BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

AUTOMATION OF DDOS ATTACK MITIGATION
AUTOMATICKÁ MITIGACE DDOS ÚTOKU

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Bc. PETER NAGY
SUPERVISOR VEDOUCÍ PRÁCE	Ing. MATĚJ GRÉGR, Ph.D.

BRNO 2018

Zadání diplomové práce/20619/2017/xnagyp01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Nagy Peter, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Automatická mitigace DDoS útoku**
Automation of DDoS Attack Mitigation
Kategorie: Počítačové sítě

Pokyny:
1. Seznamte a analyzujte nástroje pro konfiguraci sítě v zařízeních postavených nad systémem GNU/Linux (např. vyos, cumulus linux, vyatta, netx ...)
2. Navrhněte systém pro automatickou rekonfiguraci sítě na základě výstupu z nástroje pro detekci DDoS útoku.
3. Navržený systém implementujte
4. Otestujte automatickou rekonfiguraci sítě v prostředí sítě VUT.
5. Vyhodnoťte výsledky a diskutuje možná rozšíření.

Literatura:
- Marques P, Sheth N, Raszuk R, Greene B, Mauch J, McPherson D. RFC 5575: Dissemination of flow specification rules. August. 2009.
- J. Haas, Ed., "Clarification of the Flowspec Redirect Extended Community," RFC 7674, IETF, Oct 2015.
- Wei Yin Tay: Scalable DDoS mitigation using BGP Flowspec, [online], url: https://conference.apnic.net/data/37/apricot-2014-wei-yin-scalable-ddos-mitigation-using-bgp-flowspec_1393312254.pdf
- Cisco ASR 9000 Routing Configuration Guide, Release 5.2.x: Implementing BGP Flowspec.

Při obhajobě semestrální části projektu je požadováno:
- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Grégr Matěj, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstract

The aim of this thesis is automation of DDoS attack mitigation. This thesis provides an overview of GNU/Linux network platforms and different approaches for their configuration. The aim is to select a platform that could be extended to automate DDoS mitigation. DDoS attack types are explained as well. Selected methods for DDoS mitigation are described in more detail such as Remote Triggered Black Hole and BGP Flowspec. Existing tools like DDoS Defender and FastNetMon are used to detect a DDoS attack. NETX was chosen as target implementation platform. To communicate with devices, API or BGP protocol with Flowspec extension are used.

Abstrakt

Cieľom práce je automatizovaná mitigácia DDoS útokov. Táto práca sa zaoberá prehľadom jednotlivých GNU/Linux systémov používaných na konfiguráciu siete. Cieľom je výber platformy, ktorá môže byť vhodne rozšírená za účelom automatizácie mitigácie DDoS útokov. Súčasťou práce je takisto prehľad jednotlivých typov DDoS útokov. Vybrané mitigačné metódy Remote Triggered Black Hole a BGP Flowspec sú bližšie popísané. Nástroje DDoS Defender a FastNetMon sú používané na detekciu DDoS útokov. Platforma NETX bola vybraná ako implementačná platforma. Komunikácia medzi zariadeniami prebieha pomocou API alebo protokolu BGP s použitím rozšírenia Flowspec.

Keywords

DDoS, NETX, automation, mitigation, network, configuration, BGP, Flowspec, filtration

Klíčová slova

DDoS, NETX, automatizácia, mitigácia, sieť, konfigurácia, BGP, Flowspec, filtrácia

Reference

NAGY, Peter. *Automation of DDoS Attack Mitigation.* Brno, 2018. Master’s thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Matěj Grégr, Ph.D.

Rozšířený abstrakt

Cieľom tejto práce je automatizovaná mitigácia DDoS útokov. Prvá časť práce, presnejšie kapitola 2 sa zaoberá prehľadom jednotlivých platforiem založených na GNU/Linux a ich konfiguráciou. Prvým vybraným systémom bol systém Vyatta, ktorý je založený na systéme Debian. Jeho konfigurácia je podobná systému Junos OS od spoločnosti Juniper. Neskôr prišlo k odlúčeniu niektorých ďalších systémov založených na Vyatta. Najznámejším zástupcom je systém VyOS, ktorý je ďalej vyvíjaný a spravovaný s pomocou komunity. Tento systém je často používaný ako firewall alebo VPN brána. Ďalším z testovaných systémov bol systém Cumulus Linux, ide o systém používaný hlavne v dátových centrách. OpenWrt nazývané aj LEDE je ďalšou z testovaných platforiem. Využíva klasické Linux jadro a je zameraný hlavne na menšie domáce smerovače prípadne bezdrôtové prístupové body. Samotné OpenWrt nevyžaduje špecifický hardware, je ho možné nainštalovať na bežne dostupné podporované domáce smerovače. Poslednou testovanou platformou je NETX vyvíjaný na Vysokém Učení Technickém v Brně (VUT). Jej výhodou je použitie dostupného hardwaru v kombinácii s niektorými štandardne dostupnými nástrojmi. Výhodou je Netc API, ktoré poskytuje možnosť jednoduchej a centrálnej konfigurácie celého systému. Jeho modulárna štruktúra dovoluje jednoduché pridanie nových modulov a rozšírenie funkcionality. Z testovaných distribúcií bola vybraná platforma NETX vďaka modulárnej štruktúre Netc API je možné jednoduché pridanie nových modulov, ďalšou výhodou bol aj už spomínaný vývoj priamo na VUT. Posledná časť tejto kapitoly je venovaná smerovaciemu programu BIRD, ktorý sa ďalej v tejto práci používa na smerovanie v rámci systému NETX.

Kapitola 3 sa venuje prehľadu rôznych typov DDoS útokov. Vytvorenie jednoduchého DDoS útoku, prípadne jeho nákup nie je v dnešnej dobe drahou záležitosťou. V kapitole je ďalej možné nájsť spôsoby akými sa maskujú dané útoky a okrajovo sa venuje takisto metódami používanými na odhalenie opísaných útokov.

V kapitole 4 je opísaný spôsob detekcie opísaných DDoS útokov. Sú opisované rôzne nástroje používané na detekciu, ako aj ich konfiguračné a detekčné možnosti. Najvýznamnejšími nástrojmi sú DDoS Defender od spoločnosti Flowmon a Fastnetmon, ktoré sú používané ďalej ako detekčné nástroje v tejto práci.

Mitigácia, alebo zastavenie útoku nasleduje po detekcií daného DDoS útoku. Spôsoby a nástroje použité na mitigáciu sú opísané v kapitole 5. Kapitola opisuje Access Control List (ACL) ako základný kameň pomocou ktorého je možné danú prevádzku filtrovať, prípadne zahadzovať. Ďalej sú opísané ďalšie možnosti mitigácie, ktoré poskytuje protokol BGP. Ide hlavne o Remote Triggered Black Hole (RTBH) a rozšírenie protokolu BGP nazývané BGP Flowspec. Flowspec sa používa na špecifikáciu sieťových tokov a umožňuje uskutočňovať rôzne ďalšie akcie na identifikovanom toku. Za účelom ochrany proti DDoS útokom je najzaujímavejšia možnosť zahadzovania sieťovej prevádzky. Koniec kapitoly sa venuje filtrácií sieťovej prevádzky priamo na sieťových kartách od spoločnosti Intel. Na pridávanie pravidiel, ktoré sa priamo nahrávajú do sieťovej karty sa používa nástroj `ethtool`. Pomocou vyššie opísaných metód je možné dosiahnuť mitigáciu DDoS útokov s priamym využitím sieťovej karty bez potreby zaťažovať filtráciou správ samotné jadro systému.

Nasledujúca kapitola 6 sa zaoberá samotným návrhom architektúry celého systému a jej následnou implementáciou. Implementácia je rozdelená do niekoľkých častí a sú zvolené rôzne prístupy. V prvej časti je opísaná implementácia modulu do systému NETX, presnejšie `hw-filter` modul do Netc API. Tento modul je zodpovedný za pridávanie pravidiel priamo na sieťovú kartu. V ďalšej časti sa práca venuje implementácii skriptov na prepojenie samotného detekčného nástroja, v našom prípade DDoS Defender alebo Fastnetmon, so samotným systémom NETX. Distribúcia týchto pravidiel je implementovaná pomocou

NETX Netc API pomocou ktorého je možné pravidlá priamo pridávať a samotný modul sa postará o pridanie na sieťovú kartu. Ďalšou možnosťou je použitie BGP Flowspec na distribúciu daných pravidiel. Pri distribúcii pomocou BGP Flowspec bolo potrebné modifikovať smerovací program BIRD, ktorý nedokázal zavolať externý skript v prípade obdržania novej správy.

Ďalšia časť práce, kapitola 7 sa zaoberá otestovaním implementovaného systému a zhrnutiu zistených poznatkov. Všetky z vyššie popísaných prístupov boli otestované v prostredí testovacej siete VUT. Hlavným poznatkom sú možné problémy, ktoré môžu vzniknúť pri pridávaní pravidiel na sieťovú kartu. Nie všetky typy pravidiel musia byť akceptované, čo môže viesť k následnej nestabilite celého systému.

V rámci dosiahnutých výsledkov je možné konštatovať, že navrhnutý systém je funkčný a je použiteľný na mitigáciu DDoS útokov. Problém môže nastať pri pridávaní pravidiel na sieťovú kartu, kde je potrebné byť oboznámený s filtračnými možnosťami používanej sieťovej karty.

Automation of DDoS Attack Mitigation

Declaration

Hereby I declare that this master's thesis was prepared as an original author's work under the supervision of Mr. Ing. Matěj Grégr, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

. . . .
Peter Nagy
May 23, 2018

Acknowledgements

I would like to thank my supervisor Ing. Matěj Grégr, Ph.D. for advice, patience, and a considerable amount of time spent on consultations. Furthermore, I would like to thank God and my family and friends for support.

Contents

Chapter 1

Introduction

Computer networks are nowadays complex which raises the problem of stability and security. Complexity and number of end nodes also increase the number of network equipment to be configured and maintained to stay secured. Unfortunately, security is often not implemented properly and large number of devices are exposed to Internet and misused for attacks, often Distributed denial-of-service (DDoS).

DDoS attacks are a rapidly growing problem. Attacks are getting bigger and more complex. To defend a network efficiently can be a problem.

To solve this issue, new mechanisms and network protocols were designed, such as Flowspec expansion to Border Gateway Protocol (BGP). Flowspec provides a mechanism to identify data flow using IP addresses, port numbers, and other parameters. These flow specifications can be sent to a gateway router using BGP protocol. The router can use Flowspec information to drop or rate-limit the flow, thus secure the network against DDoS attack.

The goal of this thesis is to reconfigure a computer network automatically in case of a DDoS is detected. This reconfiguration should be done on an open source networking platform and implemented and tested in Brno University of Technology (BUT) campus network. Communication between DDoS detection tools, such as DDoS Defender and network device should take place via a configuration Application Programming Interface (API) or BGP protocol with a Flowspec extension. This process should be fully autonomy and without any additional approval or configuration from the side of a network engineer. To gain the best performance and to not overload the device which provides routing, technologies like hardware filtration on a network interface card (NIC) should be used.

The thesis is structured as follows: Chapter 2 provides an overview of an open source networking based on GNU/Linux and its configuration. In chapter 3 the DDoS attacks, their history and impact on the network are explained. There is an overview of the DDoS attack types and masking methods. The next chapter 4 DDoS detection tools are discussed along with method used for detection. DDoS Defender and Fastnetmon are used in this thesis for detecting a DDoS attack. After detection, mitigation takes place in chapter 5. Remote Triggered Black Hole (RTBH) and BGP Flowspec are explained. Next technology used to mitigate a DDoS attack is Hardware filtration using rules offloading to a NIC. Chapter 6 talks about the architecture design and the whole system implementation. It is divided into a few parts according to used detection tool. NETX Netc API or BGP Flowspec is used for adding rules to NIC on NETX open source routing platform. Chapter 7 discuss testing and Chapter 8 is a conclusion and discusses the future improvements.

Chapter 2

Network configuration tools on GNU/Linux

Routers based on GNU/Linux are gaining popularity in networking due to price, scalability, and reliability. There are few popular open source projects which are focused in this area, for example, OpenWrt for home-based embedded devices are VyOs and pfSense for network security. Other platforms like NETX aimed at maximal performance and shaping are presented in this chapter.

2.1 Vyatta/VyOS

Vyatta system is a specific Debian based distribution used as a software-based virtual router. In addition to the standardized management console, similar to Junos OS from Juniper. Vyatta also provides a web interface and support for traditional Linux commands. There were two versions, Vyatta core, and Vyatta subscribers edition. Vyatta core was an open source version available to freely download and use. On the other hand, Vyatta subscription edition was a paid edition which added additional support as well as advanced web interface and full API access. In 2012, Vyatta was acquired by Brocade Communications Systems and renamed to Brocade Vyatta 5400 vRouter. However, vRouter was no longer open source based project. In 2017 vRouter was bought by AT&T and no longer provides support for its customers. Based on available information, it is only used internally in AT&T as a part of their internal network infrastructure [9].

There are several other platforms which were forked from Vyatta, such as VyOS and EdgeOS developed by Ubiquiti Networks.

As most of the presented tools, also VyOS is an open source operating system based on GNU/Linux, more precisely as mentioned above, it is based on Vyatta Core. It joins multiple applications such as OpenVPN, Quagga, and others under single management interface. Comparing to NETX platform, which focuses on high performance, VyOS is more focused on virtualization and broad hardware range support.

The remote management Application programming interface (API) or graphical interface, as well as multicast and MPLS support, are the most common requsted features among users.

Unlike OpenWrt or PfSense, VyOS is focused on being more like a traditional router based on available hardware as an extension of a Linux operating system. However, VyOS is not neglecting other usages such as firewall or VPN gateway [29].

2.2 Cumulus Linux

Cumulus Linux is one of the operating systems focused on data center switches [7]. It is based on Debian and offers capacity up to 100 Gbps. Some of the data center solutions from companies like Google or Facebook are using Cumulus Linux in their switches. However, Cumulus Linux is a paid solution paired with hardware from well-known network equipment manufacturers like Dell or HP.

Cumulus is designed for use on bare metal switches. However, it is available also as a virtual appliance Cumulus VM that enables to test the platform or to create development prototypes. It uses Quagga routing suite as an implementation of routing protocols such as OSPF, RIP, RIPng, and BGP. Thanks to its use and configuration of network interfaces and services based on Debian style, the configuration is easy for administrators familiar with Linux and allows using different scripting languages.

Although the platform management is controlled by Linux, its kernel is not involved in forwarding network communication. Cumulus comes with the `switchd` daemon which controls and configures the hardware Ethernet switching interfaces and performs the internal routing and switching. Consequently, the common Linux commands like those for interface inspection may not handle all traffic.

Considering the aim of this thesis, Cumulus Linux is not the right solution, because it is paid solution and it focused more on data center-class networking.

2.3 OpenWrt

Speaking of open source networking, OpenWrt or project LEDE must be mentioned. These two projects have been merged back together at the beginning of 2018 when both projects announced their reunification under the OpenWrt name [6]. It is open source GNU/Linux distribution for embedded devices providing a fully writable file system and package management tool `opkg`. Project also maintenances official software repository with thousands of different software packages.

OpenWrt main components are Linux kernel, `musl`, and `BusyBox`. The platform is dedicated mainly to the network embedded devices and provides many customization options such as installing new custom packages and configuring the system by any user needs. Several network services have been implemented from scratch within the OpenWrt project to minimize usage of hardware resources. For example, a minimalist single threaded web server `uHTTPd` or `opkg` package manager [4]. The whole OpenWrt configuration stands on UCI (Unified Configuration Interface) system [5] and comes with pre-installed web configuration interface LuCI.

Central configuration is stored into various files according the configuration areas. UCI provides unified configuration interface for whole supported services. UCI data model allows committing more changes at one time. UCI offers two configuration styles, human-friendly and programmable style. The following is an example configuration of network interface in human-friendly style as is stored in configuration file:

```
config interface ’vlan40’
    option proto ’static’
    option ifname ’eth0.40’
    option ipaddr ’172.20.1.11’
    option netmask ’255.255.255.0’
```

```
        option broadcast '172.20.1.255'
```

The same configuration in programmable style obtained by `uci show network.vlan40` looks like this:

```
network.vlan40=interface
network.vlan40.proto='static'
network.vlan40.ifname='eth0.40'
network.vlan40.ipaddr='172.20.1.11'
network.vlan40.netmask='255.255.255.0'
network.vlan40.broadcast='172.20.1.255'
```

As mentioned above, OpenWrt is designed for embedded devices. It supports a wide range of system architectures like MIPS or ARM. It usually means devices running OpenWrt has limited computing power. It is not designed to be a core routing device. Project repository contains Quagga and bird routing daemons.

2.4 NETX

NETX is an open source routing platform used for high-performance open source routing developed at the Brno University of Technology. Thanks to rich experience in high-performance networking, this product provides a massive set of routing features as well as traffic shaping capabilities. It can handle several full BGP tables and supports a lot of other networking protocols. Routing performance of such a system is up to 60 Gbps.

It is distributed as a complete solution, which includes hardware and software. NETX uses an operating system which runs on GNU/Linux distribution with kernel patches for performance improvements. It uses standard Linux daemons, which are often slightly updated or patched to provide high performance and availability. NETX provides a full-featured configuration API which can be used for remote configuration as well as it can be integrated into other network management systems [23].

NETX also has a cloud solution which runs on VMware virtualization platform. It can be used as a service or to backup the HW NETX system. In case of hardware failure, the cloud system platform can be automatically used, and traffic is routed to this system using BGP.

There are a few key features on which NETX is focused:

- **Performance** - High performance of an open source Linux based system allows NETX to forward gigabits per seconds as well as shape the traffic,
- **Simple management** - NETX provides powerful RESTful API access as well as cisco like command line interface (CLI) for configuration named netc. This platform focuses on simple cisco like configuration interface which is well known in network engineer community. It can be used to integrate with NetOPS automatization processes,
- **High Availability** - To gain the high availability, every NETX has a redundant main processing unit design and also second power supply installed. Link redundancy has to be considered regarding high availability. For this reason, NETX uses Virtual Router Redundancy Protocol (VRRP) protocol. This protocol provides a solution to

link failure as well as it allows redundant link usage. NETX is often used in a cluster. It provides even better availability in case of whole system failure,

- **Carrier Grade NAT** - Because of the limited IPv4 address space, NAT is often used to provide an IP connection to the Internet even in the corporate network. It also provides a particular layer of security, when attacking a device placed in an internal network is harder when NAT is used. NAT can be indeed difficult and intense for device hardware, but NETX supports NAT on full link speed,

- **Shaping** - Several QoS policies like FIFO or HTB are available. These policies can be set through API. Each interface supports several thousand QoS policies per interface.

There are a lot of other technologies and supported protocols not mentioned above. Such as multicast support, L3 protocols like DHCP, DNS with DNSSEC or VxLAN and MPLS. For example, most of the NETX systems installed, to gain maximal availability are installed redundantly, one hardware-based system and backup located on the cloud. NETX uses tested hardware configurations. It is the key to stability and high availability. A few of them are shown in the table 2.1 below. All data is available on NETX website [24].

Version	NETX X1120	NETX X2460	NETX Cloud
Dimension	1U	2U	Virtual
Power consumption	80W	220W	Depends
Network interfaces	2x SFP+, 2x 1G Base-T	10x 40G QSPF+	Virtual
Routing performance	20 Gbps	60 Gbps	60 Gbps
Routing table size	5 mil.	10 mil.	5 mil.
Carrier Grade NAT	20 mil. sessions	40 mil. sessions	20 mil. sessions

Table 2.1: NETX Versions.

Netc

Netc was created to simplify the router configuration using Cisco-like commands to make the whole system configurable on one place. Netc contains a few modules, and new modules can be quickly written. Support for autocomplete has also been added to improve the user experience. There is a possibility to commit the configuration and rollback when needed. The following example showns, a detailed interface info.

Command:

```
netx# show interface bond0.110 detail
```

Result:

```
Device:                 bond0.110
HW Address:             0c:c4:7a:87:2d:53
Driver, fw:             802.1Q VLAN Support-1.8N/A, fw:N/A
Oper status:            up
Link status:            20Gb/s Full
```

```
IP Address:          10.90.110.1/24
IP Address:          10.90.114.1/24
IP Address:          2a07:6881:0:ffff::1/64
IP Address:          fe80::ec4:7aff:fe87:2d53/64
MTU:                 1500
Description:         main-backbone

STATISTICS                    RX                          TX
                        total      per/sec          total      per/sec
Bytes,bits/s             6.2T      287.8k           10.9T       27.6k
packets                  8.5G       62.0            10.5G       24.3
multicast                3.4M        1.8             --          0.0
dropped                  0.0         0.0             1.0         0.0
errors                   0.0         0.0             0.0         0.0
fifo_errors              0.0         0.0             0.0         0.0
frame_errors             0.0         0.0             --          0.0
length_errors            0.0         0.0             --          0.0
over_errors              0.0         0.0             --          0.0
crc_errors               0.0         0.0             --          0.0
missed_errors            0.0         0.0             --          0.0
aborted_errors           --          0.0             0.0         0.0
carrier_errors           --          0.0             0.0         0.0
heartbeat_errors         --          0.0             0.0         0.0
window_errors            --          0.0             0.0         0.0
```

Netc REST API

Netc also includes a powerful REST API which makes all configuration options available remotely. API supports GET, POST and DELETE methods. There are a few examples which describes each method.

Get interface detail:

```
GET interface/tge1/detail
```

Create a new vlan interface:

```
POST interface/tge1.112
```

Delete VLAN interface:

```
DELETE interface/tge1.112
```

2.5 BIRD

The name BIRD stands for BIRD Internet Routing Daemon managed by CZ.NIC [1]. CZ.NIC distributes BIRD under the GNU General Public Licence. It is designed to run on all UNIX-like operating systems. It is a daemon used for dynamic routing with the support of multiple routing protocols. Routing means forwarding packets between interconnected networks to allow hosts which are not connected directly to communicate. The second task is also to communicate with surrounding routers to exchange routes or topology information. Most of the routers are hardware devices with specific hardware which is difficult to configure. However, the closed firmware and particular hardware design allow them to gain better performance than general-purpose computers. Most of the Linux computers allows to route packets, but only if these routes are configured manually - in static tables. BIRD consist of `bird` routing daemon and `birdc` CLI interface. BIRD uses protocols for defining an instance of the desired protocol as well as tables where the rules are stored. Custom tables can also be defined, where custom rules can be added. There is also `kernel` protocol present in the system, which allows to send and receive information from the operating system routing table.

There are instants `import` and `export` commands which provide data exchanges together with advanced filtration mechanism which can control route exchange between protocols.

Unlike other routing daemons BIRD supports:

- Both IPv4 and IPv6 protocols,
- Multiple routing tables,
- Command line interface allowing online inspection,
- Border Gateway Protocol (BGPv4) support,
- Soft reconfiguration,
- Powerful language for route filtering.

BIRD is running under root privileges which can be restricted using options *-u* and *-g* [1]. Currently, two versions exists: `1.x` and `2.x`. `1.x` is considered as a stable release and `2.x` as a experimental with new features. Most of the syntax is backward compatible with the BIRD version `1.x`. The most significant change between versions is the introduction of the IPv4 and IPv6 families and some changes in the commands between them. Routing tables can be considered as the hearth of the BIRD. There are two default tables, `master4` and `master6` for IPv4 and IPv6 routes in version `2.x`. There are also other tables, but they have to be configured explicitly. These tables are not kernel forwarding tables.

BIRD uses text configuration files. It reads `/etc/bird.conf` as a default location, or custom configuration file can be defined using `-c` option. Following example describes a router with router-id `2.2.2.2` and its neighbor `1.1.1.1`. There are also added some routes into the internal routing table, and `flowtab4` is defined. This table is intended to be used for keeping information about IPv4 Flowspec messages in case there are some Flowspec rules. More information about Flowspec can be found in Chapter 5.

```
router id 2.2.2.2;
flow4 table flowtab4; # IPv4 Flowspec table
protocol static flowstat4 {
    flow4;
    route flow4 {
        src 80.79.28.1/32; dst 0.0.0.0/32;
    } drop;
    route flow4 {
        src 1.1.1.1/32; dst 0.0.0.0/32; proto = 6;
    } drop;
}
    protocol static static4 {
        ipv4;
        route 10.10.0.0/24 via 192.168.56.2;
        route 20.20.20.0/24 via 192.168.56.2;
    }
    protocol bgp peer1 { description "BGP demo"; local as 63000;

    neighbor 192.168.56.1 as 63000; ipv4 {
        import all;
        export all;
    };

    flow4 {
        import all;
        export all;
    };
}
```

As mentioned, Cisco-like command line can be used to talk with the BIRD. The commands can perform simple actions like enabling/disabling protocols or showing status. Command line interface communicates with BIRD using a UNIX socket. Example of some necessary actions using a command line interface:

- **show status** - show router status that is BIRD version, uptime and time from the last reconfiguration,
- **show interfaces** - show the list of interfaces. For each interface, print its type, state, MTU, and addresses assigned,
- **show protocol** - show the list of protocols,
- **reload** - reload a given protocol instance.

Chapter 3

DDoS Attacks

DDoS can be characterized as an attempt to prevent legitimate use of a service. DDoS attacks are launched from a large number of compromised devices, globally connected and managed, which refers to a botnet. It differs from a Denial of Service (DoS) attack which is described as one device with single Internet connectivity trying to flood a target device or network [22].

Unlike many other types of attack, the primary purpose of the DDoS attack is not to gain unauthorized access to the system or any direct benefit to the attacker. The purpose of the DDoS attack is always to disable the service. Then this may serve as a mean of extortion, masking some other types of attack, or a demonstration of force. Possible attacker benefits do not flow from the attack itself, but indirectly from its consequences. The basic DDoS attack can be managed in the order of several euros, and damage caused by, for example, a malfunctioning e-shop during the pre-Christmas time could surely be a few orders higher.

3.1 History

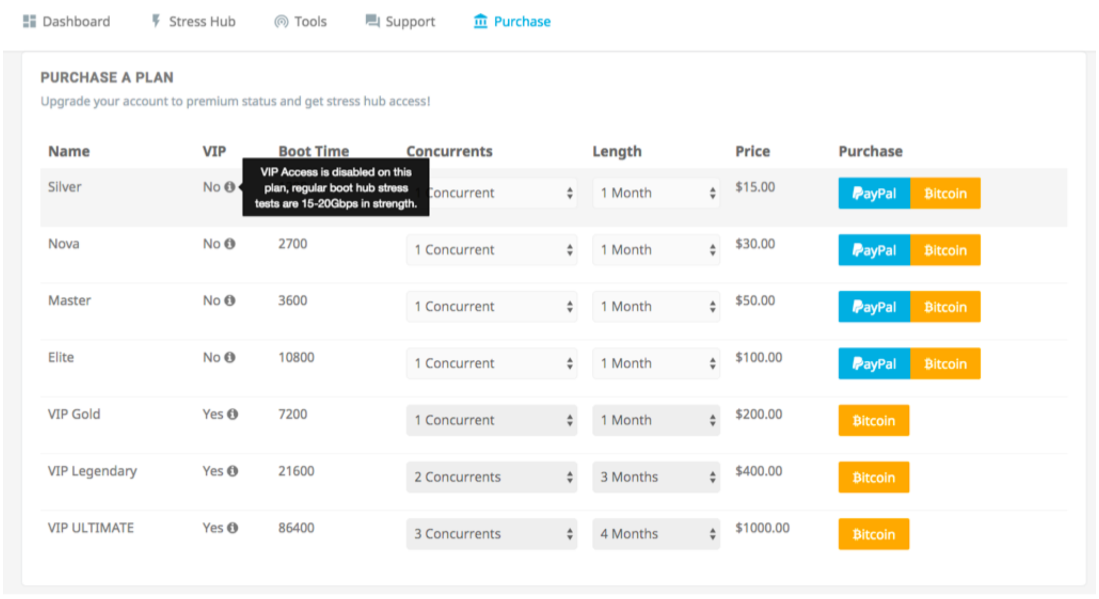
The first DDoS attack occurred in 1995 when the Italian team from the Strano Network attacked to protest against France's nuclear policy [30]. First attack did not use automation, but rather were entirely controlled by the users from their computers. Computer networks expanded, and the size of DDoS attacks has increased as well.

Nowadays, DDoS attacks are often used as a tool for unfair competition, or for political propaganda. Realizing a smaller DDoS attack is not technically challenging. Such attacks are sold as ”attack as a service“ and can be purchased anonymously and scheduled, for example, via the Tor network. Their price does not necessarily represent a huge amount of money. More and more often, DDoS attacks are used by various smart devices that are increasingly common at home. Whether it is an IP camera or a home wifi router, it can often becoming part of a DDoS attack. Those devices usually become part of a botnet, a group of computers that receive commands and attack the targets.

The most massive DDoS attack in the history was made in 2016. The target was the OVH (French Hosting Provider). The attack had a maximum capacity of 1 Tbps, and more than 150,000 Internet of Things (IoT) devices were involved in the attack [31].

3.2 Where to buy a DDoS

As it was mentioned before, buying a DDoS attack is not an expensive matter. It can be done with a few clicks using some services. It is enough to google terms like DDoS Stresser or DDoS Booter and choose an attack which best suits the needs of the attacker. There are also some reviews, which can help decide which attack provider is the best. Of course, most DDoS stresser systems declare that they should not serve for DDoS attacks in any way but only to test buyer's infrastructure. Checking whether the targets are related to the attacker does not take place, so there is no guarantee that it will be appropriately used. Many of these tools make it easy to specify the strength of attacks, lengths, and so on. An example is in Figure 3.1.



PURCHASE A PLAN

Upgrade your account to premium status and get stress hub access!

Name	VIP	Boot Time	Concurrents	Length	Price	Purchase
Silver	No		oncurrent	1 Month	$15.00	PayPal Bitcoin
Nova	No	2700	1 Concurrent	1 Month	$30.00	PayPal Bitcoin
Master	No	3600	1 Concurrent	1 Month	$50.00	PayPal Bitcoin
Elite	No	10800	1 Concurrent	1 Month	$100.00	PayPal Bitcoin
VIP Gold	Yes	7200	1 Concurrent	1 Month	$200.00	Bitcoin
VIP Legendary	Yes	21600	2 Concurrents	3 Months	$400.00	Bitcoin
VIP ULTIMATE	Yes	86400	3 Concurrents	4 Months	$1000.00	Bitcoin

VIP Access is disabled on this plan, regular boot hub stress tests are 15-20Gbps in strength.

Figure 3.1: Buying an DDoS attack [2]

There are a lot of fake pages advertising DDoS stresser tools. Their purpose is to collect the payment under the pretext of mediating the DDoS attack. Most DDoS stresser sites that Google finds is just in this category.

An exciting view of DDoS attacks can be provided by statistics regularly published by Akamai [8] - see Figure 3.2.

The ultimate goal for DDoS attacks is the game sector and far behind Telco or Finance. The reason for the popularity of gaming system attacks (such as XBox Live, Blizzard, etc.) is especially the better blackmailing of the operator. This issue is increasing with massive online games like World of Warcraft, EVE online.

These attacks are implemented by organized gaming groups or hacktivists, but it is also possible to encounter attacks by individuals who, for example, failed and want to revenge. This points to the fact that the availability of DDoS attacks are available to ordinary gamers without any qualifications or more in-depth knowledge of the issue.

If a user would like to test the resilience of his systems against the DDoS attack, a user can try to turn to the CESNET2 flab lab, which can determine the degree of resistance of

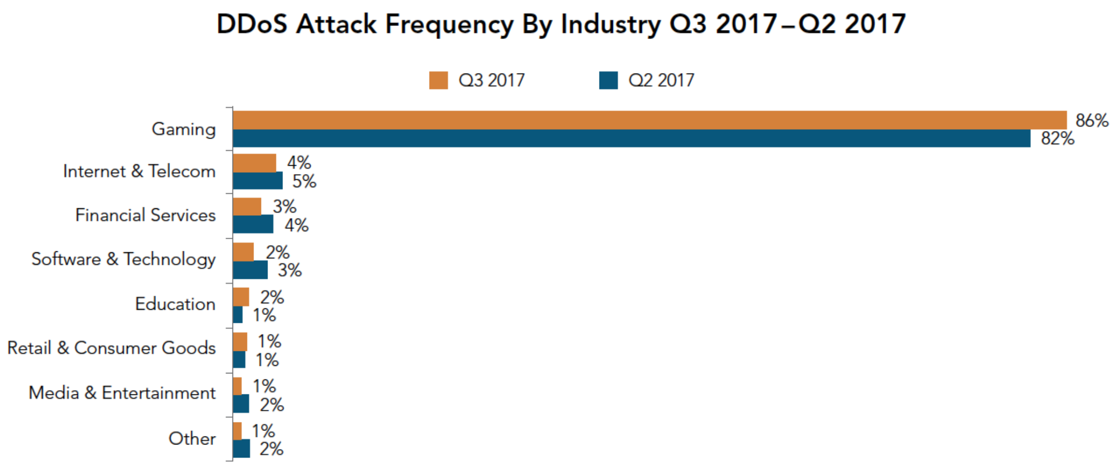


Figure 3.2: Akamain DDoS statistics [8]

such networks and services and to suggest an adjustment plan. This eliminates the risk of ”sponsoring“ real or false DDoS Stresser operators.

3.3 Types of attacks

There are several categories and types of DDoS attacks. Most DDoS attacks can be divided into three categories. Volume based or also called a volumetric attacks, these attacks are focused on saturating the bandwidth of the attacked site. Protocol attacks, which includes SYN flood or ping of death. The goal of this attack is to consume all resources of servers or other network equipment like routers or firewalls. The last type is application layer attacks. These attacks are usually slow and targeting, for example, Apache, Windows or other system vulnerabilities. There is a list of common attack types from these categories [17]:

- **UDP Flood** - This type of attack is using UDP protocol to send packets to a victim on a random port. This causes that device checking if an application is listening on a specified port answers with ICMP packet destination unreachable if no application is listening. The goal of this attack is to consume all resources.
- **SYN Flood** - Vulnerability lies in TCP protocol properties, specifically in a three-way handshake. An attacker sends only SYN packets to the target. Target responds SYN-ACK a waits for an ACK response in ”half-open“ state before the timeout expires. It consumes resources on the server.
- **Reflective attack** - An attacker sends packets with a spoofed source address to a large number of devices. These devices will accept and respond. The packet will go to a spoofed address.
- **HTTP Flood** - In this type of attack, a large number of seemingly legitimate HTTP POST or GET requests are sent to the destination server. These requests are created to consume the computing power and system resources of the target server. The server may be unavailable without overloading the server network connection.

- **DNS flood** - In this attack, many queries from different IP addresses are sent to the destination (DNS server). It is not possible to distinguish which queries come from actual clients and which do not. There may be a depletion of the line capacity or overloading of the server and its subsequent unavailability. If it is an authoritative server for a domain, this makes the entire domain unavailable.
- **DNS amplification** - In this case, the DNS server is the attacking tool, unlike the previous type where the DNS server was the target. This type of attack uses amplification factor. Mainly because the DNS response can be more verbose than the original query, so even a small capacity line can cause a massive attack.
- **Slowloris** - Slowloris is a highly-targeted attack when many network connections are opened and kept open. The attacker sends an incomplete HTTP request and immediately sends another uncompleted request just before the timeout expires.
- **Ping of Death** - An attacker sends a packet larger than the maximum size of 65535 bytes. This is only possible due to fragmentation of the packet after the defragmentation occurs, the stack overflows and the system crashes. Current systems are resistant to this attack, but large packets that cause fragmentation are still used.

3.4 Attack scenarios

In the minimum case, there are three elements of attack. The victim, the attacker and the network that connects them. In this simple case, it will not be a DDoS attack, because it will somehow miss the first D - distribution. Consider a simple volumetric attack – see Figure 3.3, where the attacker generates a large number of packets towards the victim's destination through a Internet Service Provider (ISP) network.

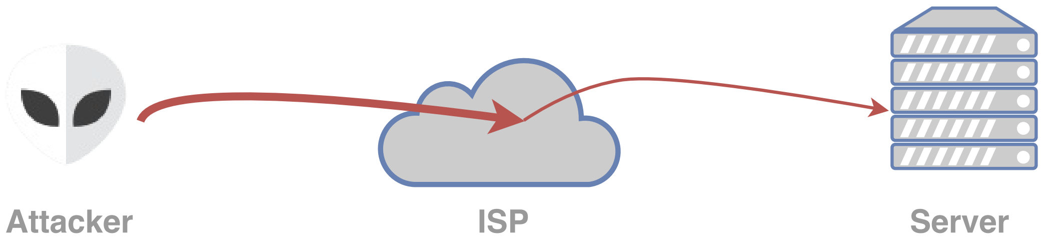


Figure 3.3: Basic DoS attack. The attacker has more capacity than the victim.

If the attacker has a line with more capacity than the victim, the attack will be successful. The line to the victim is blocked by the attackers packets, and there is no bandwidth for the legitimate traffic towards the server.

The purest form of defense is quite straightforward - more bandwidth and performance of the system. However, this simple solution has economic limits. It can usually be only used for critical services - for building high performance distributed DNS clusters [10].

If an increase in capacity is no longer sufficient or feasible, selective packet filtering can be used. However, there are some complications. The first is to identify the packets belonging to the attacker in some way. Subsequently, the network HW must support such filtration. Suppose, it is possible to determine the IP address of the attacker from NetFlow statistics, based on the most significant amount of data sended to the target network and identify the attacker.

It remains to solve where to filter. Logically, filtration would ideally be in a network that connects the attacker so that the transport network would be loaded as little as possible. However, this is a complicated issue in the real world, especially if the attack comes from a network somewhere on the other end of the world and network's abuse contact is not very communicative. In practice, filtering requires Internet service provider (ISP) which filters the traffic directly, see Figure 3.4. If it is not enough, ISP can move the filter up one level and communicate with its suppliers of connectivity, or filter on peering links.

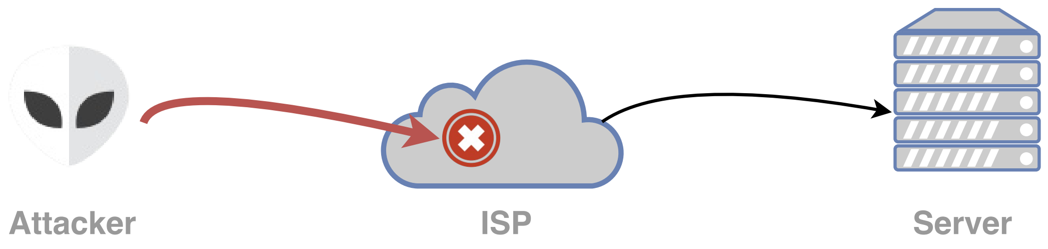


Figure 3.4: Basic DoS attack filtration. ISP doing packet filtration. Customer network only gets legitimate traffic.

As can be seen, even in the case of the most straightforward DoS attack, eliminating the attack is not an easy task. Several subjects (at least victim's connectivity provider) must be involved in the solution. This simple case demonstrates the three key features that is crucial to the DDoS attack:

1. An attacker must have the equipment to generate a significant amount of network traffic.
2. Attack suppression (mitigation) is most useful as close as possible to the source of the attack.
3. Identifying the source of the attack is necessary for effective mitigation.

3.5 Creating a DDoS attack

One of the essential elements in conducting a DDoS attack is the ability of the attacker to generate sufficient data. To do this, an attacker can use several multiplatform tools, that take care of everything else. For several massive DoS and DDoS attacks, the LOIC (Low Orbit Ion Cannon) tool [16] was used. This tool was behind some well-known DDoS attacks, where a group of people gathered and attacked the same target. The application itself does not typically provide enough power to disable the service, but it can generate the data quickly that a small number of users already have enough power to eliminate the server. Linux operating system may use the built-in `pkt_gen` packet generator within the Linux kernel that can be used as well. Sufficient performance can provide a simple program for generating UDP packets written in C. The basic pseudo-code could look like the following example.

```
/* Function for generating UDP packets */
void *gen_pkt_loop(opts_t *opts) {
...
  /* creating UDP header and data */
  udpp = create_udp_packet();
  /* loop for sending packets */
  while (1) {
    sendto(sock, udpp, pktlen, 0, dst_addr, sizeof(dst_addr));
  }
}

int main(int argc, char *argv[]) {
  ...
  for (i = 0; i < number_threads; i++) {
    // Creating threads to generate UDP traffic
    pthread_create(thread_id, NULL, gen_pkt_loop, opts );
  }
}
```

Listing 3.1: C pseudocode for generating packets.

According to Example 3.1, a simple DDoS generator, ordinary programmer can handle without problems. The intensity of the attack that can be created in this way is in the order of gigabit and higher. To create a DoS attack on a 10 Gb/s connected server or service, it is enough to get ten people with a gigabit connection, and attack can be created.

3.6 Masking techniques

An appropriate analytical system, or custom queries over NetFlow records can detect DDoS addresses sending the most data to the service. Filtering can be done for those addresses. Most network devices has a support for such filter. Thus the attacker often implements some of the camouflage techniques.

3.6.1 Traffic dilution

The attacker can increase the number of hosts to disperse the attack. If, the 10 Gb/s attack is divided into 10,000 sources, and a 1 Mb/s data stream is allocated to every source IP address, it may be close to the legitimate amount of traffic from one IP address, and ISP will not be able to distinguish between the legitimate traffic and the attacker's traffic by a simple statistical method. Thus, the identification of the source of the attack will be significantly reduced. Moreover, as we said earlier, if we can not identify the sources of the attack, we can not even filter it. However, getting 10,000 people to make the attack can be a difficult. However, there are several ways how the attacker can deal with this problem.

3.6.2 Source IP address spoofing

In the regular IP communication, the header of each packet is formed by the source and destination IP addresses. The destination address serves as an identifier based on which routers deliver the packet to the destination host. However, the source address does not perform any function from packet delivery point. It serves only for the target host to know where to send the reply packet and for sending ICMP errors as well. The attacker can send a packet using changed source IP address and it appears that the packet was sent from another source. It is often used for DDoS attacks when the attacker send a lot of packets to the victim, but does not need the response packet. It is hard to filter, because it comes from many different source IP addresses to hide the true identity of the attacker.

3.6.3 Botnet

Another way to divide the attack among multiple source addresses is to use botnet networks. The issue of botnets presents a wide-ranging problem, so this thesis will only deal with them marginally.

3.6.4 Reflective attacks

The use of a legitimate service as an intermediary is another technique. This approach also masks the real identity of the attacker. Generally, we call this technique a reflective DDoS attack. How this attack looks like is captured in the following Figure 3.5.

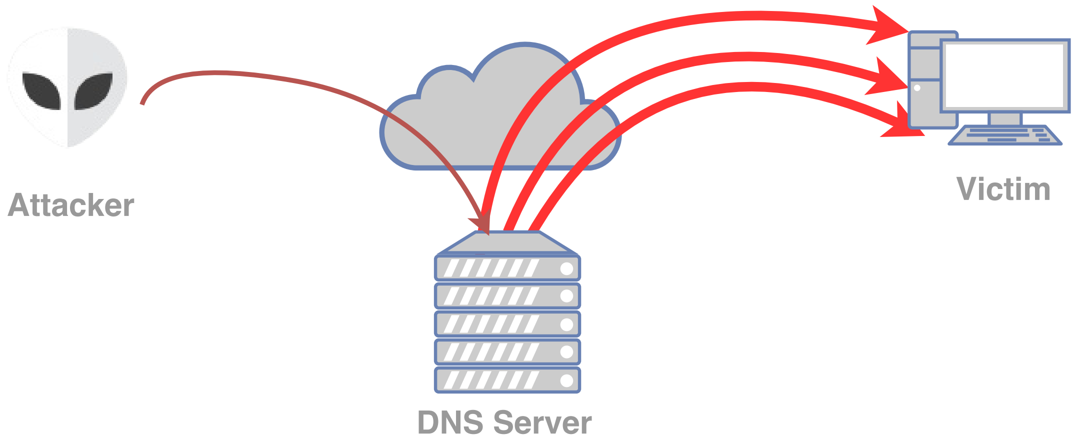


Figure 3.5: Reflective DDoS attack using DNS.

Two assumptions are necessary to make the reflective attack successful. The first is a running that answers to each packet. The second is the ability of an attacker to create a packet with a custom source address as was described in subsection 3.6.2. In Figure 3.5, an attacker creates a packet and specifies the address of the DNS server as the destination address and victim's IP address as the source address. The server receives the packet and sends a response. However, the packet is not sent back to the attacker's address, but to the victim's address.

Why it would make sense to send the packet through the reflector when an attacker could send a packet directly to the victim? The main reason is the amplification ability. Some services response packets larger than the original request packet. A typical example of such a service is DNS. A DNS queries for an MX record such as Cesnet DNS server might be a good example:

```
$ dig  @195.113.144.205 MX  www.cesnet.cz

IP (tos 0x0, ttl  64, id 23942, offset 0, flags [none]
 proto: UDP (17), length: 59)
 147.229.3.152.38768 > 195.113.144.205.53:  5373+ MX? www.cesnet.cz. (31)

IP (tos 0x0, ttl  59, id 56237, offset 0, flags [none]
 proto: UDP (17), length: 344)
 195.113.144.205.53 > 147.229.3.152.38768:  5373*- 3/3/7 www.cesnet.cz.
 MX cartero.cesnet.cz. 10, www.cesnet.cz. MX postino.cesnet.cz. 10,
  www.cesnet.cz. MX mail.cesnet.cz. 100 (316)
```

On the 59 B server query, the server generated a response of 344 B. The query-to-answer ratio is 1:5.8. Other protocols like NTP and SNMP are also effective for reflective DDoS attacks.

3.6.5 Fragmentation

Fragmentation mechanism can be misused by the attacker as well. It complicates the identification of packets belonging to the attacker. The problem is that in the case of fragmented packets, the information about the higher protocol layer is carried only in the first fragment. If a filter, for example, all UDP packets with source port 123 is created, only the first packet will be filtered, and the remaining packets will be able to pass.

Chapter 4

DDoS Attack Detection

Defending against a DDoS attacks can be divided into three phases: prevention, detection, and response. A suitable detection method should have low false positive rate and short detection time. Detecting DDoS can be done using some kind of dedicated tools, which often use a mix of different detection approaches to ensure the best result and minimum false positives [20].

In this thesis, a few DDoS detection tools are presented. Most of the products are paid, but licenses were acquired under the Brno University of Technology for research purposes. These tools are installed and used in Brno University of Technology campus network.

4.1 nScrub

nScrub is a software-based DDoS mitigation engine based on `PF_RING ZC` (zero copy), which can operate at 10 Gbps speeds using a commodity hardware. Nscrub can operate in two modes, transparent or router mode. Router mode is designed to be used with BGP routing protocol. nScrub is designed as an extensible platform, which provides the ability to extend, improve or add new detection algorithms for traffic mitigation. It can be configured using a REST API as well as command line interface [3].

It is free to use for non-profit and educational organizations.

4.2 Cloudflare Advanced DDoS Protection

Cloudflare provides a DDoS Protection service, a sophisticated solution which can be used to mitigate DDoS attacks of all sizes and forms. For example, Cloudflare platform provides protection against attacks that target UDP, ICMP or DNS amplification and Layer 7 attacks. The system is focused on a website protection and has a few different pricing plans. For example, basic protection is for free, but to gain an advanced features users have to pay. Cloudflare is using their high capacity edge network to clean the traffic from DDoS attack [15].

4.3 DDoS Defender

DDoS Defender is a solution for detection and mitigation of volumetric attacks from Flowmon. DDoS Defender does not require any topology changes, it detects such attacks automatically. It is deployed with Flowmon Collector and uses Flowmon Monitoring Center flow for attack detection. Thanks to these additional tools, DDoS Defender does not require additional flow source. The detection method is focused on volumetric attacks and detection is based on collected flow data. The flow can be from any source, for example, router or probe on the network. DDoS Defender can be deployed to the network in these three scenarios [18]:

- **Standalone -** Performing only passive attack detection and alerting. DDoS Defender is not doing any mitigation in this scenario.
- **Out-of-band elimination of DDoS attack -** Performing passive attack detection. Sends alerts and mitigation actions using BGP Flowspec rules or black hole routing to stop or rate limit the attack.
- **Scrubbing Center -** DDoS Defender, can be integrated together with a scrubbing center to mitigate traffic attack and return clean traffic. The clean traffic is returned using static ARPs with unique VRF tables or GRE tunnels.

Detection is done for every protected segment configured in DDoS Defender. Protected segments are configured using network subnets. For every segment, two methods are evaluated regularly: Baseline methods and static methods. For each baseline method, one or more baselines are defined in a configured learning period. Each baseline is then compared to the actual traffic. The attack is reported in case of crossing a threshold. Attack detection also requires configuring the minimal bits and packets per second.

Definition of two main methods to detect DDoS attack is as follows:

- **Baseline methods**
  - **Manual threshold -** Single baseline is generated for the peak value of incoming traffic. A user can specify the threshold in percents for triggering an attack. An attack is triggered in case of incoming traffic is more than baseline multiplied by the defined threshold value.
  - **Adaptive threshold -** Baseline is generated for each traffic type, for example, TCP, UDP, ICMP or TCP syn packets. For each baseline, two thresholds are generated. First is for attack and second for suspect traffic. The system then compares the amount of incoming traffic with these generated thresholds.
- **Static methods**
  - **Incoming/Outgoing traffic ratio -** Amount of outgoing, incoming packets are compared to a user-defined threshold in percents. The attack is only triggered if Incoming packets are more than outgoing multiplied by a threshold.

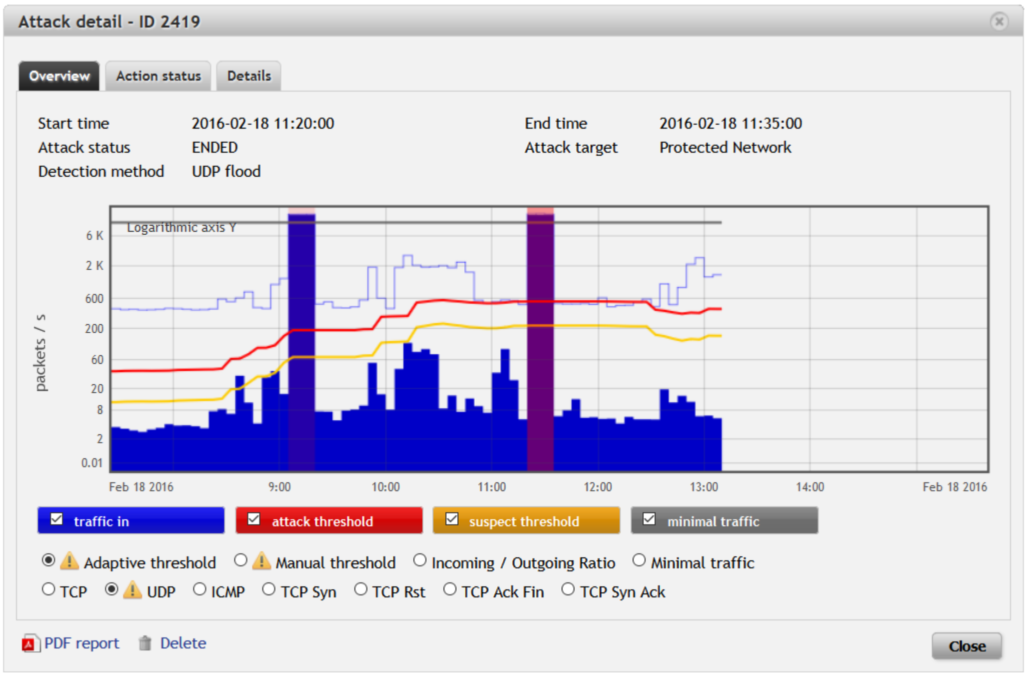
Minimal traffic can also be defined in packets or bits per second. There are two modes of limitation:

- **Trigger evaluation of detection methods -** The detection method described before is evaluated only if traffic is exceeding both minimal packets and bits per second limits.
- **Trigger attack -** Attack is triggered in case of minimal configured traffic is exceeded by the packets and bits per second limits.

If any DDoS attack is detected, the following actions can be performed. Information is based on data located on a flowmon website [18]:

- **Alert (e-mail, syslog, SNMP trap),**
- **Traffic diversion (PBR, BGP, RTBH),**
- **User-defined script initiation,**
- **Attack mitigation using BGP Flowspec.**

DDoS Defender also contains attacks visualization. This can help administrator to know the source and destination of the attack as well as to see other information about the attack like the strength or L4 protocol characteristics. A simple attack detection example with thresholds can be seen in the Figure 4.1.



Attack detail - ID 2419

Start time	2016-02-18 11:20:00	End time	2016-02-18 11:35:00
Attack status	ENDED	Attack target	Protected Network
Detection method	UDP flood		

[✓ Adaptive threshold] Manual threshold Incoming / Outgoing Ratio Minimal traffic

TCP [✓ UDP] ICMP TCP Syn TCP Rst TCP Ack Fin TCP Syn Ack

Figure 4.1: DDoS Defender attack visualization [18].

4.4 Fastnetmon

Another representative in the fight against DDoS attacks based on GNU/Linux is Fastnetmon available from [25]. It is an open source project with the support of many traffic information formats like sFlow v5, NetFlow v5, v9, jFlow, IPFIX, and others. Flowspec is also supported.

According to [26], Fastnetmon was tested on 10 Gbps link on 12.6 MPPS with the load of 5795 Mbps using Intel i7-3820 processor. This configuration used 100 % CPU.

Fastnetmon has two versions, Fastnetmon community edition is an open source version which is available to download from GitHub [25]. It should be considered more as a framework which can be used for building more complicated solutions. An advanced edition was built as a completely independent solution for businesses. In this thesis, Advanced edition will be used. The main differences are shown in a Table 4.1:

Fastnetmon features	**COMMUNITY**	**ADVANCED**
Amplification attack detection	YES	YES
Ability to block only malicious traffic	YES	YES
Ability to whitelist remote hosts	NO	YES
Threshold for each networks	NO	YES
Native BGP Flow Spec/RFC 5575	NO	YES
Call script when attack arrives	YES	YES
Grafana dashboards support	NO	YES
CLI configuration tool	NO	YES

Table 4.1: Fastnetmon versions comparison.

Attack visualization is provided by Grafana, an open visualization platform. Basic attack visualization is shown in Figure 4.2.

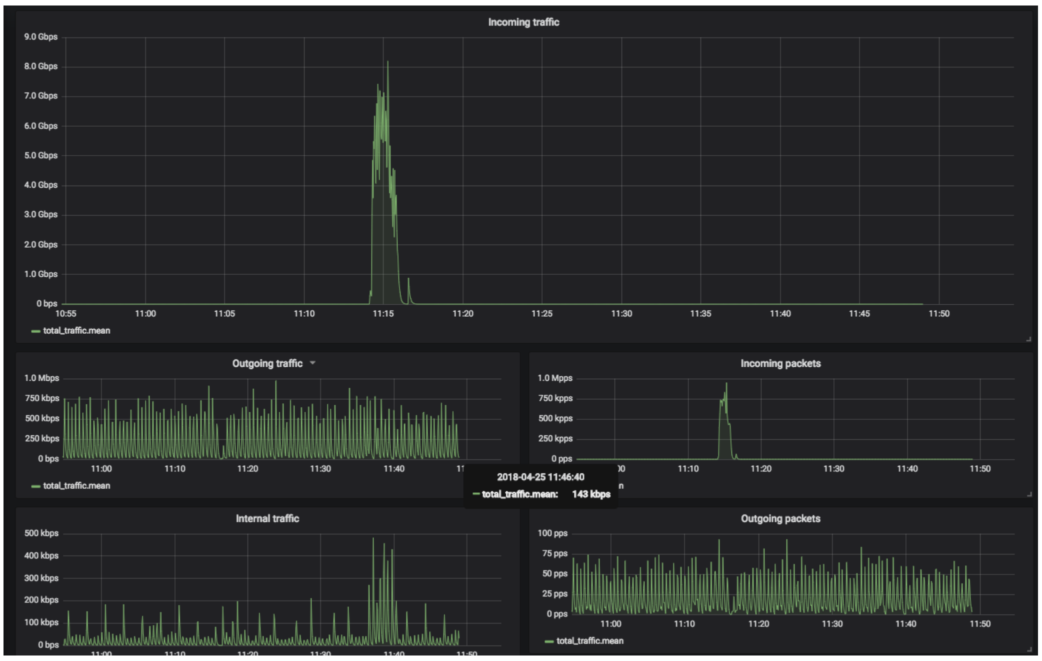


Figure 4.2: Grafana traffic visualization.

Chapter 5

DDoS Attack Mitigation

After the DDoS attack is detected, the next step is attack mitigation. Purpose of DDoS mitigation is to stop or reduce the effect of the DDoS attack on the network. One of the primary objective of the mitigation is to stop only the DDoS attacks and not regular traffic. Mitigation can be done in a Internet Service Provider (ISP) network or in exchange points.

One of the mitigation approach is to directly block addresses, or entire prefixes that are under attack using ACL or BGP Remote Triggered Black Hole Filtering (RTBH) on the border Internet routers in the direction the attack comes from. Upon agreement with ISP, RTBH information can also be propagated to the neighboring autonomous system (AS) via the BGP to block the prefixes there.

As another solution, policy-based routing (PBR) can be used. PBR allows to define rules based on source/destination IPs/ports, etc. and traffic can be discarded, rate-limited or redirected. However, these actions have to be done manually on border router in case of an attack [28].

Another option for responding to a DDoS attack is to use the BGP protocol extension, BGP Flowspec [21], which allows to signal L4 data flows with a given action (discard, rate-limit, etc.) that needs to be done. This information can be further propadated to the boundary BGP routers, or upon agreement with the ISP on its routers and selectively block, or rate to limit the detected aggressive streams. The rest of this chapter describes these techniques in more detail.

5.1 Access Control List (ACL)

ACLs can be used as a simple tool for mitigating DDoS attacks. ACL can be defined as a packet filtering mechanism. Routers look into L3 and L4 OSI model layers. ACL is a set of rules that router checks when the packet arrives at the router. Generally, the most checked field in a packet are:

- Source/Destination IP address,
- Source/Destination UDP/TCP port,
- Protocol.

5.1.1 Types of ACLs

In this section, basic types of ACLs will be explained based on Cisco terminology. There are two basic types of ACLs. ACLs can be divided in two groups:

- Standard ACL – Older and simpler ACL with fewer configuration options,
- Extended ACL – Has more configuration options

Standard ACL use numbering from 1 to 99 and from 1300 to 1999 [11]. These numbers are unique and can be used as reference elsewhere in the configuration. Standard ACLs can be used for filtration only according to the source address. Standard ACLs are usually set as close to the destination as possible. Example of standard ACL that permit only traffic from host using `192.168.1.20` IP address and blocks everything else what is following. Block is because of implicit deny.

```
access-list 20 permit host 192.168.1.20
```

Extended ACL can use numbering from 100 to 199 and from 2000 to 2699. These types of ACL can filter.

Based on fields in the L3 layer of ISO / OSI (IP address, protocol, data from ToS) and in the L4 layer (TCP header ports and protocols, UDP header ports, ICMP header types and ICMP messages). Following example shows an extended ACL which permits only traffic from host using `192.168.1.20` IP address and source port 22. Destination address with port also can be defined. It this case, it is defined to match any IP address and port.

```
access-list 120 permit udp host 192.168.1.20 eq 22 any
```

Depending on the used protocol, the parameters change, for example, the port parameter is used only for TCP and UDP. IP can be used to include all L4 protocols. A range of ports can also be set. For ports, there are operators `eq` (is equal), `neq` (not equal), `gt` (greater than), `lt` (less than), and range. Operators, including the port number, are entered as the source or destination address, a port is then used at source or destination.

5.2 Border Gateway Protocol

The Border Gateway Protocol (BGP) protocol is used to exchange routing information between individual Autonomous Systems (AS) in the Internet. In this context, an autonomous system means a set of routing prefixes managed by a single administrative entity. Every internet service provider (ISP) who wants to route on the Internet must have a unique autonomous system number registered. These AS numbers as well as IP prefixes are managed by Internet Assigned Numbers Authority (IANA), which authorized regional Internet registries (RIRs) to allocate these numbers to ISPs. BGP routing between autonomous systems is also called external BGP or eBGP. BGP selects the best route based on the path metric, network policy or ruleset configured by a network administrator. BGP can also be used within one AS, its called Internal BGP or iBGP [27].

BGP also has a community attribute which is used in order to control the routing policy in ISP network or the upstream service provider network. Communities can be used as flags to the specific route groups. Based on these flags, the service provider than decides what to do with the traffic flagged with a specific community.

5.2.1 Remote Triggered Black Hole

BGP protocol can be used as a solution for DDoS mitigation in form of RTBH (Remote Triggered Black Hole) as mentioned earlier. The concept is based on statically defined route to a `null0` at edge router. Traffic sent to this destination is dropped [28].

When an attack occurs, the administrator sends a BGP route to the border router with the affected host or network destination tagged with a BGP community. This means traffic is dropped. It prevents the traffic from hitting the server. It protects the server effectively, but it removes the ability to reach the server completely from the outside world.

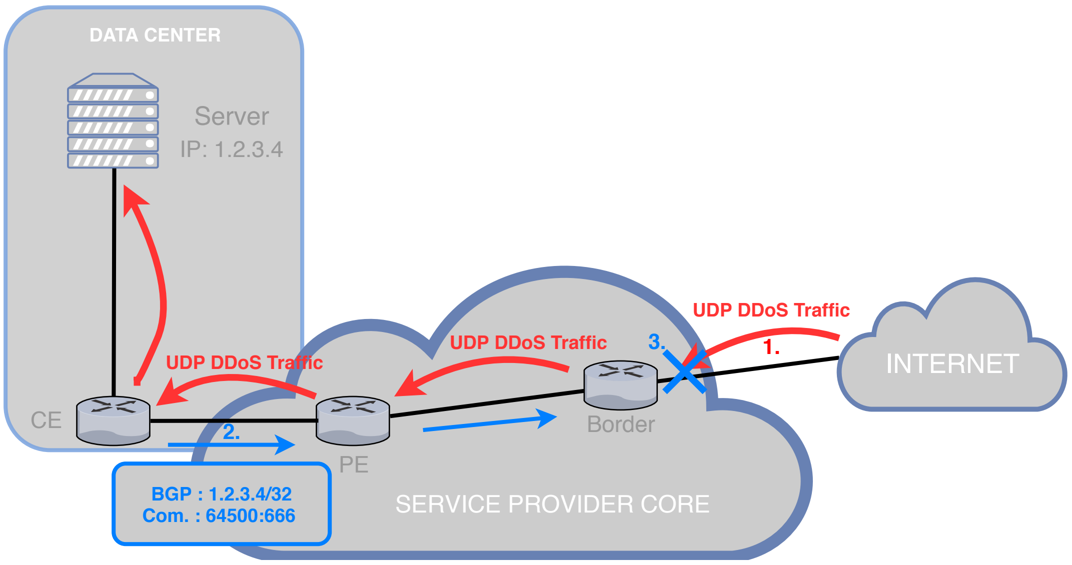


Figure 5.1: Remote Triggered Black Hole architecture.

It is possible to send a route with source address tagged with a community. It assumes administrator knows where the attack comes from. Both options work, but they are also stopping the regular traffic.

5.2.2 BGP Flowspec

BGP provides another solution for DDoS attacks mitigation. It is defined as a BGP extension [21]. BGP Flowspec defines a new Network Layer Reachability Information (NLRI) type message in BGP. This can be used for passing information about IP flows and actions, which should be done with the traffic. It provides more granular approach, and it can match a particular flow based on source, destination, L4 parameters and packet specifics such as packet length [12]. Dynamic installation of an action can be defined at the border router to either:

- inject traffic to a different Virtual Routing and Forwarding (VRF) for analysis,
- allow traffic, but police it (rate can be defined),
- drop the traffic.

Instead of sending a route with a special community as RTBH works, Flowspec defines a new attribute which is sent to border router instructing them to create something like an ACL to implement the advertised rule.

As mentioned before, BGP Flowspec is a scalable DDoS mitigation solution. Compared with policy-based routing (PBR), BGP Flowspec is a dynamic solution which allows propagating policy-based routing rules dynamically. Existing control plane communication can be used by using MP-BGP infrastructure.

Consider Figure 5.2, Flowspec message is sent to border router with the defined action. The rule is then installed and can also be dynamically removed after the attack ends [28].

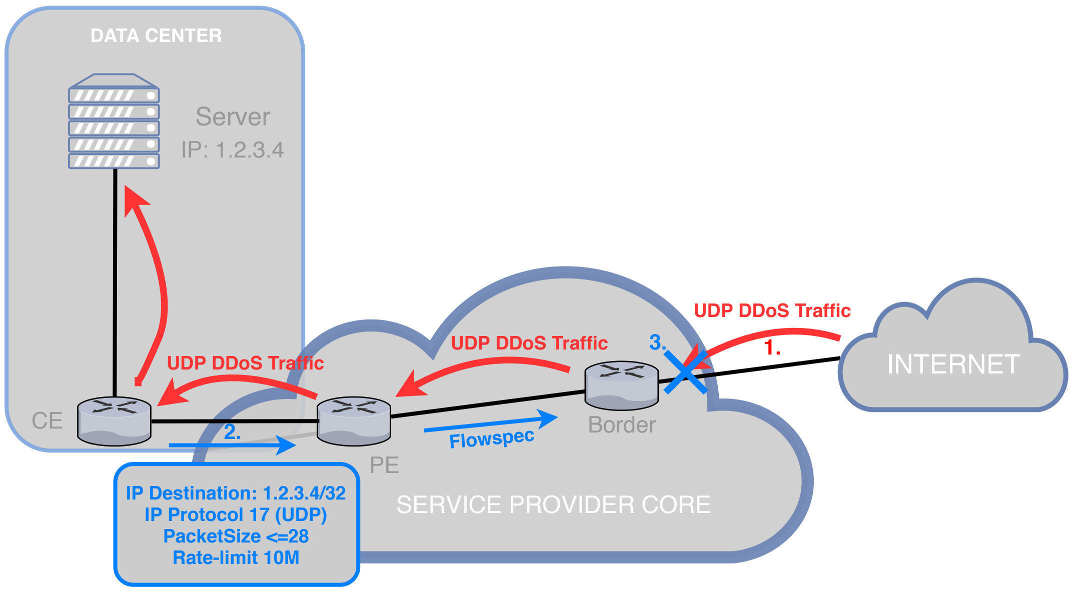


Figure 5.2: BGP Flowspec architecture.

In most of the cases service provider does not trust its customers and BGP AFI/SAFI have to be deployed between provider and customer. It results that BGP Flowspec is not being deployed between customer and provider. Instead of information exchange BGP Flowspec, a central BGP Speaker trusted and managed by service provider is only allowed to distribute Flowspec rules. Figure 5.3 shows the real-world architecture [28].

Following parameter using NLRI types can be defined in Flowspec [13]:

- **Type 1 -** IPv4 or IPv6 Destination address,
- **Type 2 -** IPv4 or IPv6 Source address,
- **Type 3 -** IPv4 last next header or IPv6Protocol,
- **Type 4 -** IPv4 or IPv6 source or destination port,
- **Type 5 -** IPv4 or IPv6 destination port,
- **Type 6 -** IPv4 or IPv6 Source port,
- **Type 7 -** IPv4 or IPv6 ICMP type,

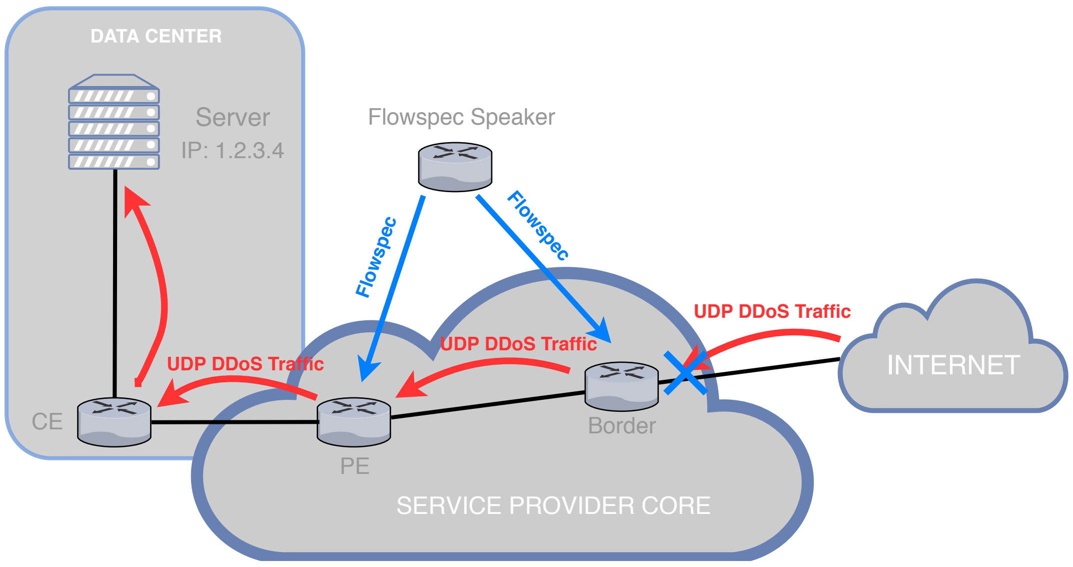


Figure 5.3: BGP Flowspec real-world architecture.

- **Type 8 -** IPv4 or IPv6 ICMP code,
- **Type 9 -** IPv4 or IPv6 TCP flags,
- **Type 10 -** IPv4 or IPv6 Packet length,
- **Type 11 -** IPv4 or IPv6 DSCP,
- **Type 12 -** IPv4 or IPv6 Fragmentation bits.

Actions which can occur have been defined in the extended community. The extended community was added to BGP as an extension, meaning of each entry is defined in RFC 5575 [21]. Various actions that can be defined can be seen in Table 5.1.

type	extended community	encoding
0x8006	traffic-rate	2-byte as, 4-byte float
0x8007	traffic-action	bitmask
0x8008	redirect	6-byte Route Target
0x8009	traffic-marking	DSCP value

Table 5.1: BGP Flowspec extended community actions.

Extended communities are described in detail in the following section:

Traffic-rate

Traffic-rate is a non-transitive extended community and uses the following encoding. First two octets carry the octet id, which can be assigned from the two-byte AS number. In case of four-byte AS number, the least significant two bytes are used. This value is only informational and should not be interpreted. The remaining four bytes are carrying the traffic-rate number in bytes per second using IEEE floating point [IEEE.754.1985] format. In case of 0, the whole traffic of the matched flow should be discarded.

Traffic-action

Traffic-action defines six bits, but only two bits are currently defined. The two least significant bits stand for:

- **Terminal Action -** If this bit is set, the traffic filtering engine applies any subsequent filtering rules. If not set, the evaluation of the traffic filter stops when this rule is applied.
- **Sample -** Enables traffic sampling and logging for matching flows.

Redirect

Redirect extended community allows to traffic to be redirected to defined VRF routing instance [19].

Traffic Marking

The traffic marking extended community allows modifying the DSCP bits on a matching IP packet from the flow. It is defined as a sequence of five zero bytes followed by DSCP value encoded in the six least significant bits of the sixth byte.

5.3 Hardware filtration

To gain maximal performance and not overload the router, hardware filtration can be used. It can be described as ACLs for network card not consuming any system resources. There is a tool called `ethtool` to control network driver as well as other hardware settings. Most of the NETX routers are using Intel 10 Gbps Ethernet network cards like Intel 540 or Intel XL710.

Those cards have an Intel Ethernet Flow Director (Intel Ethernet FD), which is mainly designed to direct the packets to the operating system core, where the consuming process or container is running according to [14]. Network cards with Intel Ethernet FD support can direct received packets to different queues, and enable tight control on each received flow. It provides the most benefit on Linux when the traffic is heavy, and packets are often small. Using Intel Ethernet FD, packet processing can be offloaded to the network interface card, hence it can be used for mitigation of DDoS attacks. To determine if the network card supports Intel Ethernet FD, the `ethtool` using the following command can be used:

```
$ ethtool --show-features <interface name> | grep ntuple
```

Supported devices are capable of hardware filtration based on these parameters:

- Source IP adress,
- Destination IP adress,
- Source UDP/TCP port,

- Destination UDP/TCP port.

Experience shows that each card has to be tested. Sometimes, there can occur some hardware issues like not allowing to add a rule or not applying the added rule. There are few actions which can be applied, but most of the card allows only to drop the traffic. To add a rule using `ethtool`, parameter *–config-ntuple* have to be used, to show currently applied rules *–show-ntuple* can be used.

Example of adding filter rule:

```
$ ethtool --config-ntuple [interface] flow-type [protocol]
  dst-ip [IP] dst-port [port] action -1
```

Example of removing filter rule:

```
$ ethtool --config-ntuple [interface] delete [rule]
```

Example of showing all active rules on interface:

```
$ ethtool --show-ntuple [interface]
```

Output of all active filtration rules on the interface. This rule drops all the traffic with `192.168.1.1` as a destination IP address and TCP as a used protocol. Other preferences like ports or VLAN tags are not defined.

```
Filter: 2044
        Rule Type: TCP over IPv4
        Src IP addr: 0.0.0.0 mask: 255.255.255.255
        Dest IP addr: 192.168.1.1 mask: 0.0.0.0
        TOS: 0x0 mask: 0xff
        Src port: 0 mask: 0xffff
        Dest port: 9000 mask: 0x0
        VLAN EtherType: 0x0 mask: 0xffff
        VLAN: 0x0 mask: 0xffff
        User-defined: 0x0 mask: 0xffffffffffffffff
        Action: Drop
```

Chapter 6

Design and implementation

The goal of this thesis is to design and implement a mechanism to automatically reconfigure the network in case of a DDoS attack. For this reason the target implementation platform has to be chosen. Due to a several reasons described in section 2.4, NETX platform was chosen. After choosing the routing platform the filtration method have to be chosen. NETX uses NICs with hardware filtration support on Intel and Solarflare network cards. These rules have to be distributed from a DDoS detection tool to the routing platform. NETX has a great API prepared to configure the system without any restrictions and Netc modular design allows to add a new module to the system easily.

6.1 Architecture

The primary goal of this thesis is to provide DDoS attack mitigation automatically. The architecture must be prepared for these automatic network reconfigurations. These reconfigurations have to minimalize an error possibility which can lead to the whole network malfunction.The primary mechanism used for dropping the DDoS attack is hardware filtration on Network Interface Controller (NIC) which was described above in previous chapter.

The most crucial part for DDoS mitigation is to not drop traffic which is useful, for example, regular traffic or traffic used for complete network reconfiguration. After an attack occurs, it should be detected as fast as possible, and the detected information about attack details should call scripts which will parse this information and send them via some channel to the border router. These attack details are the source and destination IP addresses, source and destination ports and protocol. In most cases, the filtration is based on a source IP address to drop only traffic flowing from the attacker. The problem is that in common DDoS attacks there are many attackers attacking the target in the network infrastructure. Two approaches can resolve this. First is to drop the destination IP address. This stops the attack from getting to the network, but the downside is it also stops the regular traffic, and the whole service is unreachable. The second approach is to stop top N attackers. In real network scenario, it means to stop a few source network prefixes from which is the attack ordinated. This provides mostly dropping the traffic from the attacker and not stopping the service or the server. A small portion of regular traffic can also be dropped, however it is better then dropped the whole traffic.

Detecting the whole range of attack types and not producing a large number of false positives is not an easy task. We use an existing solution int this thesis, such as DDoS Defender and Fastnetmon which were described in Chapter 4. These tools are configured to detect the attack and send a message via BGP or use an implemented script to parse and send the corresponding information.

The outputs from detection tools are sent automatically to the NETX router and the hardware filtration rule is created. These rules are then applied to the appropriate interface. In most cases, these rules are not stopping the 100 % of the attack, but they can stop the majority of the attack which can lead to reducing the impact of the attack. In case of an error, syslog message is sent to warn the user or sysadmin.

After the attack stops the detection tool starts a script which is responsible for stopping the mitigation. In case of using BGP, a withdraw BGP Flowspec update message is created and sent to the border router. The rule is deleted from the interface, and the network is configured automatically to its previous or default state.

This whole architecture is shown on the following Figure 6.1. As mentioned, NETX is used as a border router in the network. Next, there is a network traffic analyzer or DDoS detection tool. As mentioned, DDoS Defender or Fastnemon is used. This architecture and implementation are not allowing to use both tools at once. It could create a filtration rules mismatch which could end with network malfunction or not proper network reconfiguration. Network traffic is intercepted by probes in the network which sends data in NetFlow or IPFix format. There is also a possibility to mirror the ports on border routers and send the whole traffic to the traffic analyzer.

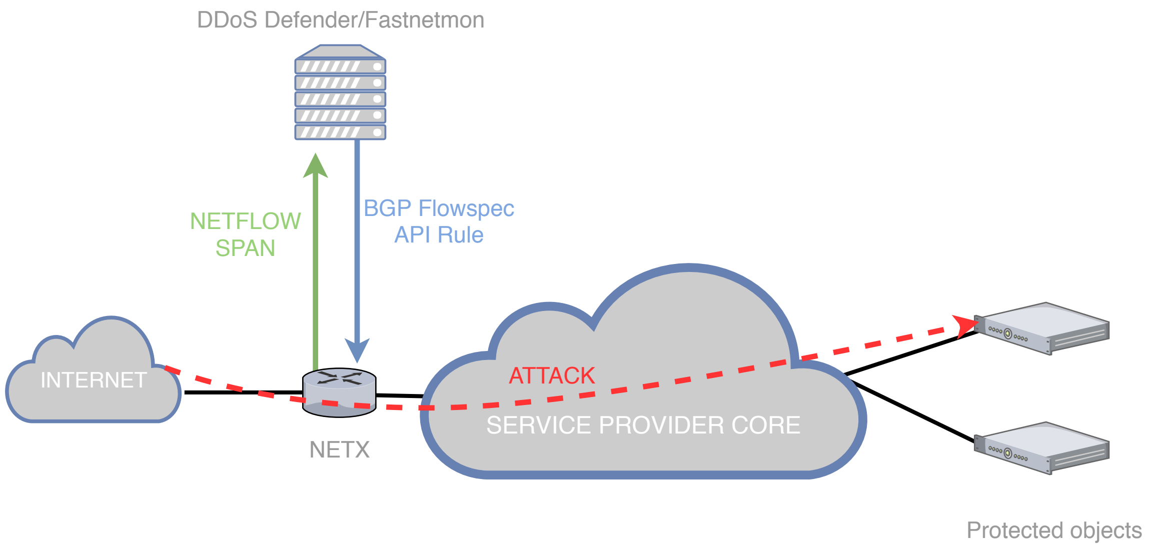


Figure 6.1: DDoS mitigation architecture.

6.2 NETX Netc module

One of the first steps to create an automation mitigation was writing a module into NETX system. The goal of this module is to make hardware filtration configurable via Netc. This module is written in Perl and is a standard part of a NETX installation. The only requirement is to have NIC which is compatible with hardware packet filtration, in our case an Intel card along with the newest drivers. Hardware filtration in NETX system can be configured either using command line interface (CLI) or RESTfull API.

The first step was to make the new module registered and visible to the whole system. This can be done by extending `Config.pm` file and define the structure where the hardware filtration configuration should be placed. First step was to make a `hw-filter` section and try to enable the hardware filtration on the NIC when the menu is accessed. In case of turning off the `hw-filter` module, the filtration is disabled, that means all the filtration rules which were enabled are removed. Following pseudo-code shows how the `hw-filter` is implemented as a Netc module.

```
'hw-filter' => {
    DESCR  => 'set hw filter to interface',
    SET  => 'xcfg-netc-hwfilter enable-filter %1',
    UNSET => 'xcfg-netc-hwfilter disable-filter %1',
    GET  => '(ethtool -k %1 | grep \'ntuple-filters: on\') 2>/dev/null',
        'action' => { CHILD => { '%STR' => {
                GET  => 'xcfg-netc-hwfilter get-filter %1',
                SET  => 'xcfg-netc-hwfilter set-filter %1 %opts',
                UNSET => 'xcfg-netc-hwfilter del-filter %1 %opts',
                OPTIONS => {
                        SET => {
                                'src' => {VALUE => '%IP4', ADD => '--src-ip %s'},
                                'smask' => {VALUE => '%IP4', ADD => '--src-mask %s'},
                                'sport' => {VALUE => '%NUM', ADD => '--src-port %s'},
                                'dst' => {VALUE => '%IP4', ADD => '--dst-ip %s'},
                                'dmask' => {VALUE => '%IP4', ADD => '--dst-mask %s'},
                                'dport' => {VALUE => '%NUM', ADD => '--dst-port %s'},
                                'proto' => {VALUE => '%STR', ADD => '--protocol %s'},
                                'id' => {VALUE => '%NUM', ADD => '--filter %s'},
                        },
                },
        }},
    }},
},
```

Listing 6.1: Perl pseudo-code for `hw-filter` Netc module.

In case of setting GET, SET or UNSET rule in Netc, the script `xcfg-netc-hwfilter` is started with given parameters. In first stage, script checks once more if the filtration is enabled and enables it in case if not. Then the `ethtool` program is used to GET, SET or UNSET the filtration rule. The main purpose of the `xcfg-netc-hwfilter` is to parse given parameter, check the hardware filtration status and GET, SET or UNSET the filtration rule using `ethtool`. Script has the following configurable parameters:

- **src-ip** <source_ip_address>,
- **src-mask** <source_mask>,
- **src-port** <source_port>,
- **dst-ip** <destination_ip_address>,
- **dst-mask** <destination_mask>,
- **dst-port** <destination_port>,
- **protocol** <protocol> (default IPv4),
- **filter** <filter_number>.

6.3 Fastnetmon

Fastnetmon 4.4 will be used for DDoS detection and to provide an output for NETX routing platform for DDoS mitigation. In the following example, configuration detection script which will be called in case of attack is shown. Text format is used instead of JSON format because text format contains flow dumps in notification message. Defined script will receive all the details which are gathered by Fastnetmon. Attack details are sent to the script via standard input.

```
notify_script_enabled: enabled
notify_script_format: text
notify_script_pass_details: enabled
notify_script_path: /usr/local/bin/notify.pl
```

Connection tracking option should be enabled. It provides an additional information about flows involved in DDoS attack. Connection tracking can help with filtration based on a source IP addresses. Source addresses can be parsed from the connections and used for DDoS attack mitigation. This information is attached bellow the attack details and sent to the script.

```
enable_connection_tracking: enabled
```

After notification about an attack is received, script is called which parses the notification data and sends the rule to the NETX API. Example of pseudo-code of a script used for connecting to NETX API 6.2.

```
my $request_url = '';
# Construct URL (encode arguments)
if ($ARGV[3] eq "ban") {
    $request_url = $APIURL_BAN;
}

if ($ARGV[3] eq "unban") {
    $request_url = $APIURL_UNBAN;
}

# Add actions
my @actions = ("SET");
$request_body->{'Actions'} = \@actions;

# Add options
my $opts = parse_extra_options("src=$ARGV[0]");
$request_body->{'Options'} = $opts;

# Send request
$client->request($METHOD, $request_url, encode_json($request_body));

if ($client->responseCode() ne '200' && $client->responseCode() ne '400') {
    print $client->responseContent();
}
```

Listing 6.2: Perl pseudo-code for accesing the Netc API.

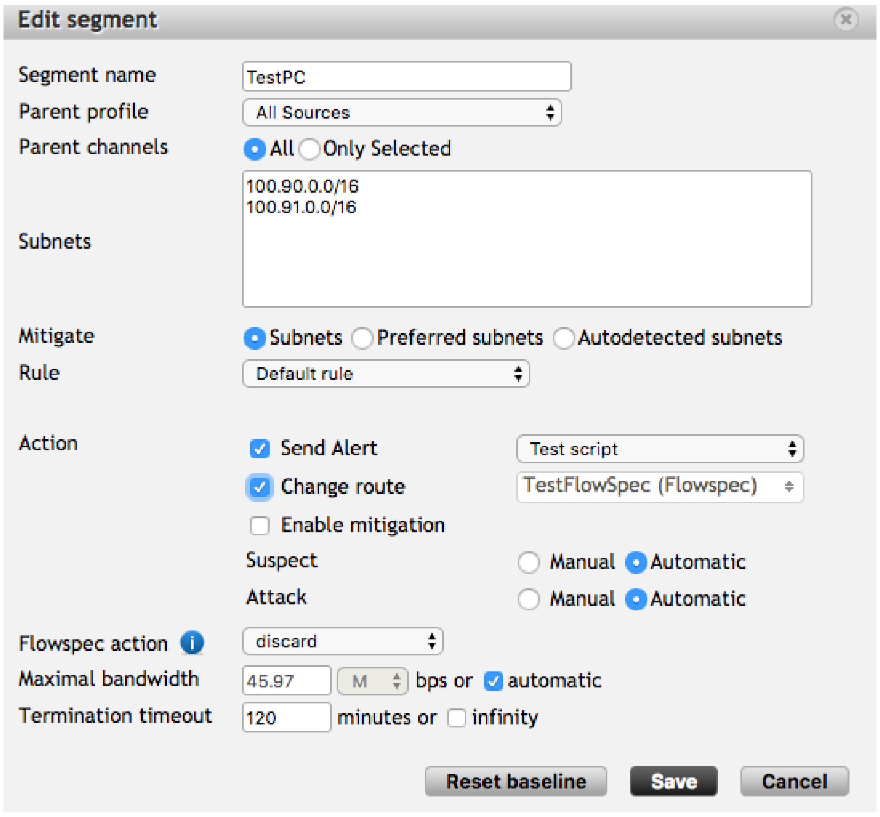
This script uses NETX API for accessing the NETX router and writing the rules. In the first part of the script, the REST API URL is chosen, and it depends on the Fastnetmon generated a notification. After choosing the URL, the options are filled, and the request is sent to the NETX router. Function `parse_extra_options()` is responsible for parsing the script options. Top N source IP addresses are extracted. If the source addresses are not available, the script uses destination address and creates filter based on the IP address. Rules are translated to `ethtool` syntax and applied to the interface. After adding the rule, it can be checked using the following command:

```
Command:
    $ netc show interface [interface-name] hw-filter

Example output:
    $ action drop dst [dst-address] dmask [dst-mask] id [ID]
```

6.4 DDoS Defender

DDoS Defender was also considered as a network analyzer. The first step was the system installation into the BUT testing network infrastructure followed by setup and configuration. The system is configured to dynamically respond to changing amount of traffic on the network and detect attacks in every condition. The following Figure 6.2 shows how can system respond to ongoing attacks.


Edit segment

Segment name: TestPC
Parent profile: All Sources
Parent channels: [✓ All] Only Selected
Subnets:
100.90.0.0/16
100.91.0.0/16
Mitigate: [✓ Subnets] Preferred subnets Autodetected subnets
Rule: Default rule
Action:
[✓ Send Alert] — Test script
[✓ Change route] — TestFlowSpec (Flowspec)
Enable mitigation
Suspect: Manual [✓ Automatic]
Attack: Manual [✓ Automatic]
Flowspec action: discard
Maximal bandwidth: 45.97 M bps or [✓ automatic]
Termination timeout: 120 minutes or infinity

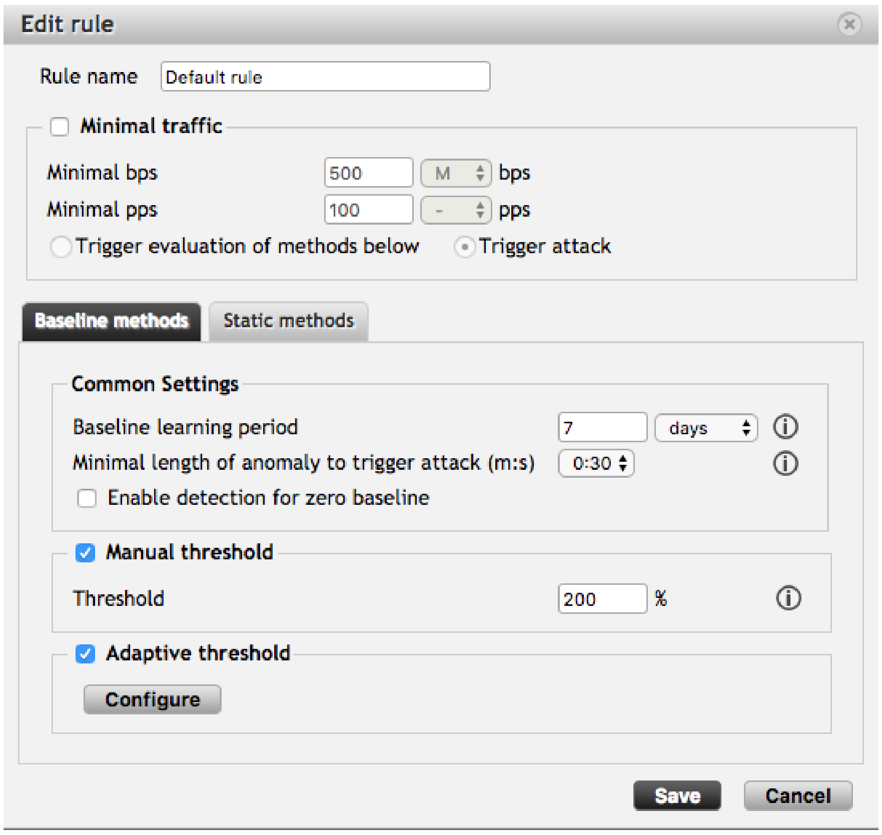
Figure 6.2: DDoS Defender segment.

DDoS Defender gets NetFlow data from a probe on the network and analyzes this traffic. There are a few threshold settings as can be seen in the following Figure 6.3. These settings are essential in deciding if the attack can be classified as DDoS. Referring to actual configuration, baseline learning period is set to 7 days to have a very accurate measurement. There is also manual threshold set which is set to 200 % of baseline traffic.

Figure 6.2 shows triggered actions after DDoS attack is detected. In this case, an alert is generated and the script named `Test script` is launched. BGP Flowspec message is also sent. The following example shows a basic script launched after DDoS attack is detected. Script parses data and stores them into variables. All the information is then printed.

```
$ /usr/local/bin/iad_alert_functions
$ parse_alert_data    # parse alert data and store them to variables
$ alert_data_print
```

Received information about the attack is parsed and saved to a *sqlite* database which contains only one table with the flow definition and a rule ID on the added network interface. Rule is sent into NETX REST API which was already presented. Top N source IP prefixes, source addresses or destination IP addresses can be used in mitigation rule. After this rule is added, the script is trying to get the rule ID via NETX API and this ID is also saved to the database. ID is important for rule removal. When attack ends, this rule has to be removed and to be removed rule ID has to be used.


Edit rule

Rule name: Default rule

☐ Minimal traffic
Minimal bps: 500 M bps
Minimal pps: 100 - pps
Trigger evaluation of methods below [✓ Trigger attack]

Common Settings
Baseline learning period: 7 days
Minimal length of anomaly to trigger attack (m:s): 0:30
☐ Enable detection for zero baseline

☑ Manual threshold
Threshold: 200 %

☑ Adaptive threshold

Figure 6.3: DDoS Defender rule configuration.

6.5 BIRD

BIRD was described in section 2.5. BIRD is also used as the main routing daemon in NETX. This section describes internals of BIRD, its internal design, architecture, and modifications made to call external scripts. For the needs of this thesis, BIRD version 2.0.2 is used, which is an experimental version, but with BGP Flowspec support.

BIRD is not allowing to call external scripts to parse data by default. This can be limiting for automatically do some action in case of any change in the routing table. How to achieve such action? The first approach would be to periodically try to get information from Flowspec table using BIRD client. For example output of the Flowspec table displaying BIRD client birdc as shown on example 6.4. This method is called polling and would increase the load of the system. Second approach is to modify the BIRD source code to be able to call external scripts in case of a change in a routing table. In this thesis, the second approach was chosen, and BIRD routing daemon is modified to call external scripts.

Modification of the routing daemon consists of several parts. In the first part, it was necessary to thoroughly study the structure and internal operations of the core of the BIRD routing daemon. The most useful part is processing the received message in the `bgp/packets.c` source file. The received message is processed in *bgp_rx()* function. This function then processes the header of the received messages and sends the complete packet to *bgp_rx_packet()* function, which further distributes the packet according to its type. Furthermore, when receiving Flowspec message, this message is processed in *bgp_rx_update()* as it is a update message type. Consequently, based on the type of NLRI, it is determined that it is a flow definition, i.e. type `BGP_AF_FLOW4` and the message goes to *bgp_decode_nlri_flow4()* for IPv4. After that, the message itself is processed and entered into the flowtab4 table.

In the *bgp_decode_nlri_flow4()* function is the right place to modify the source code and to call an external script. The following example 6.3 shows the modified code. The script bird_parser receives a message converted to a text format in the form of a parameter.

```
// Decoding received message completed − run external script
char buf;
char cmd_buf;

// Convert Flowspec table entry to String
flow4_net_format(buf, buf_len, (const net_addr_flow4 ∗) n);

if (!flowspec_table) {
  // Removed from flowspec table
  sprintf(cmd_buf, "bird_parser remove−filter −c \"%s\"", buf);
} else {
  // Added to flowspec table
  sprintf(cmd_buf, "bird_parser add−filter −c \"%s\"", buf);
}

// Run bird_parser
int status = system(cmd_buf);
```

Listing 6.3: C pseudo-code for running an external script in case of received Flowspec message in BIRD daemon.

This modification is only experimental, but after the patch will be accepted by the community, this will be a part of the routing daemon itself. It will not be hardcoded, but a configuration option will be added to define the type of message and table which should be followed.

```
root@bird2:~# birdc show route table flowtab4
BIRD 2.0.0-pre0 ready.
flow4 { dst 0.0.0.0/0; src 19.19.19.19/32; proto 6; dport 321;
sport 123; } blackhole [flowstat4 22:53:51] * (200)
```

Listing 6.4: Output of the Flowspec table using BIRD daemon client `birdc`.

ExaBGP

ExaBGP can be used instead of BIRD as it is also a BGP routing daemon. ExaBGP provides a simple way to convert individual messages into plain text or JSON objects and further process these messages, or directly call an external script API in the event of an event. For BGP retrieval, the use of this tool would be a more appropriate choice, but from a general perspective, this tool cannot modify the system routing table itself. Therefore it is not appropriate to use it as the primary routing daemon in the system. The software is licensed under BSD3.

Chapter 7

Testing and experiments

To provide the best results the testing was done incrementally after each part was implemented. Some of the test cases are demonstrated in section 7.2. Testing was done by generating real DDoS attacks on a testing machine. Attack strength was up to 4 Gbps.

Brno University of Technology (BUT) testing network was used to test each implemented part. Test_pc (see Figure 7.1) was used to generate DDoS attacks which were distributed over the network. The DoS attack was generated as follows: A script was used to generate `pcap` file which was then sent using `pfsend`. The following pseudocode shows how the traffic was generated during tests. Source/destination IP addresses, as well as ports, can be defined to generate the required attack. To generate a real DDoS attack, the range of source and destination addresses, as well as ports, can be defined.

Generating pcap - pcap writer script generating UDP connections using destination port 123 which stands for NTP:

```
for (my $i = 1; $i < 1000000; $i++) {
    my $conn = $writer->udp_conn('100.92.42.42', 30000, '100.91.56.99', 123);
    $conn->write(0,"H" x 1000);
}
```

Sending pcap using pfsend, SPEED and MAC address have to be configured:

```
DEV=zc:enp1s0f1
while [ 1 ]; do
    pfsend -f test.pcap -n 0 -r $SPEED -i $DEV -m $MAC
    sleep 1
done
```

7.1 BUT Network

The whole testing was done in Brno University of Technology (BUT) testing network. This part of the network is prepared to handle a lot of traffic. In the Figure 7.1, the whole testing environment is shown. Each line is at least 10 Gbps with the exception of the link to the Internet. Internet link is set to 1 Gbps to do not generate a DDoS attack which would travel through the Internet. Test_pc is used to generate DDoS attacks. As was mentioned before, it is a regular computer using special `pfsend` script.

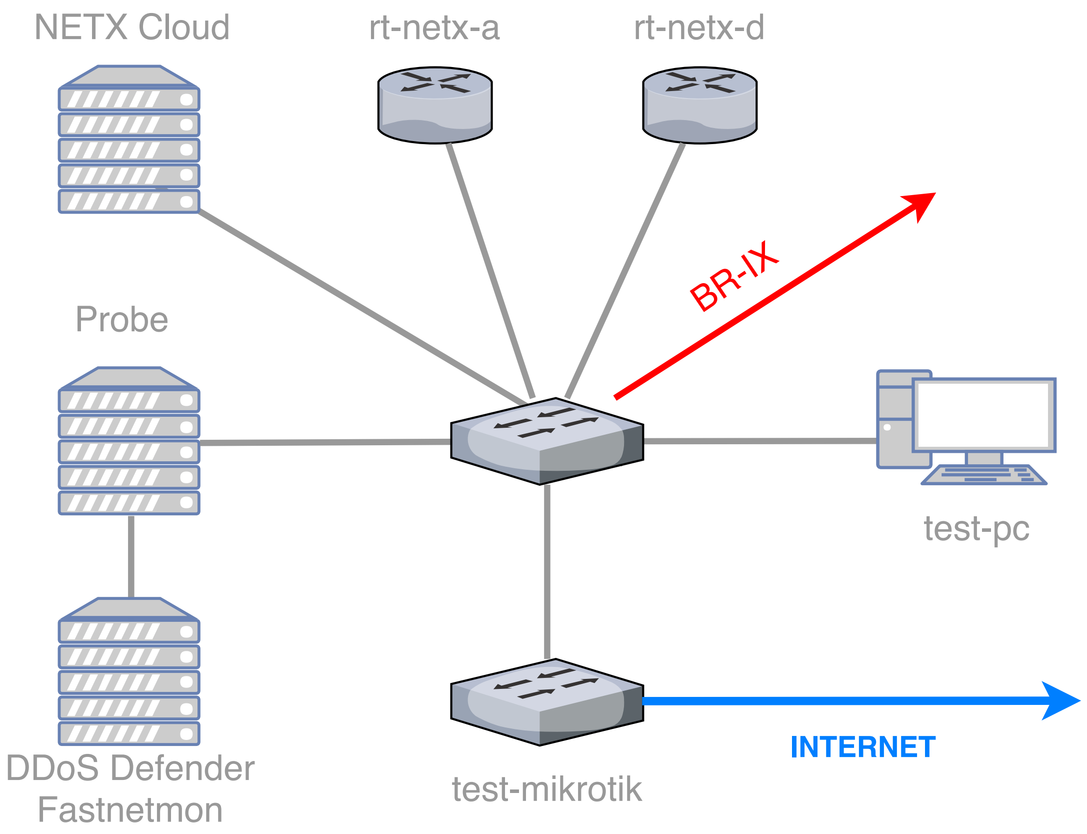


Figure 7.1: BUT Testing network.

As can be seen in Figure 7.1, DDoS detection tools are part of the topology. Traffic is mirrored from the switch to the network probe and then the NetFlow is sent to the traffic analyzer tools, in this case DDoS Defender or Fastnetmon. Detection tools detect the attack and run a custom script which gets essential information about the attack like top N source attacker IP addresses. The script sends this information to NETX using API or via BGP Flowspec. The Netc module ensures that hardware filtering is enabled and filters are correctly added.

The next step is to add the filtration mechanism to the live network. DDoS attack protection on NETX border routers in BUT campus network will be enabled within major software update.

7.2 Test cases

There are few components which were tested. Firstly, the hardware filtration test was run to verify the functionality of the filtration on each NIC. Secondly the tests of the Netc module in NETX. Lastly the tests of the whole system trying these three scenarios. First two tests are using the DDoS Defender as a detection tool, and API or Flowspec for sending the rules and the last is using Fastnetmon as a DDoS attacks detection tool. Attack generator described previously was used in each test case shown in this chapter.

All the tests use DDoS attack generator demonstrated above to generate the DDoS attack from the `test_pc` to the `rt-netx-a` or `rt-netx-c` which is located on NETX Cloud referring to 7.1. Destination IP address is set to `100.91.56.99` and source IP address to `100.92.42.42`, which is on the BUT private testing segment. The destination IP address is blackholed in BUT core network thus the traffic cannot get out to the Internet. The destination MAC address in DoS script is set to the MAC address of the `tge4` interface on the `rt-netx-a` router to get the DoS traffic to the destination.

Test cases are also recorded as a video for demonstration purposes and added to the attached media according to appendix A.

7.2.1 Hardware filtration tests

To test the hardware filtration, it was necessary to use bare metal NETX installation with network interface card (NIC). It was not possible to use a virtualized cloud version. Hardware filtration in this thesis was tested on the Intel `XL710` network card using the `i40e` driver. This NIC was used because it supports hardware filtration. However, the right drivers have to be installed, and filtration has to be enabled. Following part shows a workflow example how to verify that a new filtration rule was added to the NIC.

1. Make sure filtration on interface is enabled `tge4` in this case.

   ```
   $ ethtool -K tge4 ntuple on
   ```

2. Start the test DoS attack from the `test_pc` to the `rt-netx-a` according to 7.1.

3. Verify that the DoS attack traffic is received by the `rt-netx-a` router. For attack verification a `eth` tool is used. It monitors the received network traffic and prints the statistics. Traffic statistics from ongoing DDoS attack can be seen on the following example:

   ```
   +-----------+------------------+-------------------+
   |IFACE      |        RX        |        TX         |
   |           |  bytes/s  pkts/s |  bytes/s   pkts/s |
   +-----------+------------------+-------------------+
   |bond0      |  2454.6M  294.6k |  2454.7M  294.5k  |
   |tge3       |    73.9k    80.5 |    56.1k    59.5  |
   |tge4       |  2454.6M  294.6k |  2454.7M  294.5k  |
   +-----------+------------------+-------------------+
   ```

4. Apply a hardware filtration rule. This test is manual, so the rule for filtration is added manually. In this case the following rule is added:

   ```
   $ ethtool --config-ntuple tge4 flow-type ip4
   src-ip 100.92.42.42 action -1
   ```

5. Verify that the attack was sucesfully mitigated. This step requires to check the interfaces state again and see if the attack is stopped. Following output shows traffic after DDoS attack was stopped:

```
+-----------+------------------+------------------+
|IFACE      |        RX        |        TX        |
|           |  bytes/s   pkts/s |  bytes/s   pkts/s |
+-----------+------------------+------------------+
|bond0      |   152.0k   221.5  |   115.6k   134.0  |
|tge3       |    73.9k    80.5  |    56.1k    59.5  |
|tge4       |    78.0k   141.0  |    59.6k    74.5  |
+-----------+------------------+------------------+
```

The following example shows almost the same test, when filtration rule using source and destination IP address is tried to be used. This rule cannot be added properly and the DDoS attack is not stopped. The same problem occurs when subnet mask is defined. In this state, the card could filtrate traffic only based on single source or destination IP adderess and port.

```
$ ethtool --config-ntuple tge4 flow-type ip4
src-ip 100.92.42.42
dst-ip 100.91.56.99 action -1
```

7.2.2 Fastnetmon tests

Previous test case showed how to add a hardware filtration rule using ethtool. This test case is focused on Fastnetmon functionality and its connection with NETX router via REST API. Steps provided during the testing:

1. Start the test DoS attack from the `test_pc` to the `rt-netx-a` according to 7.1.

2. Verify that Fastnetmon detected the attack, and the attack is visible on the NETX interface. This can be done using Grafana, available in Fastnetmon. Detected attack can be seen in the Figure 7.2.

3. Verify added rules on the NETX router interface using `ethtool`. Output of added filtration rules can be obtained also using Netc `hw-filter` module. Filtration is done using destination rule. Output of the currently added filtration rules:

```
Filter: 2044
     Rule Type: TCP over IPv4
     Src IP addr: 0.0.0.0 mask: 255.255.255.255
     Dest IP addr: 100.91.56.99 mask: 0.0.0.0
     TOS: 0x0 mask: 0xff
     Src port: 0 mask: 0xffff
     Dest port: 0 mask: 0xffff
     VLAN EtherType: 0x0 mask: 0xffff
     VLAN: 0x0 mask: 0xffff
     User-defined: 0x0 mask: 0xffffffffffffffff
     Action: Drop
```

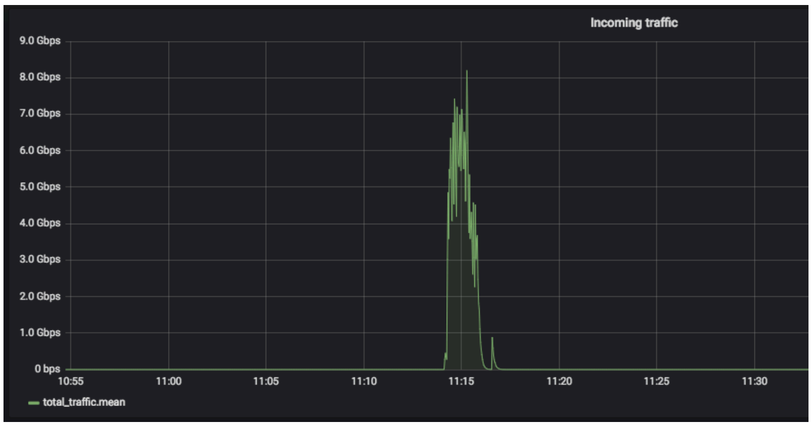
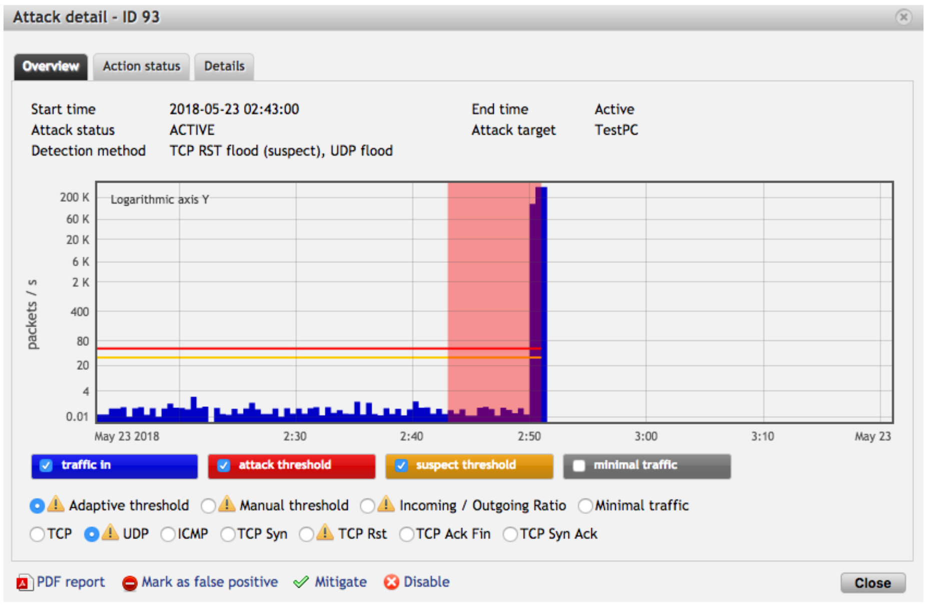


Figure 7.2: Fastnetmon - DDoS attack using Grafana.

7.2.3 DDoS Defender API tests

The following test case is using the DDoS Defender with a combination of the REST API and hardware filtration on the NETX router. This test case is similar to the previous with the difference of using DDoS Defender instead of Fastnetmon. In this case, removing rule instead of adding a new one was verified. The test steps are very similar to the previous and described in the following example:

1. Start the DoS attack from the `test_pc` to the `rt-netx-a` according to 7.1.
2. Verify that DDoS Defender detected the attack, and the attack is visible on the NETX interface. In this case, Flowmon monitoring center shown in Figure 7.3 was used for attack visualization which is a part of the DDoS Defender.



Attack detail - ID 93

Start time	2018-05-23 02:43:00	End time	Active
Attack status	ACTIVE	Attack target	TestPC
Detection method	TCP RST flood (suspect), UDP flood		

[✓ traffic in] [✓ attack threshold] [✓ suspect threshold] minimal traffic

[● Adaptive threshold] Manual threshold Incoming / Outgoing Ratio Minimal traffic

TCP [● UDP] ICMP TCP Syn TCP Rst TCP Ack Fin TCP Syn Ack

Figure 7.3: DDoS Defender attack visualization.

3. Verify added rule on database on DDoS Defender and using `netc` command line interface on NETX router:

```
Command:
    $ netc show interface tge4 hw-filter
Output:
    $ action drop src 100.92.42.42 id 2045
```

4. Stop the DoS attack – wait for attack end detection by DDoS Defender.

5. Verify the rule removal after attack ended.

```
Command:
    $ netc show interface [interface-name] hw-filter
Output:
    $
```

7.2.4 Flowspec tests

As in the previous test case, this test case also uses DDoS Defender for attack detection, but with the help of the Flowspec protocol. DDoS Defender is reconfigured to send BGP Flowspec messages to a router in case of DDoS attack. In this test case, the DDoS Defender is connected via BGP with the `rt-netx-c` router, which has a modified BIRD routing daemon installed. As mentioned earlier, modified BIRD is capable of running external scripts in case of Flowspec message is received. This test case is providing the following steps:

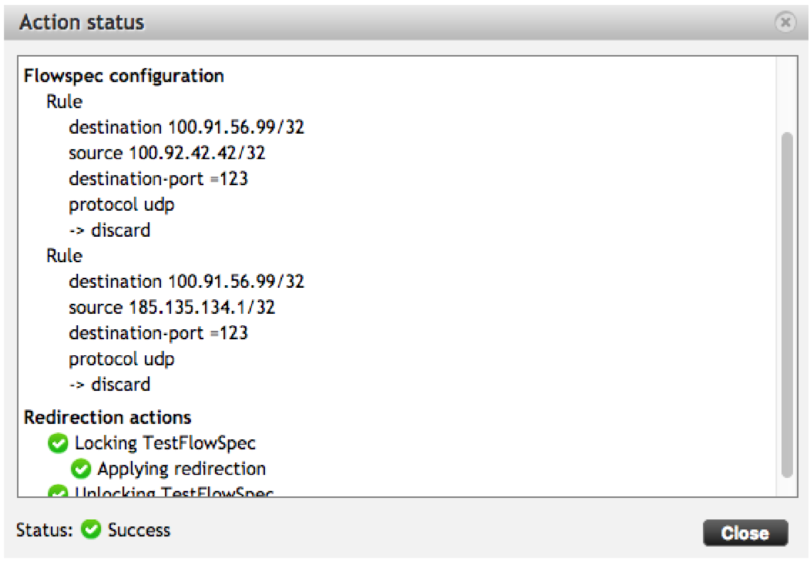
1. Configuring and generating the DoS attack as shown in previous tests.

2. Verification of the attack detection by DDoS Defender as well as started mitigation. Figure 7.4 shows the mitigation in DDoS Defender platform:

3. Verification of received Flowspec message on BIRD routing daemon:

```
root@bird2:~# birdc show route table flowtab4
BIRD 2.0.2 ready.
flow4 { dst 100.91.56.96/32; src 185.135.134.1/32;
proto 17; dport 123; } [IBGP_DDOS_DEFENDER
12:12:39.037 from 147.229.3.139] * (100) [i]

flow4 { dst 100.91.56.96/32; src 100.92.42.42/32;
proto 17; dport 123; } [IBGP_DDOS_DEFENDER
12:12:39.030 from 147.229.3.139] * (100) [i]
```



```
Flowspec configuration
  Rule
    destination 100.91.56.99/32
    source 100.92.42.42/32
    destination-port =123
    protocol udp
    -> discard
  Rule
    destination 100.91.56.99/32
    source 185.135.134.1/32
    destination-port =123
    protocol udp
    -> discard
Redirection actions
  ✓ Locking TestFlowSpec
    ✓ Applying redirection
  ✓ Unlocking TestFlowSpec
Status: ✓ Success
```

Figure 7.4: DDoS Defender DDoS attack mitigation - BGP Flowspec.

7.3 Test results

Test scenarios have shown, the NIC is not adding the filtration rules correctly. System can not add rules where both source and destination is defined, whether it is an address or a port. This is not so important in trying to protect against DDoS attacks. It is usually sufficient to use only the source or destination address. However, the impossibility of defining a mask, which in practice leads to the impossibility of filtering packets from the entire range of addresses, is a more serious problem. This can be limiting in case of administrator trying to block the whole subnet of attacking addresses.

There are some other test outputs, DDoS Defender can detect DDoS attacks faster than Fastnetmon in similar conditions. However, both tools are very reliable, and false positive detections were rare in tested environment.

Distributing the rules over Flowspec is without any problems, but when comparing to using a custom solution with use of REST API, Flowspec can be limiting. It can be limiting, when defining some advanced flow specification rules. For example, filtration based on higher layers of the OSI model. However, this is not a real problem, because the hardware filtration on Intel NICs is also similarly limited.

Chapter 8

Conclusion

This thesis provides an overview of the router distributions based on GNU/Linux and their configuration options. Platforms like VyOs, OpenWrt, and NETX were tested. NETX was chosen as the target platform for DDoS attack mitigation. The reason was its simple, configurable modular design that can be extended with a custom new module. Optimization for high-performance networks and REST API are also benefits.

Next chapters are focused on a review of the DDoS attacks, their types and how to create and detect a DDoS attack. Running a basic DDoS attack nowadays is not difficult or expensive. There are many techniques to mitigate the DDoS attack, some of them are discussed in this thesis. DDoS Defender and Fastnetmon have been used as tools for DDoS detection.

After studying DDoS attacks, the mitigation system was designed and implemented. It uses the output from DDoS load analyzers such as DDoS Defender from Flowmon or Fastnetmon. Outputs from these analyzers were parsed using scripts, which were implemented, and useful information such as source or destination IP addresses were used for mitigation. Information about attacks was then sent to NETX border routers which are responsible for the mitigation. Hardware filtration module for NETX was implemented to support hardware filtration in NETX system. The advantage of this approach is the impact on system performance is limited. One of the most significant disadvantages can be considered the problem with network card support from hardware manufacturers. Lack of documentation or not working filtration rules are only a few problems which occurred while implementing this mitigation system.

For communication with NETX routing platform sending hardware rules from analyzers to NETX border router, two mechanisms were tested. Firstly sending hardware rules using NETX Netc API. The second solution was using BGP protocol for sending rules over the network. BGP added a Flow Specification rules extension defined in RFC 5575 [21]. To use Flowspec, BIRD modification in a routing daemon written in C was required to be able to call an external script after the Flowspec table was updated. These changes are free to use for the whole community.

The last part of the work is devoted to result of the testing and checking the functionality of the independent parts of the project. Tests indicate that the developed scripts and whole designed architecture can be used as a system for effective DDoS mitigation. However, the support from hardware manufacturers can cause a problem and can lead to the whole system instability. The drawback is also the limited number of rules which can be loaded to the hardware network card. These drawbacks can be circumvented using software filtration.

8.1 Future work

The DDoS attack mitigation system can be improved in many ways. The most critical is to try hardware filtration on cards from another vendor than Intel. Providing feedback to these manufacturers is also important to get better results. The problem with lack of documentation could be resolved with the help of the community and support of the hardware card manufacturers.

There are also some new features which could be implemented. For example, the NETX module for filtration could automatically decide if the rule should be loaded to the hardware or use a software filtration based on for example `iptables`. This decision could be made based on a rule complexity. In case the rule is too complex, it is not possible to use hardware for filtration, the software filtration mechanism would be used.

Currently, this solution is used in BUT network.

Bibliography

[1] The BIRD Internet Routing Daemon. [Online; visited 2018-04-21].
Retrieved from: http://bird.network.cz

[2] The cost of launching a DDoS attack. [Online; visited 2018-02-11].
Retrieved from:
https://securelist.com/the-cost-of-launching-a-ddos-attack/77784

[3] NScrub. [Online; visited 2018-03-11].
Retrieved from: https://www.ntop.org/products/ddos-mitigation/nscrub

[4] OpenWrt – operating system architecture. [Online; visited 2018-05-18].
Retrieved from: https://wiki.openwrt.org/doc/techref/architecture

[5] The UCI System. [Online; visited 2018-05-18].
Retrieved from: https://wiki.openwrt.org/doc/uci

[6] Welcome to OpenWrt. [Online; visited 2018-01-13].
Retrieved from: https://openwrt.org/start

[7] Cumulus Linux. 2018. [Online; visited 2018-02-18].
Retrieved from: https://cumulusnetworks.com/products/cumulus-linux/

[8] Akamai: Q3 2017 State of the Internet – Security Report. 2017. [Online; visited 2017-08-11].
Retrieved from: https://content.akamai.com/us-en-pg10101-q3-17-state-of-the-internet-security-report.html

[9] AT&T: AT&T to Acquire Vyatta Software Technology from Brocade. 2018 [Online; visited 2017-10-20].
Retrieved from: http://about.att.com/story/att_to_acquire_vyatta_software_technology_from_brocade.html

[10] Brůna, Z.: Jak vylepšujeme DNS infrastrukturu pro .CZ? 2017. internet a Technologie 17.2.
Retrieved from: https://www.nic.cz/konference-it/#Photo[it17.2]/0/

[11] Cisco: Configuring IP Access Lists. 2007 [Online; visited 2018-1-5].
Retrieved from: https://www.cisco.com/c/en/us/support/docs/security/ios-firewall/23602-confaccesslists.html

[12] Cisco: ASR9000/XR: Understanding BGP flowspec. 2014 [Online; visited 2018-4-6].
Retrieved from: https://supportforums.cisco.com/t5/service-providers-documents/asr9000-xr-understanding-bgp-flowspec-bgp-fs/ta-p/3139916

[13] Cisco: Cisco ASR 9000 Series Aggregation Services Router Routing Configuration Guide. 2018 [Online; visited 2018-4-5].
Retrieved from: https://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k_r5-2/routing/configuration/guide/b_routing_cg52xasr9k/b_routing_cg52xasr9k_chapter_011.html

[14] Clayne Robison: How to Set Up Intel® Ethernet Flow Director. 2017 [Online; visited 2018-05-18].
Retrieved from: https://software.intel.com/en-us/articles/setting-up-intel-ethernet-flow-director

[15] Cloudflare: Cloudflare Advanced DDoS Protection. 2017 [Online; visited 2018-05-20].
Retrieved from:
https://www.cloudflare.com/media/pdf/cloudflare-whitepaper-ddos.pdf

[16] Darknet: Low Orbit Ion Cannon DDoS Booter. 2017. [Online; visited 5-January-2018].
Retrieved from: https://www.darknet.org.uk/2017/10/loic-download-low-orbit-ion-cannon-ddos-booter

[17] Dzurenda, P.; Martinasek, Z.; Malina, L.: Network Protection Against DDoS Attacks. *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*. vol. 4, no. 1. 2015: pp. 8–14. ISSN 1805-5443.

[18] Flowmon: DDoS Defender. 2018 [cit. 2017-01-10]. [Online; visited 2018-02-11].
Retrieved from: https://www.flowmon.com/en/products/flowmon/ddos-defender

[19] Haas, J.: Clarification of the Flowspec Redirect Extended Community. RFC 7674. RFC Editor. October 2015.
Retrieved from: http://www.rfc-editor.org/rfc/rfc7674.txt

[20] Lee, Y.-J.; Baik, N.-K.; Kim, C.; et al.: Study of detection method for spoofed IP against DDoS attacks. *Personal and Ubiquitous Computing*. vol. 22, no. 1. Feb 2018: pp. 35–44. ISSN 1617-4917. doi:10.1007/s00779-017-1097-y.
Retrieved from: https://doi.org/10.1007/s00779-017-1097-y

[21] Marques, P.; Sheth, N.; Raszuk, R.; et al.: Dissemination of Flow Specification Rules. RFC 5575. RFC Editor. August 2009.
Retrieved from: http://www.rfc-editor.org/rfc/rfc5575.txt

[22] Mirkovic, J.; Reiher, P.: A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Comput. Commun. Rev.*. vol. 34, no. 2. April 2004: pp. 39–53. ISSN 0146-4833. doi:10.1145/997150.997156.
Retrieved from: http://doi.acm.org/10.1145/997150.997156

[23] NETX: NETX at a glance. 2017. [Online; visited 2017-12-04].
Retrieved from: https://netx.as/netx-at-a-glance

[24] NETX: NETX Smart Router Series Comparison. 2017. [Online; visited 2017-12-04].
Retrieved from: http://netx.as/netx-smart-router-series

[25] Pavel Odintsov: Fastnetmon. 2017 [Online; visited 2017-12-18].
Retrieved from: https://github.com/pavel-odintsov/fastnetmon

[26] Pavel Odintsov: Fastnetmon Documentation. 2017 [Online; visited 2017-12-19].
Retrieved from: https://fastnetmon.com/docs-fnm-advanced

[27] Rekhter, Y.; Li, T.; Hares, S.: A Border Gateway Protocol 4 (BGP-4). RFC 4271. RFC Editor. January 2006.
Retrieved from: http://www.rfc-editor.org/rfc/rfc4271.txt

[28] Tay, W. Y.: Scalable DDoS mitigation using BGP Flowspec. 2014. [Online; visited 2017-12-11].
Retrieved from: https://conference.apnic.net/data/37/apricot-2014-wei-yin-scalable-ddos-mitigation-using-bgp-flowspec_1393312254.pdf3

[29] VyOS: New to VyOS? 2018 [Online; visited 2017-11-13].
Retrieved from: https://vyos.io

[30] Winder, D.: A brief history of DDoS… and how to defend yourself and your customers. [Online; visited 2018-04-10].
Retrieved from:
https://www.solarwindsmsp.com/blog/brief-history-ddos-defend-customers

[31] World's largest 1 Tbps DDoS Attack. 2016 [Online; visited 2018-01-13].
Retrieved from: https://thehackernews.com/2016/09/ddos-attack-iot.html

Appendix A

CD Content

- **src/** - Application source codes,
- **test-cases/** - Test cases used to test the application,
- **thesis.pdf** - Thesis in pdf format,
- **src-latex/** - Source files of the thesis in latex,
- **video/** - Video demonstrations.