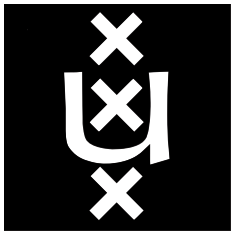

EQUENS

University of Amsterdam
System & Network Engineering

DDoS protection measures for Electronic Payment Systems

Authors:
Joris Claassen and Sean Rijs

Supervisor:
Stefan Dusée, Equens

February 9, 2014

Abstract

This research was done within a month, in which it focussed on Distributed Denial of Service (DDoS) prevention measures: whitelisting, robust Domain Name System (DNS) resolving, and scrubbing. The effectiveness and the difficulty of implementation is researched in the context of electronic payment systems for the payment processor Equens. Electronic payment systems process financial transactions, which require only small data to be sent. The scope does not include layer 7 attacks and focusses only on high volume layer 3 attacks and the aforementioned measures.

The hypothesis for the effectiveness of whitelisting (i.e. only allowing packets to the local network from a set of Internet Protocol (IP) addresses) is that it does not scale during a high volume DDoS attack. If the link from the internet to the electronic payment system network has to send more traffic then the links capability it will drop packets. A test environment was created to simulate the a simple electronic payment system environment. In the test environment a high volume layer 3 attack was created with a maximum of 14 attack computers, and one computer sending small legitimate packets. The results show that, when using six attack computers generating 1 Gbps each, on a 1 Gbps link it will drop drop 85% and 90% packets with whitelisting off and on respectively. Implementing whitelisting entails simply adding addresses to an Access Control List (ACL) and is, therefore, easy to implement.

In order to provide robust DNS resolving two possible solutions are looked into: distributing the DNS servers up to the end of the zone and changing the transport protocol from the default User Datagram Protocol (UDP) to Transmission Control Protocol (TCP). The former is looked into by a literature research and the latter by testing. The DNS root servers have suffered high volume DDoS attacks in the past. One of the attacks was analysed by Internet Corporation for Assigned Names and Numbers (ICANN) and proves that distributing DNS servers using anycast is the most successful measure against high volume DDoS attacks. The implementation of distributing DNS servers with anycast requires changes in the network design and implementation is therefore considered difficult. However distributing DNS servers can be outsourced by a third party. The other solution of changing the transport protocol is tested with the same test as the whitelist test, but the measurement is replaced by counting the timeouts the requests returned when resolving an A record. The results show that using six computers for the attack, its TCP timeouts are 97% as opposed to 14% when using UDP. It is concluded that using TCP for resolving is most likely not effective.

The final prevention measure is scrubbing, which is simply cleaning DDoS attack traffic from the legitimate traffic. Our research focusses on scrubbing centres designed for high volume attacks. By using anycast it can change the traffic path to a high traffic capacity data centre. It scrubs the DDoS traffic (e.g. whitelisting) and sends it via a tunnel to the destination network. As long as the tunnel endpoint address is not known this solution works. However the solution is completely depending on the hiding of the IP address. If it is known to the attacks, the high volume attack can simply be change its target to that address, which effectively creates the aforementioned whitelisting test results.

Contents

1 Introduction

Electronic payment systems are services provided by payment processors to enable exchange of financial transactions. Since financial transactions are a crucial part of the society, its electronic payment systems require a high availability. Additionally financial transactions require little data communication. Some parts of electronic payment systems are accessible from the internet. These parts are just as vulnerable to Distributed Denial of Service (DDoS) attacks as any other service on the Internet.

There are different kinds of DDoS attacks mainly split up in network (layer 3) and application (layer 7) attacks. Mitigation of a layer 7 attack requires a specialised DDoS Defense System (DDS) appliance, which is configured specifically for certain Denial of Service (DoS) vulnerabilities in applications.

This report analyses measures that are available today to mitigate high volume (layer 3) DDoS attacks. The techniques to detect and prevent DDoS attacks vary in levels of effectiveness, and ease of implementation.

1.1 Research question

The research question for this report is:

> *What is the implementation difficulty and how effective is a subset of DDoS protection measures to keep electronic payment systems available?*

The subset of DDoS protection measures consists of:

1. Whitelisting
2. Robust Domain Name System (DNS) resolution
3. Scrubbing

2 Whitelisting

In this chapter the implementation difficulty and effectiveness of whitelisting, i.e. only accepting packets from a set of hosts, will be researched. The inverse is blacklisting, i.e. dropping packets from a set of hosts.

With blacklisting the addresses of the Distributed Denial of Service (DDoS) attackers are required. Getting these addresses is only possible after or during a DDoS attack, and it is not trivial to distinguish malicious packets[1] from legitimate packets. Preventing is always more feasible than waiting for an attack, therefore this chapter omits blacklisting.

Whitelisting is confined to the edge of an electronic payment system as shown in figure 2.1, which is generally the Access Control List (ACL) of a router or firewall. The ACL contains Internet Protocol (IP) addresses which are allowed to communicate with the network.

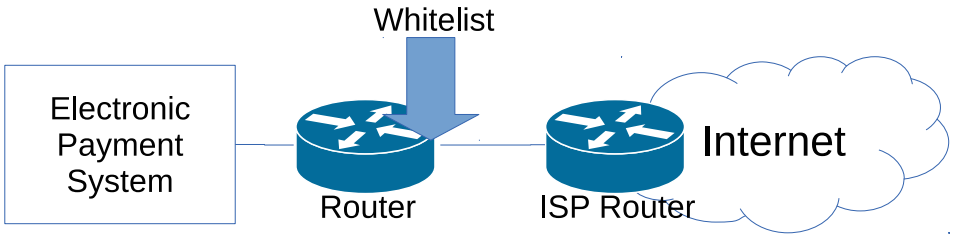

Figure 2.1: Where the whitelist is assumed to be applied

2.1 Implementation difficulty

Implementing a whitelist is simple and quick. The administrators only specify the addresses required for its network and apply it to its ACL.

For example: we want to implement a whitelist with iptables[2] to only allow IP address block 145.100.0.0/15 and 2001:610::/32. The iptables commands shown in listing 2.1 are required.

```
iptables -A FORWARD -i eth0 -s 145.100.0.0/15 -j ACCEPT
iptables -A FORWARD -i eth0 -j DROP
ip6tables -A FORWARD -i eth0 -s 2001:610::/32 -j ACCEPT
ip6tables -A FORWARD -i eth0 -j DROP
```

Listing 2.1: Implementing a whitelist

Unfortunately being able to communicate with every host on the Internet can be a requirement for electronic payment systems. As clients on the Internet need to be connected as soon as possible with minimal tasks or could be connected with dynamic IP addresses.

2.2 Effectiveness

To understand the effectiveness we assume a simple large volume attack on a network. In this scenario the router has a 100 Mbps link protected by a whitelist. The attackers simply sends traffic more than 100 Mbps to the router. The router drops the attackers packets once received, because they are not listed on the whitelists. However, the link connected from the internet to the router only has the capacity to transmit 100 Mbps. This could cause the port connected from the internet to the router to drop incoming packets because its queues are full consequently making the electronic payment system unavailable. Figure 2.2 shows the aforementioned scenario.

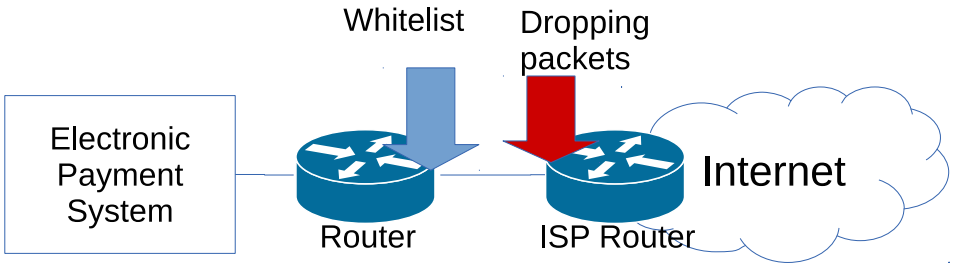

Figure 2.2: A large volume attack protected by whitelisting

2.2.1 Test

In order to test the scenario in figure 2.2 we need to reproduce the situation from the legitimate users and attackers up to the router. It is impossible to exactly reproduce the same situation(e.g. hardware routers) with the available resources for this research. However it is possible to create a similar environment on a smaller scale.

We created the test environment shown in figure 2.3. The roles in figure 2.2 are similar to our test environment. The virtualisation host acts as the router, the target Virtual Machine (VM) acts as the electronic payment system, and the Internet is the switch and the desktops. The specifications of the used hardware and software are described in more detail in appendix A.

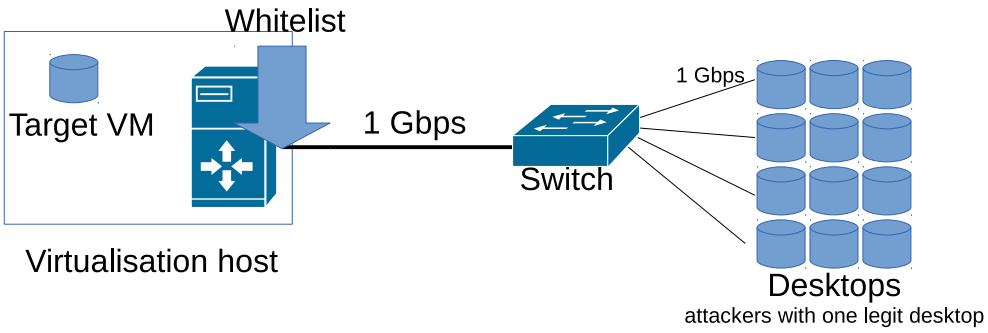

Figure 2.3: Reproducing a large volume attack protected by whitelisting

The DDoS attack consists of 14 desktop computers which floods packets to the VM. The goal is to send 1 Gbps from every desktop over the virtualisation host. Although Transmission Control Protocol (TCP) is used in the packets it is not a goal to do a SYN attack on the end host.

There is one desktop computer taking the role of a legitimate users. To test if the desktop can still communicate with the target VM we send a 1000 packets with the

speed of 10 packets per second to the target. The sending packets use TCP and the amount of unreceived TCP replies is our result value. There are the two parameters:

1. The amount of attackers that send traffic
2. If whitelisting ACL is enabled or disabled

The returning TCP replies of the legitimate traffic are shown in figure 2.4. The listing in A.2 on page 21 show more details of what commands are used.

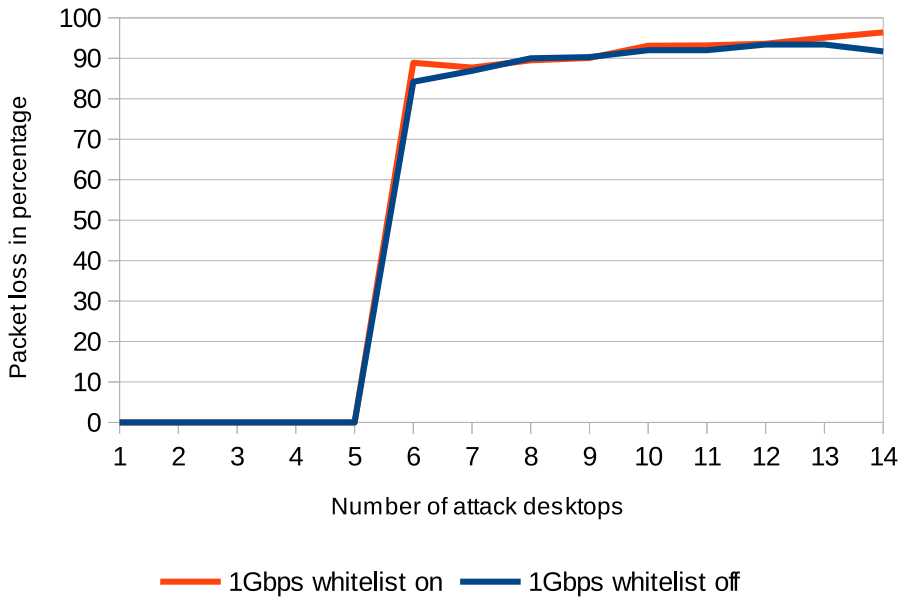

Figure 2.4: Results of an attack using 1 Gbps links

The results shows that generating at least 6 Gbps traffic to a 1 Gbps link causes legitimate traffic to be dropped and consequently not returning a reply. If we try a similar test using only 100 Mbps links the results shown in figure 2.5.

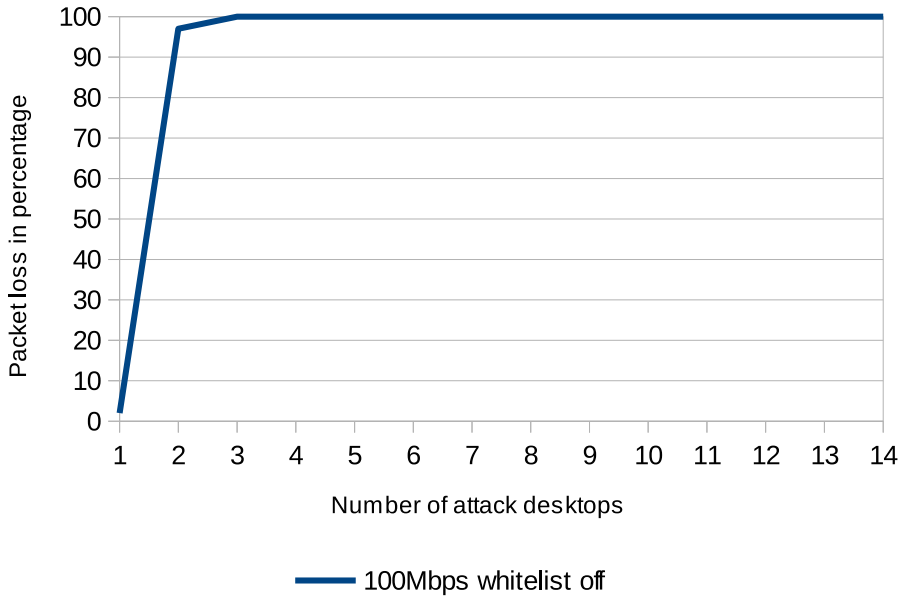

Figure 2.5: Results of an attack using only 100 Mbps links

The cause of the packet loss can be seen in the switches port counter using the Simple Network Management Protocol (SNMP). The counter *ifOutDiscards* is defined in RFC1213[3] as:

> *"The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space."*

After first clearing the counters, a four minute attack with all 14 desktops using 1 Gbps links was performed. The ifOutDiscards counter show that the 272624186 outgoing packets are dropped. This makes it highly likely that the reason for the packet drops is indeed full buffer space in the port. However, it is unknown why packets start to drop using relatively less desktops using only 100 Mbps links.

Listing 2.2 show the exact commands used. The same counters of all other ports are shown in listing A.5 on page 22.

```
user@client:~$ snmpget -Os -c public -v 1 switchaddress \
                                                ifOutDiscards.21
ifOutDiscards.21 = Counter32: 272661695
```

Listing 2.2: Switch drops packets on virtualisation host port

2.3 Sub-conclusion

It is very easy to implement a whitelist as an example implementation in listing 2.1 shows. However electronic payment systems could need new clients to be connected as fast as possible without needing to edit the whitelist first.

If electronic payment systems decide its acceptable to edit the ACL before connecting clients whitelisting still does not guarantee availability. The used test setup is not the

same as figure 2.1, but it gives an indication of the effectiveness during a real, larger scale, attack. The conclusion is that it is highly probable that whitelisting has a low effectiveness in a large volume DDoS attack.

3 Robust DNS resolution

This chapter is about robust Domain Name System (DNS) resolution. Robust means: keeping the service available. Note that robust DNS does not concern, confidentiality and integrity, which are also part of the general security attributes; Confidentiality, Integrity and Availability (CIA). DNS is one of the older protocols of the Internet. To illustrate, at the time of development of DNS, designing the protocol against DDoS attacks was not even remotely imaginable. Since DNS resolution requires a server to accept requests it is vulnerable for a DDoS attack as shown in figure 3.1.

History has proven[4] that distributing the DNS service is the most effective (i.e. physical dispersion and IP anycast). However, an alternative could be that the use of TCP instead of the default User Datagram Protocol (UDP) would improve the reliability of the queries[5]. The arguments for using TCP instead are its inherit features: it can retransmit packets when not received, and at every failed Acknowledgement (ACK) it slows down the sending.

This chapter will look into two different measures that provide possible robust DNS resolution during a high volume DDoS attack.

1. Compare returning query answers TCP and UDP
2. What is required to implement IP anycast

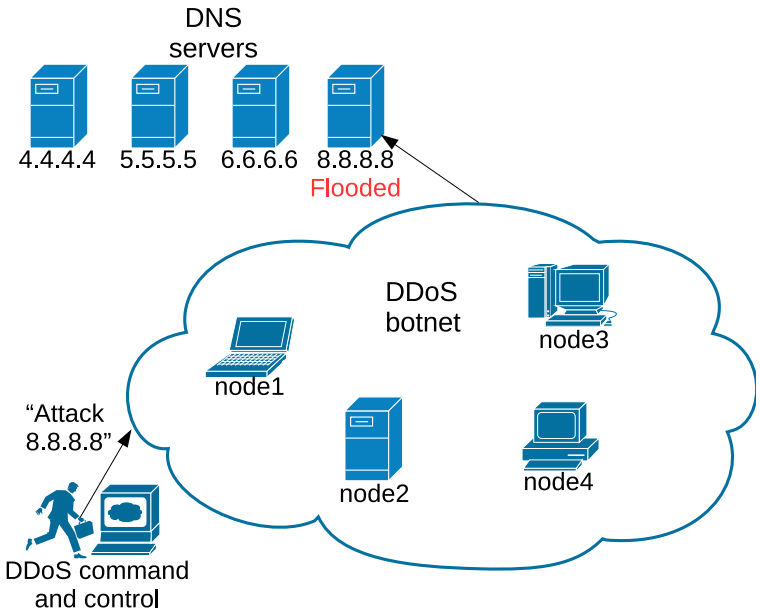

Figure 3.1: DDoS attack on a DNS server

3.1 Implementation difficulty

This section describes the implementation difficulty of migrating from TCP to UDP and implementing IP anycast.

3.1.1 TCP and UDP resolving

The difficulty of changing the DNS resolving transport protocol from UDP to TCP is that it could be difficult, depending on the resolver implementation. If it requires the clients to change the standard library (e.g. *libc* in Linux) it would imply maintaining upcoming security updates of the software library. However one could use a different library for their specific software consequently bypassing the standard library, which would not require extra maintainability of software but protects only a specific application.

3.1.2 Anycasting

IP anycast is made possible with the Border Gateway Protocol (BGP) which is the routing protocol on the Internet. Anycast makes it possible to advertise an IP network (e.g. 145.100.0.0/15), consequently enabling one IP address to resolve to different servers depending on the source of the sending IP[6].

Implementing anycast is complicated. It requires multiple servers, and preferably on geographically different locations. The most important requirement is the distribution of both the software and the network services. All the servers use the same IP network and the end user should be able to connect to the nearest server. Implementation therefore requires changes in the application, and the routers that advertise the network.

3.2 Effectiveness

This section looks into the effectiveness of BGP anycast and migrating from TCP to UDP.

3.2.1 Comparing TCP and UDP

To test the two transport protocols the same attack is launched as described in figure 2.2 on page 6. During the attack 1000 queries for a single A record is send at a speed of 10 queries per second. The test consists of two parameters:

1. The amount of attackers that send traffic
2. Using TCP or UDP

The results of this test are shown in 3.2 showing that TCP does not scale well during a DDoS attack. Two possible arguments for it can be: packets are retransmitted if no ACK is received, and at every failed ACK it slows down the flow of sending packets. The former argument only generates more traffic, as more packets are sent over the network it congests the link with even more packets which causes more packets to be dropped. The latter argument does not matter as the DDoS has a constant rate of traffic.

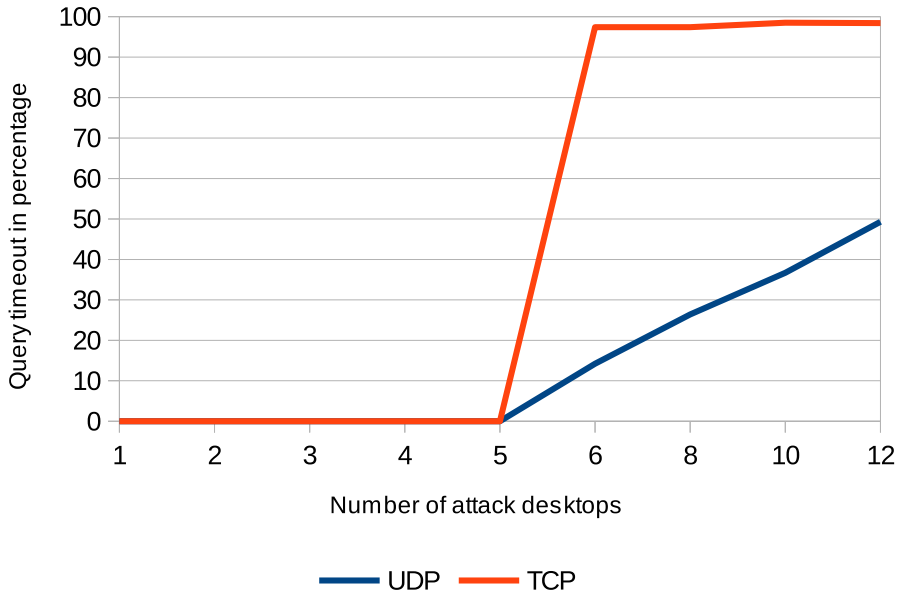

Figure 3.2: Comparison of query timeouts between TCP or UDP

3.2.2 Anycasting

History has proven that anycast is an effective measure against DDoS attacks on DNS servers. For example: the DNS root servers were under a DDoS attacks in both 2002 and 2007. After the attacks in 2002, further measures were taken to prevent outages. One of the main components of these measures was the implementation of anycast on most of the root servers as shown in figure 3.3. The servers on which anycast was not implemented suffered the highest outages during attacks in 2007[4].

RFC 4786 (Anycast Best Current Practices)[6] also suggests the use of anycast to mitigate DDoS attacks in the "Security Considerations" chapter.

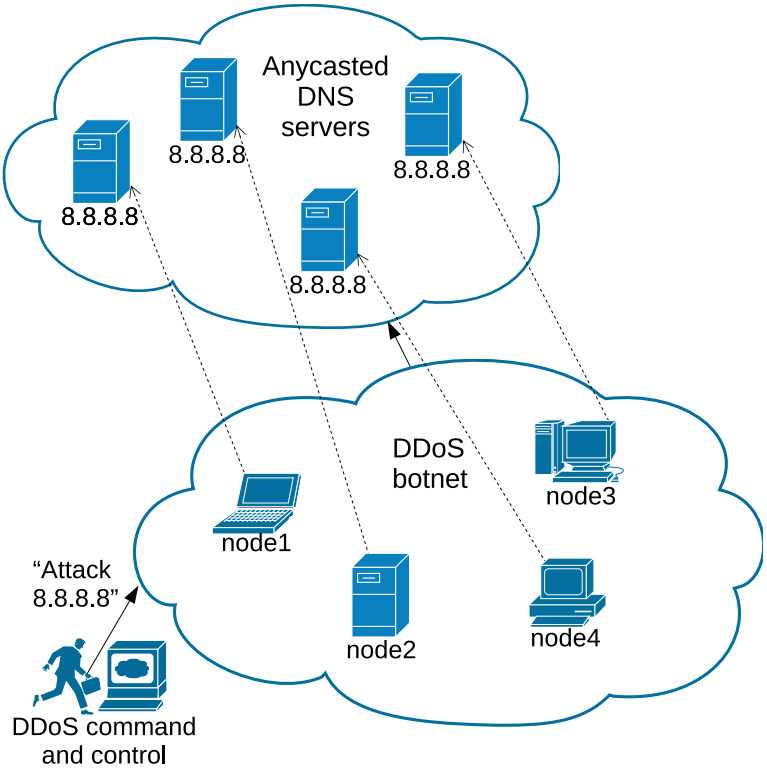

Figure 3.3: Physical dispersion and anycast

3.3 Sub-conclusion

To conclude this chapter we need to come back to the beginning of this chapter; availability of the DNS service. In this case: the availability during a DDoS attack should be as high as possible, ideally with no impact for the end-user of the service.

Changing the transport protocols from the default UDP to TCP causes the DNS resolution to be less available.

The only measure which has been proven to provide more availability is to distribute an already distributed service like DNS even further, i.e. up to the endpoints of the DNS tree. Technical measures to mitigate DDoS attacks on the DNS service can be taken by the owner of the service, or they can be bought from a third party which owns a sufficient infrastructure to provide (global) distribution.

4 Scrubbing

The term scrubbing in the context of DDoS attacks applies to the cleaning of attack traffic. Since DDoS attacks come in all shapes and sizes, scrubbing does as well. There are DDoS Defense System (DDS)s which work in line with the current network topology, reinforcing them against layer 7 attacks. These appliances can be fine-tuned for the environment they are protecting.

A scrubbing service is usually bought from a third party, as it is not feasible to maintain an infrastructure required to mitigate the amounts of bandwidth a DDoS attack generates. Scrubbing services can handle vast amounts of data from all over the world and are often distributed using anycast. Scrubbing services have existed since 2003[7], as the urge for DDoS protection began to rise after attacks on the DNS root servers[4].

4.1 Implementation difficulty

Both DDSs and Scrubbing services make use of several DDoS mitigation techniques: most fall back to the principles of blackholing and sinkholing. Blackholing refers to "black holes" as in astronomy. Black holes are places where network packets are dropped, without informing the sender of the packet that it has not arrived. In scrubbing centres a technique called "black hole filtering" is applied, which drops packets at layer 3.

Sinkholing refers to analysing data containing a DDoS attack to distinguish attack packets from legitimate packets. This can be done by saving data on large data volumes in the data centre where the scrubbing centre resides, or by making use of a network telescope or a darknet. Saving the data in the data centre is quite straightforward, and allows for inspection after or even during the attack.

A network telescope[8] or a darknet[9] are two names for the same thing; namely (a set of) probes to accept data packets in a "dark", unused part of the Internet. The information they gather is often used by research institutes to get a better overview in what sort of attacks are in the wild (visualised in figure 4.1).

The high volume traffic sent during a DDoS attack causes a lot of replies to be sent from the network that is under attack. These replies are sent to spoofed addresses. Some of these spoofed addresses may be part of the network telescope. The data collected from these probes may contain valuable information about DDoS attacks. Because it is not limited to just a attack on a single network, it can even gather insight about other networks that are under attack.

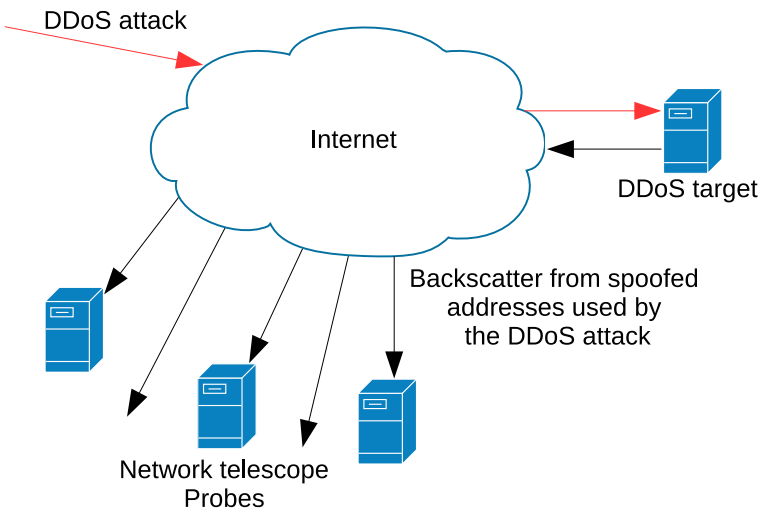

Figure 4.1: Network telescope

When a company tries to protect itself from DDoS attacks they have three options: Implementing a DDS, contract a third party scrubbing service, or do both. Doing nothing would be taking a huge service availability risk.

Implementing a DDS is about as hard as implementing a firewall in an infrastructure; install and then configure the appliance to match the specifications of the network.

Scrubbing services rely on routes in the network, that can be changed using BGP if necessary. In addition to routes, there also has to be a tunnel from the scrubbing centre to the network that is to be protected. When all the data is sent through the scrubbing centre both black- and sinkholing is done. The filtered traffic is then passed through the tunnel, after which the server sends out the replies back to the legitimate addresses.

4.2 Effectiveness

The effectiveness of scrubbing depends on the attack. A DDS inside a local network might be a very good mitigation strategy if a company is experiencing layer 7 attacks. But when it is experiencing a high volume attack the same DDS stands no chance, simply because the ingress link is fully saturated with traffic.

Because of the distributed way a scrubbing service is set up, it can handle vast amounts of data without losing packets. It can then sort out the "good" packets using a multitude of measures. Our hypothesis is that the tunnel endpoint is still vulnerable to high volume attacks, because the scrubbing centre needs a endpoint to send the filtered traffic to. It should be possible to hide this endpoint by dropping all incoming *and* outgoing packets, while whitelisting the tunnel endpoint at the border router of the company.

4.2.1 Test

To test our hypothesis we will do a series of traceroute tests, to see whether we can hide the tunnel endpoint. Figure 4.2 shows the test environment used to simulate these tests.

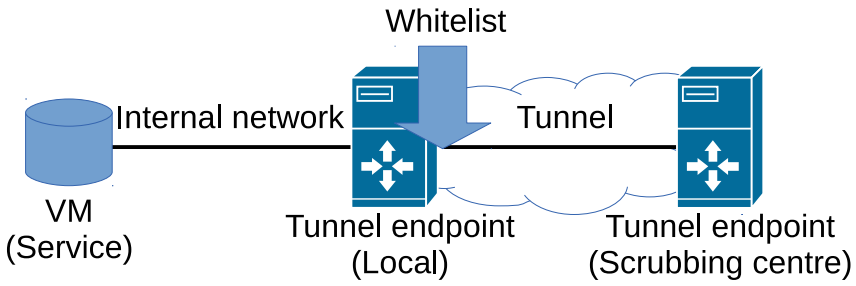

Figure 4.2: Test environment

In listing 4.1 an Internet Control Message Protocol (ICMP), UDP and TCP traceroute are shown respectively, with no whitelist enabled. All the routers in between including the tunnel endpoints reply, consequently making them known.

```
user@client:~$ traceroute 172.16.1.2
traceroute to 172.16.1.2 (172.16.1.2), 30 hops max, 60 byte packets
 1  172.16.1.1 (172.16.1.1)  0.267 ms  0.255 ms  0.246 ms
 2  172.16.1.2 (172.16.1.2)  0.401 ms  0.356 ms  0.338 ms
user@client:~$ traceroute -U 172.16.1.2
traceroute to 172.16.1.2 (172.16.1.2), 30 hops max, 60 byte packets
 1  172.16.1.1 (172.16.1.1)  0.293 ms  0.268 ms  0.250 ms
 2  172.16.1.2 (172.16.1.2)  0.358 ms  0.342 ms  0.326 ms
user@client:~$ sudo traceroute -T 172.16.1.2
traceroute to 172.16.1.2 (172.16.1.2), 30 hops max, 60 byte packets
 1  172.16.1.1 (172.16.1.1)  0.235 ms  0.207 ms  0.183 ms
 2  172.16.1.2 (172.16.1.2)  0.347 ms  0.326 ms  0.320 ms
```

Listing 4.1: Whitelist disabled

Listing 4.2 shows the ip(6)tables commands used to implement the whitelist.

```
iptables -A INPUT -i vmbr1 -j DROP
ip6tables -A INPUT -i vmbr1 -j DROP
iptables -A FORWARD -i vmbr1 -s 172.16.1.0/24 -j ACCEPT
iptables -A FORWARD -i vmbr1 -j DROP
ip6tables -A FORWARD -i vmbr1 -s 2001:610::/32 -j ACCEPT
ip6tables -A FORWARD -i vmbr1 -j DROP
iptables -A OUTPUT -i vmbr1 -j DROP
ip6tables -A OUTPUT -i vmbr1 -j DROP
```

Listing 4.2: Implementing a whitelist

In listing 4.3 an ICMP, UDP and TCP traceroute are shown again, but without the whitelist enabled.

```
user@client:~$ traceroute 172.16.1.2
traceroute to 172.16.1.2 (172.16.1.2), 30 hops max, 60 byte packets
 1  * * *
 2  172.16.1.2 (172.16.1.2)  0.309 ms  0.324 ms  0.317 ms
user@client:~$ traceroute -U 172.16.1.2
traceroute to 172.16.1.2 (172.16.1.2), 30 hops max, 60 byte packets
 1  * * *
 2  172.16.1.2 (172.16.1.2)  0.519 ms  0.530 ms  0.525 ms
```

```
user@client:~$ sudo traceroute -T 172.16.1.2
traceroute to 172.16.1.2 (172.16.1.2), 30 hops max, 60 byte packets
 1  * * *
 2  172.16.1.2 (172.16.1.2)  0.386 ms  0.352 ms  0.394 ms
```

Listing 4.3: Whitelist enabled

As can be seen in these results, it is possible to hide a tunnel endpoint from the rest of the Internet.

4.3 Sub-conclusion

Depending on the threat faced by DDoS attacks on the network a different form of scrubbing can be chosen. If there is only a layer 7 threat for one or more applications, the most effective approach would be to install a custom configured DDS on the premises.

In case of a high volume attack, it would make more sense to contract a third party scrubbing service and let them cope with the huge amounts of traffic. The filtered traffic can then be tunnelled through to the company network, and data can be sent out through the normal IP routes. Because the tunnel has an endpoint that needs to function at all times, it is *crucial* that this endpoint is hidden during both normal operation and while being under attack.

5 Conclusion

After analysing the three different kind of measures none of the solutions provide complete DDoS mitigation against high volume attacks. A brief summary of our sub-conclusions:

1. Whitelisting: does not provide high volume DDoS mitigation
2. Robust DNS resolution: distributing the DNS of the network reduces the effects of a high volume DDoS attack
3. Scrubbing: a scrubbing centre is effective against a high volume DDoS attack, as long as the tunnel end-point remains hidden to the attacker

A proven and successful measure against high volume DDoS attack are distributed systems, as the DNS root servers have proven multiple times in the past. It might not be feasible to distribute the entire environment depending on the complexity of a system. An on-demand scrubbing centre, distributed DNS service and correctly implemented whitelisting on tunnel endpoints should provide an acceptable level of availability.

6 Future research

This report focused on a subset of measures against high volume layer 3 attacks and how it effects the availability. Electronic payment systems include services that use layer 7 protocols. A combination of the two, also known as a smoke and mirrors attack, could be an other interesting attack vector.

When combining layer 3 with layer 7 attacks a local DDS might not be able to process all the attack data. This forces it to let it pass, consequently exposing applications to possible layer 7 Denial of Service (DoS) attacks. More attack vectors could arise when including other security attributes such as confidentiality and integrity.

Figure 2.5 showed that using 100 Mbps in our tests gave relatively sooner packets loss using less attack desktops. It is unknown what the exact cause is. It could for instance be researched if this might has something to do with the switches internals or something else. However full access to the switches code is probably required to be certain of the cause.

Setting up DDoS test setups is not trivial. In this research, most time was spent creating a DDoS setup. It is not known what the best practices could be in order to create a deterministic DDoS setup. One could define a setups in order for future research to spend less time in creating DDoS setups.

A Setup whitelisting

In this appendix the test setup we used for the whitelisting chapter will be described.

A.1 Hardware

Manufacturer	Dell
Model	PowerEdge R210 II
CPU	1x Intel(R) Xeon(R) CPU E3-1220L V2 @ 2.30GHz
Memory	4x DM0KY - 2GB DIMM, 1333MHz
Ethernet	2x Broadcom Corporation NetXtreme II BCM5716 Gigabit

Table A.1: Virtual host server running the target VM

CPU	1-2
Memory	256-512 MB
Ethernet	virtio

Table A.2: Target VM running on KVM using dynamic resources

Manufacturer	Dell
Model	Optiplex 7010
CPU	1x Intel(R) Core(TM) i5-3570S CPU @ 3.10GHz
Memory	3x 531R8 - 4G DIM, 1600MHz
Ethernet	1x Intel Corporation 82579LM Gigabit

Table A.3: Attacking desktops machines

Manufacturer	Dell
Model	PowerConnect 6224
Ports	24x Gigabit ethernet ports

Table A.4: Gigabit switch

A.2 Software

This section only shows software versions that are relative to the scenario.

Operating System	Ubuntu Linux 13.10 server 64-bit
Linux kernel	3.11
qemu	1.5.0
libvirt-bin	1.1.1

Table A.5: Virtual host server using KVM

Operating System	Debian 7.3 64-bit
Linux kernel	3.2

Table A.6: Target VM

A.3 Throughput

This is a throughput test with the command *iperf*[10]

```
%desktop machine to the target
```

```
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
```

```
[  4] local:5001 connected with 172.16.1.65:56342
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  1.08 GBytes   930 Mbits/sec
```

Listing A.1: iperf throughput output from an attack desktop to the target VM

A.4 The simulated attack

In short a TCP port that the target is not listening on was attacked, therefore not creating a SYN attack but still saturating the ingress link. A program called *hping*[11] packets was used to generate and send the attack.

A.4.1 Software

To generate the attack traffic the hping parameters shown in listing A.2 were used. It sends TCP packets with 8000 byte data to the target on destination port 5001, which the target is not listening on.

```
parallel-ssh -h nodes \
sudo hping3 --flood -S 172.16.1.10 --destport 5001 \
--data 8000
```

Listing A.2: hping command to generate an attack on port 5001

To test the packet loss on a whitelisted address the hping parameters shown in listing A.3 were used during a DDoS attack.

```
sudo hping3 -c 1000 --fast 172.16.1.10
```

Listing A.3: hping command to test packet loss whitelisted desktop

The iptables commands in listing A.4 was used to create a whitelisting ACL to allow one desktop sending legitimate traffic to the attacked server.

```
iptables -A FORWARD -i vmbr1 -s 172.16.1.10/32 -j ACCEPT
iptables -A FORWARD -i vmbr1 -s 172.16.1.201/32 -j ACCEPT
iptables -A FORWARD -i vmbr1 -j DROP
```

Listing A.4: Implementing a whitelist

A.4.2 Results

These results show the counters of the switch and server and the processor load of only the switch. Listing A.5 show all discarded packets of every port on the switch egress and ingress respectively.

```
user@client:~$ snmpwalk -Os -c public -v 1 switchaddress \
                                                ifOutDiscards
ifOutDiscards.1 = Counter32: 3248
ifOutDiscards.2 = Counter32: 3256
ifOutDiscards.3 = Counter32: 3250
ifOutDiscards.4 = Counter32: 3259
ifOutDiscards.5 = Counter32: 3251
ifOutDiscards.6 = Counter32: 3246
ifOutDiscards.7 = Counter32: 0
ifOutDiscards.8 = Counter32: 4153
ifOutDiscards.9 = Counter32: 3246
ifOutDiscards.10 = Counter32: 3244
ifOutDiscards.11 = Counter32: 0
ifOutDiscards.12 = Counter32: 0
ifOutDiscards.13 = Counter32: 3249
ifOutDiscards.14 = Counter32: 3247
ifOutDiscards.15 = Counter32: 3684
ifOutDiscards.16 = Counter32: 3243
ifOutDiscards.17 = Counter32: 3244
ifOutDiscards.18 = Counter32: 3496
ifOutDiscards.19 = Counter32: 0
ifOutDiscards.20 = Counter32: 3251
ifOutDiscards.21 = Counter32: 272661695
ifOutDiscards.22 = Counter32: 3248

user@client:~$ snmpwalk -Os -c public -v 1 switchaddress \
                                                ifInDiscards
ifInDiscards.1 = Counter32: 19499611
ifInDiscards.2 = Counter32: 19431655
ifInDiscards.3 = Counter32: 19553175
ifInDiscards.4 = Counter32: 19548517
ifInDiscards.5 = Counter32: 19553610
ifInDiscards.6 = Counter32: 19554255
ifInDiscards.7 = Counter32: 0
ifInDiscards.8 = Counter32: 263
ifInDiscards.9 = Counter32: 19523887
ifInDiscards.10 = Counter32: 19554357
ifInDiscards.11 = Counter32: 0
ifInDiscards.12 = Counter32: 0
ifInDiscards.13 = Counter32: 19380978
ifInDiscards.14 = Counter32: 19531682
ifInDiscards.15 = Counter32: 42
```

```
ifInDiscards.16 = Counter32: 19524560
ifInDiscards.17 = Counter32: 19264616
ifInDiscards.18 = Counter32: 1185
ifInDiscards.19 = Counter32: 0
ifInDiscards.20 = Counter32: 19519823
ifInDiscards.21 = Counter32: 117
ifInDiscards.22 = Counter32: 19182124
```

Listing A.5: SNMP ifOutDiscards and ifInDiscards of all switch ports

```
switch#show process cpu

Memory Utilization Report

status       bytes
------ -----------
  free    26754320
 alloc   188061760



CPU Utilization:

 PID      Name                          5 Sec      1 Min      5 Min
-------------------------------------------------------------------
 335fc10  tNetTask                      0.00%      0.02%      0.00%
 354c0f0  ipnetd                        0.00%      0.04%      0.00%
 355e690  tXbdService                   0.00%      0.06%      0.00%
 35796d0  osapiTimer                    1.11%      0.89%      1.09%
 366aca0  bcmL2X.0                      0.15%      0.20%      0.30%
 3680130  bcmCNTR.0                     0.15%      0.34%      0.20%
 36b3600  bcmTX                         0.00%      0.06%      0.01%
 3caa800  bcmRX                         0.00%      0.06%      0.07%
 3ee0a30  MAC Send Task                  0.00%      0.02%      0.00%
 3ee9f30  MAC Age Task                   0.00%      0.02%      0.00%
 4a6a5a0  bcmLINK.0                     0.47%      0.37%      0.35%
 4ca7050  cpuUtilMonitorTask            0.15%      0.02%      0.00%
 516b880  tL7Timer0                     0.00%      0.06%      0.00%
 5191160  osapiMonTask                  0.00%      0.00%      0.09%
 5e7fd60  simPts_task                   0.00%      0.14%      0.15%
 62b4280  dtlTask                       0.00%      0.14%      0.06%
 6320e70  tEmWeb                        0.00%      0.10%      0.03%
 6401b70  hapiRxTask                    0.00%      0.06%      0.02%
 6966950  DHCP snoop                    0.00%      0.04%      0.00%
 69fce60  Dynamic ARP Inspection        0.00%      0.04%      0.00%
 6a07250  SNMPTask                      0.00%      0.00%      0.02%
 763dd40  dot1s_timer_task              0.63%      0.41%      0.48%
 8617240  snoopTask                     0.15%      0.02%      0.00%
 90142b0  trapTask                      0.00%      0.00%      0.03%
 95d8290  ipMapForwardingTask           0.00%      0.06%      0.15%
 cbc8bb0  lldpTask                      0.31%      0.16%      0.30%
 d873850  DHCP Client Task              0.00%      0.02%      0.00%
 d8a8f00  isdpTask                      0.00%      0.04%      0.00%
 e0aa6b0  RMONTask                      0.15%      0.09%      0.01%
 e0b6d50  boxs Req                      0.00%      0.05%      0.00%
```

```
e5e7f50 sshd                          0.00%     0.06%     0.00%
e61be60 sshd[0]                       0.00%     0.00%     0.43%
-------------------------------------------------------------------
Total CPU Utilization                 3.27%     3.59%     3.79%
```

Listing A.6: CPU load on switch

```
em2   Link encap:Ethernet  HWaddr d4:ae:52:bf:e3:d5
      inet6 addr: fe80::d6ae:52ff:febf:e3d5/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:4452765 errors:0 dropped:7 overruns:0 frame:0
      TX packets:278225 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:6118904480 (6.1 GB)  TX bytes:18040839 (18.0 MB)
```

Listing A.7: ifconfig output of the virtualisation host port to the switch

Ping results during a DDoS attack while whitelisting is on.

B Setup DNS resolution comparing TCP and UDP

This appendix describes the test setup used for comparing UDP with TCP availability. The same hardware and software is used as described in appendix A except for the DNS query traffic and the DNS resolver at the target VM.

B.1 Software

Table B.1 shows the related software used for the target VM.

Operating System	Debian 7.3 64-bit
Linux kernel	3.2
DNS resolver	Unbound 1.4.17

Table B.1: Target VM

The goal to measure the timeouts is to generate 10 queries per second and stop after a total of 1000 queries. Listing B.1 shows the Python script generating the queries and listing B.2 shows the shell command to start the script and measure the timeouts.

```
#!/usr/bin/python
import dns.resolver
import time
import threading


class ThreadedClass(threading.Thread):
    def run(self):
        global count
        my_resolver = dns.resolver.Resolver()
        my_resolver.nameservers = ['172.16.1.100']
        my_resolver.lifetime = 30
        try:
            #change boolean "tcp" for switch
            answer = my_resolver.query('www.os3.nl', \
                                       tcp=False)
            for rdata in answer:
                print rdata
        except dns.resolver.Timeout:
            print "Thread timed out."
```

```
start_time = time.time()
interval = 0.1

for i in range(1000):
    t = ThreadedClass()
    t.start()
    time.sleep(int(start_time + i*interval - time.time()))
```

Listing B.1: Python code to generate the queries

```
user@client:~$ python dns_test.py |grep \
                "Thread timed out." | wc -l
```

Listing B.2: Bash command to count the amount of timeouts

C | Acro

DDoS Distributed Denial of Service

DoS Denial of Service

TCP Transmission Control Protocol

ACK Acknowledgement

UDP User Datagram Protocol

DNS Domain Name System

VM Virtual Machine

IP Internet Protocol

BGP Border Gateway Protocol

ACL Access Control List

CIA Confidentiality, Integrity and Availability

ICMP Internet Control Message Protocol

SNMP Simple Network Management Protocol

DDS DDoS Defense System

ICANN Internet Corporation for Assigned Names and Numbers

D References

[1] N. Jeyanthi, N. Iyengar, P. C. M. Kumar, and K. A, "An Enhanced Entropy Approach to Detect and Prevent DDoS in Cloud Environment," 2013. http://www.ijcnis.org/index.php/ijcnis/article/view/367.

[2] "The netfilter.org project (a.k.a. iptables)." http://www.netfilter.org/.

[3] K. McCloghrie and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II," 1991. http://www.ietf.org/rfc/rfc1213.

[4] "Factsheet: Root server attack on 6 february 2007." http://www.icann.org/en/about/learning/factsheets/factsheet-dns-attack-08mar07-en.pdf.

[5] K. Park, V. S. Pai, L. L. Peterson, and Z. Wang, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," 2004. http://static.usenix.org/events/osdi04/tech/full_papers/park/park.pdf.

[6] J. Abley, A. Canada, and K. Lindqvist, "Anycast BCP," 2006. http://www.ietf.org/rfc/rfc4786.

[7] "Prolexic technologies," https://www.prolexic.com/company.html.

[8] CAIDA, "The ucsd network telescope." http://www.caida.org/projects/network_telescope/.

[9] Team CYMRU, "The darknet project." https://www.team-cymru.org/Services/darknets.html.

[10] "iperf website." http://iperf.fr/.

[11] "hping website." http://www.hping.org/.