MAZEBOLT

Top 10 DDoS Attacks That Can Bring Down Your Business

Executive Summary

DDoS attacks have become the weapon of choice for threat actors worldwide, mainly because they are relatively simple to execute. For various reasons, stemming from ideological motives to plain greed, DDoS attackers seek to shut down organizations' activity, and sometimes escalate to ransom attacks. But as networks become more complex, DDoS attacks evolve to become more sophisticated and malicious in the damage they inflict.

Given the dramatic rise of DDoS attacks in recent years, with an incline of over 60%, many official reports predict that the average of DDoS attacks per year will surpass 16 million[1]. In addition, DDoS-as-a-service subscriptions have become very popular among threat actors, and can cost as little as $500, making it easy to launch a DDoS attack on vulnerable organizations worldwide, with little effort – and sometimes, without advanced technical experience.

The dynamic nature of cloud environments and the workflows that accompany them make it easier for threat actors to bypass protection services. Thus, DDoS perpetrators continue to launch attacks that severely impact organizations' uptime. As experts in DDoS security, we've encountered many cases throughout the years that showcase complex security and protection postures, that actually create misconfigurations that lead to vulnerabilities. These vulnerabilities eventually lead to damaging DDoS attacks. But many of these DDoS protection blind spots can be uncovered with continuous and non-disruptive testing, and once uncovered, they can be easily remediated. With just a few fixes, an organization can improve their DDoS security ten-fold.

In this eBook, we will shed light on the most common DDoS attack vectors that tend to go unnoticed, unchecked, and overlooked by security teams. We strongly recommend looking for these DDoS vulnerabilities and properly configuring the DDoS protection posture – because sometimes, it's the simplest DDoS attack vectors that bypass an organization's protection layers, wreak havoc, and take down crucial services.

1. Cyber Defense Magazine's February Edition, p 50-52.

1.Brobot: The DDoS Attack That Never Went Away

Any cybersecurity professional with a background in DDoS protection will be happy to tell you all about HTTP floods. But not many people are well acquainted with a very similar attack vector - similar, but in some cases, much more dangerous - the Brobot attack.

An HTTP Flood is one of the most well-known and "reliable" DDoS attack vectors: a layer 7 DDoS attack that targets web servers and applications. HTTP Floods are designed to overwhelm web servers' resources by continuously requesting single or multiple URLs from many source-attacking machines, which simulate HTTP clients, such as web browsers.

Designed by the Izz ad-Din al-Qassam Cyber Fighters (QCF), an Islamist hacking group that was traced back to Iran, Brobot was an uncommonly powerful botnet that was mainly used in the early 2010s to attack banks and the entire financial sector. Although the FBI issued an official warning concerning the Brobot DDoS attack vector and the botnet, many DDoS protection services are not configured to tackle this destructive attack vector nowadays.

What happens during the Brobot DDoS attack?

Brobot is similar to an HTTP Flood as it's designed to overwhelm online services' resources by constantly requesting single or multiple URLs from many source-attacking machines. Brobot dynamically changes its user's identity, as well as the HTTP method type (GET/POST). In addition, Brobot can add a suffix to the end of URLs, thus enabling the request to bypass many CDN systems. If this happens, the server will reach its concurrent connection limits and will no longer respond to legitimate requests from other users.

Like any other HTTP Flood, the Brobot DDoS attack starts with the standard TCP handshake. Once established, the Brobot will send a POST or a GET request with a random URL.

No.	Time	Source	Destination	Protocol	Length	TCP Flags	Info
1	0.000000	10.0.0.2	10.128.0.2	TCP	74	SYN	37763→80 [SYN] Seq=0 Win=28400 Len=0 MSS=1420 SACK_PERM=1 TSval=1416769 TSec
2	0.021146	10.128.0.2	10.0.0.2	TCP	74	SYN-ACK	80→37763 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=6
3	0.021188	10.0.0.2	10.128.0.2	TCP	66	ACK	37763→80 [ACK] Seq=1 Ack=1 Win=28416 Len=0 TSval=1416774 TSecr=63474759
4	0.030444	10.0.0.2	10.128.0.2	HTTP	1127	PSH-ACK	POST /AXoYQkfQPw2XoAv/BMFvHB04N-G72Em31/Qq-VWnf/_tNu/o7N?nMjEzNg72XbxnG9f=Y
5	0.051290	10.128.0.2	10.0.0.2	TCP	66	ACK	80→37763 [ACK] Seq=1 Ack=1062 Win=31104 Len=0 TSval=63474766 TSecr=1416777
6	0.051706	10.128.0.2	10.0.0.2	HTTP	515	PSH-ACK	HTTP/1.1 404 Not Found (text/html)
7	0.051726	10.0.0.2	10.128.0.2	TCP	66	ACK	37763→80 [ACK] Seq=1062 Ack=450 Win=29568 Len=0 TSval=1416782 TSecr=63474766
8	0.071003	10.0.0.2	10.128.0.2	TCP	66	FIN-ACK	37763→80 [FIN, ACK] Seq=1062 Ack=450 Win=29568 Len=0 TSval=1416787 TSecr=634
9	0.091755	10.128.0.2	10.0.0.2	TCP	66	FIN-ACK	80→37763 [FIN, ACK] Seq=450 Ack=1063 Win=31104 Len=0 TSval=63474777 TSecr=14
10	0.091780	10.0.0.2	10.128.0.2	TCP	66	ACK	37763→80 [ACK] Seq=1063 Ack=451 Win=29568 Len=0 TSval=1416792 TSecr=63474777

POST request

No.	Time	Source	Destination	Protocol	Length	TCP Flags	Info
31	0.327012	10.0.0.2	10.128.0.2	TCP	74	SYN	37767→80 [SYN] Seq=0 Win=28400 Len=0 MSS=1420 SACK_PERM=1 TSval=1416851 TSec
32	0.347790	10.128.0.2	10.0.0.2	TCP	74	SYN-ACK	80→37767 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=6
33	0.347841	10.0.0.2	10.128.0.2	TCP	66	ACK	37767→80 [ACK] Seq=1 Ack=1 Win=28416 Len=0 TSval=1416856 TSecr=63474841
34	0.357492	10.0.0.2	10.128.0.2	HTTP	508	PSH-ACK	GET /FQ/RHwSF/EwBGwYk7w7vA13?jMzE4MQb-J=ezx HTTP/1.1
35	0.378141	10.128.0.2	10.0.0.2	TCP	66	ACK	80→37767 [ACK] Seq=1 Ack=443 Win=30080 Len=0 TSval=63474848 TSecr=1416858
36	0.378344	10.128.0.2	10.0.0.2	HTTP	488	PSH-ACK	HTTP/1.1 404 Not Found (text/html)
37	0.378381	10.0.0.2	10.128.0.2	TCP	66	ACK	37767→80 [ACK] Seq=443 Ack=423 Win=29568 Len=0 TSval=1416864 TSecr=63474848
38	0.398180	10.0.0.2	10.128.0.2	TCP	66	FIN-ACK	37767→80 [FIN, ACK] Seq=443 Ack=423 Win=29568 Len=0 TSval=1416869 TSecr=6347
39	0.418681	10.128.0.2	10.0.0.2	TCP	66	FIN-ACK	80→37767 [FIN, ACK] Seq=423 Ack=444 Win=30080 Len=0 TSval=63474858 TSecr=141
40	0.418727	10.0.0.2	10.128.0.2	TCP	66	ACK	37767→80 [ACK] Seq=444 Ack=424 Win=29568 Len=0 TSval=1416874 TSecr=63474858

GET request

But unlike other HTTP Floods, the BroBot attack will randomly use one of a small number of User Agents:

```
$ua = array 'Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.1.16) Gecko/20110929 Iceweasel/3.5.16',
       'IE/5.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 1.1.4322;)',
       'GooglePocket/2.1 ( http://www.googlePocket.com/Pocket.html)',
       'msnPocket-Products/1.0 (+http://search.msn.com/msnPocket.htm)',
       'Opera/9.00 (Windows NT 5.1; U; en)','Safari/5.00 (Macintosh; U; en)',
       'DoCoMo/2.0 SH902i (compatible; Y!J-SRD/1.0; http://help.yahoo.co.jp/help/jp/search/indexing/indexing-27.html)',
       'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4b) Gecko/20030505 Mozilla Firebird/0.6');
```

The Brobot DDoS attack can also be used to attack sites with HTTPS, which will then include an SSL Handshake.

2. DNS Response Flood: Not Only for Beginners

As the number of sophisticated DDoS attacks rises, it can be easy to forget about the basic attacks. From banks and governments to software vendors and insurance companies - organizations invest a lot in their DDoS protection solutions, which are often misconfigured and exposed to the simplest DDoS attacks such as the DNS Response Flood.

DDoS attackers realize that even the best DDoS protection can be penetrated and sometimes, they can use simple attacks to bring down enterprises and shut down online services. Let's review the DNS Response Flood DDoS attack, one of the simplest and most effective DDoS attacks that is still commonly used today.

What is a DNS Response Flood?

A DNS Response Flood is a layer 7 DDoS attack that floods the target with DNS responses, originating from different attackers. The DDoS attacker aims to disrupt Domain Name System (DNS) servers by targeting one or more sub-zones. DNS servers are the "roadmap" of the Internet, aiding in the location of requested servers. A DNS zone represents a distinct portion of the domain name space within the DNS. Each zone is managed by a single server cluster. During a DNS flood attack, the DDoS attacker attempts to overwhelm a particular DNS server, or servers, with seemingly legitimate traffic. This excessive traffic exhausts server resources and damages the servers' ability to correctly route legitimate requests to resources within the zone.

A DNS Response Flood is a symmetrical DDoS attack. The objective is to overwhelm server-side resources, such as memory or CPU, by flooding them with a high volume of UDP requests. These requests are generated by scripts running on multiple compromised botnet machines.

What happens during a DNS Response Flood DDoS attack?

During a DNS Response Flood attack, the attacker generates Standard DNS query response packets, with a random record. These random records will include one of the following types: "A" for IPv4 addresses, "CNAME" (Canonical Names) which specifies a domain name that has to be queried to resolve the original DNS query or "MX" (Mail eXchange), which requests information about the mail exchange server for a specific DNS domain name.

The DDoS attacker utilizes a script that is typically executed from multiple servers. These scripts send packets with incorrect formats and spoofed IP addresses. The DNS primarily uses the User Datagram Protocol (UDP), and the DNS responses contain the query and the answers, with some answers that may contain the IP of the FQDN record in the query.

```
Questions: 1
Answer RRs: 7
Authority RRs: 0
Additional RRs: 0
Queries
  iejyo.pmepyv.com: type ANY, class IN
    Name: iejyo.pmepyv.com
    [Name Length: 16]
    [Label Count: 3]
    Type: * (A request for all records the server/cache has available) (255)
    Class: IN (0x0001)
Answers
  iejyo.pmepyv.com: type A, class IN, addr 197.218.170.195
    Name: iejyo.pmepyv.com
    Type: A (Host Address) (1)
    Class: IN (0x0001)
    Time to live: 2332800
    Data length: 4
    Address: 197.218.170.195 (197.218.170.195)
  iejyo.pmepyv.com: type SOA, class IN, mname ns1.whois.com
    Name: iejyo.pmepyv.com
    Type: SOA (Start Of a zone of Authority) (6)
    Class: IN (0x0001)
    Time to live: 2332800
    Data length: 57
    Primary name server: ns1.whois.com
    Responsible authority's mailbox: agoghujsxwc.gkzlgon
    Serial Number: 2015032905
    Refresh Interval: 7200 (2 hours)
    Retry Interval: 7200 (2 hours)
    Expire limit: 172800 (2 days)
    Minimum TTL: 38400 (10 hours, 40 minutes)
```

In layer 7 attacks like DNS floods, the effectiveness does not rely on receiving a response. Therefore, the DDoS attacker can send packets that are neither accurate nor properly formatted.

3. DDoS Reflection Attacks: The Perfect Opportunity to Get Proactive

The effect of a sophisticated DDoS attack on any organization can be devastating and cause a significant threat to business activity. There are two specific types of Reflection DDoS attacks that have been successful and harmful to different organizations recently, that any security team must take into consideration. But first, we must understand the difference between Amplification and Reflection DDoS attacks.

An Amplification attack overwhelms the target by exploiting vulnerabilities in various internet protocols. Amplification attack work by sending a relatively small number of requests that are designed to trigger large responses from intermediary systems. In turn, they amplify the volume of traffic directed towards the target rendering it inaccessible.

A Reflection attack exploits the functionality of certain internet protocols to bounce traffic towards a target. In this attack, the attacker spoofs their source IP address and sends requests to vulnerable servers or devices, which then responds to the target (the spoofed IP) with larger volumes of traffic than the original request.

A Reflection attack relies on two main components: reflection and amplification:

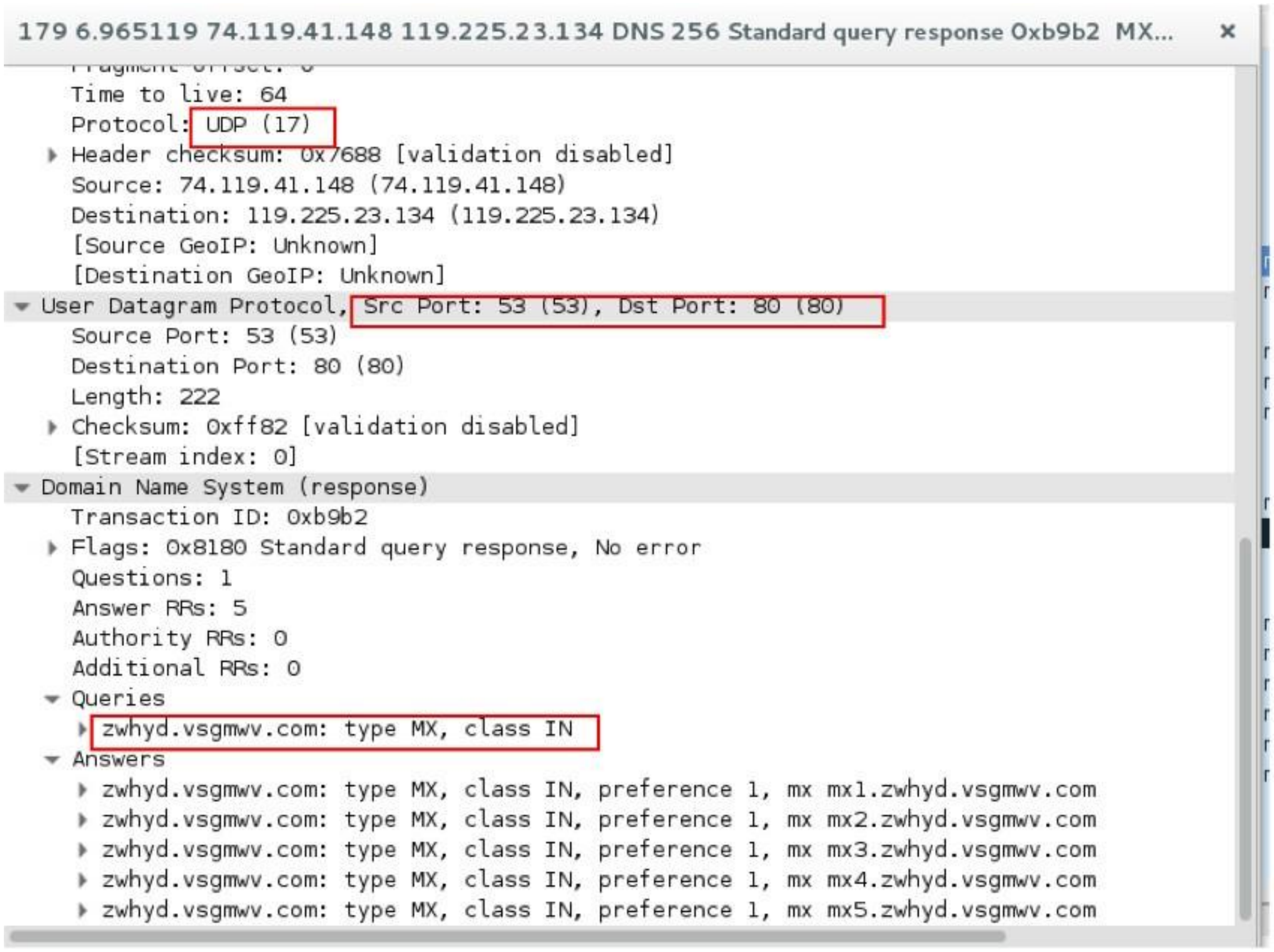
- Reflection: The attacker sends requests to a third-party server, known as a reflector. These reflectors are either misconfigured or have been purposely placed to bypass censorship and will respond to requests from any source IP address. The attacker spoofs the source IP address to make it appear as if the requests originate from the target they wish to attack.

- Amplification: The "reflector" responds to requests with much larger responses, amplifying the volume of traffic directed towards the target. This amplification effect occurs when the response generated by the reflector is significantly larger than the size of the initial request.

The DNS Response Flood

A DNS Response Flood is a layer 7 attack that targets Domain Name System (DNS) infrastructure and floods it with DNS responses from different attackers. This attack aims to overwhelm DNS servers by flooding them with an enormous volume of DNS response packets, thereby disrupting the DNS resolution process and rendering the targeted domain or network unavailable.

During the attack, an attacker generates standard DNS query response packets with a random record from one of the following types: "A" IPv4 addresses, "CNAME" that specifies a domain name that has to be queried in order to resolve the original DNS query, and “MX” to request information about the mail exchange server for a specific DNS domain name.



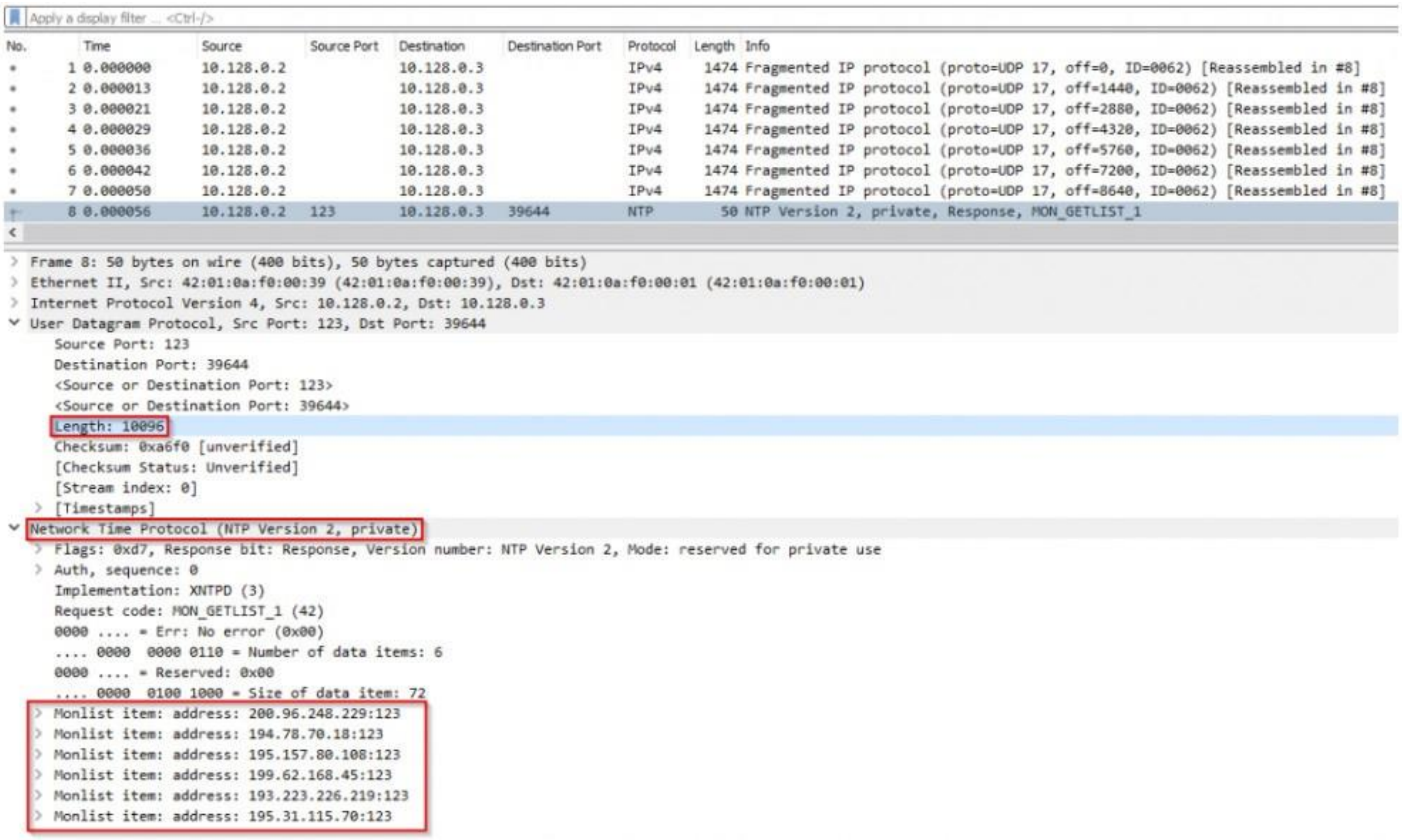
DNS Response Packet Structure

In a DNS Response flood, the attacker will generate multiple DNS responses for random records, and the target will respond with an ICMP error message stating that its destination port is unreachable. The response might also include the IP of the FQDN record in the query.

NTP Monlist Amplification Reflection Flood

An amplification reflection attack is an attack vector that allows DDoS attackers to magnify the amount of malicious traffic they generate but also to obscure the sources of the attack traffic. Such an attack vector is the Network Time Protocol (NTP) Monlist Amplification attack, which tries to saturate bandwidth to disrupt services.

The NTP Monlist Amplification Reflection flood uses publicly accessible NTP servers to overwhelm a victim's services with NTP (UDP-based) traffic. By sending a rapid succession of NTP requests datagrams with spoofed source IP to an NTP server. Thus, the attack vector makes the target reply with large NTP response datagrams to the spoofed IP address, which is, of course, the DDoS attack target. This flood contains a lot of NTP information in the data section of the NTP response datagrams, turning this attack into an Amplified Reflection DDoS attack. When the targeted server's bandwidth is overwhelmed, legitimate traffic cannot reach its destination, thus causing a denial of service.

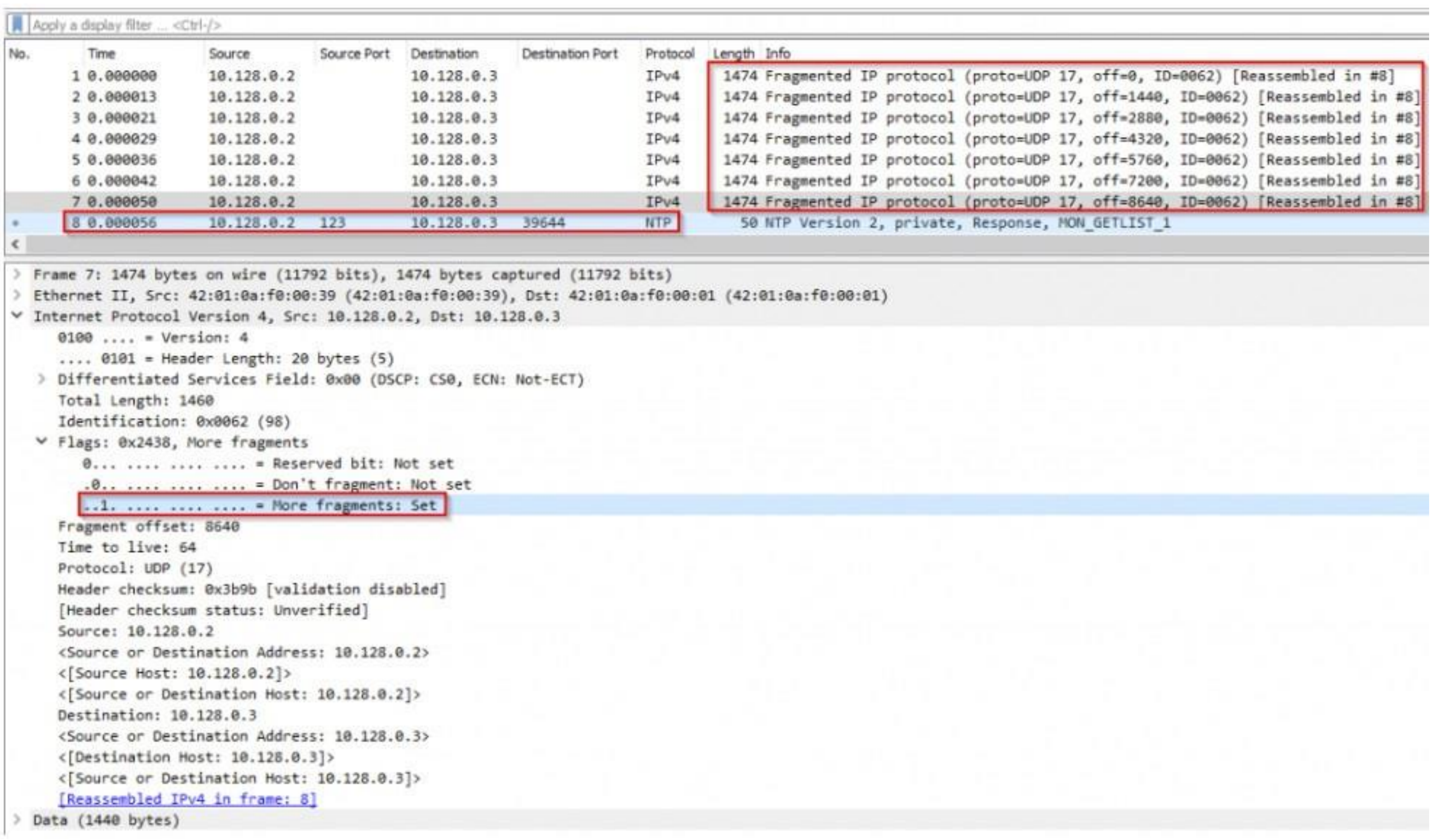


NTP Monlist Amplification Reflection Flood – Packet data section

Because the NTP Monlist Amplification Reflection Floods use standard NTP responses, it will be quite challenging to differentiate malicious traffic from valid traffic. As the NTP response datagram size is too high, network components fragment them into smaller packets that will be reassembled back to the original NTP response datagram on the attacked target.

Multiple fragmented packets will be sent to the target destination, with the “More Fragments” flag set - but the size of the fragmented packet tends to be almost equal to the Maximum Transmission Unit (MTU).

It is important to remember that packet sizes and traffic rates will vary from attack to attack, but the basic principle of the NTP Monlist Amplification Reflection Flood will always remain the same - overwhelming the target with packets while concealing the origin.



NTP Monlist Amplification Reflection Flood – single response packet

4. SSL Decryption is DDoS-Vulnerable

Due to the growing need for online services, for practically every organization in any field and industry, enterprises are constantly trying to minimize risk and protect their services. Among the many strategies and technologies incorporated into protecting online services, SSL is a basic and essential security measure.

Secure Sockets Layer (SSL) is a cryptographic protocol that controls encryption and transmission of data between two points. Sometimes referred to as SSL Visibility, SSL Decryption decrypts traffic and routes it to various inspection tools to identify threats –targeting both inbound and outbound applications from users to the internet. Implementing SSL decryption is a common and useful tool for security teams to protect end users, customers, and the organization's data safe. Using SSL decryption will prevent data breaches, monitor outgoing information from within the organization, meet regulatory compliance requirements, and support a multi-layered security protocol.

Is SSL Decryption DDoS-Vulnerable?

Security teams that use SSL decryption must take into account that there are several common DDoS vulnerabilities that are a common gateway for two major DDoS attack vectors: the THC-SSL Attack and the SSL Negotiation/ Re-Negotiation Attack.

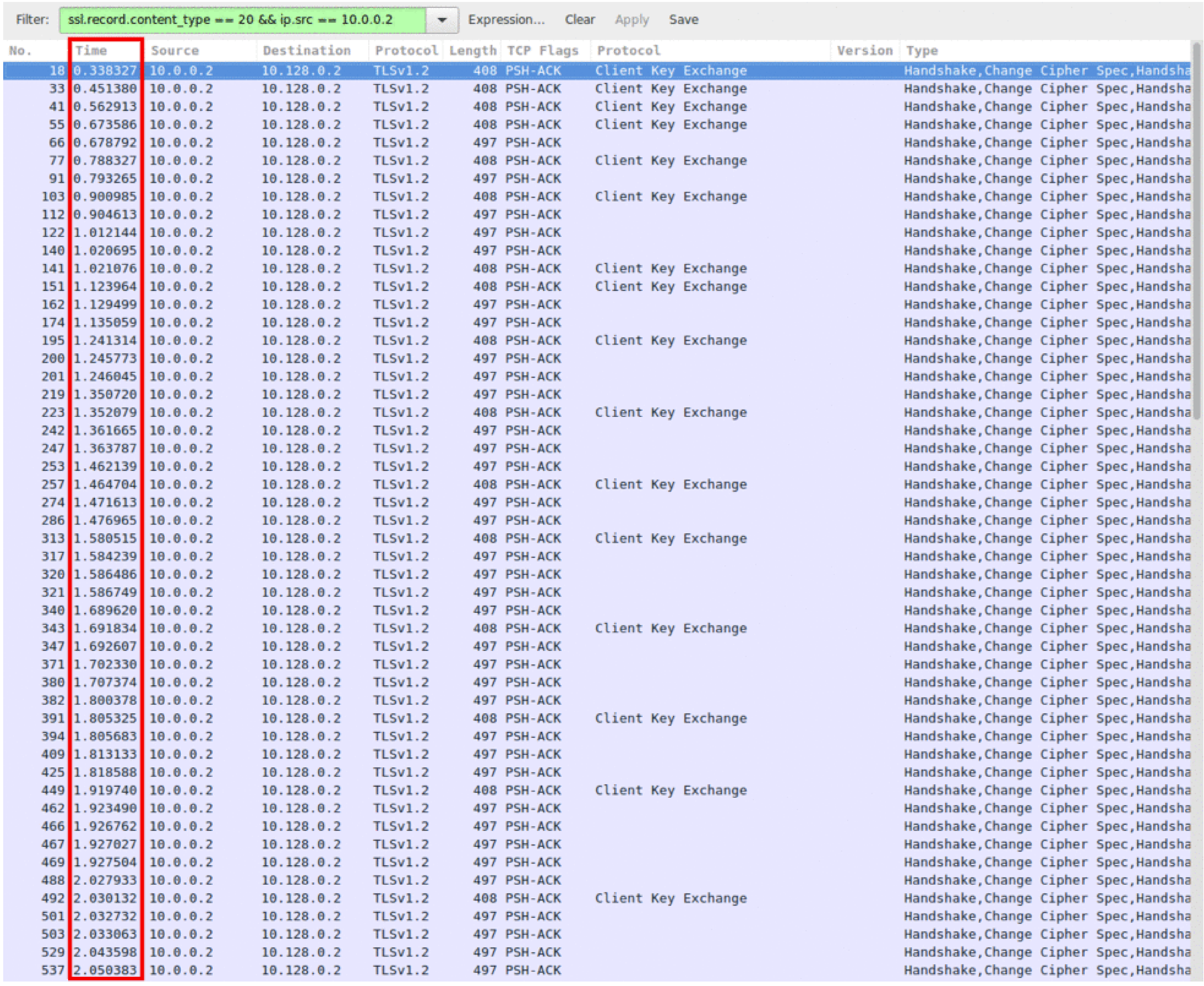
THC-SSL Attack

The THC-SSL attack uses a single TCP connection to constantly renegotiate new encryption keys. What makes this attack vector unique and malicious is that with one single connection, the server "allows" the client to request a new SSL handshake in the same TCP connection. The THC-SSL attack will work effectively on a server, which will allow the clients to initiate a new handshake at the time of their choosing - but leaving such behavior in the server is considered a vulnerability to DDoS attacks.

First, the attacker initiates a connection to the server using the TCP handshake. Once established, the attacker will begin the attack. If the server hasn't disabled client-initiated cipher renegotiation, the attacker will request a cipher spec change. The server will then compute what is required for the cipher spec change and send the data to the client – but the client is actually the attacker.

As soon as the server is done, the attacker will request another cipher spec change and will continue to do so, at a high rate. The PCAP will be filtered for the attacker's single source IP and for the SSL content type that matches the renegotiation request. The THC-SSL attack is such a simple yet devastating attack, that a single computer can take down a web server because the DDoS attacker gains a direct route to the victim's CPU.

The computations required for the renegotiation are expensive, and the attacker can trigger those computations with a single PSH-ACK packet, without ever needing to initiate a new TCP or SSL connection.



Filter: ssl.record.content_type == 20 && ip.src == 10.0.0.2 Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	TCP Flags	Protocol	Version	Type
18	0.338327	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
33	0.451380	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
41	0.562913	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
55	0.673586	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
66	0.678792	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
77	0.788327	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
91	0.793265	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
103	0.900985	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
112	0.904613	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
122	1.012144	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
140	1.020695	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
141	1.021076	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
151	1.123964	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
162	1.129499	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
174	1.135059	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
195	1.241314	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
200	1.245773	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
201	1.246045	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
219	1.350720	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
223	1.352079	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
242	1.361665	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
247	1.363787	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
253	1.462139	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
257	1.464704	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
274	1.471613	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
286	1.476965	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
313	1.580515	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
317	1.584239	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
320	1.586486	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
321	1.586749	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
340	1.689620	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
343	1.691834	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
347	1.692607	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
371	1.702330	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
380	1.707374	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
382	1.800378	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
391	1.805325	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
394	1.805683	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
409	1.813133	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
425	1.818588	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
449	1.919740	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
462	1.923490	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
466	1.926762	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
467	1.927027	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
469	1.927504	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
488	2.027933	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
492	2.030132	10.0.0.2	10.128.0.2	TLSv1.2	408	PSH-ACK	Client Key Exchange		Handshake,Change Cipher Spec,Handsha
501	2.032732	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
503	2.033063	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
529	2.043598	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha
537	2.050383	10.0.0.2	10.128.0.2	TLSv1.2	497	PSH-ACK			Handshake,Change Cipher Spec,Handsha

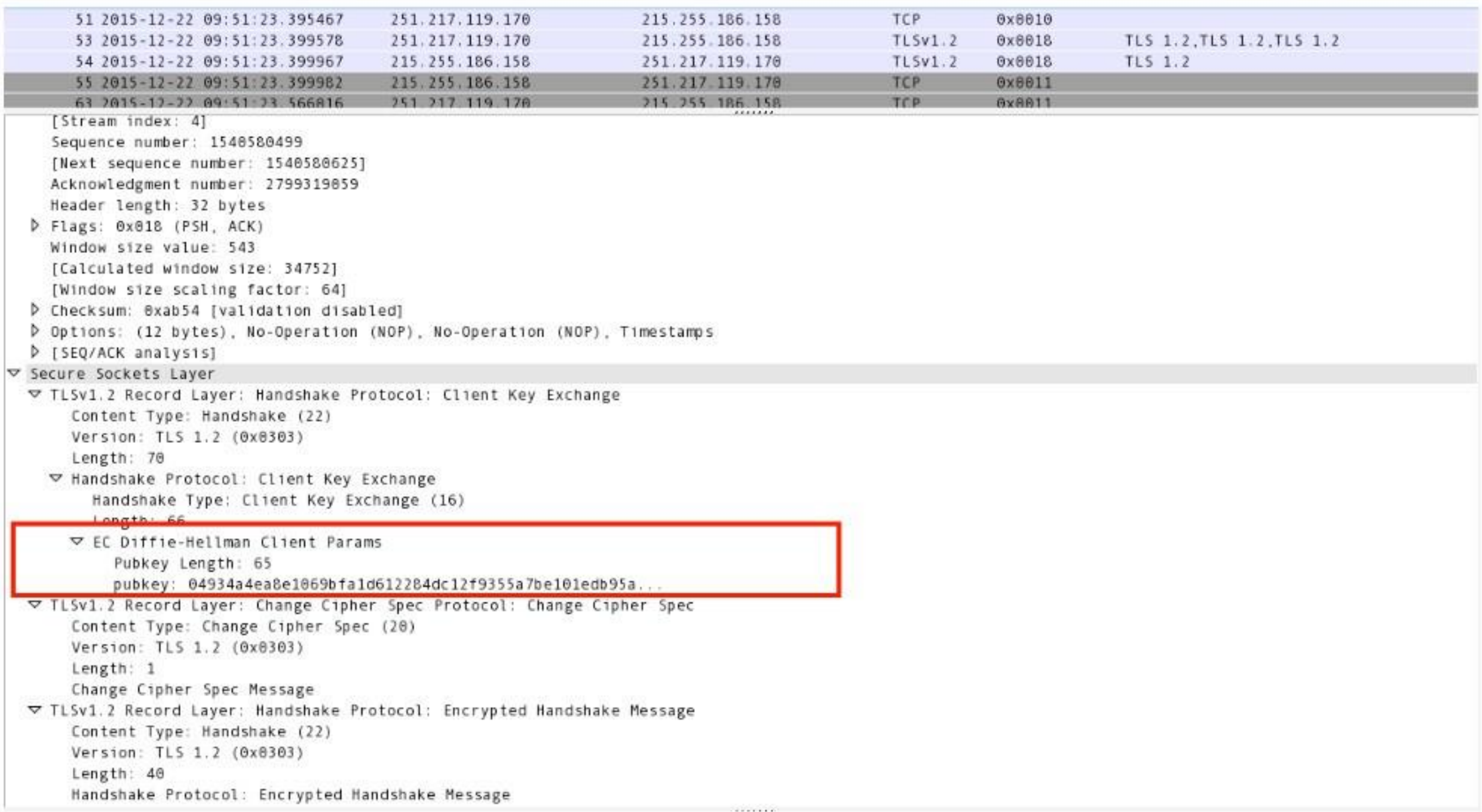
THC-SSL - Constant renegotiation

SSL Negotiation/Re-Negotiation Attack

The SSL Negotiation attack is a DDoS attack that attempts to establish many new SSL handshakes with the targeted server. Each handshake is a new TCP connection that affects the target server by opening and closing many such connections.

SSL/TLS handshakes can get up to 15 times more CPU-intensive on the server than on the client, so while the server may not be down following this attack, it may be unable to establish any new SSL connections, effectively leaving that SSL service unavailable. It is important to note that technically, this attack may be referred to as a layer 6 attack and not layer 7. Following the initial TCP handshake, the server will respond with a “server Hello” packet which contains the Cipher Suite chosen by the server from the list of cipher suites supported by the client, and also the session ID, as well as a random string. Next, client key exchange will take place, using the server's public key.

Once done with the key exchange, all messages passed subsequently will be encrypted. Each key exchange takes about 15 times more computing power on the server, due to it needing to handle the new client SSL handshake initiation. Another reason is that a new TCP session is needed for all the SSL daemon server side. Thus, the attack saturates both the server’s CPU and the TCP session table.



SSL Client key exchange

5. ACK-SYN Flood: The Most Intriguing DDoS Attack of Them All

According to various reports from recent years[2], global organizations are expected to increase their spending on DDoS protection from $3.8 billion in 2022 to $13 billion in 2032, due to the increase in DDoS attacks. The DDoS threat has become more sophisticated and complex, for example, a common DDoS attack vector like the ACK-SYN Flood can be used as a smoke screen or in combination with other attack vectors.

But it's not only multi-vector attacks that cause chaos in organizations' online services. It only takes one unknown DDoS vulnerability that allows the simplest DDoS attack to penetrate the organization's protection layers, for the company's services to be disrupted - and the damages to start piling up. This simple and common attack can be the ACK-SYN Flood.

What is the ACK-SYN Flood?

The ACK-SYN flood is a DDoS attack designed to disrupt inline services' activity by saturating bandwidth and resources on stateful devices in its path. Usually, a host server responds to incoming SYN requests by generating SYN-ACK packets. However, during the ACK-SYN flood attack, the targeted host server is bombarded with a large number of fake SYN-ACK packets.

In an attempt to handle this influx of fake packets, the attacked server expends additional computing power (such as RAM and CPU) to evaluate each SYN-ACK packet and compare it with the existing connection table entries. But these actions will overload the target server, which will result in unavailability and possibly a fail-open mode, similar to the effects of the SYN-Flood attack.

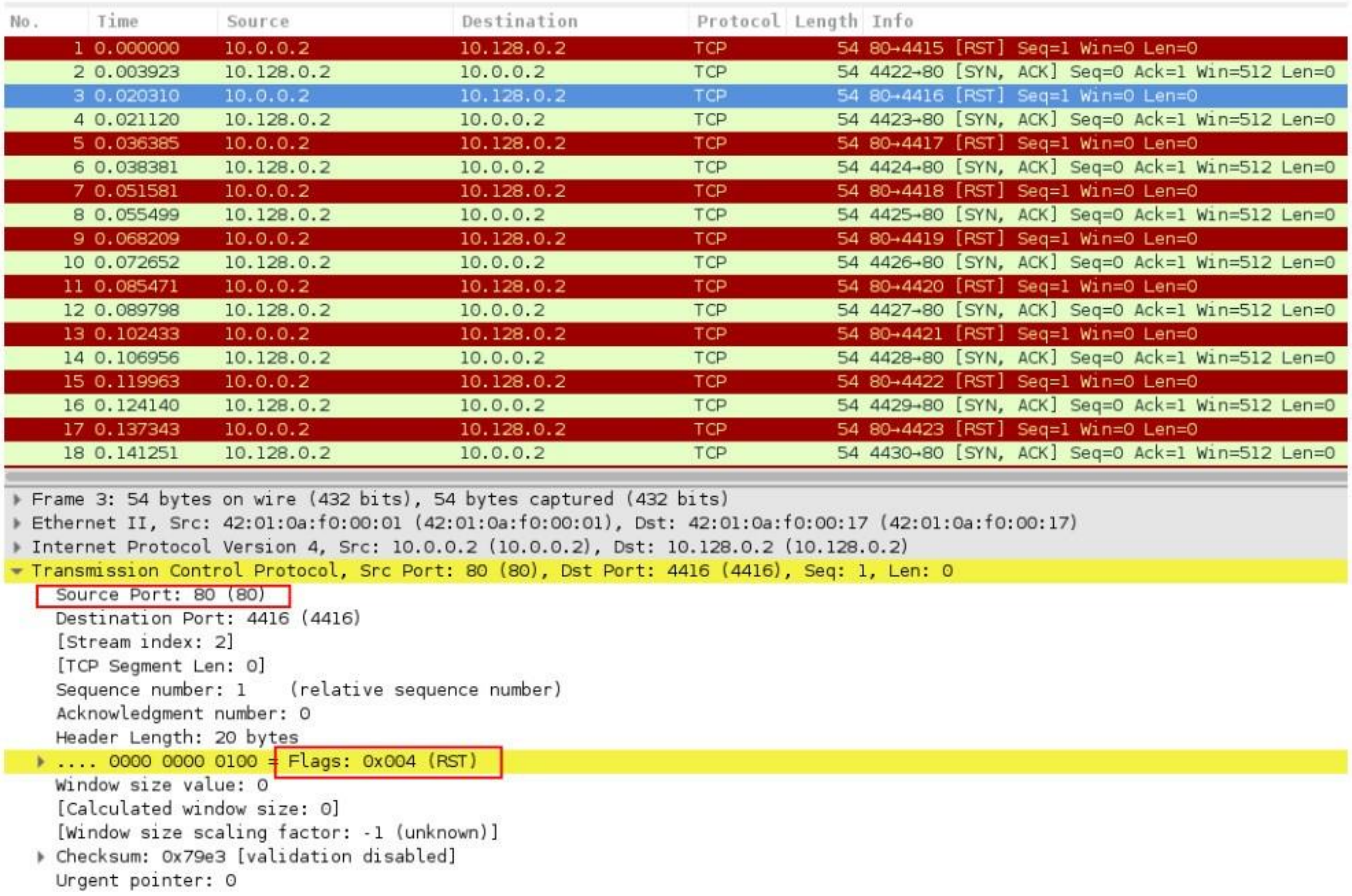
2. The DDoS Protection and Mitigation Global Market Report, 2023

What happens during an ACK-SYN Flood DDoS attack?

During an ACK-SYN Flood DDoS attack, a high rate of ACK-SYN packets will be sent from a single source IP toward a single destination IP. In many cases, the target will respond with an RST packet because the TCP stack receiving the ACK-SYN packet might not have a corresponding sequence of SYN – SYN+ACK +ACK, which is basically the TCP handshake.

Some environments may opt not to send an RST packet back to the source of the attacker's ACK-SYN packet because the ACK-SYN packet is known as an out-of-state packet. A typical ACK-SYN Flood targeting an unsuspecting host will most likely have a high rate of ACK-SYN packets (not preceded by a TCP handshake) and a slightly lesser rate of RST packets coming from the targeted server.

The overwhelmed target server will not be able to sustain the computing activity, and the result will be a partial or full unavailability of services.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.2	10.128.0.2	TCP	54	80→4415 [RST] Seq=1 Win=0 Len=0
2	0.003923	10.128.0.2	10.0.0.2	TCP	54	4422→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0
3	0.020310	10.0.0.2	10.128.0.2	TCP	54	80→4416 [RST] Seq=1 Win=0 Len=0
4	0.021120	10.128.0.2	10.0.0.2	TCP	54	4423→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0
5	0.036385	10.0.0.2	10.128.0.2	TCP	54	80→4417 [RST] Seq=1 Win=0 Len=0
6	0.038381	10.128.0.2	10.0.0.2	TCP	54	4424→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0
7	0.051581	10.0.0.2	10.128.0.2	TCP	54	80→4418 [RST] Seq=1 Win=0 Len=0
8	0.055499	10.128.0.2	10.0.0.2	TCP	54	4425→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0
9	0.068209	10.0.0.2	10.128.0.2	TCP	54	80→4419 [RST] Seq=1 Win=0 Len=0
10	0.072652	10.128.0.2	10.0.0.2	TCP	54	4426→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0
11	0.085471	10.0.0.2	10.128.0.2	TCP	54	80→4420 [RST] Seq=1 Win=0 Len=0
12	0.089798	10.128.0.2	10.0.0.2	TCP	54	4427→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0
13	0.102433	10.0.0.2	10.128.0.2	TCP	54	80→4421 [RST] Seq=1 Win=0 Len=0
14	0.106956	10.128.0.2	10.0.0.2	TCP	54	4428→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0
15	0.119963	10.0.0.2	10.128.0.2	TCP	54	80→4422 [RST] Seq=1 Win=0 Len=0
16	0.124140	10.128.0.2	10.0.0.2	TCP	54	4429→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0
17	0.137343	10.0.0.2	10.128.0.2	TCP	54	80→4423 [RST] Seq=1 Win=0 Len=0
18	0.141251	10.128.0.2	10.0.0.2	TCP	54	4430→80 [SYN, ACK] Seq=0 Ack=1 Win=512 Len=0

```
Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
Ethernet II, Src: 42:01:0a:f0:00:01 (42:01:0a:f0:00:01), Dst: 42:01:0a:f0:00:17 (42:01:0a:f0:00:17)
Internet Protocol Version 4, Src: 10.0.0.2 (10.0.0.2), Dst: 10.128.0.2 (10.128.0.2)
Transmission Control Protocol, Src Port: 80 (80), Dst Port: 4416 (4416), Seq: 1, Len: 0
  Source Port: 80 (80)
  Destination Port: 4416 (4416)
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 1    (relative sequence number)
  Acknowledgment number: 0
  Header Length: 20 bytes
  .... 0000 0000 0100 = Flags: 0x004 (RST)
  Window size value: 0
  [Calculated window size: 0]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x79e3 [validation disabled]
  Urgent pointer: 0
```

RST packet received because of "out of state" ACK-SYN packet sent

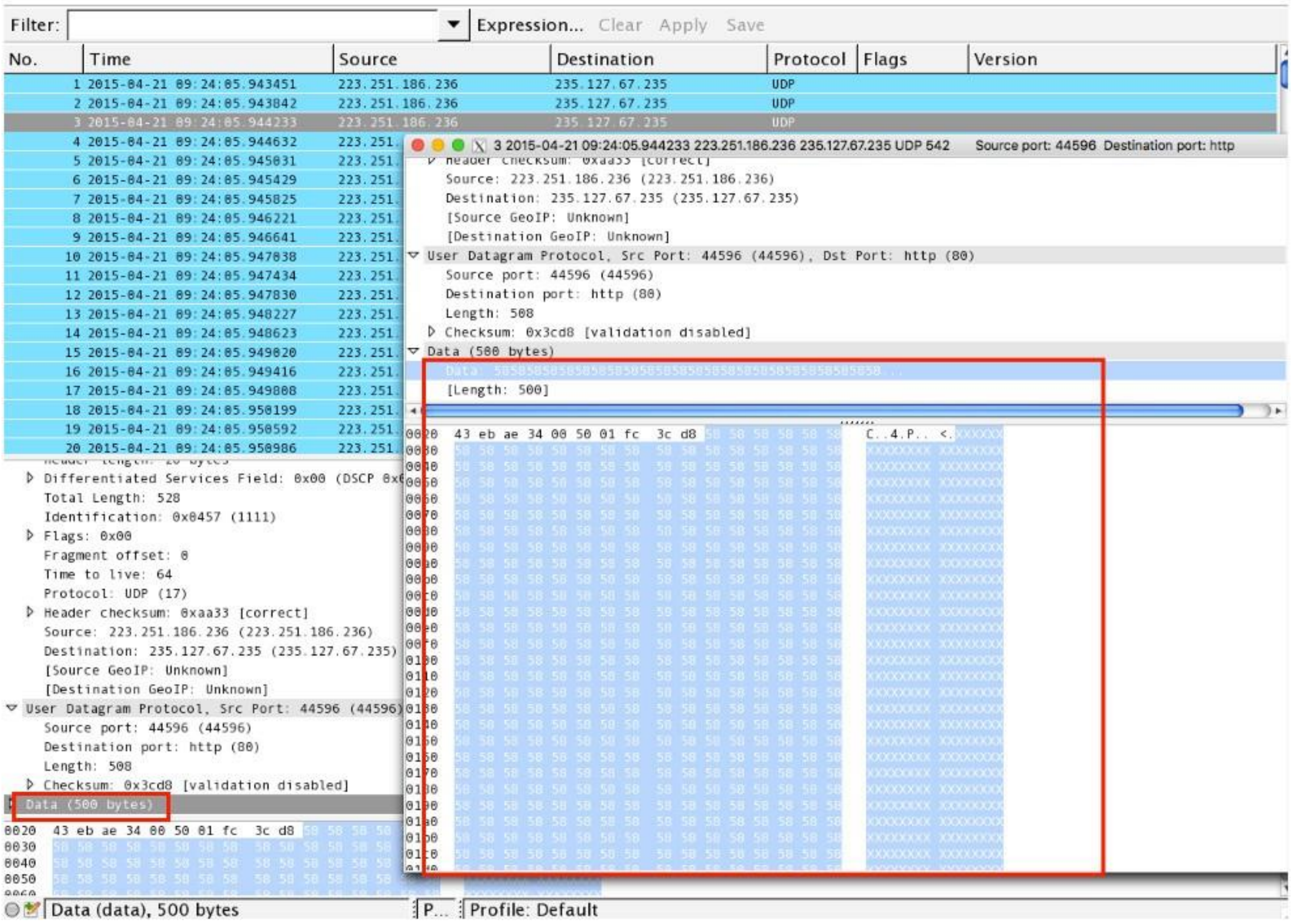
6. A New DDoS Botnet with an Old-School Approach

Early 2023 saw several major DDoS attacks targeting gaming companies, online streaming services, game server hosting providers, and gaming community members. These attacks were executed using a new botnet, Dark Frost, which contains Mirai, QBot, and Gafgyt malware source code.

The Dark Frost botnet was used in a number of malicious and sophisticated DDoS attacks, mainly targeting the gaming sector, but with a clear intention of its user (the DDoS attacker) to prove the botnet's capabilities, and perhaps set the foundation to become a DDoS-for-hire provider. Whatever the reason for the attacks may be - it is clear that these attacks are not something to be ignored: the Dark Frost botnet has the capability to launch UDP flood attacks, and the attacker published live recordings of the attacks, clearly in an attempt to build themselves as a major threat.

What is a UDP Flood?
A UDP flood tries to saturate bandwidth in order to disrupt online services, in a very "reliable" manner for DDoS attackers. This DDoS attack vector is usually done by sending a rapid succession of UDP datagrams with spoofed IPs to a target server using various ports, forcing the server to respond with ICMP traffic.

The saturation of bandwidth happens both in the ingress and the egress direction. This flood has some garbage in the data section of the datagram. In a UDP flood attack, the UDP packets will be sent to a port in the destination target IP. In most cases, the UDP will be sent to an unusual port, which will be the first sign of a UDP flood attack. In the data section of the packet, security teams will notice the "XXXXXXXXXXXX" - bytes of "garbage", which is further evidence the packet is an attacking packet. In addition, in a UDP flood attack, the PPS (Packet Per Second) rate will be higher than expected, which is another indication of an attack.



UDP Flood Data section of packet

UDP Flood is a high-volume flood due to the size of packets that will be generated per attacking machine. However, it is relatively easy to detect, as this attack vector stands out in normal online services communications.

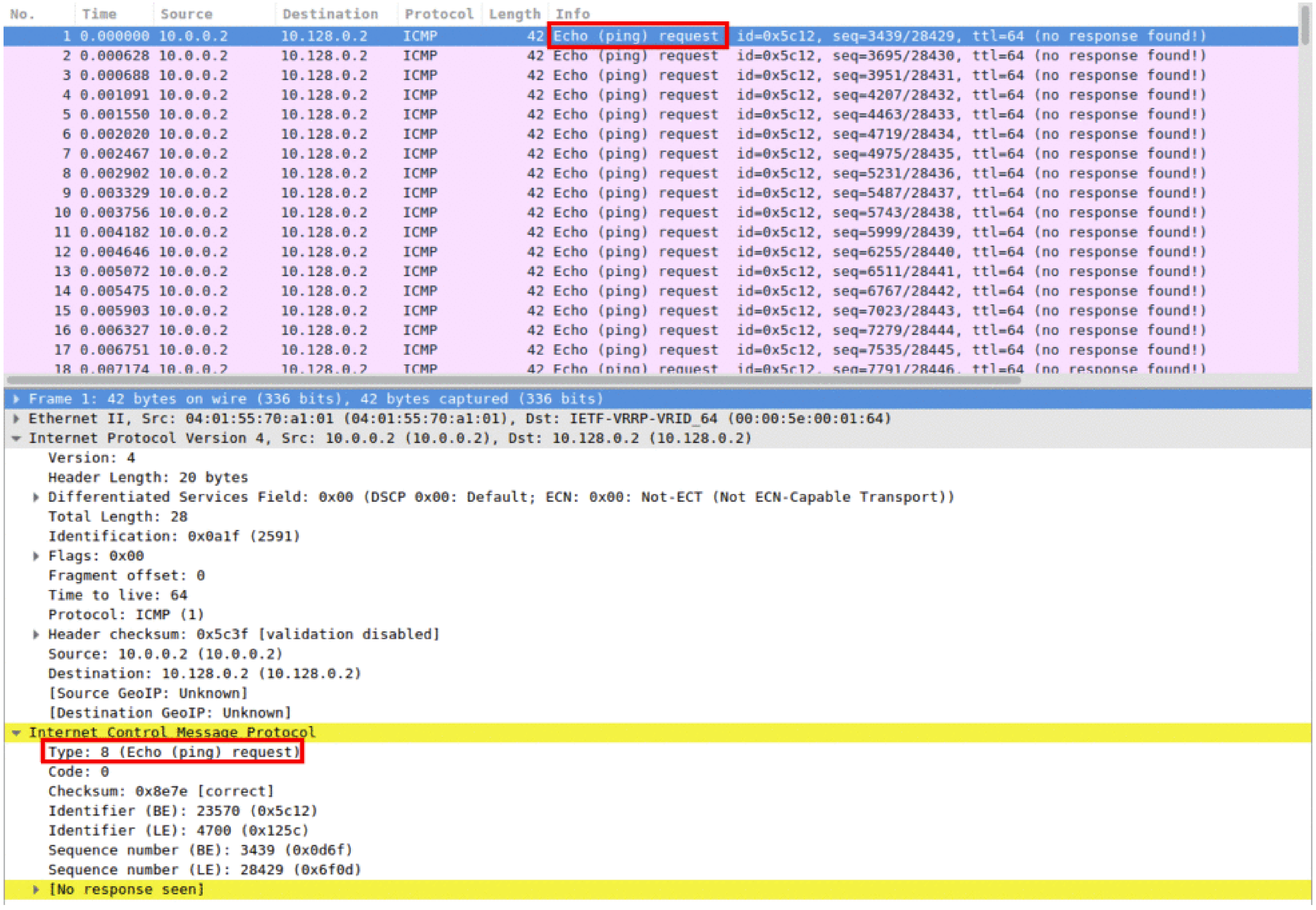
7. ICMP Ping Flood: Basic, But Critical

Recent years have shown a dramatic incline of DDoS attacks on virtually every vertical and industry. From financial institutions and governments to gaming, organizations are constantly targeted, and the number of successful DDoS attacks that result in severe downtime is alarming. DDoS attackers understand that even the best DDoS protection can be bypassed relatively easily - and sometimes, they use basic attack vectors to accomplish their malicious goals. One of these basic attack vectors is the ICMP Ping)Type 8) Flood.

What is the ICMP Ping Flood?

ICMP Ping Floods are DDoS attacks that consume computing power and saturate bandwidth. They are generally spoofed attacks that are used at a high rate, but more specifically, ICMP Ping Floods are echoed requests that may elicit echo responses such as ICMP Type 0. If not mitigated easily by on-site DDoS protection devices, ICMP Ping Floods may overwhelm the internal network architecture.

ICMP Ping Floods can also generate outgoing traffic because servers are answering the echo request, thus, they are commonly used as a basic yet effective flood to shut down on-premises devices or saturate bandwidth. It is important to remember that because ICMP Ping Floods, sometimes referred to as “Ping Flood Attacks,” are a basic tool in an attacker’s toolbox, they have fallen out of favor as a major DDoS attack vector. Nevertheless, they can be used alongside other attack vectors to create complex DDoS attacks that are more difficult to mitigate, i.e., multi-vector attacks.



ICMP Ping Flood (type 8)

What happens during an ICMP Flood?

ICMP Ping (Type 8) consists of a high volume of ICMP Echo packets. These packets have a source IP which is normally spoofed to reduce the effect of the IP reputation mechanism, and the destination IP of the victim. The echo replies are sent back to the original requesting source IP with the same number of reply packets. Generating large volumes of attack traffic, the attacker will consume all available bandwidth on the targeted device, thus making it inaccessible to normal and legitimate traffic. In addition, any network devices that are connected to that targeted endpoint will also be overwhelmed.

But many DDoS protection services and devices are currently not configured to notice such a basic attack because it is considered too simple and “old fashion”. Yet, the reality is that sometimes, it’s the basic attacks that will penetrate security layers. Thus, even if an organization is using top-of-the- line DDoS protection services, it must perform continuous DDoS tests to gain full visibility into its online services. Having full visibility allows the security and network teams to mitigate and decrease vulnerabilities, and with that, reduce and eliminate successful DDoS attacks.

8. Cloudscraper HTTP/S-GET Flood: The Most Vulnerable HTTPS Vector

In recent years, DDoS attacks have become one of the weapons of choice for threat actors who wish to wreak havoc on leading organizations' online services. DDoS attacks are a simple yet highly effective tool for any attacker who wants to disrupt and deny availability. Due to various DDoS attack vectors, such as the Cloudscraper HTTP/S-GET Flood, these attacks often succeed because traditional DDoS protection is not regularly updated with evolving attack vectors.

When it comes to DDoS security, organizations lack the necessary visibility into their online services. With adding misconfigurations, such organizations operate under a false sense of security, which leaves them exposed to successful DDoS attacks, resulting in losses and damages.

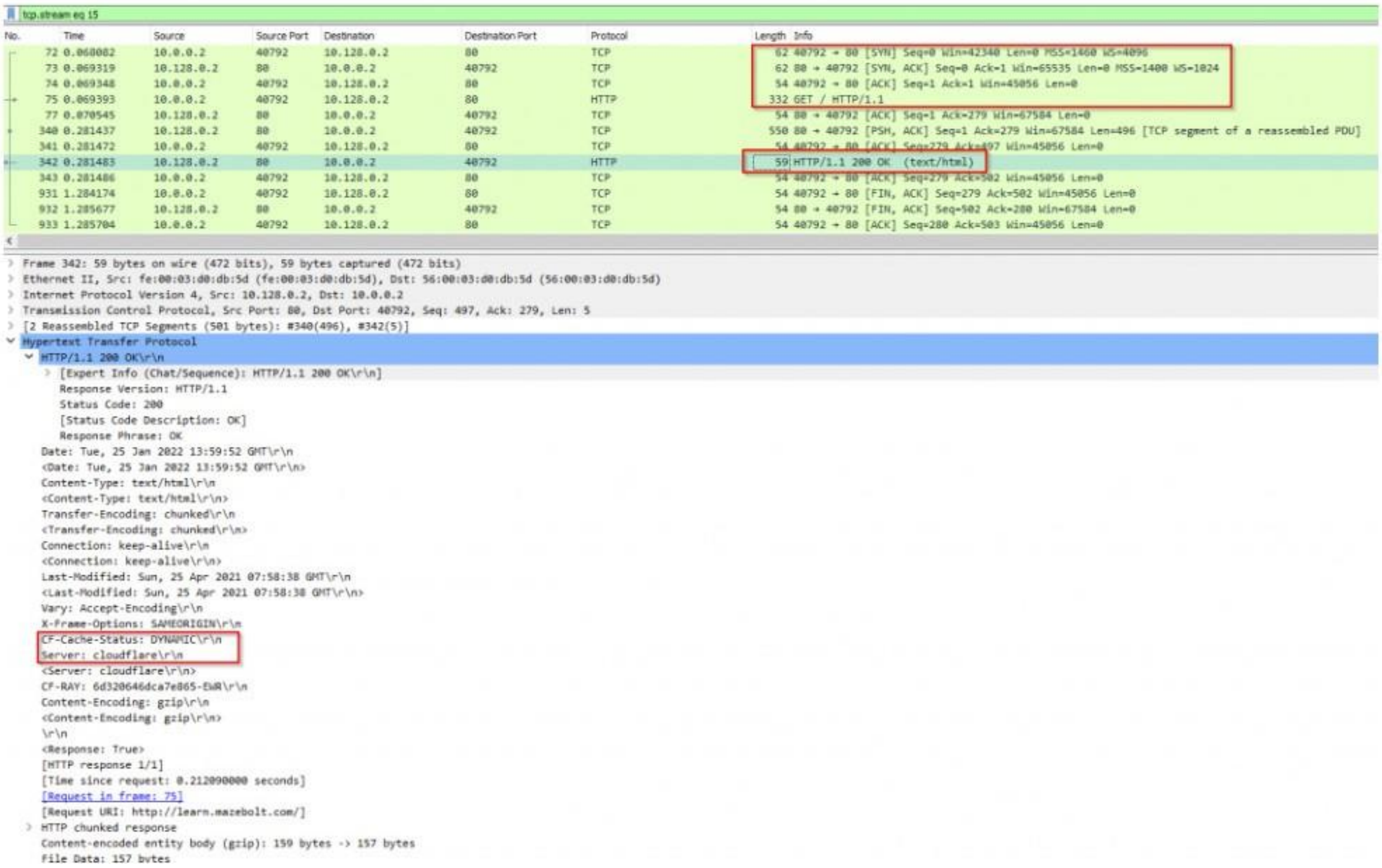
What is the Cloudscraper HTTP/S-GET Flood?

The Cloudscraper HTTP/S-GET Flood attack is one of the most dangerous HTTPS DDoS attacks today because it is sophisticated enough to bypass many layer 7 protection protocols. The Cloudscraper HTTP/S-GET Flood attack is an HTTP flood designed to overwhelm web servers' resources by continuously requesting a chosen URL from many attacking sources. This DDoS attack vector will be explained in this article in its HTTP nature, but it can also be used by attackers over HTTPS by encapsulating its packets with a secure protocol such as SSL/TLS.

The Cloudscraper HTTP-GET Flood sends HTTP GET requests to online web services. It will bypass the CDN's anti-bot protections by implementing multiple different parameters inside the HTTP packets of each request. In addition, the attack vector is able to successfully pass web-based challenges, such as Captcha. This makes the CDN service deliver HTTP requests to the back-end origin server, and when the server reaches its limits of concurrent connections, it will no longer respond to legitimate requests from other users - thus, creating a service disruption.

What happens during a Cloudscraper HTTP/S-GET Flood attack?
As in many DDoS attacks, the first step is the TCP handshake. Before requesting a web resource with an HTTP GET request, the TCP connection between the client and the server is established, using a 3-Way Handshake (SYN, SYN- ACK, ACK).

Once a TCP connection is established between the client and the server, an HTTP GET request will be transported inside a PSH/ACK packet from the client to the server, for example, a CDN-protected online web service. Multiple HTTP GET request packets will be sent by the DDoS attacker to the server, with the attacker opening a single TCP-based connection for each HTTP GET request.



HTTP session exchange using Cloudscraper HTTP-GET flood:
Attacker is IP 10.0.0.2

Due to its nature, the server takes time to respond back to each HTTP GET request, but the DDoS attacker will continue to flood it with more and more HTTP GET requests, until the server will no longer be able to keep up with the request attempts, at which point the attack is successful.

9. HTTP/s Flood with Browser Emulation: Your Worst DDoS Nightmare

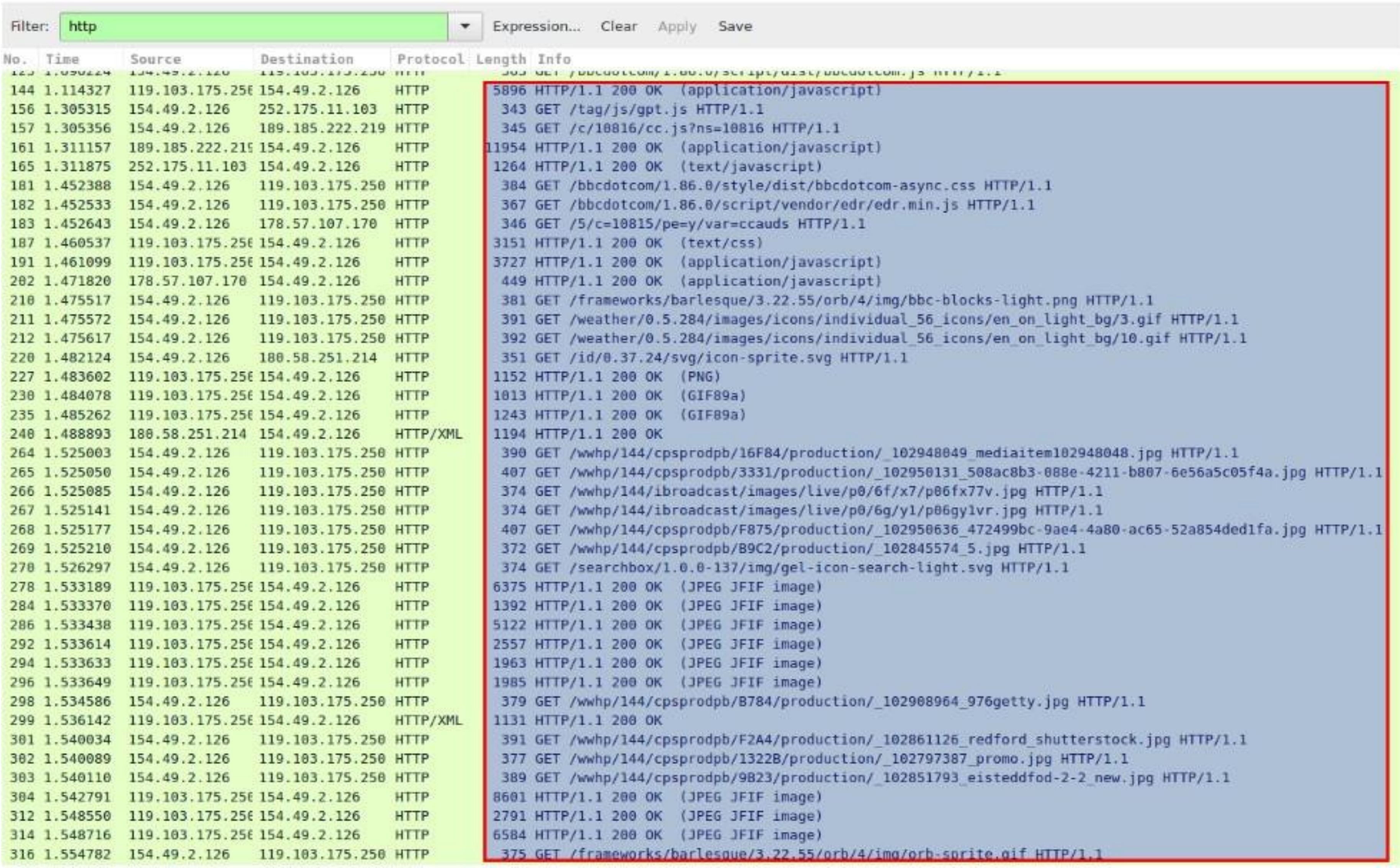
Browser Emulation mimics the functionality of popular web browsers available in the market. For example, an Internet Explorer emulator would emulate the look, feel and behavior of a real Internet Explorer. But browser emulation is also used for DDoS attacks and is one of the most difficult to detect attack vectors a security team can encounter.

Browser emulators are mainly used in cross-browser testing when developers and QA teams don't want or can't install real browsers. They will then install a browser emulator software package. While delivering close results like their original, browser emulators can't render a web page exactly like the original native browser. But that is not the case with the HTTP/s Flood with Browser Emulation DDoS attack.

What is an HTTP/s Flood with Browser Emulation?
HTTP/s Flood with Browser Emulation is a layer 7 DDoS attack that targets web servers and applications. The HTTP protocol is an internet protocol that is the basis of browser-based internet requests and is commonly used to send form content over the internet or to load web pages. HTTP/s Floods with Browser Emulation is a DDoS attack designed to overwhelm web servers' resources by continuously requesting single or multiple URLs from many source-attacking machines.

Unlike the common HTTP Flood, attacking machines with Browser Emulation will interpret Javascript and fetch all page-related resources (such as images and CSS), thus maintaining proper sessions and cookies.

Such behavior makes the HTTP/s Flood with Browser Emulation capable of bypassing simple Javascript challenges and similar DDoS protection protocols. A custom DDoS attack using Browser Emulation can easily be enhanced with form submissions, mouse movement emulations, and other malicious operations that fall under the definition of normal visitor behavior.



Filter: http | Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
144	1.114327	119.103.175.25	154.49.2.126	HTTP	5896	HTTP/1.1 200 OK (application/javascript)
156	1.305315	154.49.2.126	252.175.11.103	HTTP	343	GET /tag/js/gpt.js HTTP/1.1
157	1.305356	154.49.2.126	189.185.222.219	HTTP	345	GET /c/10816/cc.js?ns=10816 HTTP/1.1
161	1.311157	189.185.222.21	154.49.2.126	HTTP	11954	HTTP/1.1 200 OK (application/javascript)
165	1.311875	252.175.11.103	154.49.2.126	HTTP	1264	HTTP/1.1 200 OK (text/javascript)
181	1.452388	154.49.2.126	119.103.175.250	HTTP	384	GET /bbcdotcom/1.86.0/style/dist/bbcdotcom-async.css HTTP/1.1
182	1.452533	154.49.2.126	119.103.175.250	HTTP	367	GET /bbcdotcom/1.86.0/script/vendor/edr/edr.min.js HTTP/1.1
183	1.452643	154.49.2.126	178.57.107.170	HTTP	346	GET /5/c=10815/pe=y/var=ccauds HTTP/1.1
187	1.460537	119.103.175.25	154.49.2.126	HTTP	3151	HTTP/1.1 200 OK (text/css)
191	1.461099	119.103.175.25	154.49.2.126	HTTP	3727	HTTP/1.1 200 OK (application/javascript)
202	1.471820	178.57.107.170	154.49.2.126	HTTP	449	HTTP/1.1 200 OK (application/javascript)
210	1.475517	154.49.2.126	119.103.175.250	HTTP	381	GET /frameworks/barlesque/3.22.55/orb/4/img/bbc-blocks-light.png HTTP/1.1
211	1.475572	154.49.2.126	119.103.175.250	HTTP	391	GET /weather/0.5.284/images/icons/individual_56_icons/en_on_light_bg/3.gif HTTP/1.1
212	1.475617	154.49.2.126	119.103.175.250	HTTP	392	GET /weather/0.5.284/images/icons/individual_56_icons/en_on_light_bg/10.gif HTTP/1.1
220	1.482124	154.49.2.126	180.58.251.214	HTTP	351	GET /id/0.37.24/svg/icon-sprite.svg HTTP/1.1
227	1.483602	119.103.175.25	154.49.2.126	HTTP	1152	HTTP/1.1 200 OK (PNG)
230	1.484078	119.103.175.25	154.49.2.126	HTTP	1013	HTTP/1.1 200 OK (GIF89a)
235	1.485262	119.103.175.25	154.49.2.126	HTTP	1243	HTTP/1.1 200 OK (GIF89a)
240	1.488893	180.58.251.214	154.49.2.126	HTTP/XML	1194	HTTP/1.1 200 OK
264	1.525003	154.49.2.126	119.103.175.250	HTTP	390	GET /wwhp/144/cpsprodpb/16F84/production/_102948049_mediaitem102948048.jpg HTTP/1.1
265	1.525050	154.49.2.126	119.103.175.250	HTTP	407	GET /wwhp/144/cpsprodpb/3331/production/_102950131_508ac8b3-088e-4211-b807-6e56a5c05f4a.jpg HTTP/1.1
266	1.525085	154.49.2.126	119.103.175.250	HTTP	374	GET /wwhp/144/ibroadcast/images/live/p0/6f/x7/p06fx77v.jpg HTTP/1.1
267	1.525141	154.49.2.126	119.103.175.250	HTTP	374	GET /wwhp/144/ibroadcast/images/live/p0/6g/y1/p06gy1vr.jpg HTTP/1.1
268	1.525177	154.49.2.126	119.103.175.250	HTTP	407	GET /wwhp/144/cpsprodpb/F875/production/_102950636_472499bc-9ae4-4a80-ac65-52a854ded1fa.jpg HTTP/1.1
269	1.525210	154.49.2.126	119.103.175.250	HTTP	372	GET /wwhp/144/cpsprodpb/B9C2/production/_102845574_5.jpg HTTP/1.1
270	1.526297	154.49.2.126	119.103.175.250	HTTP	374	GET /searchbox/1.0.0-137/img/gel-icon-search-light.svg HTTP/1.1
278	1.533189	119.103.175.25	154.49.2.126	HTTP	6375	HTTP/1.1 200 OK (JPEG JFIF image)
284	1.533370	119.103.175.25	154.49.2.126	HTTP	1392	HTTP/1.1 200 OK (JPEG JFIF image)
286	1.533438	119.103.175.25	154.49.2.126	HTTP	5122	HTTP/1.1 200 OK (JPEG JFIF image)
292	1.533614	119.103.175.25	154.49.2.126	HTTP	2557	HTTP/1.1 200 OK (JPEG JFIF image)
294	1.533633	119.103.175.25	154.49.2.126	HTTP	1963	HTTP/1.1 200 OK (JPEG JFIF image)
296	1.533649	119.103.175.25	154.49.2.126	HTTP	1985	HTTP/1.1 200 OK (JPEG JFIF image)
298	1.534586	154.49.2.126	119.103.175.250	HTTP	379	GET /wwhp/144/cpsprodpb/B784/production/_102908964_976getty.jpg HTTP/1.1
299	1.536142	119.103.175.25	154.49.2.126	HTTP/XML	1131	HTTP/1.1 200 OK
301	1.540034	154.49.2.126	119.103.175.250	HTTP	391	GET /wwhp/144/cpsprodpb/F2A4/production/_102861126_redford_shutterstock.jpg HTTP/1.1
302	1.540089	154.49.2.126	119.103.175.250	HTTP	377	GET /wwhp/144/cpsprodpb/1322B/production/_102797387_promo.jpg HTTP/1.1
303	1.540110	154.49.2.126	119.103.175.250	HTTP	389	GET /wwhp/144/cpsprodpb/9B23/production/_102851793_eisteddfod-2-2_new.jpg HTTP/1.1
304	1.542791	119.103.175.25	154.49.2.126	HTTP	8601	HTTP/1.1 200 OK (JPEG JFIF image)
312	1.548550	119.103.175.25	154.49.2.126	HTTP	2791	HTTP/1.1 200 OK (JPEG JFIF image)
314	1.548716	119.103.175.25	154.49.2.126	HTTP	6584	HTTP/1.1 200 OK (JPEG JFIF image)
316	1.554782	154.49.2.126	119.103.175.250	HTTP	375	GET /frameworks/barlesque/3.22.55/orb/4/img/orb-sprite.gif HTTP/1.1

Fetching page-related resources

What happens during the HTTP/s Flood with Browser Emulation DDoS attack?

Similar to HTTP Flood attacks, it may be quite challenging to differentiate the actual attack from valid traffic. Traditional rate-based volumetric detection is ineffective when it comes to detecting HTTP Flood attacks since traffic volume in HTTP Floods is often under the common detection thresholds.

When the attack begins, the browser engine establishes TCP connections in order to send HTTP(s) requests. The TCP connection between the client and the server is usually established using 3-Way Handshake (SYN, SYN- ACK, ACK). The HTTP request packet will normally be in a (PSH+ACK). Unlike a common HTTP Flood that would usually carry to a random or predefined URL, the HTTP/s Flood with Browser Emulation DDoS attack will fetch all page related resources, like JS, CSS and images. It will also execute the page's Javascript and will likely cause all possible page requests to be performed.

In addition, the attack can fetch all the hyperlinks from the page and start following them in a predefined or random order. This will generate an intense "browsing" of the attacked site, in a very high volume, which will eventually lead to the site being overwhelmed, thus creating a disruption of online services.

The HTTP/s Flood with Browser Emulation DDoS attacks are especially dangerous, as they have the potential to "fool" almost every DDoS protection system and security layer. When the limit of concurrent connections is reached on the attacked server, it will no longer respond to legitimate requests from other users, effectively causing a denial of service, disruption of production and business activity, and possibly severe financial damages.

10. Close Ports - Stop the DDoS Attack

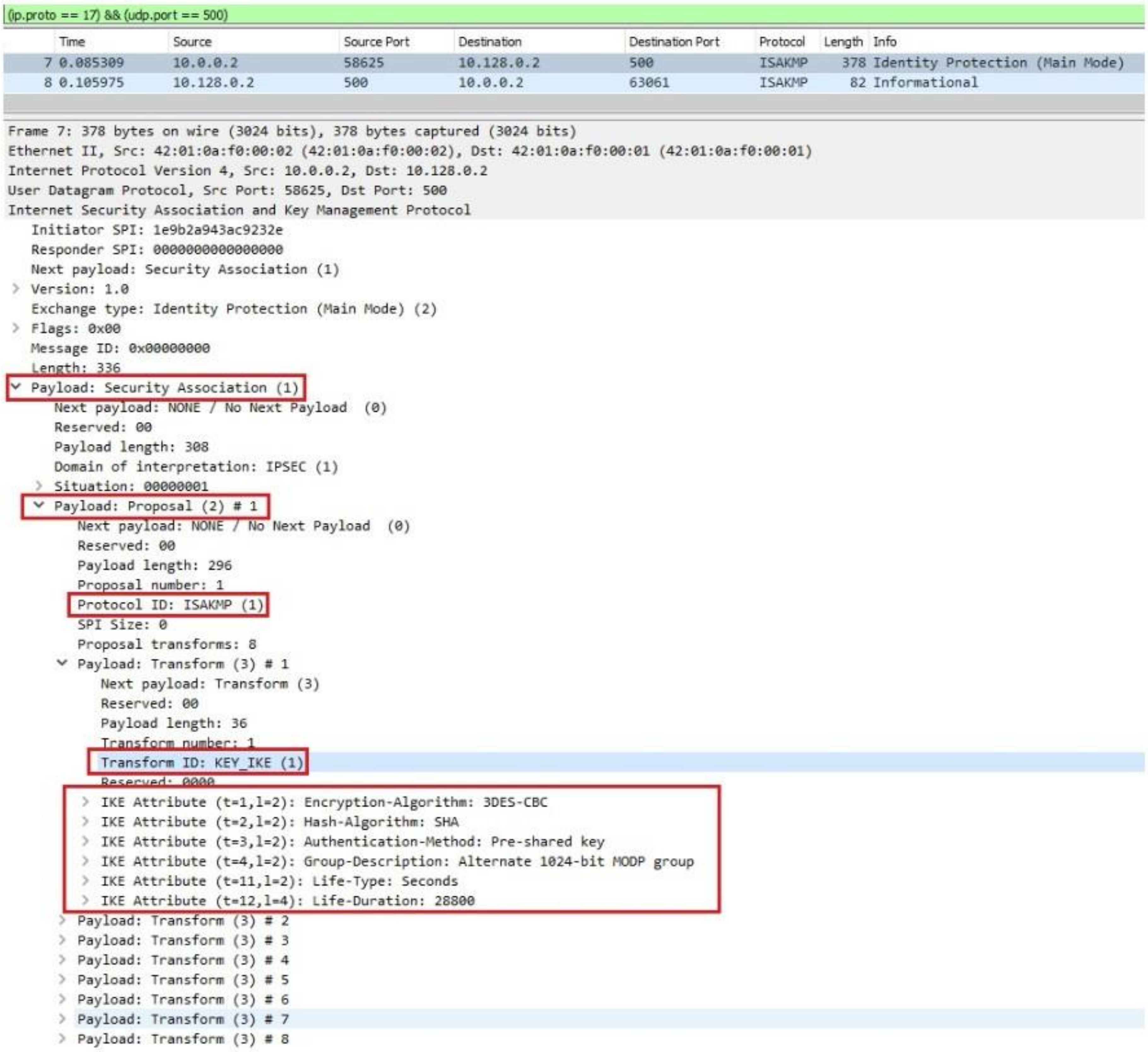
Many organizations lack the necessary visibility into their online services' security posture, thus leaving the protection layers with misconfigurations that eventually lead to successful DDoS attacks. In order to adequately protect online services against DDoS threats, organizations must be proactive and constantly conduct DDoS tests, exposing vulnerabilities and performing prioritized remediation actions.

In particular, security teams must be aware of two common open ports that should be closely monitored and closed if not actually needed. These two ports tend to be open to incoming and outgoing traffic, even if the organization doesn't necessarily provide those types of services from the corresponding parties. If an organization has open service ports like DNS and IKE, they should be either restricted, properly configured, or closed. Otherwise, the organization is at high risk to be targeted for a DDoS attack using one of these two common attack vectors:

IPSEC IKE Flood

The IPSEC IKE Flood is a layer 5 DDoS attack that tries to consume a targeted victim's VPN server resources in order to disrupt a VPN service. This attack is normally performed by sending rapid IPSEC IKE requests to a VPN server via port 500, possibly with a spoofed source IP. This turns the VPN server's response to one containing IKE traffic, and the resource consumption takes place on the victim's VPN server.

Typically, multiple IKE requests will be sent to the target's VPN server. The IKE requests will be sent as a legitimate offer in an ISAKMP payload and the victim's VPN server will respond with a "NO-PROPOSAL-CHOSEN" response, which indicates there is a mismatch of proposals during the negotiation phases. Once noticing a high PPS (Packets Per Second) value per source IP, a security specialist can assume it is a strong indication that an attack is taking place. IPSEC IKE Flood is a resource consumption flood because of the actions executed on the victim's VPN server following the IKE requests. To identify this attack vector, the security team needs to count the number of PPS toward the target VPN server and look at the ISAKMP payload information.



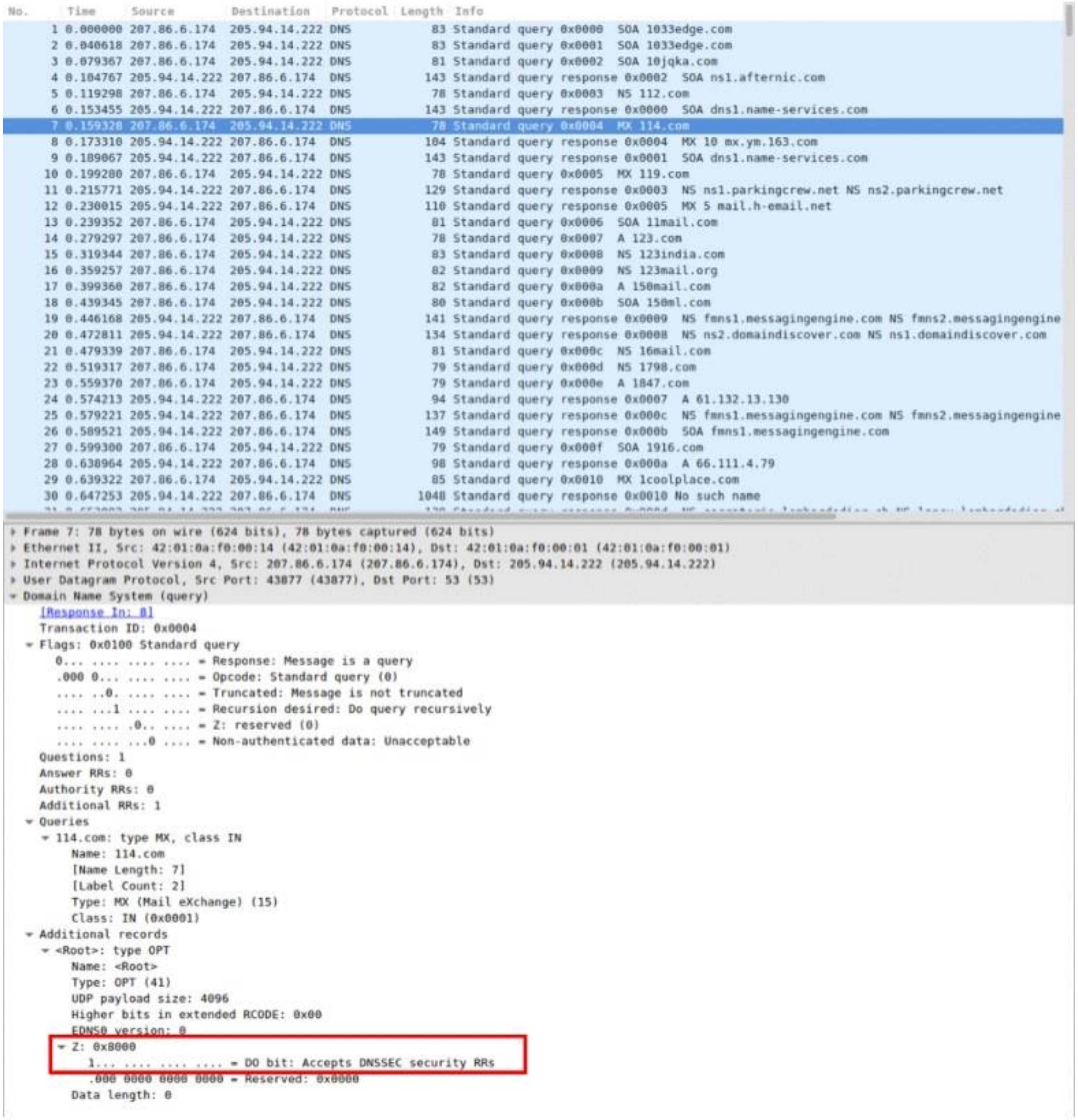
MP Payload

DNS SEC Flood

The DNS SEC Request Flood is a DDoS attack vector that sends DNS SEC request packets to a DNS server in an attempt to overwhelm the server's ability to respond to legitimate DNS requests. DNS services that are unavailable to legitimate users can completely cripple most online services since domain names are used to provide most services. DNS SEC sets the DNS SEC bit to 1, which may cause some servers to process security rules differently.

The attack begins when a DNS SEC request uses the UDP protocol with a specific destination port. The UDP packet contains the query information (name, type, and class).

The server then responds with the query's result, and identifying the request- response pair can be done using the Transaction ID. Depending on the request type, the server may respond differently, but the result will still be the same - a flood of traffic and disruption of services.



DNS SEC Bit

The Bottom Line: How to Uncover DDoS Vulnerabilities and Achieve Fully Automated DDoS Protection?

In order to uncover vulnerabilities in DDoS protection layers and invest the proper prioritized efforts in remediation, an organization must take the new and proactive approach to DDoS security: non-disruptive testing for DDoS vulnerabilities, without compromising business operations. Regardless of what DDoS protection services the organization employs, the security team must be confident they have complete visibility into their DDoS security posture.

Most DDoS protection providers are still reactive in their approach, mitigating known threats well, but unable to mitigate unknown or misconfigured DDoS attack vectors. Thus, DDoS protection providers and security teams must stay updated and configure their security layers properly, through continuous testing of every attack vector against every target, with no operational downtime. DDoS vulnerabilities should be constantly identified, remediated, and then validated to ensure full DDoS resilience. These steps will ensure that an organization has full visibility into its online services and can maximize the full potential of its DDoS protection.

MAZEBOLT

MazeBolt is pioneering a new standard in DDoS security.
RADAR™ enables organizations to leave behind unexpected manual mitigation and SLAs, and move forward to transformative, reliable, and automated DDoS protection that does not require damaging downtime and response scenarios.

RADAR is an industry-first patented solution that identifies how attackers succeed in bypassing existing protection systems through vulnerabilities, through continuous non-disruptive DDoS attack simulations. RADAR's autonomous risk detection allows cybersecurity teams to go light-years beyond traditional DDoS testing and identify and remediate vulnerabilities in every layer of DDoS protection.

Global enterprises, including financial services, insurance, gaming, and high-security government environments rely on MazeBolt to avoid damaging DDoS attacks.

LEARN MORE