Protecting Internet Networks Against DoS Attacks

Adam Mirre

Bachelor's thesis
2021

Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Adam Mirre**
Osobní číslo:	**A18065**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Implementace ochrany sítě proti DoS útokům**
Téma práce anglicky:	**Protecting Internet Networks Against DoS Attacks**

Zásady pro vypracování

1. Prostudujte a popište nejčastěji používané typy útoků typu DoS a DDoS.
2. Ke každému typu útoku najděte a popište možné způsoby ochrany pro zamezení škod.
3. Vyzkoušejte a popište některé z bezplatných i komerčních služeb pro provádění DDoS útoků.
4. Vyberte jeden z volně dostupných systémů ochrany proti DDoS útokům a nainstalujte jej.
5. Otestujte funkcionalitu vámi vybraného systému v reálném nasazení a popište dosažené výsledky.

Forma zpracování bakalářské práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. COLLIER, Ben, Daniel R. THOMAS, Richard CLAYTON a Alice HUTCHINGS, 2019. Booting the booters: evaluating the effects of police interventions in the market for denial-of-service attacks. In: Internet Measurement Conference: IMC 2019 – Proceedings of the 2019 ACM Internet Measurement Conference. s. 50–64. ISBN: 9781450369480
2. NAWROCKI, Marcin, Jeremias BLENDIN, Christoph DIETZEL, Thomas C. SCHMIDT a Matthias WÄHLISCH, 2019. Down the Black Hole: Dismantling Operational Practices of BGP Blackholing at IXPs. In: Proceedings of the Internet Measurement Conference. New York, NY, USA: Association for Computing Machinery, s. 435–448. IMC ’19. ISBN 978-1-4503-6948-0.
3. SADEGHIAN, A. a M. ZAMANI, 2014. Detecting and preventing DDoS attacks in botnets by the help of self triggered black holes. In: 2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE): s. 38–42.
4. SANTANNA, José Jair Cardoso de, 2017. DDoS-as-a-Service: Investigating Booter Websites. ISBN: 978-90-365-4429-0.
5. SANTANNA, J. J., R. O. De SCHMIDT, D. TUNCER, J. de VRIES, L. Z. GRANVILLE a A. PRAS, 2016. Booter blacklist: Unveiling DDoS-for-hire websites. In: 2016 12th International Conference on Network and Service Management (CNSM): s. 144–152. ISSN 2165-963X.

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **17. května 2021**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

I hereby declare that:

- I understand that by submitting my Bachelor's thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Bachelor's Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Bachelor's Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlín.
- I am aware of the fact that my Bachelor's Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, Tomas Bata University in Zlín has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work – Bachelor's Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlín with a view to the fact that Tomas Bata University in Zlín must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Bachelor's Thesis include the use of software provided by Tomas Bata University in Zlín or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Bachelor's Thesis cannot be used for commercial purposes.
- I understand that, if the output of my Bachelor's Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- The submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlín; dated: .....

Student's Signature

ABSTRAKT

V tejto práci sme analyzovali bežné spôsoby prevádzkovania Denial of Service útokov, ako aj niektoré zo známych nástrojov, ktorými sa útoky vykonávajú. Na druhej strane sme sa pojednávali o regulérne používaných spôsoboch ochrany a to tak z pohľadu sieťových operátorov, ako aj koncových užívateľov. Pokúsili sme sa zautomatizovať vytváranie testovacieho laboratória a vyskúšali sme si sprevádzkovať útok na jednoduchej topológii s cieľom precvičit si black hole techniku.

Klíčová slova: DoS, Sítě, BGP, Black-holing

ABSTRACT

In the thesis it is explored how Denial of Service attacks are usually performed, what techniques are used and some of the popular tools used to conduct attacks. On the other side we looked at the commonly used mitigation methods with both network operators and end users. We attempted to automate creating a testing lab and tried out performing an attack in a simple topology with the aim to exercise the black hole technique.

Keywords: DoS, Networks, BGP, Black-holing

TABLE OF CONTENTS

INTRODUCTION

In this day and age, there has probably been nothing else considered as inevitable and impactful on one's personal or professional life, media, politics, culture or science as reliable Internet connection, allowing seemless interaction and access to information. Networking infrastructure powering the Internet has been the target of various attacks almost since day one. While the early network interconnections had been based on mutual trust and strong sharing spirit for advancement of humanity, with increasing amount of participants from different backgrounds (essentially democratization) and proliferating deeds of mischief, trust could no longer be taken for granted. Although trolling has always been an integral part of the Internet culture, it had not taken long until criminals found their way into the online world with the intention of taking advantage of lacking legislation and collecting easy (though huge) economic profit. Furthermore, another popular activity has become causing real harm to anybody on the way with the sense of lawlessness.

It is this willful damage in its many shapes and forms brought upon the Internet service providers and users that our thesis deals with. What we have commonly observed as being used in the wild in recent years may be called attack 'professionalizing', by which we mean that the tools typically available 10 years ago cannot match to practically anything you can get ready in minutes to cause real harm today. Since the networks compose the medium of communication, it is essential they remained secured and protected, which in fact poses a challenging task for us of coming up with new solutions to rapidly evolving problem.

On the side of the user, unavailability of the service often means inability to proceed with fulfilling their tasks. For the provider, an attack that cripples the network or its resources means inability to provide the promised service.

We would like to take a look on some of the conventional methods and tools which attackers resort to as well as methods and tools both users and network operators may (and should) utilise to protect the network. The main objectives of our thesis aim at researching key ways of DoS attack mitigations and performing a DoS attack and a BGP blackhole reaction simulation.

Our thesis comprises two main parts - theoretical and practical. The theoretical part discusses relevant context, different types of attack methods and tools, mitigation attack methods and tools. The practical part describes a simulated simple topology and

an attack in lab conditions. In the first two sections of the theoretical part, we were focused on attack definitions and providing context. Section 3 attempts to categorise attack methods based on several metrics and explain some of the mentioned. The fourth section presents a list of some of the popular DoS tools that are available. Next on, the section 5 traces various ways to mitigate attacks, both on the user and provider level and the final section of the theoretical part, section 6, marks out some of the tools that can be used to mitigate an attack. First of the practical part, section 7 gives an overview of the tools used to construct the lab infrastructure and configure the systems, as well as specifics of the configuration. It also focuses on setting up the infrastructure and the tools and processes to achieve a reproducible outcome. In section 8, we set up mitigation tools in preparation of an attack. In section 9 we go through the process of preparing attack tools. The final section 10 describes performing the attack itself.

I. THEORETICAL PART

1 DEFINITION

While denial of service can be caused in a multitude of different ways and impact any part of the stack, we are predominantly going to look at the ones pertaining Internet networks. First, we shall define what a denial of service (*DoS*) attack in fact *is* and we can achieve it by understanding what it does.

A DoS attack is an action that harms a *service* in such a way that it can no longer serve legitimate requests as a result of being occupied by bogus or excessive requests from the attacker.

A DDoS is a DoS that is distributed among many participant devices (and operators).

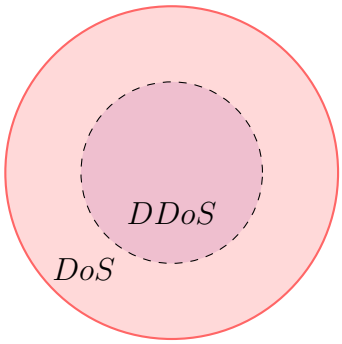

Figure 1.1 Illustration of relationship between DoS and DDoS attacks.

The devices participating are generally also victims in this, most of the attacks are performed with open DNS resolvers, home routers left to rot by vendors, misconfigured web services or IoT devices as involuntary participants. All one has to do is open Shodan and look for specific ports open (ports of protocols with good reflection ratio such as DNS, CLDAP, or SSDP), start probing and then reap the easy harvest. A quick search for devices running with port 123 (NTP) open is certain to return a mind-blowing number [1].

2 CONTEXT

In the last decade only we have witnessed many large DoS/DDoS attacks, some of them against critical infrastructure services like cloud hosting, DNS, git hosting services or even CCTV cameras. All of the attacks weaponized poorly managed endpoints, unpatched IoT devices or malware-infected-hosts-turned-botnet-zombies. The intensity and frequency have also been sharply increasing with the latest of attacks passing over the Tbps threshold (Akamai mitigated a 1.44Tbps DDoS in 2020 [2]), with data from Cisco Annual Internet Report showing that overall there was a **776%** growth in attacks between 100 Gbps and 400 Gbps from 2018 to 2019, and with predictions for the total number of DDoS attacks to double from 7.9 million in 2018 to 15.4 million by 2023 [3]. The question is: why?

The motifs will probably more often than not stay a mystery; however, a proliferation of DDoS-for-hire websites [4], even on *clearnet*[1)], points us to a plausible answer.

Somebody is making money selling abusive services that are being used for putting competitors out of business or just plain extortion. According to Akamai, extortion attacks have seen a widespread return, with a new wave launching in mid-August 2020 [5].

Akamai went on to note that DDoS attackers are expanding their reach across geographies and industries, with the number of targeted entities now being 57% higher than the year before that.

[1)] the surface web; i.e. not even attempting to hide

3 ATTACK METHODS

There are generally several different ways to categorise a method of attack:

By layers, in which the attacks are performed:

- Link layer
- Internet layer
- Transport layer
- Application layer

By the nature of their distribution:

distributed the effort is collectively advanced by a group of devices

1. deliberate
   (a) remotely coordinated devices (IRC C&C) - so called *voluntary botnets*
   (b) each operating their own computer, performing a premeditated operation in a synchronized manner
2. involuntary - hijacked devices

not distributed there is a single source of badness

By the kind of remoteness necessary to successfully execute the attack:

close-proximity (physical engagement, i.e. sabotage) requires physical presence in/near e.g. a datacenter, networking equipment (cutting cables, playing a pyro)

local network access such as over a WiFi access point or on LAN

remote such as over the Internet

By specific features:

- IP fragmentation
- SYN flood - a rapid sequence of TCP protocol SYN messages
- volumetric DDoS attack

- amplification attack (also called 'reflection attack')
  - memcached (up to 1:51200)
  - DNS with a formula [6]

$$R = answersize/querysize \tag{3.1}$$

  - SNMP (theoretically 1:650)
  - NTP
- exploits
  - 0days
  - simply running unpatched versions of software
- physical network destruction/crippling

3.1 IP fragmentation

This is the type of attack whereby an attacker attempts to send a fragmented payload (TCP, UDP or even ICMP) that the client is supposed to reassemble at the destination, by doing of which their system resources (CPU and mainly memory) would quickly get depleted, ultimately crashing the host.

It is usually necessary for IP datagrams (packets) to get fragmented in order to be transmitted over the network. If a packet being sent is larger than the maximum transmission unit (MTU) of the receiving side (e.g. a server), it has to be fragmented to be transmitted completely.

ICMP and UDP fragmentation usually involves packets larger than the MTU, a simple attempt to overwhelm the receiver that is unable to reassemble such packets, ideally even accompanied by a buffer overflow that the attacker can exploit further. Fragmenting TCP segments, on the other hand, targets the TCP mechanism for reassembly. Reasonably recent Linux kernel implements protection against this [7]. In either case, this is a network layer attack, since it targets the way the Internet Protocol requires data to be transmitted and processed.

3.2 SYN flood

To establish a TCP connection, a *three way handshake* must be performed.
That is the opening sequence of a TCP connection that any two machines - let's call

them TCP A and TCP B - perform, whereby TCP A wanting to talk sends a *segment* with a SYN control flag, TCP B (assuming also willing to communicate) responds with a segment with SYN and ACK control flags set and finally, TCP A answers with a final ACK [8].

Using `tcpdump` we can capture an outgoing SYN packet on interface `enp0s31f6`.

```
# tcpdump -Q out -n -N -c 1 -v -i enp0s31f6 "tcp[tcpflags] == tcp-syn"
```

A malicious actor is able to misuse the handshake mechanism by posing as a legitimate *client* (or rather many legitimate clients) and sending large number of SYN segments to a *server* willing to establish a connection (*LISTEN* state). The server replies with a [SYN, ACK], which is a combined acknowledgement of the client's request *and* a synchronization request of its own. The client responds back with an ACK and then the connection reaches the *ESTABLISHED* state.

There is a state in which a handshake is in the process but connection has not yet been ESTABLISHED. These connections are referred to as embryonic (half-formed) sessions. That is precisely what happens when an attacker sends many SYNs but stops there and leaves the connection hanging.

One particularly sly method aimed at causing as much network congestion near/at the victim as possible is setting a private IP address (these are unroutable, or rather, *should not be routed* over public Internet) or an address from deallocated space as the source IP address. For the sake of the argument suppose it is an address from deallocated space, what then ends up happening is the server responds with a [SYN, ACK] and since no response comes from an address that's not currently allocated to a customer (no response *can* come because nobody is using it), TCP just assumes that the packets lost on the way and attempts packet *retransmission* [9]. Obviously, this cannot yield a successful result so in the end the server just added onto the already congesting network.

Current recommended practice as per RFC 3704 is to enable strict mode when possible to prevent IP spoofing from DDoS attacks. If asymmetric routing or other kind of complicated routing is used, then loose mode is recommended [10].

That way the spoofed traffic never leaves the source network (responsibility of the transit provider/ISP) and does not aggregate on a single host's interface. For this to be a reality the adoption rate of the subject RFC recommendations would need to see

a proper increase.

As is true for anything, if countermeasures are set up improperly, legitimate traffic could end up being blocked as a result.

3.3 Amplified Reflection Attack

The name suggests this type of attack is based on two concepts: amplification and reflection. The amplification part pertains the fact that certain protocols answer even a relatively small query with a sizable response. The reflection part is usually taking advantage of session-less protocols. One such protocol is UDP with session-less meaning that hosts are not required to first establish a *session* to communicate, a response is simply sent back to the address that the packet arrives from (source address). Except for the fact that if a malicious player is not interested in communication but only wants to cause harm, a packet's source address does not necessarily have to, in fact *cannot* (from an attacker's point of view) correspond to the source address of their machine.

Since overwriting fields of the packet header (where the information important to routing reside) is trivial and there's nothing easier than supplying a UDP request with (either a bogus but more commonly) a victim IP address as the source address instead of our own that's present there by default. The response is then returned *back* - not to the actual sender, but simply according to the source address.
Since UDP has no concept of a *connection* or any verification mechanism, the response arrives at the door of the victim that has never asked for it - in the worst case an unsolicited pile of them.

This is why the three-way handshake is used with TCP, which was developed later than UDP, as it reduces the possibility of false connections.

The goal of the attacker is then clear: get the largest possible response and have it delivered to the victim (in good faith of the server even).

Spoofing the source address is done with the purpose of evading detection as a blocking or rate-limiting mechanism at the destination would likely identify any above-threshold-number requests coming from a single IP and ban them, thus decreasing the impact of the attack when the intent was to achieve congestion at the designated destination - the victim.

A perfect example for how bad this can get is unpatched or misconfigured `memcached`

software, that is very commonly being used as e.g. a database caching system and has an option to listen on UDP port. Cloudflare say they have witnessed amplification factors up to 51200 times [11].

As has already been mentioned in 3.2, this entire suite of issues could be if not entirely prevented then largely mitigated if the very sound recommendations of RFC 3704 gained greater adoption among ISPs.

Until then, brace yourselves for the next assault.

3.4 Slowloris

The principle of this attack is to first open as many connections as possible, aiming to fill the capacity of the server, and then keep them open for as long as possible by sending periodic keep-alive packets.
This attack works at the application layer but the principle can easily be reapplied elsewhere.

3.5 BGP hijacking

BGP is an inter-Autonomous System routing protocol, whose primary function is to exchange network reachability information with other BGP systems. Furthermore, this network reachability information "includes information on the list of Autonomous Systems (ASes) that reachability information traverses. This information is sufficient for constructing a graph of AS connectivity for this reachability, from which routing loops may be pruned and, at the AS level, some policy decisions may be enforced. This information is sufficient for constructing a graph of AS connectivity for this reachability, from which routing loops may be pruned and, at the AS level, some policy decisions may be enforced." [12].

BGP hijacking, in some places spoken of as prefix hijacking, route hijacking or IP hijacking is a result of a intentional or unintentional misbehavior in which a malicious or misconfigured BGP router originates a route to an IP prefix it does not own and Zhang et al. find it is becoming an increasingly serious security problem in the Internet [13].

3.6 Low-rate DoS on BGP

As shown by Zhang et al. in their "Low-Rate TCP-Targeted DoS Attack Disrupts Internet Routing" paper, BGP itself is prone to a variation of slowloris due to the fact that it runs over TCP for reliability. Importantly, this is a low-bandwidth attack and a more difficult one to detect because of that. Beyond the attack's ability to further slow down the already slow BGP convergence process during route changes, it can cause a BGP session reset. For the BGP session to be reset, the induced congestion by attack traffic needs to last sufficiently long to cause the BGP Hold Timer to expire [14]. On top of all that, this attack is especially hideous in that it can be launched remotely from end hosts without access to routers or the ability to send traffic directly to them.

4 ATTACK TOOLS

Believe it or not there actually exists a DDoS attack tools topic on GitHub `https://github.com/topics/ddos-attack-tools?o=desc&s=stars`.

4.1 HOIC

LOIC successor HTTP flooding High Orbit Ion Cannon, affectionately referred to as 'HOIC' is a *free software*[1)]tool which enables one to stress-test the robustness of their infrastructure by applying enormous pressure on the designated target in form of high number of requests. It operates with HTTP and users are able to send 'GET' or 'POST' requests to as many as 256 sites simultaneously. While it is relatively easily defeated by a WAF (see 5.5), the possibility to target many sites at once makes it possible for users to coordinate the attack, consequently making detection and mitigation efforts more difficult.

4.2 slowloris.py

`slowloris.py` is a python script available from `github.com/gkbrk/slowloris` that is able to perform a slowloris attack. It seeks to extinguish file descriptors needed for opening new connections on the server and then keeping the connections for as long as it can.
Legitimate requests cannot be served as a result, since there is no way for the server to facilitate them until resources bound by bogus requests are freed, i.e. the attack ceases to be.

4.3 Metasploit Framework

Metasploit is a penetration testing framework with an open source community version and a commercial version (Metasploit Unleashed) available. It enables security researchers to automate workflows of probing vulnerable services or devices via use of so called modules - smaller programs with definable inputs that perform prefined actions. Modules are often community-contributed and one can even write a module ourselves.a The SYN-flooding funtionality has been implemented - `aux/synflood` an auxiliary module. Auxiliary modules do not execute payloads and perform arbitrary

[1)] free as both in freedom and free beer

actions that may not be related to exploitation, such as scanning, fuzzing and denial of service attacks [15].

4.4 Web browser

Depending on our point of view (more fittingly, our scaling capabilities), sometimes all that is needed to cause a denial of service is tons of people behind a web browser. Numerous requests quickly overload a small server, eventually causing it respond so slowly that the impact is indistinguishable from a DoS attack.

That is because in principle *a DoS attack* is practically the same thing as discussed above, the only difference is the malicious intent, imperceivable to a machine.

5 MITIGATION METHODS

Drastic times require drastic measures and since a DDoS attacks coming at us practically every other month classify as *drastic* quite easily, we're forced to act accordingly [5].

Still, it is more reasonable to prepare than to improvise, therefore the following write-up mentions of commonly used mitigation methods at different levels, from a hobbyist server to an e-commerce service to an ISP. The list is inconclusive and of course if reading this at a later date, always cross-check with the current best practices at the time.

5.1 Blackhole routing (black-holing, null routing)

Black-holing is a technique that instructs routers that traffic for a specific prefix is to be routed to the null interface, i.e. be dropped and is used to cut attack traffic before it reaches the destination AS.
Assuming the router is properly configured to direct RFC 1918 destined traffic to a null interface, traffic destined to the attacked network gets dropped, making the attacked network unreachable to the attacker and everyone else. Matter of factly, we actually conclude the DoS for the attacker ourselves.[16][17]

In case of a DDoS, the traffic is likely to come from all over the world [2]. The idea here is to announce to our upstream (ingress provider) that supports RTBH (remotely-triggered black hole) signalling (critical) that we do not need any traffic for the victim IP anymore. They would then propagate the announcement further and in no time we'd stop seeing malicious traffic coming to a victim IP in our network. In fact, we would not see any traffic coming to the victim anymore, because we just broadcast a message that we do not wish to receive traffic for it. For the entire time we're announcing it, the victim host stays unreachable.

We should make sure to announce the smallest possible prefix to minimise the collateral damage. Generally, a /21 or /22 prefix is assigned to an AS (the average prefix per AS being 22.7866 as of 11 May 2021 [18]) announcing a black hole for such a large space would likely cause more damage than the attack itself.

To reduce BGP overhead, prefixes are usually announced aggregated, with the exception of "a situation", such as when we wish to only stop receiving traffic for one IP

address. Smallest possible accepted prefix size tends to be /24 (which is still a lot) with average prefix size updated being 23.11 [19], however, some upstream providers might even support a /32 in case of emergency, effectively dropping traffic only for the victim.

When an attack hits, all we have to do is:

1. deaggregate prefixes
2. withdraw hit prefixes.

In case our upstream provider did not support RTBH and we could not lose them (e.g. the only one around), we could still make use of Team Cymru's new BGP-based solution that distributes routes to participating networks using only vetted information about current and ongoing unwanted traffic - the **Unwanted Traffic Removal Service** (UTRS). It is a free community service, currently only available to operators who have an existing ASN assigned and publicly announce one or more netblocks with their own originating ASN into the public Internet BGP routing tables.

If only there was a way to just shut down the bad traffic but keep the good one flowing[1)]!

Behold, this is what *selective black-holing* actually is. Some upstream providers define multiple different blackhole communities each followed by a predefined action on the upstream. One is able to announce to these communities as needed. Assume we would announce to a community that would in response announce the blackhole to Internet exchanges in, say, North America and Asia and but allow traffic coming from Europe, would be a perfect example of selective black-holing. This causes two things to happen. First, our customer's IP is still reachable from our local area (e.g. Europe) and since our fictitious customer mostly serves European customers that's fine. Second, outside of the prefedined radius (Europe in this exercise) any traffic destined for our network (of which the victim IP is a part of) is immediately dropped at the remote IXPs, long before it ever comes anywhere near our geographical area, let alone our network.

I believe this approach is superior to indiscriminate black-holing and, given it is reasonably automated and quick to respond, in combination with other mitigation methods it can provide a viable protection for the network.

[1)] other than scrubbing

5.2 Sinkholing

Moving on, this method works by diverting only malicious traffic away from its target, usually using a predefined list of IP addresses known to be part of malicious activities to identify DDoS traffic. False positives can occur more rarely and collateral damage is lesser than with black-holing but since botnet IPs can be also used by legitimate users this is still prone to false positives. Additionally, sinkholing as such is ineffective against IP spoofing, which is a common feature in network layer attacks.

5.3 Scrubbing

An improvement on arbitrary full-blown sinkholing, during the scrubbing process all ingress traffic is routed through a security service, which can be performed in-house or can even be outsourced. Malicious network packets are identified based on their header content, size, type, point of origin, etc. using heuristics or just simple rules. The challenge is to perform scrubbing at an inline rate without impacting legitimate users.

If outsourced, the scrubber service has the bandwidth capacity (either on-demand or permanently) to take the hit that we do not have. There are at least two ways to go about this - the BGP and the DNS way, we will cover the BGP one. Once an attack is identified, we stop announcing the prefix that is currently being hit, contact our scrubbing provider (usually automatically/programatically) to start announcing the subject prefix, receiving all its traffic (including the attack traffic), the scrubbing service does the cleaning and sends us back the clean traffic [20].

When performing the scrubbing in-house, we have to clean the traffic on our own appliance that has to have sufficient bandwidth (usually on par with upstream).

A poor man's scrubber:

- utilizing hardware accelerated ACLs on switches,
- switches can do simple filtering at *inline rate* (ASICs)
- this can be effective when attack protocol is easily distinguishable from real traffic
- hit by NTP/DNS/SNMP/SSDP amplification attack

We should be performing network analysis and once higher rates of packets with source ports of protocols known to be misused for DoS/DDoS start arriving to our network (such as 123 or 53), we start signalling to our upstream providers, since they can probably handle it better than us and have as much interest in doing so as us (or should).

Volumetric attacks sending traffic in smaller packet sizes will, however, still result in higher CPU utilization, especially on non-dedicated networking equipment.

One thing we should do no matter whether we are currently suffering an attack (and scrubbing it ourselves) is to rule out and drop and never receive traffic appearing to come from *our own network*, since such traffic could not exist naturally and is obviously spoofed.

Team Cymru has got now a long tradition of maintaining bogons lists called the **Bogon Reference**. Bogon prefixes are routes that should never appear in the Internet routing table. A packet with an address from a bogon range should not be routed over the public Internet. These ranges are commonly found as the source addresses in DoS/DDoS attacks.
Bogons are netblocks that have not been allocated to a regional Internet registry (RIR) by the Internet Assigned Numbers Authority (IANA) and Martian packets (private and reserved addresses defined by RFC 1918, RFC 5735, and RFC 6598 [16], [21], [22]).

To get help with bogon ingress and egress filtering, we should set up automated obtaining of updated and curated bogon lists via HTTP, BGP, RIRs and DNS from Team Cymru [23].

In case we have our own ASN, are connected directly at an IXP, have no RTBH support upstream and basically have no other choice, we just need to find out who is sending the malicious traffic and, if possible, drop the session and receive traffic from other peers.

5.4 IP masking

This is technique is widely used (e.g. CloudFlare flagship service), relying solely on not divulging sensitive information - in this case server IP - to attackers and the capacity of the *fronting* service to withstand the attack due to having access to more badwidth than the attacker can produce. All traffic - including potentially harmful traffic - flows through what is basically a giant proxy. However, before declaring it a net win for us,

it is important to acknowledge that it also comes with heavy privacy implications, as now some other service performs TLS termination in our behalf and **sees everything** (that was encrypted only *in transit* and is not additionally encrypted) that *anyone* sends us, before finally forwarding it back.

5.5 WAF

WAF - *Web Application Firewall* - is an appliance used to protect (as name suggests) web applications. In this day and age, this is especially necessary and enables system administrators to craft protection logic in one place and shield potentially vulnerable applications. This method works on the application layer of the OSI model and is commonly deployed as part of a web proxy or a module of a web proxy, which means network layer attacks cannot be handled in this way. While not negligible, as always, it is crucial to not have any assumptions and know exactly what *layer* of protection using of WAF brings.

Generally or at least as per CBP (current best practices), applications are not deployed with ports exposed directly to the Internet. A sane approach of having access to resources *proxied* yields multiple possibilities in terms of authentication/authorization and protection scenarios and also several ways to more effectively use resources available. For one, where any web content *caching* is required, it is easily achieved with a *caching* proxy server. It commonly also enables specifying custom access policies.

There are also hosted (cloud) WAF offerings, however, they come with exactly the same privacy implications as IP masking solutions (see 5.4).

5.6 Rate-limiting

As a general precaution, it is sane to limit number of connections a client is able to make in a predefined amount of time (based on the requirements of the service). The same applies to a limit on how many connections a client can have open simultaneously, which can even prevent Slowloris (see 3.4). Rate-limiting is usually set either on a proxy or a WAF, but some form of rate-limiting can even be built into an app.

A well known rate-limiting pluggable solution that can be used with SSHd, HTTP or multitude of other endpoints is `Fail2Ban`.

5.7 Decreased-TIME_WAIT connection closing

This can help withstand a situation when conntrack table fills up and the server refuses to accept any new connections. There is absolutely no reason to keep connections in the conntrack table long after they become inactive. The Linux kernel's NetFilter actually has a scrubbing mechanism, that is supposed to be getting the conntrack table rid of the timed-out entries. Except practice shows they can linger for much longer than necessary.

When dealing with massive amounts of traffic it is very reasonable not only to increase the size of the conntrack table (memory trade-off), which is the generally recommended solution, but also to decrease the TIME_WAIT timeout to force-evict connections that have stopped sending data. It is also an easy way to mitigate slowloris (see 3.4). More on the workings of conntrack in 6.1.2

Nginx is a widely used proxy. It uses two FDs (file descriptors) for each connection. The limit of max open FDs can indeed be increased easily, howerever, we might still just be delaying the inevitable (FD exhaustion) and inefficiently wasting precious compute resources needed when an attack comes. If Nginx is unable to allocate FDs necessary to track a connection, the connection attempt will fail. By resetting connections that timed out we prevent such a situation from occurring easily. In Nginx this is set with a single line: `reset_timedout_connection on;`

6 MITIGATION TOOLS

No tools are going to remedy for a bad design decision and that applies equally to physical and Internet engineering.

6.1 Firewall

No matter the specific implementation, it is presumably safe to say that any firewall is better than no firewall.

There are two main types of firewalls:

- software,
- appliance (hardware-accelerated).

A software firewall is just another program running on the operating system, apart from the fact that it is typically running with system-level privileges. It can be run on a general-purpose computer. In fact most of the consumer-grade operating systems nowadays incorporate or by default enable a firewall solution.

In contrast, an appliance firewall is a dedicated piece of hardware purpose-build specifically for the sole role of behaving as a firewall and is typically running a custom and very minimal operating system and no userspace programs. Usually the system does not have a userspace, since it is vendored to run as an appliance.

6.1.1 Software firewall

Solutions available as software firewalls are typically specific to a given operating system.

Usually, there exist several tools that enable communication with the core implementing the logic, commonly by a way of embedding deeply in the networking stack of the OS or utilizing kernel APIs. In Linux distributions, the Linux kernel is the one that sees all. Each packet arriving at the network interface is inspected by the kernel and a decision is made regarding it.

Historically, `ipset` and later `iptables` used to be the de-facto standard, however, a more recent successor emerged quite some time ago and is replacing (in modern distributions has replaced, although backward compatibility has been preserved) the former two - the `nftables` tool.

6.1.2 Netfilter

The Linux kernel subsystem named `Netfilter` is part of the Internet protocol stack of the kernel and is responsible for packet manipulation and filtering [24]. The packet filtering and classification rules framework frontend tools `iptables` as well as the newer `nftables` can be interacted with via a shell utility and since they also expose APIs of their own, it is common that they have graphical frontends as additional convenience as well, most notably `firewalld`, which can be used in conjunction with both of them.

Although newer versions of the Linux kernel support both `iptables` and `nftables` just the same, only one of them can be used at a time. This can be arbitrarily changed at runtime, a reboot is not necessary since they are userspace tools) and interact with the kernel using *loadable kernel modules*.

Part of the `Netfilter` framework responsible for connection tracking is fittingly named Conntrack. Connection, or a *flow* is a tuple defined by a unique combination of source address, destination address, source port, destination port a and the transport protocol used. Conntrack keeps track of the flows in a special fixed-size (tunable[1)]) in-kernel hash table structure with a fixed upper limit.

On Linux devices functioning as router devices (especially when you add NAT to the mix) a common issue is the depletion of space in the conntrack table. Once the maximum number of connection is reached, Linux simply logs an error message `nf_conntrack: table full, dropping packet` to the kernel log and "all further new connection requests are dropped until the table is below the maximum limit again." [25]. That, as Westphal further notes, is indeed very unfortunate, especially in DoS scenarios.

Unless the router also functions as a NAT, this can be remedied in two ways: decreasing the timeout until an established connection is closed and decreasing the timeout until an inactive connection in the TIME_WAIT state is evicted from the conntrack table. By default, the TIME_WAIT timeout is several hours long and leaves the router

[1)] via `net.netfilter.nf_conntrack_buckets`

vulnerable to packet floods or Slowloris.

Netfilter is here to help again with conntrack treating entries that have not (yet) seen two-way communication specially – they can be evicted early if the connection tracking table is full. In case insertion of a new entry fails because the table is full, "...the kernel searches the next 8 adjacent buckets of the hash slot where the new connection was supposed to be inserted at for an entry that has not seen a reply. If one is found, it is discarded and the new connection entry is allocated."[25]. Randomised source address in TCP SYN floods becomes a non-issue because now most entries can be early-evicted because the TCP connection tracker sets the "assured" flag only once the three-way handshake has completed.

In case of UDP, the assured flag is set once a packet arrives after the connection has already seen at least one packet in the reply direction, that is the request/response traffic does not have the assured bit set and can therefore be early-dropped at any time.

6.2 FastNetMon DDoS Mitigation toolkit

Originally created by Pavel Odintsov, this program can serve as a helper on top of analysis and metric collection tools, evaluate data and trigger configurable mitigation reactions [26], [27], [28].

FastNetMon can run on most popular architectures an several different general-purpose and specialised platforms such as Linux distributions, VyOS, FreeBSD or Mikrotik devices. Most of the program is written in C++ but it uses many C libraries for additional functionality and also Perl install scripts. It has got a command line client and an API and analysis server. Its detection engine is able to identify many types of flood attacks, fragmentation and amplification attacks. The metrics can be exported into an InfluxDB engine for visualization. It is capable of interacting directly with BGP-enabled equipment and even supports Flow Spec.

II. PRACTICAL PART

7 INFRASTRUCTURE SET-UP DESCRIPTION

The testing was performed in a virtual lab comprised of five virtual machines (VMs) running on a KVM-enabled Fedora 34. Since the expectation was to frequently tweak various system settings of the guests (VMs) as part of the verification process, we decided to take the *infrastructure as code* approach. Every piece of infrastructure - down to the details of how many virtual CPUs are allocated to a host, what is the disk size and the filesystem, etc. - is declared as code, can be versioned and used to provision resources.

The industry standard tool `Terraform` was chosen due to a broad support of infrastructure and providers, great documentation, large user base and the tool being open source.

For bootstrapping, `cloud-init` has been used mainly because of the fact that it can integrate with terraform quite smoothly, works on many Linux distributions and allows us to pre-setup things like copy over SSH pubkeys so that a secure connection can be established right after first boot, set VM hostname, locale, timezone, add users/groups, install packages, run commands and even create arbitrary files, such as program configurations.

The disk sizes of the VMs were determined by the size of their base image. The VM naming convention is specified as follows: a prefix `r_` for routers and `h_` for other hosts, in our case the attacker, victim and defender machines.

7.1 VM specifications

Table 7.1 VM specifications

VM name	vCPU(s)	RAM	disk space	net ifaces	operating system
r_upstream	1	768MB	4.3GB	outer,DMZ	Fedora 33
r_edge	1	768MB	4.3GB	DMZ,inner	Fedora 33
h_victim	1	768MB	11GB	inner	CentOS 8
h_attacker	1	1GB	5.37GB	outer	Fedora 34
h_defender	1	1GB	5.37GB	DMZ	Fedora 34

The inner (our edge) and the upstream (our transit provider) routers are each part of different *AS*. They are directly connected and communicate using BGP. The outer router and the inner router are BGP peers.

We assume our upstream provider supports RTBH signalling. In this scenario the attacker is directly connected to our UPSTREAM router, and while in reality there would probably be a greater distance between us and them, this is fine for our simulation purposes, since malicious traffic will be cut before it reaches us.

If our upstream provider did not support RTBH signalling, in case we were attacked we could still use a scrubbing service but it is preferred that such a provider is picked that has the RTBH capabilities.

The cunning plan is to watch for traffic anomalies, when the attack comes detect it as soon as possible, trigger a reaction (announce a black hole), wait for some time, withdraw the black hole if the attack is gone and reintroduce it if it is still going on. Automatically, of course.

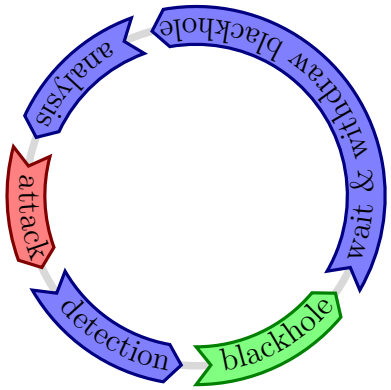

Figure 7.1 The Cunning Plan

Initially, two approaches for setting up infrastructure were considered. While both proposed to use KVM and `Terraform` with `libvirt` provider [29], the first one planned to continue configuring the VMs with `cloud-init`, which is compatible with any GNU/Linux distribution. The finishing touches would be done using `ansible`. The second one, on the other hand, considered a newer technology - Fedora CoreOS, which uses a different paradigm where hosts are only customised on initial boot via `ignition` (although he configuration format used is called *Butane*, which uses YAML and is then transpiled to Ignition's JSON) and ran as configured, by default without a separate install disk. This provides the much needed system immutability during runtime but also makes it more difficult to configure. Configuration changes *can* be done during runtime, however, the CoreOS provisioning philosophy teaches that there is no need to deploy once and endlessly configure (by hand or e.g. with `ansible`) when boot times are short (Ignition runs before the userspace begins to boot) and the system is as minimal as possible (container-like).

approach 0:

- KVM
- `terraform` with `libvirt` provider
- `cloud-init`
- `ansible`

approach 1:

- KVM
- `terraform`
- `ignition`
- Fedora CoreOS

We decided to go with the approach 0 as we felt it was more appropriate for our use case, which involves relatively lot of configuration.

VMs required:

- victim
- router - inner
- router - edge
- attacker
- defence machine

See tab. 7.1 for details.

To reiterate, simulating multiple physical devices performing different roles (routing, attacking, playing victim) in our attack-defence/mitigation scenario has been achieved by creating a test-lab virtual infrastructure.
The tried-and-true way, state-of-the-art Linux kernel-native virtualization solution has been chosen to tackle the hypervisor task - the KVM technology.

Testing has been performed on our personal laptop - Dell Latitude 5480 machine equipped with a ULV dual-core Intel i5 Core 6300U processor with `mitigations=off`, 24GB (8+16) of RAM and a 512GB SATA SSD (TLC).

The host operating system from the perspective of VMs was `Fedora 34`. Both `updates` and `updates-testing` repositories have been enabled, which allowed us to use latest (at the time) stable Linux kernel Fedora had to offer directly without too much of a hassle, as of the time of writing in version `5.11.20`.

File system in use on the host was Btrfs on top of LVM (LUKS+LVM to be precise) and a Btrfs subvolume has been created specifically for the libvirt storage pool for better logical isolation (subvolumes are omited during snapshotting etc.; that way the lab does not end up in system snapshots. Since all of the system images for our VMs have been downloaded in a QCOW2 format, the CoW (Copy-on-Write) feature of Btrfs has been turned off for the subject subvolume, just as recommended in the Brfs Sysadmin Guide [30] for improved storage performance (and decreased flash wear).

Notably, the system has also been using the `nftables` backend of `firewalld`, for which, luckily, `libvirt` was already prepared and played nice together.

The whole infrastructure code resides in a git repository available at `https://git.dotya.ml/mirre-bt/tf-libvirt`.

7.2 Terraform and Cloud-Init

Thanks to the `libvirt` provider for Terraform, VMs could easily be brought up and torn down. Terraform works by tracking state of the resources that together create the infrastructure, which can be described in numerous ways. Every resource that is absent needs to be created. Terraform uses a *plan* of the infrastructure as it wants to *apply* changes to the current *state*. Every VM or a network, every backing storage is a resource that is managed for us, which is very convenient especially for testing.

The initial boot configuration has been performed by Cloud-Init [31]. It enabled us to configure users with SSH public keys to secure further connections, install packages and create arbitrary files. Scenario-role-specific configuration was made really easy thanks to this tool.

7.3 Ansible

What has not been prepared by either Terraform or Cloud-Init was left to Ansible. It was used to deploy configurations of GoBGPd and FastNetMon, as well as to install and enable services.

8 MITIGATION TOOLS SET-UP

8.1 FastNetMon

An open-source DDOS mitigation toolkit named `fastnetmon` was picked to serve as an attack detection tool. It supports analysing traffic from multiple different exporter types, including Netflow (v5 and v9), sFlow and port mirrors. We tried the installer from projects website, however, the installation did not appear to succeed. Therefore, we decided to build it from sources. That is when we uncovered some curious information.

The project's master branch on GitHub, where it is hosted, appears to have been modified on 22 June 2016 [32] [26]. Since we knew about some propagation activities [33] and we found the repo as "updated" on 17 Feb 2021 (that is what GitHub shows when e.g. a description is updated) [34] at the same time as we saw the last commit pushed with a 2016 date, we went digging over the project's forks to find out what happened to the project. They perfectly preserve it, from a point in time when the "fork" (essentially a clone) was created, up until the latest updates that have either been integrated from upstream or changed by the owner of the fork. While some forks have been abandoned a long time ago, several more showed the same tree with the exact same commits, referring to a common history. Furthermore, the one thing giving away what apparently happened (not why) with almost certainty is the Pull Requests tab. It showed some 164 Closed PRs, some of them *merged* as late as 23 Dec 2020 [35]. That is, the history *has* indeed been overwritten and we have no information as to why, but that was also a reason why we chose to use one of the latest updated forks [36] [27] as a base for our work [28] instead of the original upstream. The claim that the history has been overwritten is also supported by an earlier grab (snapshot) of the project by the Internet Archive's awesome *Wayback Machine* [37], the rest have been triggered by us to help support the arguments in case anything changed.

Ad setup itself, several changes had to be made to the project to make it even compile on recent hosts (Fedora 33/34, Arch Linux). A Drone CI build job has been set up to help make sure nothing breaks with our changes. Fastnetmon needs quite a lot of dependencies for its full functionality and the way to go about it in the project was using custom versions of the third party libraries to link against, all downloaded using the Perl install script mentioned earlier. While that might seem like a sound idea, we feel in the long run it is always better to use a reasonably recent distribution with sizable repositories and active community that, combined, are able to provide for

most developer needs. The only major overhaul that had to be done was patching the `CMakeLists.txt` file to instruct CMake/Make to use system locations to look for headers and (dynamic) libraries (`.so` files) instead of download location from the install script. Further, to accomodate using newer version of `nDPI` (Open Source Deep Packet Inspection Software Toolkit) that aids FastNetMon with traffic sorting (if enabled), an interface and `fast_dpi.h` header had to be updated. The history of all changes can be found in the repository at `https://git.dotya.ml/wanderer/fastnetmon-ng`.

For building and deployment to the defender host, an Ansible role has been created. It makes sure that FastNetMon is built correctly, installed and its `systemd` service is *enabled* (at boot time) and started.

8.2 GoBGPd

We attempted to peer the two router VMs, however, we were not able to pin down the proper configuration that would allow us to define a community tied to a black-holing action.

8.3 Netflow

FastNetMon supports collecting Netflow metrics, therefore the edge router has been configured to listen on the upstream interface and to send traffic information to this FastNetMon's collector. The ansible role `github.com/juju4/ansible-fprobe` has been edited and used to deploy the configuration to the edge router.

9 ATTACK TOOLS SET-UP

When considering the way to simulate an attack locally, we weren't primarily looking for a tool, which would enable a decentralised (the first "D" of DDOS) attack, instead the objective was mainly to congest the weakest link, which would happen to live inside our network (that's why we're concerned in the first place).

9.1 slowlorispy

With this python script there are a couple of flags to tweak the intended behaviour. Flag `-p` sets the port we want to target, `-sleeptime` allows us to rate-limit the tool so that we open as many connections as possible without getting banned. The `-s` flag specifies the number of sockets we want to open.

9.2 iperf3

This tool works in server-client mode. To receive traffic, it `runs in server mode` and listens on port 5201 by default, although both the traffic direction and port and multitude of other parameters can be configured. To listen on the victim server, we simply ran `iperf3 -s -P 8` to receive 8 parallel streams. Then on the client - the attacker - we entered `iperf3 -c {server ip} -P 8` to send 8 parallel streams to the victim. This filled the available link bandwidth but, unfortunately, we haven not observed FastNetMon ban action even though the threshold has been crossed.

10 PERFORMING AN ATTACK

As mentioned previously, the attack was planned to be performed in the controlled environment of a virtual lab with practically no natural traffic occuring, that would generally be present on a reasonably large ISP network and thus with essentially no load on the network equipment/virtual devices. However, it should not have had any major impact on the results of the subject attack and our chosen way to mitigate.

As is the case when using e.g. a selective blackhole technique, the suspect traffic first has to be identified as highly abnormal/malicious, either by analysing network metrics over a period of time collected by sFlow or Netflow protocols, or by direct packet capture and inspection aided by tools such as nDPI. The analysis and detection mechanism in our scenario is left to FastNetMon and so is the reaction (mitigation) logic.

With the first attack we performed, we basically attempted to naively overflow the conntrack table of our server host. We were not able to extinguish connections on the server using `slowloris.py`, presumably because the inactive connections were quickly being closed. While the number of used sockets steadily grew, after about 5000 this tool was not able to open any new connections and the server worked fine. The key was to set the `net.netfilter.nf_conntrack_tcp_timeout_time_wait` parameter to a lower value, such as 10 or 5 and the `net.netfilter.nf_conntrack_tcp_timeout_established` parameter to somewhere between 300 and 1200 (we chose 600). Both values are seconds and they are set either in the global *sysctl* file (`/etc/sysctl.conf`) or they can be placed in an arbitrary complementary file inside the sysctl include directory at `/etc/sysctl.d/`. The *TIME_WAIT* timeout value affects for how long the connections in the TIME_WAIT (see 5.7) TCP state are kept before they are completely cleared and stop consuming resources. The ESTABLISHED timeout merely sets for how long the *established* connections are kept in the ESTABLISHED state before processing them further.

As per running FastNetMon attack traffic detection, we have experienced unexpected issues in the form of FastNetMon incorrectly reporting 0pps for both outbound and inbound traffic. We were not able to uncover the root cause of the issue, which might very well even be attributed to a configuration error. Due to this and the fact that we could not establish BGP peering in our virtual lab, neither the other attack nor the mitigation could be properly performed.

CONCLUSION

Recent past has seen an immense increase in the number of DoS attacks of all kinds. A large part of these attacks flood service resources and ultimately end up blocking or delaying responses for legitimate user requests. As we touched on in the beginning, the motivations for these attacks can range all from a business model with extortion plans to hacktivism to simply somebody choosing a wrong place to have fun.

The goal of our work was to describe some of the popular types of DoS attacks, including DDoS, casually used attack methods, techniques and tools most popular and most widely used among attackers to annoy Internet users, harm businesses and worry Internet service providers with small to medium capacities. We have dived a little into the workings of several potential attack vectors, but have not stayed there.

Further in the theoretical part, we also outlined various mitigation methods readily available to network operators and end users alike, along with their scope and reach. Several pros and cons of black-holing, selective black-holing, scrubbing and rate-limiting were considered in section 5 - Mitigation methods. Next, we looked at some of the concrete tools that aid mitigating DoS attacks.

In the practical part, we set out to build a virtual lab using tools like Libvirt, Terraform, Cloud-Init and Ansible on top of KVM in an automated manner by applying the *infrastructure as code* principles. The virtual lab was running on a Fedora 34 host, each virtual machine has been provisioned using Terraform, pre-configured using Cloud-Init and further configured with Ansible after the initial configuration has finished. We explored setting up both the mitigation tools used to protect Internet networks and tools used by adversaries.

Finally, we have attempted to perform several attacks in our controlled virtual environment, which has been described in the practical part of this work. The attempts were partially successful in that our proposed mitigation methods showed that a certain kind of attack *can* easily be mitigated and the unhappy consequences averted.
Sadly, in the next part of attack simulation we have arrived at unexpected results, where we were not able to fully simulate the black hole propagation among ASes due to configuration inconsistencies, presumably on multiple levels.

This, as well as improving and potentially reworking the virtual lab stays as a challenge for us for the future, and I believe performing these attack simulations could aid us in better understanding the threats and preparing for what we *surely are* going to face.

REFERENCES

[1] Shodan: NTPd devices. https://www.shodan.io/search?query=ntpd, March 2021, [online] Accessed: 2021-03-06.

[2] Emmons, T.: PART I: RETROSPECTIVE 2020: DDOS WAS BACK – BIGGER AND BADDER THAN EVER BEFORE. https://blogs.akamai.com/2021/01/part-i-retrospective-2020-ddos-was-back-bigger-and-badder-than-ever-before.html, January 2021, [online] Accessed: 2021-05-03.

[3] Cisco: Cisco Annual Internet Report (2018–2023) White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, March 2020, [online] Accessed: 2021-05-02.

[4] Santanna, J. J.; Vries, J. D.; Schmidt, R.; et al.: Booter list generation: The basis for investigating DDoS-for-hire websites. *Int. J. Netw. Manag.*, volume 28, 2017, doi:10.1002/nem.2008.

[5] Emmons, T.: 2021: VOLUMETRIC DDOS ATTACKS RISING FAST. https://blogs.akamai.com/2021/03/in-our-2020-ddos-retrospective, March 2021, [online] Accessed: 2021-05-03.

[6] Akamai: DNS Amplification Attacks and Truncated Responses. https://blogs.akamai.com/2015/06/dns-amplification-attacks-and-truncated-responses.html, June 2015, [online] Accessed: 2021-04-03.

[7] The kernel development community: Linux Networking Documentation » SNMP Counter. https://www.kernel.org/doc/html/latest/networking/snmp_counter.html#tcp-retransmission-and-congestion-control, [online] Accessed: 2021-05-10.

[8] Transmission Control Protocol. Technical report 793, Internet Engineering Task Force, September 1981, doi:10.17487/RFC0793, Also available as https://datatracker.ietf.org/doc/html/rfc793.

[9] Sargent, M.; Chu, J.; Paxson, D. V.; et al.: Computing TCP's Retransmission Timer. Technical report 6298, Internet Engineering Task Force, June 2011, doi:10.17487/RFC6298, Also available as https://datatracker.ietf.org/doc/html/rfc6298.

[10] Baker, F.; Savola, P.: Ingress Filtering for Multihomed Networks. Technical report 3704, Internet Engineering Task Force, March 2004, doi:10.17487/RFC3704, Also available as `https://datatracker.ietf.org/doc/html/rfc4271`.

[11] Cloudflare: Memcached DDoS Attack. `https://www.cloudflare.com/en-gb/learning/ddos/memcached-ddos-attack/`, [online] Accessed: 2021-05-03.

[12] Rekhter, Y.; Hares, S.; Li, T.: A Border Gateway Protocol 4 (BGP-4). Technical report 4271, Internet Engineering Task Force, January 2006, p. 4, doi:10.17487/RFC4271, Also available as `https://datatracker.ietf.org/doc/html/rfc4271`.

[13] Zhang, Z.; Zhang, Y.; Hu, Y.; et al.: Practical defenses against BGP prefix hijacking. In *CoNEXT '07*, 2007, doi:10.1145/1364654.1364658.

[14] Zhang, Y.; Mao, Z. M.; Wang, J.: Low-Rate TCP-Targeted DoS Attack Disrupts Internet Routing. In *NDSS*, 2007, doi:10.1.1.137.5004.

[15] rapid7: Metasploit Framework. `https://github.com/rapid7/metasploit-framework`, 2021, [online] Accessed: 2021-04-03.

[16] Moskowitz, R.; Karrenberg, D.; Rekhter, Y.; et al.: Address Allocation for Private Internets. Technical report 1918, Internet Engineering Task Force, February 1996, doi:10.17487/RFC1918, Also available as `https://datatracker.ietf.org/doc/html/rfc1918`.

[17] Turk, D.: Configuring BGP to Block Denial-of-Service Attacks. Technical report 3882, Internet Engineering Task Force, October 2004, doi:10.17487/RFC3882, Also available as `https://datatracker.ietf.org/doc/html/rfc3882`.

[18] Huston, G.: Average prefix length. `https://bgp.potaroo.net/cgi-bin/plota?file=%2fvar%2fdata%2fbgp%2fas2%2e0%2fbgp%2daverage%2dprefix%2etxt&descr=Average%20prefix%20length&ylabel=Average%20prefix%20length&with=step`, [online] Accessed: 2021-05-11.

[19] Huston, G.: Average prefix size updated. `https://bgp.potaroo.net/cgi-bin/plota?file=%2fvar%2fdata%2fbgp%2fas2%2e0%2fbgp%2dupd%2davgprefsize%2etxt&descr=Average%20prefix%20size%20updated&ylabel=Average%20prefix%20size%20updated&with=step`, [online] Accessed: 2021-05-11.

[20] Akamai: DDoS Defense in a Hybrid Cloud World. `https://www.akamai.com/us/en/multimedia/documents/ebooks/`

ddos-defense-in-a-hybrid-cloud-world.pdf, 2021, [online] Accessed: 2021-05-03.

[21] Cotton, M.; Vegoda, L.: Special Use IPv4 Addresses. Technical report 5735, Internet Engineering Task Force, January 2010, doi:10.17487/RFC5735, Also available as https://datatracker.ietf.org/doc/html/rfc5735.

[22] Weil, J.; Kuarsingh, V.; Donley, C.; et al.: IANA-Reserved IPv4 Prefix for Shared Address Space. Technical report 6598, Internet Engineering Task Force, April 2012, doi:10.17487/RFC6598, Also available as https://datatracker.ietf.org/doc/html/rfc6598.

[23] Team Cymru: The Bogon Reference. https://team-cymru.com/community-services/bogon-reference/, [online] Accessed: 2021-05-02.

[24] Boye, M.: Netfilter Connection Tracking and NAT Implementation. https://wiki.aalto.fi/download/attachments/69901948/netfilter-paper.pdf, 2012, [online] Accessed: 2021-05-05.

[25] Westphal, F.: improvements to conntrack table overflow handling. https://netdevconf.info/2.1/papers/conntrack.pdf, April 2017, [online] Accessed: 2021-05-06.

[26] Odintsov, P.: FasNetMon - very fast DDoS sensor with sFlow/Netflow/IPFIX/SPAN support. https://github.com/pavel-odintsov/fastnetmon, 2021, [online] Accessed: 2021-04-13.

[27] Odintsov, P.: FastNetMon fork with preserved history. https://github.com/Wofbit/fastnetmon, 2021, [online] Accessed: 2021-04-13.

[28] Odintsov, P.; Mirre, A.: FastNetMon NG. https://git.dotya.ml/wanderer/fastnetmon-ng, 2021, [online] Accessed: 2021-05-04.

[29] Duncan Mac-Vicar P.: Terraform provider to provision infrastructure with Linux's KVM using libvirt. https://github.com/dmacvicar/terraform-provider-libvirt, 2021, [online] Accessed: 2021-04-08.

[30] The kernel development community: Linux Btrfs Sysadmin Guide. https://btrfs.wiki.kernel.org/index.php/SysadminGuide#Copy_on_Write_.28CoW.29, [online] Accessed: 2021-03-12.

[31] Canonical: The standard for customising cloud instances. https://github.com/canonical/cloud-init, 2021, [online] Accessed: 2021-04-09.

[32] The Internet Archive: Archived view of FastNetMon - very fast DDoS sensor with sFlow/Netflow/IPFIX/SPAN support. https://web.archive.org/web/20210330122630/https://github.com/pavel-odintsov/fastnetmon/tree/master, 2021, [online] Accessed: 2021-04-01.

[33] The Internet Archive: Archived view of FreeBSD forum thread on FastNetMon. https://web.archive.org/web/20210407104407/https://forums.freebsd.org/threads/fastnetmon-open-source-tool-to-detect-ddos-ddos.62032/, 2017, [online] Accessed: 2021-04-07.

[34] The Internet Archive: Archived view of GitHub search for FastNetMon. https://web.archive.org/web/20210330135951/https://github.com/search?utf8=%E2%9C%93&q=fastnetmon, 2021, [online] Accessed: 2021-04-02.

[35] The Internet Archive: Archived view of FastNetMon's closed Pull Requests. https://web.archive.org/web/20210329183006/https://github.com/pavel-odintsov/fastnetmon/pulls?q=is%3Apr+is%3Aclosed, 2021, [online] Accessed: 2021-03-29.

[36] The Internet Archive: FastNetMon - Archived view of Wofbit's fork with preserved history. https://web.archive.org/web/20210516225746/https://github.com/Wofbit/fastnetmon, 2021, [online] Accessed: 2021-05-16.

[37] The Internet Archive: Archived view of FastNetMon from January 2021. https://web.archive.org/web/20210111231449/https://github.com/pavel-odintsov/fastnetmon/, 2021, [online] Accessed: 2021-04-02.

LIST OF ABBREVIATIONS

Mpps/MPPS	*millions of packets per second*
pps/PPS	*packets per second*
ACL	*access-control list*
AS	*Autonomous System*
ASN	*Autonomous System Number*
DoS	*Denial of Service*
DDoS	*Distributed Denial of Service*
HTTP	*Hyper Text Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IoT	*Internet of Things*
IP	*Internet Protocol*
ISP	*Internet Service Provider*
IXP	*Internet Exchange Point*
LVM	*Logical Volume Management*
RFC	*Request For Comment*
SSH	*Secure Shell*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
ULV	*Ultra Low Voltage*
VM	*Virtual Machine*

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

1. ansible-gobgp-master.tar.gz
2. fastnetmon-ng-development.tar.gz
3. ansible-fprobe-master.tar.gz
4. tf-libvirt-main.tar.gz