  - CPU cycles
  - Memory
    - # of Processes, # of half-open connections, # of semaphores, etc.
- Recovery may be immediate (after the attack ends) or take some time(out) period

TCP/IP Vulnerability Attacks

www.geant.org

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

Vulnerability Attack: *LAND*

- Local Area Network Denial
  - First discovered in 1997
  - Nearly all OS of that time were vulnerable
  - Similar application layer attacks have been found in other services (SNMP, Kerberos)
- Source and destination IP address are that of the victim
- Source and destination port are the same, needs open port on victims machine
- Attacker needs one packet to start, victims system then endlessly replies to itself, eventually locking up
- Simulate: `hping3 -S -p 80 -a target target`

Vulnerability Attack: *Teardrop*

- Very old: 1997 – today's OSes are not vulnerable
- Attackers sends TCP/IP packets with specially crafted IP fragments
  - Overlapping fragments
    - I.e. fragment offset + payload size ≠ fragment offset of next packet
  - Assembled payload will be bigger than the maximum IP packet size (65536 bytes)
- Triggers bug in kernel packet re-assembly code → OS crash
  - Windows 3.1x, Windows 95, and Windows NT
  - Linux < 2.0.32 and < 2.1.63
- Related vulnerabilities appeared 2018
  - ***Fragment Smack***, (IP) fragment re-assembly queue, CVE-2018-5391
  - ***Segment Smack***, TCP Segments with random offsets, CVE-2018-5390

Vulnerability ICMP Attack: *Ping of Death*

- Very old (mid 90s) attack that crashed systems
  - Exploits bug in the IP stack, when dealing with reassembled packets being larger than 65536 bytes, the maximum size of IP packets
  - Host OS needs to check that fragment offset + fragment size < 65536
- Attacker sends a number of fragmented IP-Packets to the victim host
  - Usually done with ICMP Echo Reply packets, hence the name
  - Last fragment has data part longer than 7 bytes
  - `ping -l 65510` ***`target`***
- Re-appeared in 2013 with ICMPv6 and in 2020 with ICMPv6 Router Advertisements
  - CVE-2013-3183 and CVE-2020-16898

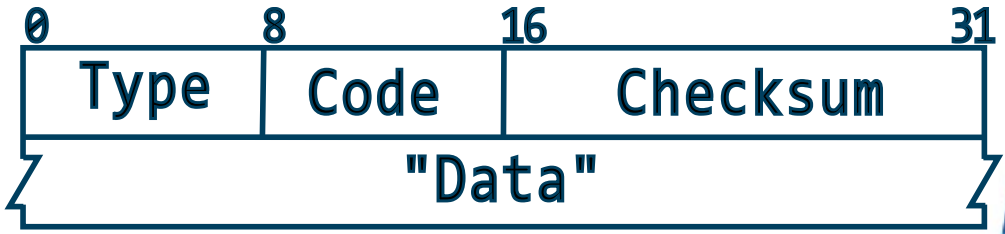
ICMP Attacks

www.geant.org

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

ICMP – Internet Control Message Protocol

- Management protocol integrated into IP layer
  - Most important: Type and Code
  - Checksum & “Data” irrelevant here



- Use cases
  - Connectivity test (Echo Request/Reply) type 8/0
  - Destination unreachable report: type 3, code 0 – 15
  - Re-routing of packets (Redirect) type 5, code 0 - 2
  - Router Advertisement/Solicitation: type 9/10 (usually not for IPv4)
  - Error Reporting (Time exceeded, Parameter Problem): type 11/12, code 0-2
  - Time synchronisation (Timestamp Request/Reply): type 13/14
- Everything else is not legit and can be discarded safely
  - Router adv. and Time sync can be blocked too

ICMP Flood Attacks

- PING: Pair of ICMP Echo Request & -Reply packets
- Attacker spoofs ICMP Echo Reply packets (type 0, code 0)
  - Echo request would DoS himself
  - Arbitrary “data”, checksum doesn’t need to be correct
- Other ICMP floods with other types/codes
- Destination address is that of the victim (unless reflection attack)
- Source address is usually spoofed, i.e. random
  - But may be constant or even that of the attacking bot
  - Depends on egress filtering at the source network
- Packet size varies, usually minimum (64) or maximum (1500)
  - Depending on whether the attacker wants to exhaust switch’s packet switching capacity or line bandwidth

Reflected ICMP Flood: *Smurf*

- One of the earliest Reflection/Amplification DDoS attacks
  - Attacker spoofs victims IP address as source address
  - Sends packet to directed broadcast address of a network, the **reflector**
  - E.g. 192.168.12.255 (if net mask is 255.255.255.0)
- All active hosts on the network would reply back to victim
  - Amplification factor varied in practice: usually 10 – 100
- Since deprecation of directed broadcast forwarding, this has lost significance in favour of other amplifiers
  - Host are also discouraged from answering directed broadcasts
- Simulate: `hping3 --icmp --flood -a` ***`target reflector`***

ICMP Attack: *Ping Flood*

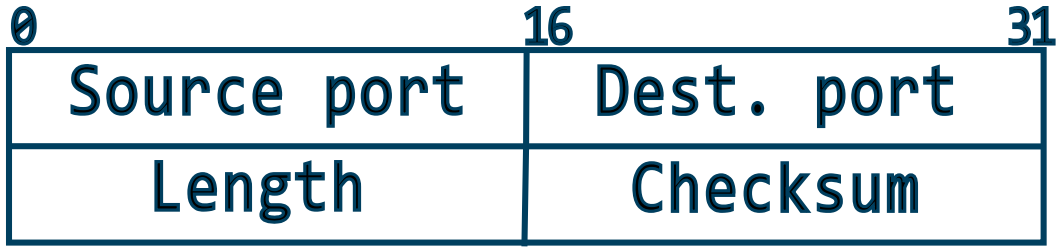
- Saturation attack against line bandwidth
  - Secondary effect on CPU usage of the victims host
- Attacker sends ICMP Echo **Request** packets as fast as possible
  - Will elicit Echo Responses from Victim host
- Goal is to saturate both downlink (to the victim) and uplink
- Works well with asymmetric Lines (DSL)
- Source address is spoofed (or attacker would DoS herself)
- Other characteristics as in ICMP floods

UDP Attacks

www.geant.org

UDP – User Datagram Protocol

- Basically a minimum layer 4 when one just wants to send raw IP packets
  - Header is just source and destination port numbers, length, and checksum



- Used for a wide variety of purposes, mostly when minimum overhead is desired
  - Security considerations most often **not** a priority
- Connection-less nature of UDP makes applications vulnerable to IP address spoofing

UDP Amplification Attacks

- Same principle as with IP/ICMP flood attacks
  - I.e. volumetric attack against bandwidth and/or packet switching capacity
- In most of today's cases, makes use of amplifiers/reflectors
  - I.e. attacker uses victims IP address as query source address (spoofing)
  - Vulnerable service answers (reflecting), hiding attackers IP address in the process
  - Reply of the service is usually much bigger than the query → bandwidth amplification
    - *Bandwidth Amplification Factor (BAF)*
  - Sometimes, service sends out multiple packets for one query packet → packet per second (pps) amplification
    - *Packet Amplification Factor (PAF)*

UDP Amplifiers (and how to find them)

Protocol	BAF	PAF	Port No (udp)	Scenario/Command
DNS auth. NS	54.6	2.08	53	`dig @target +edns +ignore com ANY`
DNS open res.	28.7	1.32	53	`dig @target +edns +ignore com ANY`
mDNS	2 - 10		5353	`dig @target -p 5353 +ignore +noedns wpad`
NTP	556.9	3.84	123	`ntpdc –nc monlist target`
SNMPv2	6.3	1.00	161	`snmpbulkget -v2c -c public target 1.3`
NetBIOS NS	3.8	1.00	137	`nmblookup –A target`
SSDP	30.8	9.92	1900	M-SEARCH request
cLDAP	56 - 70	---	389	---
TFTP	60	---	69	`tftp` command trying to download well known files
Memcached	10K - 51K	---	11211	---
WS-Discovery	10 - 500	---	3702	---

Source: https://us-cert.cisa.gov/ncas/alerts/TA14-017A

UDP Amplifiers: P2P, Games & Other

Protocol	BAF	PAF	Port No (udp)	Scenario/Cmd
BitTorrent (P2P)	3.8	1.58	6671	File search
Kademlia (P2P)	16.3	1.00	varies	Peer list exchange
Quake (Game)	63.9	1.01	27950+	Server info exchange
Steam (Game)	5.5	1.12	27015, 27005	Server info exchange
CharGen	358.8	1.00	19	Character gen. request
Quote of the Day	140.3	1.00	17	Quote request
RIPv1	131.24	---	520	Malformed request
Portmap (RPCbind)	7 - 28	---	111	Malformed request
ARMS (Apple Net Assistant)	35.5	2.00	3283	---
Microsoft RDP	85	---	3389	---

Source: https://us-cert.cisa.gov/ncas/alerts/TA14-017A

UDP Fragmentation (Attacks)

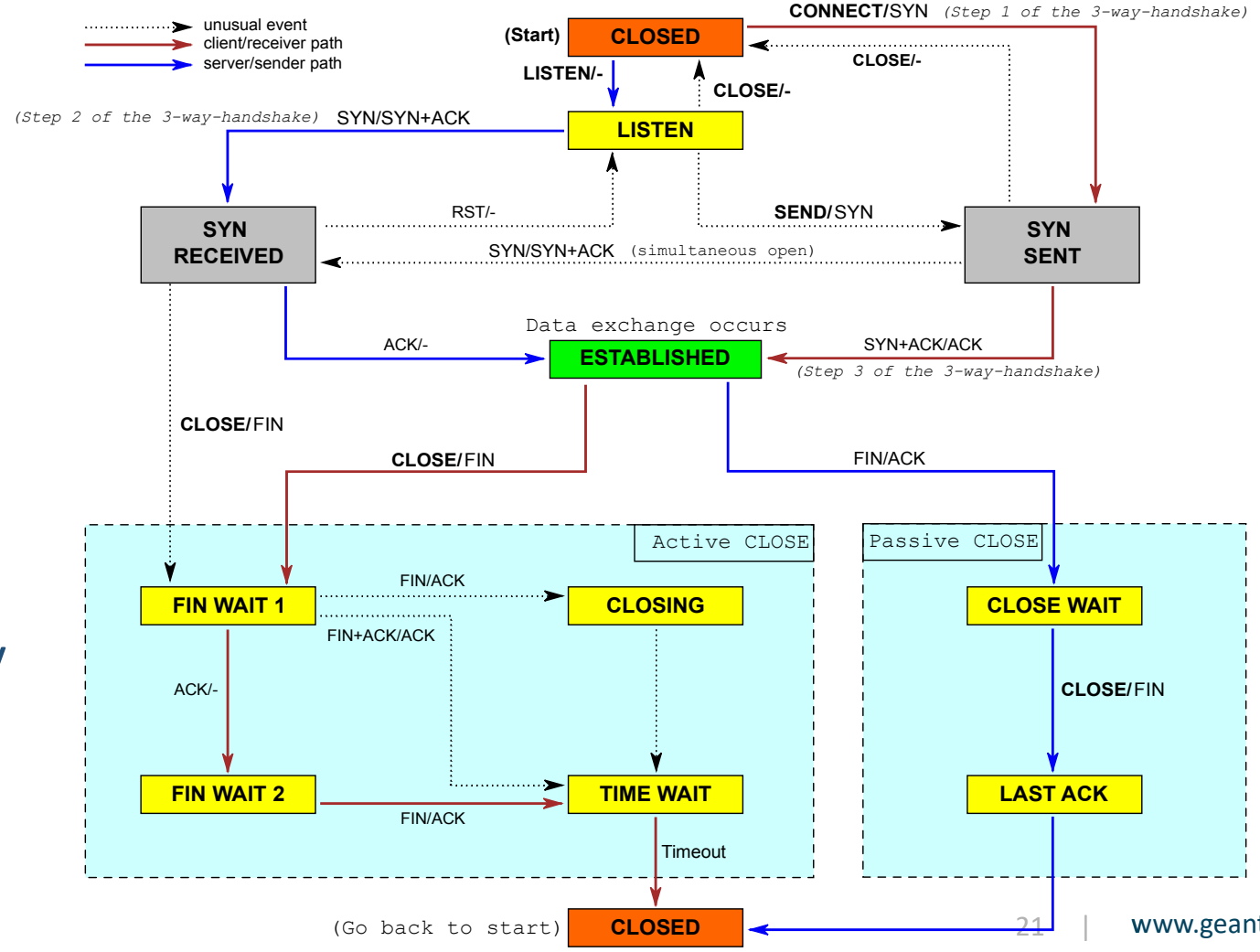
- Seen as fragmented packets with UDP source and destination port 0
- An artefact of buggy/stateless packet counting/reporting
  - 2nd and later fragments don't carry the UDP header
  - Probe only sees IP header with IP protocol number 17 (UDP)
  - Without UDP header to look for port numbers, port 0 is reported
  - Unless probe is smart enough to re-assemble fragments
  - Would require memory for fragment storage → DoS vulnerability in the probe
  - Can't be done in hardware/at wire speed

TCP Attacks

www.geant.org

TCP: Transport Control Protocol

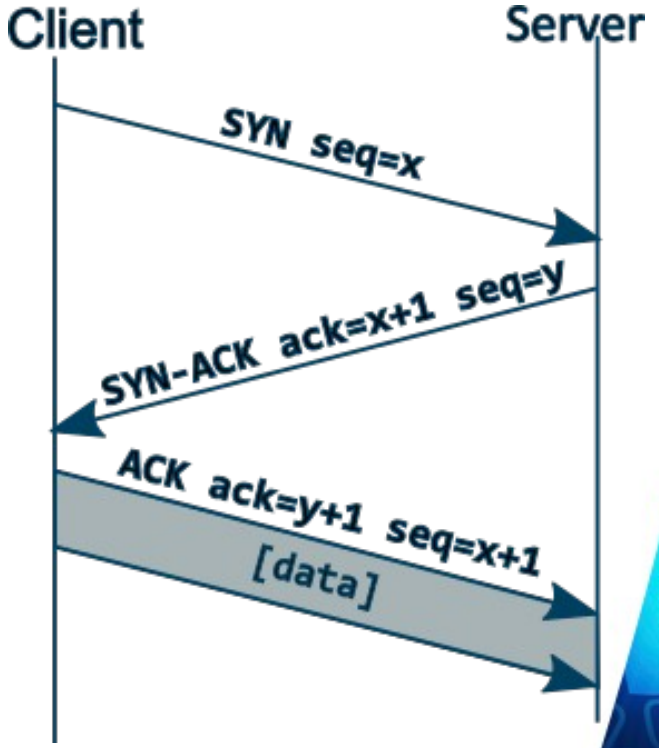
- Layer 4 Protocol for a bi-directional stream between applications
  - State of a TCP connection is modelled through a finite automaton, aka *state-machine*
  - Weaknesses in the automaton are exploited by attackers



Source: https://en.wikipedia.org/wiki/Transmission_Control_Protocol

TCP Attacks: *SYN-Flood*

- One of the oldest and still used attacks
- Exploits memory limit for the TCP 3-way handshake
  - Server has to keep state (seq & ack numbers, options, src/dst ip address & ports)
  - Attacker sends initial TCP SYN packet but never replies to SYN-ACK from server
    - Open port at victim IP address needed
    - Source IP address must not reply for timeout
  - Leaves connection in SYN_RECEIVED state
  - *Total timeout = timeout per SYN-ACK * # resends*




```
> sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 2048
```

TCP Attacks: *SYN-Flood* Detection

- Server side detection
  - Very high number of sockets in SYN-RECV state
- Client side detection:
  - Server doesn't respond / high number of connection failures
- Simulate: `hping3 -S -p` *`Port Target`* `(--flood)`
  - Only little bandwidth is needed
- Self-inflicted SYN-Floods
  - Routing misconfiguration drops SYN-ACK packet on the way back to clients
- How to tell apart from high traffic load
  - Server also has high number of normal (ESTAB) connections

TCP Attacks: *Syn-Flood* Example

```
 > ss -anto
State        Recv-Q   Send-Q        Local Address:Port                  Peer Address:Port

LISTEN       0        128                 0.0.0.0:22                       0.0.0.0:*
SYN-RECV     0        0              XX.XX.245.77:22                   ZZ.ZZZ.173.2:30487  timer:(on,10sec,7)
FIN-WAIT-1   0        1              XX.XX.245.77:22                   ZZ.ZZZ.173.2:50289  timer:(on,17sec,6)
ESTAB        0        0              XX.XX.245.77:22                  ZZ.ZZZ.173.21:13933  timer:
(keepalive,119min,0)
LISTEN       0        128                      *:80                             *:*
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3166   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3148   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3170   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3134   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3177   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3159   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3181   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3141   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3191   timer:(on,12sec,4)
ESTAB        0        36    [2001:xxxx:xxxx:xxx:1001::2002]:22 [2001:yyy:yyy:yyyy:6::4]:39666   timer:(on,240ms,0)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3169   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3174   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3136   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3188   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3137   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3186   timer:(on,12sec,4)
SYN-RECV     0        0     [::ffff:XX.XX.245.77]:80          [::ffff:YYY.YYY.208.44]:3153   timer:(on,12sec,4)
    […]
```

TCP Attacks: *SYN ACK Reflection*

- Reflection attack that was gaining popularity in 2019
  - TCP was thought to give not enough amplification for DoS attacks
- Perceived as a SYN-ACK flood at the victims host
- Attacker TCP SYN packets to the reflector host with victim's (spoofed) IP address
  - Open port needed
- Reflector sends SYN-ACK packet to victim
- If victim does not respond, will repeat SYN-ACK several times
  - Usually 2-5 times, but vulnerable hosts will re-try up to 255 times
  - See `net.ipv4.tcp_synack_retries` (Linux)
- Targeted resource is the networks packet switching capacity (pps)

Application Layer Attacks

www.geant.org

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

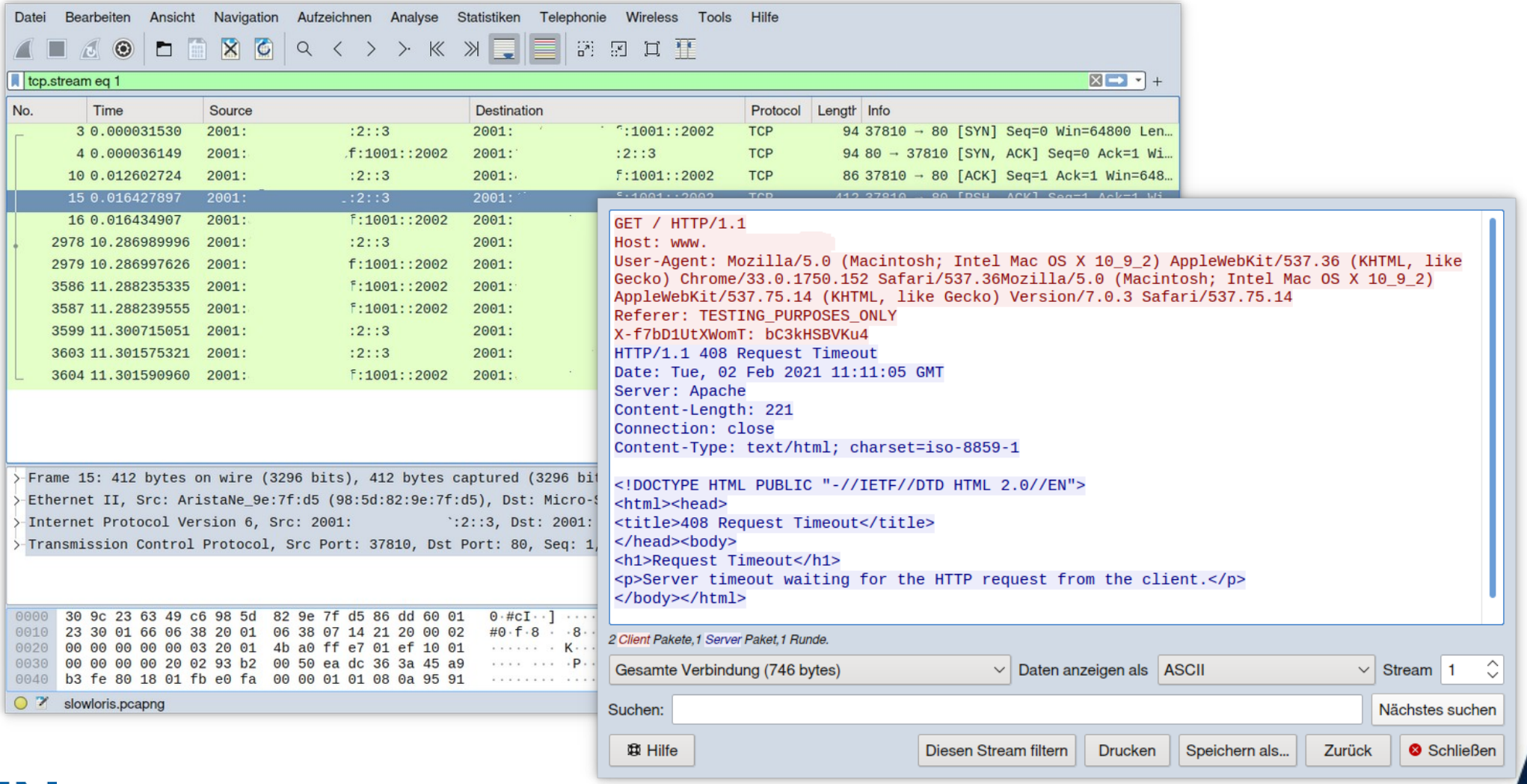
Application Attacks: *Slowloris*

- Attack directed against web servers
  - Destination port 80, 443, etc.
  - Exhausting number of open HTTP sessions
    - TCP connections
  - System will accept no new HTTP connections, hangs
- Attacker opens a number of TCP connections to a web server
- Each HTTP request is sent in small TCP packets, very slowly
  - Very low bandwidth usage
- Simulate
  - PyLoris: Supports connections through SOCKS and TOR
  - SlowHTTPTest: DDoS test tool for web servers

David Haring / Duke Lemur Center

Slow loris primates

Application Attacks: *Slowloris Packet Capture*



Application Attacks: *Slowloris* Variants/Derivates

- ***SlowDroid:*** Uses high number of spaces instead of HTTP (GET) requests
- ***SlowRead:*** Accept server response very slowly (TCP window size 0)
  - Forcing server to send response in very small packets, with a high number of seconds in between
- ***Slow HTTP Post:*** with HTTP POST requests
- ***R-U-Dead-Yet (R.U.D.Y.):*** Filling out HTTP forms very slowly (POST requests)
  - Sending chunks of the POST with a high number of seconds in between
- ***THC-SSL-DoS:*** Immediately request re-negotiation after SSL handshake
- ***#RefRef:*** Exploits SQL injection vulnerability to execute recursive SQL statements

Famous Application Attacks: *Billion Laughs*

- Targeted at XML parser code in web apps or elsewhere
- Memory exhaustion while parsing specially crafted XML files
  - CPU load as secondary effect
  - Aka XML Bomb, similar principle as in *Fork Bombs, Zip Bombs*, etc.
  - Works with other XML based formats too, like YAML

Example:

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
 <!ENTITY lol "lol">
 <!ELEMENT lolz (#PCDATA)>
 <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
 <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
 <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
 <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
 <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
 <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
 <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
 <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
 <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

Application Attacks: *ReDoS*

- Regular Expression Denial-of-Service
- Attacker sends a specially crafted message
  - ***evil RegExes*** **–** e. g. grouping with repetition, like **`([a-zA-Z]+)*`**
- Exploits weaknesses in regular expression library
- Exhausting/consume CPU on server when
  - Server regular expression is run against attacker supplied input strings
  - Server executes a regular expression supplied by the attacker

Application Attacks: *Hash Collision DoS*

- Exploiting Collisions in applications hash tables
  - Not cryptographic hash collisions
- Web application frameworks often use hash tables to index POST session parameters
  - Dealing with collisions in hash tables is computationally (much) more CPU intensive than lookups
- Attacker sends a specially crafted POST request with many specially crafted parameters
  - Built in a way that causes hash collisions
- Attack slows down responses, exhausting CPU
  - Vulnerabilities in hash-table code may also allow code execution in the application

Attack Tools: *LOIC & HOIC*

- **L**ow **O**rbit **I**on **C**annon
- Developed as a network stress test tool, modified and released as open-source by "Anonymous" group
- Flood attacks with TCP, UDP, or HTTP packets
- "Hive Mind Mode" connects to IRC channel for remote control/coordination

- **H**igh **O**rbit **I**on **C**annon
- Same group as LOIC
- Used for HTTP request floods
- "booster" (.hoic) files contain list of URLs, referrers, user agents, and HTTP headers, used randomly to avoid IPS filters
- No spoofing of source addresses

What have you learned?

- Denial-of-Service attacks come in many colours
  - And they keep changing and evolving
- How some of the most ”famous” DDoS attacks work technically
- Patching will help against vulnerability-based attacks
  - Ping-of-Death, LAND, Teardrop, etc.

- **Question 1:** If you can’t feel the impact of a DoS attack, was it real?
- **Question 2:** How can we discriminate attacks from self-inflicted Denial-of-Service?

→Next session: DDoS Detection

Thank you

Any questions?

Next course: ***DDoS Detection***

15th of February 2021

www.geant.org

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 731122 (GN4-2).

References:

- D-SCAP: DDoS Attack Traffic Generation Using Scapy Framework https://link.springer.com/chapter/10.1007%2F978-981-13-1882-5_19
- Attacks to be performed using Hping3 and Scapy – Packet Crafting

  https://hackers-factory.com/2021/01/01/attacks-to-be-performed-using-hping3-and-scapy/
- Ping-of-Death, https://insecure.org/sploits/ping-o-death.html
- CISA Alert (TA14-017A) UDP-Based Amplification Attacks: https://us-cert.cisa.gov/ncas/alerts/TA14-017A
- Shadowserver foundation network reporting, https://www.shadowserver.org/what-we-do/network-reporting/
- Netscout: A Call to ARMS: Apple Remote Management Service UDP Reflection/Amplification DDoS Attacks: https://www.netscout.com/blog/asert/call-arms-apple-remote-management-service-udp

References:

- Cloudflare cLDAP Threat Advisory: https://www.akamai.com/de/de/multimedia/documents/state-of-the-internet/cldap-threat-advisory.pdf
- Cloudflare blog about UDP reflection attacks: https://blog.cloudflare.com/reflections-on-reflections
- OWASP ReDoS page: https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS
- Russ Cox: “Regular Expression Matching in the Wild”: https://swtch.com/~rsc/regexp/regexp3.html

Tools & Packet captures

- Packet Traces (Lab): https://rickfreyconsulting.com/downloads/
- hping3 (alpha) http://www.hping.org/hping3.html
- Nping (part of nmap): https://nmap.org/nping/
- Scapy: https://scapy.net/
  - Ufonet https://github.com/epsylon/ufonet
- SlowHTTPTest: https://github.com/shekyan/slowhttptest
- PyLoris: https://sourceforge.net/projects/pyloris/
- THC-SSL-DoS: https://thehackerschoice.wordpress.com/2011/10/24/thc-ssl-dos/,
  - Kali source mirror: https://gitlab.com/kalilinux/packages/thc-ssl-dos