VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

ODVOZOVÁNÍ PRAVIDEL PRO MITIGACI DDOS ÚTOKŮ

INFERENCE OF DDOS MITIGATION RULES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DANIEL JACKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ŽÁDNÍK MARTIN, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Jacko Daniel**
Program: Informační technologie
Název: **Odvozování pravidel pro mitigaci DDoS útoků**
Inference of DDoS Mitigation Rules
Kategorie: Počítačové sítě
Zadání:

1. Nastudujte dostupnou literaturu o DDoS útocích a způsobech jejich potlačení.
2. Seznamte se s vysokorychlostním zařízením pro čištění síťového provozu DDoS Protector.
3. Navrhněte algoritmus pro odvození pravidel pro blokování v DDoS paketů, které jsou majoritně zastoupeny ve sledovaném DDoS útoku.
4. Navržený algoritmus implementujte.
5. Implementaci ověřte v laboratorním i reálném prostředí, pokud to bude možné.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Žádník Martin, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Táto práca sa zaoberá DDoS útokmi, ich typmi a spôsobmi ich potláčania. Cieľom práce je navrhnúť a implementovať algoritmus, ktorý by bol schopný odvodiť pravidlá pre blokovanie DDoS útoku. Pre túto úlohu bol zvolený algoritmus strojového učenia, rozhodovací strom, ktorý sa spustí pri detekcií útoku. Pracuje so vzorkou dát zachytených pri útoku a vzorkou legitímnej komunikácie. Súčasťou práce je taktiež opis formátu BPF a prehľad vykonaných experimentov.

Abstract

This thesis focuses on DDoS attacks, their types and means of their mitigation. The aim of the thesis is to design and implement an algorithm which would be able to derive rules to block DDoS attacks. For this, we chose the algorithm of machine learning, a decision tree, which starts operating as soon as the attack is detected. The algorithm operates with a sample of data detected during the attack, and with a sample of legitimate communication. A part of this thesis is also a description of a BPF format and an overview of executed experiments.

Kľúčové slová

DDoS, mitigácia, strojové učenie, automatizácia, DDoS Protector, filtrácia, sieť, BPF

Keywords

DDoS, mitigation, machine learning, automatization, DDoS Protector, filtration, network, BPF

Citácia

JACKO, Daniel. *Odvozování pravidel pro mitigaci DDoS útoků*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Žádník Martin, Ph.D.

Odvozování pravidel pro mitigaci DDoS útoků

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Martina Žádníka, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Daniel Jacko
11. mája 2021

Poďakovanie

Chcel by som sa poďakovať pánovi Ing. Martinovi Žádníkovi za jeho odborné rady, cenné pripomienky, ľudský prístup a veľa trpezlivosti pri vedení tejto bakalárskej práce.

Obsah

Kapitola 1

Úvod

Internet sa stal neoddeliteľnou súčasťou života. Možnosti jeho používania sa rozširujú, čím sa zvyšuje problém jeho bezpečnosti. Bezpečnosť počítačových sietí nie je vždy dostatočne implementovaná, a tak sa vytvára priestor pre sieťové útoky. Jedným z najbežnejších útokov je práve Distributed Denial of Service (DDoS), ktorému je venovaná kapitola 2. Jeho cieľom je znemožniť prístup užívateľom k sieťovému zariadeniu. Toto dosiahne vyčerpaním hardvérových zdrojov zariadenia, čo v určitej miere znemožní odpovedať na požiadavky užívateľov. Tento typ útokov je dnes možné kúpiť za niekoľkonásobne menšiu sumu, než je suma, ktorá vznikne obeti takéhoto útoku. V rámci boja proti DDoS útokom vzniklo mnoho komerčných aj open source nástrojov, ktoré sú schopné stlmiť dopad útoku, jedným z nich je aj DDoS Protector. Je to aktívne sieťové zariadenie vyvíjané združením CESNET v rámci projektu Liberouter. Dokáže detekovať a filtrovať rôzne druhy útokov, ale zameriava sa primárne na volumetrické útoky. Viac o DDoS Protectore ako aj o iných spôsoboch obrany pred DDoS útokmi je v kapitole 3. Táto kapitola sa tiež zaoberá jednotlivými krokmi pre úspešnú mitigáciu DDoS útoku. Cieľom tejto práce je návrh a implementácia algoritmu, ktorý by bolo možné spustiť v momente, kedy je na sieti detekovaný útok a dokázal by nájsť vzor v paketoch DDoS útoku a odvodiť pravidlá pre blokovanie týchto paketov. Formát, v ktorom má byť odvodené pravidlo je vysvetlený v kapitole 4. Tieto pravidlá budú následne použité na odfiltrovanie útočiacej prevádzky pomocou DDoS Protectoru. Kapitola 5 sa zaoberá strojovým učením a jedným konkrétnym algorimtom, ktorý bude využitý pri odvodzovaní pravidiel. V kapitole 6 bude vysvetlený výber metódy strojového učenia, objasnený princíp pre hľadanie vzorov v pakete a objasnení princíp tvorby mitigačných pravidiel. Kapitola 7 sa venuje experimentom, ktoré boli vykonané pomocou implementovaného algoritmu.

Kapitola 2

DDoS útok a jeho typy

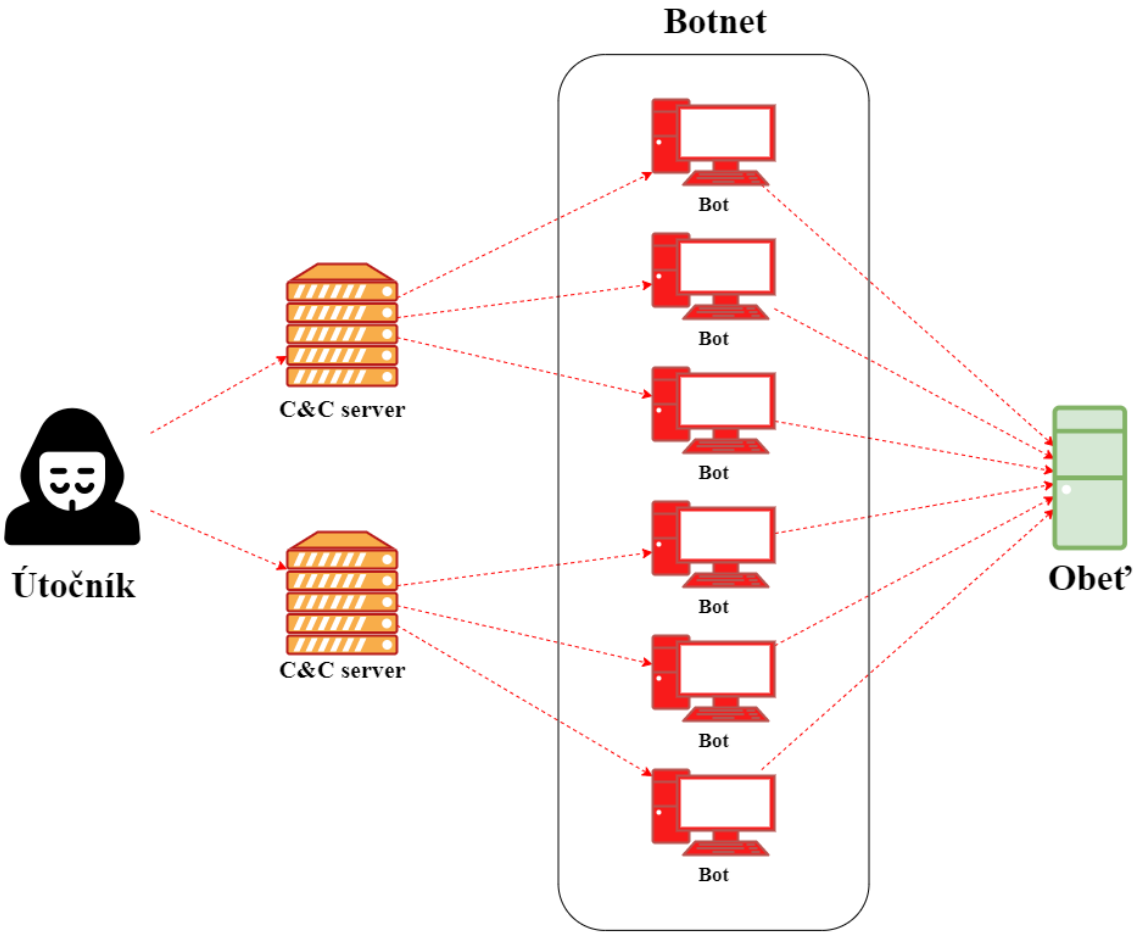
Táto kapitola sa zaoberá DDoS útokmi. Vysvetľuje samotný pojem a motiváciu k tomuto typu útoku. V úvode je opísaný jeho priebeh a následne sa zameriava na rolu botneta v útoku. Ďalej sú uvedené kategórie útokov ako aj jednotlivé druhy útokov.

DDoS (Distributed Denial of Service) útoky sú rastúcim rizikom stability a bezpečnosti internetu. Sú lacné a ľahko dostupné, vďaka čomu ich popularita rastie. DDoS útok by sa dal opísať ako veľký nával áut na cestnú komunikáciu, ktorej prevádzka sa dôsledkom toho spomalí. Cieľom DDoS útoku je dočasne alebo na neurčitý čas vyčerpať zdroje obete. Medzi zdroje môže patriť šírka pásma siete, výpočetný výkon alebo dátové štruktúry operačného systému. Útok, ktorý pochádza z jedného zdroja je nazývaný Denial of Service (DoS) [16]. Z jedného zariadenia je však pre útočníka ťažké dosiahnuť dostatočný objem útoku, ktorý je potrebný na znemožnenie služby. Preto sú Distributed Denial of Service (DDoS) útoky oveľa populárnejšie. Na zaistenie dostatočnej šírky pásma alebo výpočetného výkonu je DDoS spúšťaný z viacerých kompromitovaných zariadení, ktorými sú počítače alebo iné zariadenia, ktoré sú globálne prepojené a spravované, takzvaný botnet [18]. Na rozdiel od iných typov útokov, DDoS neslúži k získaniu neoprávneného prístupu k systému, stále má za úlohu iba znemožniť prístup k zariadeniu alebo službe pre ostatných užívateľov. Tento cieľ uskutočňuje zahltením cieľa množstvom požiadaviek. Útočník pošle infikovaným zariadeniam inštrukcie o útoku, a každé zo zariadení začne posielať požiadavky na cieľ. Server sa snaží obslúžiť všetky požiadavky, ale pokiaľ presiahnu jeho výkon, začne sa voči ostatným užívateľom správať pomaly alebo sa stane nedostupným. Pretože tieto zariadenia sú legitímne, je často ťažké rozlíšiť útočiacu a legitímnu prevádzku.

2.1 Botnet

Botnet môže pozostávať z tisícok zariadení. Pre vytvorenie botnetu musí útočník nájsť zraniteľné zariadenia, ktoré by mohol infikovať. Zraniteľné zariadenia sú väčšinou tie, ktoré nemajú žiadny antivírusový softvér, majú neaktualizovaný antivírusový softvér alebo neboli správne nastavené. Útočník tieto zariadenia vyhľadáva skenovaním internetu. Po tom, čo útočník tieto zariadenia objaví, ich infikuje softvérom, vďaka ktorému ich môže využívať v botnete a vzdialene spravovať. Vo väčšine prípadov si vlastníci týchto zariadení nie sú vedomí, že je ich zariadenie zneužité. Okrem infikovaných počítačov sa botnety čoraz viac skladajú z IoT (Internet of Things) zariadení [17], nakoľko počet týchto zariadení na sieti neustále rastie. Veľa týchto zariadení je navrhnutých bez akejkoľvek ochrany alebo je ich ochrana len veľmi slabá. Získať veľké množstvo zariadení do botnetu nemusí útočníkovi

zabrať veľa času, nakoľko existujú programy, ktoré sú pripravené na automatické vyhľadávanie zraniteľných systémov a ich infikovanie. Po tom, čo je zariadenie takto infikované, je taktiež schopné vyhľadávať zraniteľné zariadenia v okolí a infikovať ich. Vďaka tomuto je možné vytvoriť veľký botnet pomerne rýchlo. Napriek tomu, útočníci nemusia botnety vytvárať, keďže je možné si prenajať botnet vytvorený na účel prenajímania. Takáto služba nemusí stáť veľa na krátkodobé použitie, no tento typ krátkych útokov je podľa [12] stále najpopulárnejší, nakoľko sú tieto útoky vysoko efektívne. V momente, keď má útočník pripravený botnet z jedného alebo viacerých počítačov, ktoré zastávajú úlohu command and control (C&C) server, útočník odošle inštrukciu „launch“ botom[18].



Obr. 2.1: DDoS útok a botnet

2.2 Typy DDoS útokov

Existuje veľa typov DDoS útokov, no väčšina z nich by sa dala rozdeliť do troch kategórií [19]. Tými sú volumetrické útoky, útoky na protokol a útoky na aplikačnú vrstvu. Často je však možné jeden útok zaradiť do viacerých kategórií. V tejto sekcii bude vysvetlený princíp fungovania jednotlivých kategórií DDoS útokov a uvedený stručný zoznam DDoS útokov s ich krátkym popisom.

Volumetrické útoky sú najčastejším typom útoku. Ich cieľom je preťaženie serverových zdrojov ako šírka pásma siete, výpočetný výkon alebo dátové štruktúry operačného systému, veľkým množstvom paketov. Tento proces zapríčiní odmietnutie prístupu pre legitímnych používateľov. Volumetrické útoky zvyčajne stoja na slabých stránkach protokolov.

Útoky na protokol zneužívajú normálne správanie protokolov a ich slabé stránky, väčšinou sa jedná o 3. a 4. vrstvu OSI modelu. [21] Ich cieľom je vyčerpať výpočetné zdroje serveru a iných sieťových zariadení, ako napríklad firewally a balancery.

Útoky na aplikačnú vrstvu sa zameriavajú skôr na webové servery, platformy webových aplikácií a konkrétne webové aplikácie než sieť samotnú. Ich cieľom je spraviť webovú lokalitu nedostupnú pre užívateľov. Tento typ útokov zneužíva hlavne protokoly HTTP, HTTPS a SNMP. Využívajú o niečo nižšiu šírku pásma a tak ich je niekedy ťažšie detekovať.

Existuje veľké množstvo typov DDoS útokov. V nasledujúcich odsekoch bude uvedená iba časť z nich podľa [18] a [13].

- **SYN flood** – útok využíva protokol TCP a spolu s niektorými ďalšími, ktoré budú nasledovať, patrí pod tzv. záplavové útoky. Nadväzovanie spojenia cez TCP, tiež známe ako three-way handshake[1], v normálnom prípade prebieha nasledovne: Klient pošle serveru SYN paket, čím požiada o spojenie. Server odpovie paketom SYN-ACK, čím dáva najavo, že je pripravený nadviazať spojenie a čaká na potvrdzujúci ACK paket od klienta. Útok prebieha tak, že útočník pošle SYN paket s podvrhnutou IP adresou serveru (obeti). Server si rezervuje systémové zdroje na toto potencionálne spojenie a odpovie SYN ACK paketom podvrhnutej IP adrese. Server čaká na potvrdenie paketom ACK, no ten nikdy nedorazí a útočník naďalej posiela ďalšie SYN pakety. Server po určitom čase tieto zdroje uvoľní, avšak pri veľkom počte SYN paketov, môže nastať moment, kedy server vyčerpá všetky zdroje a nedokáže viac spracovať požiadavky od legitímnych používateľov. A práve to je cieľom tohto útoku.
- **UDP flood** – protokol UDP nevytvára spojenie, ktoré trvá (ako TCP), a teda neoveruje IP adresu, z ktorej bol zaslaný paket. Tento fakt využíva útočník a posiela množstvo paketov na náhodné porty serveru (obete). Server sa snaží zistiť, či nejaká aplikácia počúva na danom porte a následne server odosiela ICMP paket „destination unreachable“ pokiaľ na danom porte žiadna aplikácia nepočúva. Toto zaberá zdroje na serveri.
- **ICMP flood** – útočník využíva Internet Control Message Protocol (ICMP), ktorý slúži primárne na komunikáciu zariadení na sieťovej vrstve. Útočník posiela množstvo paketov typu Echo Request. Obeť túto žiadosť spracuje a odošle odpoveď Echo Reply, čo jej zaberie určité zdroje. Šírka pásma je zaberaná ako Echo Request, tak aj Echo Reply paketmi. Cieľom útoku je vyčerpať zdroje obeti alebo zahltiť linku.
- **DNS flood** – The Domain Name System (DNS) je systém, ktorého hlavnou funkciou je preklad doménových mien na IP adresy, keď sa užívateľ pokúša pripojiť na doménu. Pri tomto type útoku útočník posiela množstvo DNS dotazov na DNS resolver. Cieľom útoku je, aby bol server natoľko preťažený, že nedokáže ďalej odpovedať. Čo v konečnom dôsledku môže spôsobiť, že sa užívatelia nebudú vedieť dostať na určité domény.
- **DNS amplifikácia** – na rozdiel od DNS flood, kde je obeťou priamo DNS resolver, je v tomto prípade DNS resolver naopak použitý ako zbraň. Útočník použije voľne

[1] https://www.sciencedirect.com/topics/computer-science/three-way-handshake

dostupný DNS resolver, na ktorý posiela dotazy so zdrojovou IP adresou obete. Nakoľko paket vyzerá, že prišiel od obete, DNS resolver pošle odpoveď obeti, aj keď tá žiadny paket na DNS resolver neposlala.

- **Slowloris** – je útok, ktorý využíva aplikačnú vrstvu. Útočník sa pokúša otvoriť čo možno najviac spojení so serverom zároveň a udržať ich otvorené najdlhšie ako sa dá. Útok prebieha nasledovne: Útočník posiela čiastočné žiadosti serveru (obeti). Server má nastavený time-out, po ktorom začne spojenia zatvárať, preto útočník vždy tesne pred vypršaním time-outu pošle ďalší paket. Server dokáže poňať len určité množstvo spojení naraz, preto v momente, keď je toto množstvo vyčerpané, útočník dosiahol svoj cieľ. Server nedokáže komunikovať s ostatnými užívateľmi, ktorí sa snažia nadviazať spojenie. Tento typ útoku je ťažké identifikovať, nakoľko zaberá malú šírku pásma a funguje na aplikačnej vrstve, keď je TCP spojenie nadviazané, a preto vyzerá legitímne.
- **HTTP flood** – Pri tomto type útoku posiela útočník na server (obeť) množstvo HTTP GET a POST dotazov. Útočník často vyberá výpočetne najdrahšie časti webstránok. HTTP dotaz je síce malý, ale jeho spracovanie a odpoveď môže zaberať oveľa viac zdrojov. Napríklad, keď server musí opakovane pristupovať do databázy alebo posielať obrázky a súbory. Server môže byť nedostupný okrem zahltenia siete aj z dôvodu nedostatku výpočetných zdrojov[9].

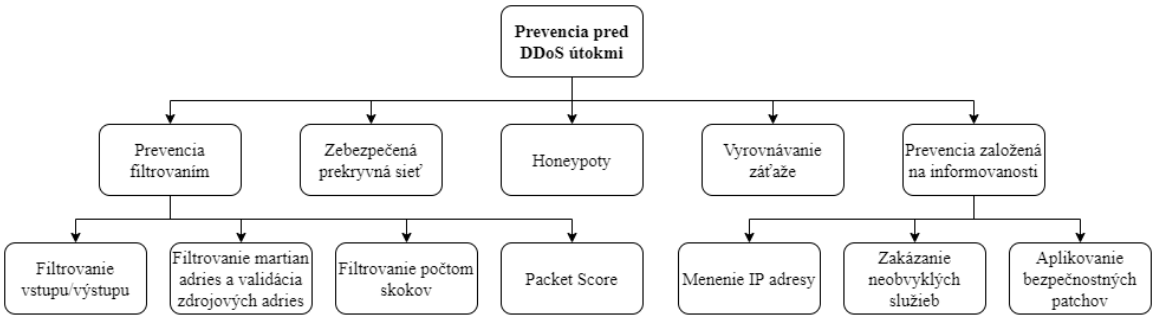
Kapitola 3

Obrana pred DDoS útokmi

Táto kapitola sa zaoberá spôsobmi obrany pred DDoS útokmi. Obranné spôsoby môžeme rozdeliť do 2 hlavných kategórií a to prevencia a mitigácia. Vzhľadom na to, že sa útočníci snažia svojimi útokmi napodobniť legitímnu prevádzku, narážame na problém, kedy môžeme nevedome pomôcť útočníkovi. Môže nastať situácia, že aj napriek tomu, že sa nám podarí pred útokom obrániť a naša služba ostane dostupná, spôsoby, ktoré použijeme na prevenciu a mitigáciu, zároveň obmedzia prístup časti legitímnych používateľov. Preto je jednou z hlavných výziev obrany pred DDoS útokmi nízka miera falošne pozitívnych označení prevádzky. DDoS útok nemusí trvať dlho, aby splnil svoj účel. Podľa [8] sú najčastejším cieľom DDoS útokov pri zákazníkoch Akamai Security práve online hry. Stačí krátky DDoS útok, ktorý nemusí siahať ani len do desiatok minút a prebiehajúci zápas v online hre bol prerušený a hráči odpojení. Práve kvôli takýmto prípadom je okrem miery falošne pozitívnych označení veľkou výzvou aj rýchla reakcia na útok. V tejto kapitole budú opísané najčastejšie typy prevencie a obrany pred DDoS útokmi podľa [13].

3.1 Prevencia pred DDoS útokmi

Prevencia pred DDoS útokmi je veľmi žiadúcou ochranou. Ako bolo spomenuté skôr, niekedy je cieľom útokov narušiť službu a spôsobiť jej nedostupnosť na krátky čas, bez nutnosti dlhotrvajúceho výpadku siahajúceho na desiatky minút až hodiny. Práve v takých prípadoch záleží na tom, aby bola odozva na útok okamžitá. Práve v tom vyniká prevencia pred DDoS útokmi, nakoľko reakcia neprichádza až po tom, ako bol útok spustený, ale už predtým. Vďaka tomu je možné zamedziť, aby sa služba stala nedostupnou čo i len na okamih. Ako príklad môžeme uviesť e-športy, kde je veľmi žiadúce, aby zápas nebol chybou prerušený alebo nedostupný čo i len jednému hráčovi na krátku chvíľu, keďže často nie je možné určitú udalosť, ktorá sa stala, vrátiť späť a tá tým pádom ovplyvní celý zápas. V nasledujúcich odsekoch budú opísané jednotlivé metódy, ich delenie možno vidieť na obrázku 3.1.



Obr. 3.1: Prevencia pred DDoS útokmi

3.1.1 Prevencia filtrovaním

Ak chceme zabrániť tomu, aby sa útočiaca prevádzka nachádzala v sieti, je veľmi dôležité ju filtrovať. Filtrovanie, okrem toho, že predchádza tomu, aby sa v sieti nachádzala útočiaca prevádzka, tiež zabezpečuje to, aby sa užívateľ nestal nevedomým útočníkom tým, že by bolo jeho zariadenie zneužité ako súčasť botnetu. Medzi techniky filtrovania patria napríklad:

- **Filtrovanie vstupu/výstupu** – Je definované v RFC 2267[1]. Táto technika zabraňuje podvrhnutým IP adresám vstúpiť a opustiť sieť. Filtrovanie vstupu filtruje prevádzku smerujúcu do lokálnej siete a filtrovanie výstupu zahadzuje podvrhnutú prevádzku, ktorá opúšťa lokálnu sieť. Filtrovanie vstupu filtruje tú prevádzku, ktorá nespadá do preddefinovaného rozsahu doménového prefixu siete. Pre úspešnosť tejto techniky je teda potrebné poznať rozsah očakávaných IP adries. Toto však nie je vždy možné, vzhľadom na používané zložité topológie v jednotlivých sieťach. Táto forma filtrovania, je teda účinná len proti podvrhnutým IP adresám, ktoré nepatria do rozsahu siete z ktorej sú odosielané.

- **Filtrovanie martian adries a validácia zdrojových adries** – Je definované v RFC 1812. Udáva, že smerovač nemá posielať ďalej žiadne správy, ktoré majú nevalidnú zdrojovú alebo cieľovú IP adresu. Zdrojová adresa je nevalidná, ak patrí medzi špeciálne adresy, definované v RFC 1812[2] odsek 4.2.2.11 a 5.3.7 alebo nie je adresou typu unicast. Taktiež sa nemajú posielať ďalej pakety, ak ich zdrojová IP adresa je zo siete 0 a 127(výnimkou je rozhranie loopback). Cieľová adresa je nevalidná, ak patrí medzi adresy definované ako ilegálne v RFC 1812 odsek 4.2.3.1 alebo je to adresa triedy E s výnimkou 255.255.255.255. . Taktiež sa nemajú posielať ďalej pakety, ak ich cieľová IP adresa je zo siete 0 a 127(výnimkou je rozhranie loopback).

- **Filtrovanie počtom hopov** - je metóda, ktorá sa odráža od faktu, že nie je možné zmeniť počet skokov - hopov paketu, keď cestuje zo zdroja k cieľu. Na základe tohto počtu táto metóda rozhoduje, či je paket validný. Metóda používa TTL na počítanie hopov. Tieto hopy sú uložené v mapovacej tabuľke pre každú zdrojovú adresu. Pri príchode paketu je vypočítaný potrebný počet hopov pre tento paket a toto je porovnané s mapovacou tabuľkou. Pokiaľ sa hodnoty nerovnajú, paket je označený za podvrhnutý. Avšak počiatočná hodnota TTL sa môže v rôznych operačných systémoch líšiť, tým pádom vzniká veľa falošne pozitívnych označení. Taktiež pri veľkom objeme

[1] https://datatracker.ietf.org/doc/html/rfc2267
[2] https://datatracker.ietf.org/doc/html/rfc1812

paketov sa môže stať, že budú vyčerpané výpočetné zdroje systému pri výpočte hopov a tak sa stane obeťou útoku.

- **Packet Score** - je proaktívna filtračná technika, ktorá používa Bayesovu vetu na výpočet skóre pre každý paket na základe atribútov, ktoré obsahuje. Keď je skóre vypočítané, táto technika začne vyberať pakety na zahodenie na základe ich skóre. Používa prahovú hodnotu na rozhodovanie, či je treba zahadzovať. Táto hodnota je dynamicky upravovaná na základe preťaženia systému a rozdelenia skóre prichádzajúcich paketov. [11]

3.1.2 Zebezpečená prekryvná sieť – Secure Overlay

Táto technika funguje na základe vytvorenia virtuálnej vrstvy siete nad fyzickou sieťou. Táto virtuálna sieť je vstupným bodom pre vonkajšiu sieť na vytvorenie komunikácie s chránenou sieťou. Predpokladá sa, že izolácia môže byť dosiahnutá, ak chránená sieť skryje svoje IP adresy alebo použije distribuovaný firewall. Tento firewall zaručí, že iba dôveryhodná komunikácia s virtuálnej vrstvy bude pustená do chránenej siete.

3.1.3 Honeypoty

Sú to časti siete, ktoré sú menej zabezpečené a teda náchylnejšie na útoky. Ich cieľom je nalákať útočníka aby na ne zaútočil. Honeypoty sa tvária ako legitímna sieť, aby si útočník myslel, že zaútočil na pravú sieť. Vďaka tomu zostáva pravá sieť chránená. Informácie získané na honeypote môžu byť následne využité na analýzu a tým pádom prevenciu pred ďalšími útokmi. Je avšak problém honeypoty zamaskovať ako pravú sieť. Taktiež, pokiaľ honeypot nezaregistruje útok, preposiela pakety ďalej do skutočnej siete.

3.1.4 Vyrovnávanie záťaže – Load Balancing

Hlavnou funkciou load balanceru je rozloženie pracovného zaťaženia, na viacero serverov, aby sa predišlo zaťaženiu konkrétneho servera. Vďaka tomu je taktiež možné zvýšiť produktivitu systému. V momente keď server čelí DDoS útoku, load balancer zaistí presmerovanie prevádzky na iný server, ktorý nie je pod útokom. Pre správne fungovanie load balanceru je potrebné mať dostatočnú šírku pásma v kritických uzloch siete, ako aj vhodný počet serverov a dáta centier.

3.1.5 Prevencia založená na informovanosti – Prevention based on awareness

Tento typ prevencie pred DDoS útokmi je možné okrem obete útoku uplatniť tiež na užívateľa, ktorého zariadenie môže byť zneužité ako súčasť botnetu. Ako príklad môžeme uviesť IoT zariadenia, ktoré sú často zneužívané ako súčasť botnetu, práve kvôli ich veľmi slabému zabezpečeniu. Žiaľ na vine sú často aj užívatelia týchto zariadení, ktorý nevedia o existencii DDoS útokov. Nastavenia IoT zariadení ponechávajú často pôvodné, bez zmeny hesla, čo bolo zneužité napríklad v útoku mirai [20]. Ďalej je uvedených niekoľko techník, ktoré môže užívateľ použiť a predísť tak DDoS útoku alebo participovaniu na ňom.

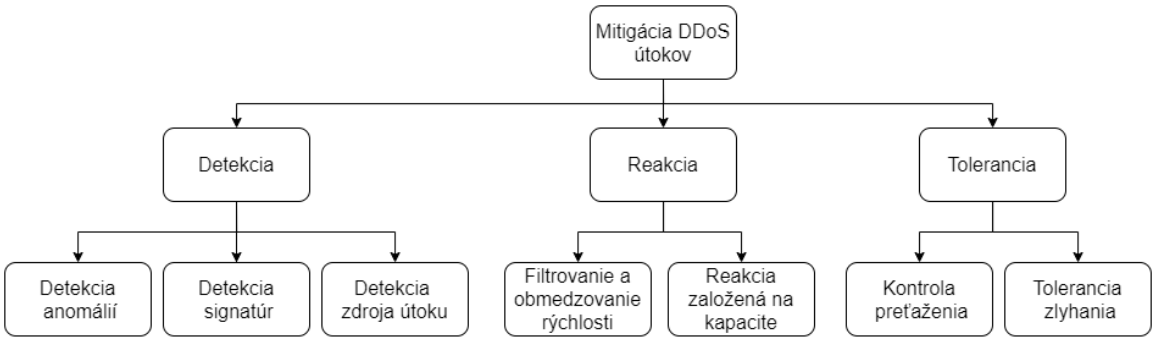
- **Menenie IP adresy** – ako už názov naznačuje, táto prevencia spočíva v menení IP adresy. Pokiaľ bol teda útok mierený na určitú IP adresu a útočník si nevšimne

zmenu, touto technikou je možné vyhnúť sa DDoS útoku. Samozrejme to so sebou nesie aj nevýhody súvisiace so zmenou IP adresy.

- **Zakázanie neobvyklých služieb** – Ide o prevenciu pred DDoS útokmi zakázaním služieb, ktoré môžu byť zneužité k útoku. Napríklad UDP echo, ktoré môže spôsobiť UDP flood. Na prevenciu pred participovaním v botnete sa zase odporúča na IoT zariadeniach zakázať protokoly na vzdialený prístup ako Telnet a SSH.
- **Aplikovanie bezpečnostných patchov** – Útočníci nemusia nutne ostať dlho pri jednej technike na prelomenie obrany, preto je nutné samotné zariadenie alebo bezpečnostný softvér na ňom pravidelne aktualizovať, aby bolo zariadenie chránené aj pred najnovšími hrozbami.

3.2 Mitigácia DDoS útokov

Prevencia pred DDoS útokmi je dôležitou súčasťou obrany pred DDoS útokmi, ale samotnú ju je ťažko považovať ako dostatočnú. Útočníci stále prichádzajú s novými hrozbami a po každom zákroku proti nim sa snažia zmene prispôsobiť. DDoS útoky sú účinným nástrojom na znemožnenie prístupu k službe, za čo sú ľudia stále ochotní platiť. Preto je možné predpokladať, že útočníci majú dostatočnú motiváciu na hľadanie zraniteľností a vyvíjanie nových techník ako obranu prelomiť. Preto sa na obranu pred DDoS útokmi používajú mitigačné techniky, ktoré sú navrhované tak, aby si dokázali poradiť aj s novými, zatiaľ nepreskúmanými typmi útokov. Mitigácia DDoS útokov je veľmi často podnetom k výskumnej činnosti. Mitigácia DDoS útokov sa skladá z troch rôznych mechanizmov. Tými sú: detekčný mechanizmus, reaktívny mechanizmus a mechanizmus tolerancie. Ďalej budú opísané jednotlivé mechanizmy a metódy, ktoré sa v nich používajú, ich prehľad je zobrazený na obrázku 3.2.



Obr. 3.2: Mitigácia DDoS útokov

3.2.1 Detekcia

Detekcia umožňuje odhalenie útoku. Je to preto dôležitý krok mitigácie DDoS útokov, nakoľko sa mitigácia od prevencie líši aj práve tým, že neprebieha neustále, ale čaká na povel na spustenie. Detekovať DDoS útok sa nezdá byť ťažké, nakoľko pri útoku môžeme pozorovať zhoršenie výkonu systému alebo zariadenia. Niekedy sa od detekcie požaduje označiť aj zdroj útoku. Dva základné typy mechanizmu detekcie sú: detekcia signatúr a detekcia anomálií.

Detekcia signatúr

Detekcia signatúr funguje na princípe porovnávania signatúr. Ide o metódu, ktorá je účinná voči už poznaným a dobre analyzovaným útokom, ktoré vykazujú určitý vzor. To, že útok neprebieha prvý krát, automaticky neznamená, že je možné ho presne detekovať. Pre správny popis útoku je potrebná analýza a pochopenie podstaty útoku. Hlavnou nevýhodou tejto techniky je, že detekuje iba už poznané útoky. Okrem toho, že nedetekuje nové útoky, je taktiež možné, že zlyhá detekcia už poznaného útoku, ak v ňom nastala zmena, ktorou sa vzor tohoto útoku zmenil.

Signatúra predstavuje množinu pravidiel opisujúcich vlastnosti prevádzky. Môže ísť o popis samotných paketov alebo popis vlastností celej prevádzky. V tejto metóde je kladený veľký dôraz na špecifikáciu jednotlivých pravidiel. Čím presnejšie je útok opísaný, tým sa zvyšuje šanca správneho označenia a znižuje šanca falošne pozitívnych označení.

Ak nastane situácia, že sa niektorá z uložených signatúr zhoduje s aktuálne prebiehajúcou prevádzkou, je označená za podozrivú. Pokiaľ sa ukáže, že išlo o útok, je vhodné previesť jeho analýzu, aby bolo možné navrhnúť pravidlá opisujúce tento útok. Tie budú následne pridané ako signatúra na porovnávanie.

- **IDS systémy** – IDS (Intrusion Detection System) je systém, ktorý monitoruje počítačovú sieť a vyhľadáva potenciálne bezpečnostné hrozby. Tento systém môže byť použitý aj na detekciu iných hrozieb ako DDoS útoky, a to napríklad útoky hrubou silou alebo skenovanie portov. Každá zachytená hrozba by mala byť oznámená administrátorovi alebo systému spravujúcemu bezpečnosť systému. Oznámenie by v sebe malo zahŕňať informácie o hrozbe, ako napríklad jej typ, rozsah a veľkosť.

  - **Zeek (v minulosti pod názvom Bro)** je open source nástroj na analýzu sieťovej prevádzky a detekciu potencionálnych hrozieb v reálnom čase. Interpretuje prevádzky, ktorú analyzuje, a vytvára kompaktné transakčné logovacie súbory a má plne prispôsobiteľný výstup, vhodný na manuálnu analýzu.
  - **Snort** je veľmi populárnym IDS zároveň IPS (Intrusion Prevention System) systémom vyvíjaným spoločnosťou Cisco. Dokáže vyhľadávať hrozby na základe signatúr aj na základe anomálií v reálnom čase, čím sa zväčšuje záber hrozieb, ktoré dokáže odhaliť.

- **Korelácia premenných premávky MIB** Táto metóda dokáže odhaliť štatistické nezrovnalosti špecifické pre TCP, UDP a ICMP pakety.

- **Spektrálna analýza** – techniky spadajúce do tejto kategórie vykonávajú spektrálnu analýzu na rozlíšenie legitímnej prevádzky od útočiacej. Napríklad v práci [4] je použitá výkonová spektrálna hustota paketov na detekovanie útočiacej prevádzky.

Detekcia anomálií

Líši sa od vyššie spomenutej detekcie signatúr tým, že nemá vopred zadanú sadu pravidiel, podľa ktorých by sa snažila detekovať útok. Detekcia anomálií sa pozerá na celkový profil prevádzky a hľadá odlišnosti od bežného profilu prevádzky. Vďaka tomu vie detekcia anomálií detekovať aj nový typ útoku alebo útok s pozmeneným vzorom.

Hlavnou nevýhodou tejto metódy je vysoká miera falošne pozitívnych označení. Keďže je detekcia anomálií citlivá na výkyvy od bežného profilu prevádzky, každá zmena je označená za útok. Takže aj legitímna prevádzka, ktorá sa nejakým štýlom vymyká bežnému

profilu prevádzky bude pravdepodobne detekovaná ako útok. Preto musí systém využívajúci túto metódu disponovať veľkým množstvom informácií o bežnej prevádzke. S tým súvisí aj nastavenie prahovej hodnoty označujúcej útok, čo predstavuje ďalšiu výzvu.

- **MULTOPS** z anglického Multi-Level Tree for Online Packet Statistics je heuristická metóda navrhnutá na detekciu volumetrických útokov. MULTOPS ukladá štatistiky o rozsahu prevádzky pre toky medzi dvomi koncovými stanicami a ukladá ich na základe IP adries. Útoky detekuje na základe nepomeru veľkosti tokov. V momente veľkého rozdielu medzi tokmi je táto situácia vyhodnotená ako útok. Dokáže rozlíšiť, či je konečná stanica obeťou alebo útočníkom. Má však dve hlavné nevýhody. Prvou je, že je kvôli ukladaniu dát o tokoch náchylná na útoky, pri ktorých môže dôjsť k vyčerpaniu pamäte. Druhou je, že predpokladá rovnomerné množstvo prichádzajúcej a odchádzajúcej prevádzky, čo však vo veľa prípadoch a hlavne pri využívaní multimediálnych služieb nemusí byť pravda.

- **DWARD** je detekčná metóda schopná obmedziť útok blízko zdroja. Funguje na princípe neustáleho monitorovania komunikácie medzi sieťou a zvyškom internetu. Periodicky porovnáva aktuálnu prevádzku s modelom normálnej prevádzky. Informácie o normálnej prevádzke zbiera počas fázy monitorovania. Na základe porovnávania odhalí anomálie v prevádzke a proporcionálne k ich veľkosti obmedzí ich tok. Avšak keďže metóda funguje blízko zdroja útoku, je potrebné, aby bola dostatočne rozšírená po celej sieti, aby spĺňala svoj účel.

Detekcia zdroja útoku

Úlohou detekčných techník je rozpoznať sieťový tok, ktorý predstavuje útok. Preto sú často zapájané blízko potencionálneho cieľa útoku. Avšak po detekcii, keď nastane zahadzovanie paketov na strane obete, nie je možné zaručiť, že nebudú zahodené žiadne legitímne pakety. Keby sa zahadzovanie presunulo bližšie k zdroju útoku, je väčšia šanca, že miera falošne pozitívnych označení bude nižšia. To zaručí hlavne to, že sa k obeti dostane viac legitímnej prevádzky nie len zo siete, z ktorej pochádza útok, ale aj z iných sietí, na ktoré nebol aplikovaný filter. Keďže sa snažíme určiť zdroj útoku, musíme poznať cestu medzi obeťou a útočníkom. Ide teda o hľadanie cesty na sieti, známe tiež ako IP traceback (IP trasovanie). Útočníci často zasielajú pakety s podvrhnutými IP adresami a inými položkami, preto ich nie je ľahké trasovať a ďalej len stručne opíšeme niekoľko metód, ktoré sa využívajú na detekciu zdroja útoku.

- **Probabilistic packet marking** (PPM) Táto technika pravdepodobnostne kóduje informácie do hlavičky paketu. Vďaka tomu je príjemca schopný rekonštruovať cestu, ktorú paket prešiel a teda bez ďalšej komunikácie, vytvárania inej prevádzky alebo inej pomoci, je možné určiť zdroj útoku vďaka rekonštrukcií cesty paketu. Vyžaduje avšak určitú výpočetnú kapacitu a preto pri útoku, do ktorého sú zapojené stovky zariadení nie je veľmi efektívna.

- **Deterministic packet marking** používa deterministické metódy na označovanie paketov, na vstupných rozhraniach. Oproti PPM je menej náročné na výkon. Podobne ako PPM nevytvára žiadnu ďalšiu prevádzku. Pomocou tejto metódy je možné trasovať tisícky útočiacich zariadení naraz.

- **ICMP traceback message** Táto technika zasiela ICMP správy smerovačom, vďaka čomu dokáže následne vytvoriť cestu paketu. Nevýhodou je však to, že táto technika vytvára prevádzku.

3.2.2 Reakcia

Po tom ako je útok detekovaný je potrebné vykonať akciu, ktorá dokáže útok potlačiť. Ako už bolo spomenuté skôr, je potrebné, aby bola reakcia okamžitá s čo najmenším oneskorením. V čase, kedy proti útoku nie je vykonaná žiadna akcia a útok tým pádom zneprístupní službu, je možné ho považovať za úspešný. Pri reakcii sa nejedná čisto o zakročenie proti útoku, môže taktiež rozšíriť poznatky o útoku dodatočnou analýzou dát nazbieraných pri detekcii. V tejto sekcii opíšeme najčastejšie techniky používané na mitigáciu DDoS útokov.

Filtrovanie a obmedzovanie rýchlosti

Ide o veľmi jednoduché, zato veľmi uplatniteľné techniky. Pokiaľ vďaka dátam získaným po detekcii útoku je možné s určitosťou povedať odkiaľ útok pochádza alebo sú známe znaky, podľa ktorých je možné útok dostatočne dobre odlíšiť od legitímnej prevádzky, je použité filtrovanie. Pokiaľ výsledok detekcie avšak nie je jednoznačný a hrozí vysoká miera falošne pozitívnych označení, bude použité obmedzovanie rýchlosti.

Reakcia založená na kapacite

Podstata DDoS útokov spočíva vo vyčerpaní zdrojov obete. Na to, aby boli tieto zdroje vyčerpané, sú taktiež potrebné určité zdroje. Tie útočník často nazbiera vďaka botnetu, vysvetlenom v kapitole 2.1. Každé zo zariadení v botnete môže posielať ľubovoľné množstvo paketov, do veľkosti jeho zdrojov. Reakcia založená na kapacite sa snaží riešiť tento problém, obmedzovaním kapacity paketov, ktorú môže užívateľ odoslať na koncový bod.

3.2.3 Tolerancia

Tolerancia je alternatívou k reakcii, pokiaľ sa vo fáze detekovania neprišlo na žiadne relevantné informácie, ktoré by sme mohli využiť v reakcii. Tolerancia nie je ideálnou metódou, ale pokiaľ zlyhá prevencia a detekcia tiež nesplní svoj účel, tak je to posledná možnosť, ako útok zmierniť a ponechať službu aspoň čiastočne dostupnú. Je na ňu treba len veľmi málo alebo žiadne informácie o útoku. Ďalej budú opísané často používané techniky pri tolerancii.

Kontrola preťaženia

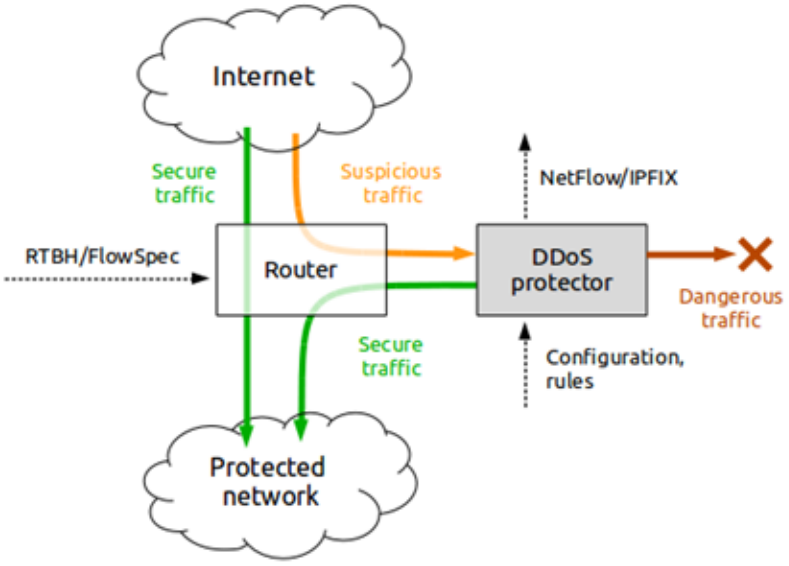
Aplikovaním mechanizmov kontroly preťaženia ako sú napríklad Re-feedback a NetFence je možné obmedziť DDoS útok.

Tolerancia zlyhania

Táto metóda zaisťuje vysokú dostupnosť systému. Jej základným princípom je replikovať a znásobovať zdroje systému.

3.3 DDoS Protector

DDoS Protector je aktívne sieťové zariadenie vyvíjané združením CESNET v rámci projektu Liberouter. Jeho úlohou je filtrovať rôzne typy DDoS útokov. Protector sa zameriava primárne na volumetrické útoky, ktoré fungujú na princípe zahltenia obete dostatočne veľkým množstvom dát. Skladá sa zo serveru a dedikovanej sieťovej karty s FPGA čipom. FPGA implementuje preposielanie a filtrovanie dát a server implementuje riadenie, ktoré nepretržite vyhodnocuje parametre sieťovej prevádzky a v prípade útoku povolí filtrovanie pomocou FPGA. Za normálnych okolností Protector nezahadzuje žiadne pakety. Iba keď je zaznamenaný DDoS útok, začne Protector fungovať ako filter. Jeho cieľom je zahadzovať DDoS pakety, aby sa znížilo množstvo prevádzky v sieti. V prípade DDoS útokov je najlepšie zasahovať čo najbližšie k zdroju, aby touto útočiacou prevádzkou nebola sieť zahltená. Nakoľko spracovať DDoS útok v malej sieti so zahltenou linkou je problém. Protector je teda najlepšie zapojiť priamo do infraštruktúry siete poskytovateľa. Protector dodržiava pravidlá stanovené administrátorom. Pravidlo pozostáva z podmienok, limitov (množstvo prevádzky na sieti, pre spustenie) a optimálnej veľkosti prevádzky na sieti. Podmienky špecifikujú, ktoré pakety odpovedajú pravidlu (napr. zdrojový port, dĺžka paketu apod.). Limity špecifikujú množstvo paketov za sekundu alebo bajtov za sekundu, ktoré musí byť prekročené na detekovanie DDoS útoku a následné spustenie filtrácie. Optimálna prevádzka vyjadruje požadované množstvo paketov za sekundu alebo bajtov za sekundu, na ktoré sa má prevádzka po spustení filtrácie obmedziť. Protector je zapojený v páre so smerovačom, ktorý do neho smeruje podozrivú prevádzku. Protektor preverí každý prijatý paket, aktualizuje štatistiku o použitých pravidlách a rozhodne či paket zahodiť alebo ho smerovať späť do smerovača.



Obr. 3.3: DDoS Protector. Prevzaté z [6]

Kapitola 4

Berkeley Packet Filter

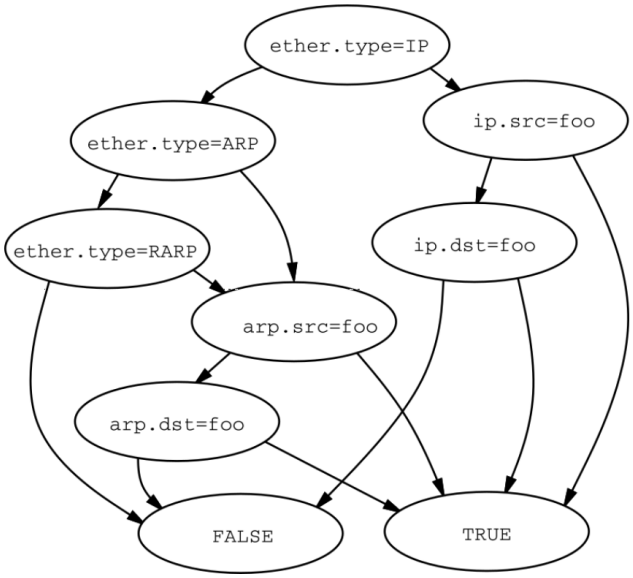
Ako bolo spomenuté v kapitole 3.3, DDoS Protector slúži na filtrovanie DDoS útokov. Cieľom tejto práce je navrhnúť algoritmus pre odvodenie pravidiel v DDoS útoku, ktoré by bolo následne možné aplikovať v DDoS Protectore. V tejto kapitole bude preto opísané fungovanie a formát BPF (Berkeley Packet Filter), nakoľko po konzultácii s vedúcim tejto práce bolo upresnené, že výsledné pravidlá majú byť práve vo formáte BPF, ktoré dokáže DDoS Protector použiť na filtrovanie.

The Berkeley Packet Filter bol pôvodne predstavený v roku 1992 prácou od Lawrence Berkeley Laboratory [14] ako systém pre filtrovanie prichádzajúcich paketov čo najskôr po ich príchode na cieľové zariadenie. Výstupom filtru je booleovská hodnota.

4.1 Model filtru

Berkeley Packet Filter používa graf toku riadenia (angl. Control flow graph, ďalej len CFG) namiesto stromovej štruktúry. CFG má oproti stromu výhodu vo výkone, nakoľko strom môže vyžadovať redundantne parsovať paket viackrát. CFG umožňuje parsovanej informácii, aby bola "vstavaná"do grafu toku a to takým spôsobom, že stav analýzy paketu je zapamätaný v grafe, pretože poznáme cestu, ktorou sme sa dostali k určitému uzlu.

Na obrázku 4.1 môžeme vidieť príklad, kedy funkcia filtra prijíma všetky pakety so zdrojovou alebo cieľovou adresou *foo*. V tomto scenári prehľadáva protokoly IP, ARP a RARP.



Obr. 4.1: CFG filtrovacia funkcia, ktorá implementuje filter "host foo". Prevzaté z [14]

4.2 Formát filtru

Formát BPF sa používa na filtráciu paketov v linuxe dodnes, ako sa píše v dokumentácii kernelu:

Linux Socket Filtering (LSF) je odvodený z BPF. Napriek tomu, že existujú určité rozdiely medzi filtrovaním kernelu v BSD a Linuxe, keď hovoríme o BPF alebo LSF v kontexte Linuxu, máme na mysli ten istý mechanizmus filtrovania v kerneli Linuxu. BPF umožňuje programu užívateľského rozhrania pripojiť filter na akýkoľvek soket a umožniť alebo zakázať určitým typom údajov prechádzať soketom. LSF sleduje presne tú istú štruktúru kódu filtra ako BPF v BSD, takže odkaz na manuálnu stránku BSD bpf.4 je pri vytváraní filtrov veľmi užitočný.[1]

Následne budú stručne vysvetlené pravidlá pre tvorbu vzorov v BPF podľa dokumentácie zo stránky tcpdump [2]. Výraz filtru pozostáva z primitív, každé primitívum pozostáva z jedného alebo viac kvalifikátorov a id. Id je názov alebo číselná hodnota. Väčšinou sa používa práve číselná hodnota, napríklad dĺžka paketu. Kvalifikátory môžu byť troch rôznych typov a spôsoby ich správneho kombinovania sú vysvetlené v dokumentácii. Typy kvalifikátorov:

- **Protokol** - kvalifikátor obmedzí toto primitívum na konkrétny protokol. Možno vyberať spomedzi: *ether, fddi, tr, ip, ip6, arp, rarp, decnet, tcp* a *udp*. Pokiaľ nie je

[1] Preložené z originálu: *Linux Socket Filtering (LSF) is derived from the Berkeley Packet Filter. Though there are some distinct differences between the BSD and Linux Kernel filtering, but when we speak of BPF or LSF in Linux context, we mean the very same mechanism of filtering in the Linux kernel. BPF allows a user-space program to attach a filter onto any socket and allow or disallow certain types of data to come through the socket. LSF follows exactly the same filter code structure as BSD's BPF, so referring to the BSD bpf.4 manpage is very helpful in creating filters.* Dostupné na: https://www.kernel.org/doc/Documentation/networking/filter.txt

[2] https://web.archive.org/web/20190711165251/https://alumni.cs.ucr.edu/~marios/ethereal-tcpdump.pdf

definovaný žiadny protokol, budú všetky protokoly uvážené konzistentné s hľadaným typom. Napríklad *port 80* znamená to isté ako *(tcp or udp) port 80*.

- **Smer** - tento kvalifikátor špecifikuje smer primitíva. Na výber sú: *src, dst, src or dst* a *src and dst*. Pokiaľ nie je zadaný smer, uvažuje sa *src or dst*. Napríklad *ip host 192.168.50.1* bude vyhľadávať všetky pakety s IPv4 adresou 192.168.50.1.
- **Typ** - vyjadruje na čo sa id odkazuje. Možné typy sú: *host, net, port* a *portrange*. Ak nie je špecifikovaný typ, uvažuje sa *host*.

Primitíva sa môžu navzájom kombinovať a spolu vytvárajú výraz. Kombinovať sa môžu pomocou:

- Negácie ('!' alebo 'not').
- Zreťazenia ('&&' alebo 'and').
- Alternatívy ('||' alebo 'or').
- Zátvorkovaním skupín primitív a operátorov.

V dokumentácii môžeme dohľadať množstvo ďalších kľúčových slov, ktoré je možné využiť pri zostavovaní primitív. Nie je avšak možné popísať každú položku v hlavičkách protokolov tretej vrstvy a vyšších, len pomocou kľúčových slov. Taktiež nie je možné všetkým položkám zadávať rozsah, pri niektorých je možná iba konkrétna hodnota. Primitíva je možné tvoriť ešte jedným spôsobom a to pomocou vzorca *expr relop expr*, kde:

- *relop* je jedno z >, <, >=, <=, =, !=
- *expr* je aritmetický výraz zložený z celočíselných konštánt (vyjadrený v syntaxe jazyka C), binárnych operátorov: +, -, *, /, &, |, «, », operátora dĺžky a vzorca na prístup k dátam paketov. Vzorec na prístup k dátam paketov je nasledovný:
  - *proto [expr : size]*. Kde *proto* je jedno z *ether, fddi, tr, ppp, slip, link, ip, arp, rarp, tcp, udp, icmp, ip6* alebo *radio* a vyjadruje hlavičku do ktorej chceme nazrieť pomocou indexu. Podľa dokumentácie je žiaľ možné pristúpiť k udp, tcp a iným vyšším protokolom len v ipv4 pakete. *expr* vyjadruje relatívnu polohu v hlavičke, kam chceme nazrieť a *size* je počet bajtov, ktoré chceme porovnávať a táto hodnota je voliteľná, predvolená je hodnota 1 a maximálnou je hodnota 4.

Kapitola 5

Strojové učenie a rozhodovací strom

V tejto kapitole bude vysvetlená podstata strojového učenia, jeho základné delenia a bude podrobnejšie opísaná metóda rozhodovacieho stromu, keďže práve s touto metódou budeme pracovať v kapitole 6.

Strojové učenie je podmnožinou počítačovej vedy, je považované za súčasť vednej oblasti umelá inteligencia. Strojové učenie využíva algoritmy, ktoré sa dokážu učiť a vytvárať predikcie na základe dát. Tieto algoritmy vytvárajú klasifikačný model na základe dát, ktorý je ďalej používaný na predikciu, namiesto toho, aby predikciu vykonávali na základe pevne daného algoritmu. Najčastejšie využívanými druhmi v strojovom učení sú učenie s učiteľom, učenie bez učiteľa, čiastočné učenie s učiteľom a posilované učenie.

5.1 Kategórie strojového učenia

Odborná literatúra nie je úplne jednotná v delení strojového učenia do kategórií, ďalej teda bude uvedené delenie podľa [3]. Informácie boli čerpané taktiež z [15].

- **Učenie s učiteľom (Supervised Learning)** - v tomto type učenia je algoritmu dodaná vzorka vstupných dát a im odpovedajúce požadované výstupné dáta, tieto dáta sa označujú ako trénovacie. Cieľom tohto typu učenia je nájsť obecné pravidlo, na základe ktorého by bolo možné predloženým vstupným dátam priradiť výstupné dáta. Kvalita výsledného klasifikačného modelu často závisí od kvality, veľkosti a rôznorodosti dát na trénovanie.
- **Učenie bez učiteľa (Unsupervised Learning)** - tomuto typu učenia sú predané iba vstupné dáta. Algoritmy spadajúce do tejto kategórie dokážu v týchto dátach nájsť určitú štruktúru. Tieto algoritmy hľadajú v trénovacích dátach podobnosti, rozdielnosti a anomálie. Hľadajú spôsoby na dosiahnutie cieľa. Typickými úlohami pre tento typ učenia sú zhlukovanie, detekcia anomálií, odhad rozloženia pravdepodobnosti a iné.
- **Čiastočné učenie s učiteľom (Semi-supervised learning)** kombinuje učenie s učiteľom a učenie bez učiteľa. Trénovacie dáta sa skladajú z tých, ktorým je priradená výstupná hodnota ale taktiež z neohodnotených dát.

- **Posilované učenie (Reinforcement learning)** - algoritmus je umiestnený do dynamického, často komplexného prostredia, v ktorom musí splniť určitý cieľ. Vopred mu nie je známe, aké budú výsledky jeho rozhodnutí. Za akcie ktoré vykonáva mu je dodaná pozitívna alebo negatívna spätná väzba, vďaka tomu dokáže smerovať sériu rozhodnutí smerom k požadovanému výsledku. Algoritmus tu často čelí prostrediu podobnému hre. Na začiatku robí algoritmus úplne náhodné rozhodnutia a často sa mu nakoniec podarí vyvinúť lepšiu taktiku k dosiahnutiu cieľa, než je naša ľudská. Využíva sa napríklad v hernom priemysle.

5.2 Rozhodovací strom

Informácie v tejto sekcií boli čerpané z [10]. Rozhodovacie stromy patria medzi najpopulárnejšie algoritmy učenia, vďaka ich zrozumiteľnosti a jednoduchosti. Rozhodovací strom sa zaraďuje medzi učenie s učiteľom. Používa sa na klasifikáciu a regresiu. Jeho cieľom je vytvoriť klasifikačný model stromového typu z ohodnotených trénovacích dát naučením sa jednoduchých rozhodovacích pravidiel.

Najpopulárnejšími algoritmami rozhodovacieho stromu sú ID3(Iterative Dichotomiser 3), C4.5, a CART. Tieto algoritmy fungujú na rovnakom princípe a rozdiely medzi nimi nie sú veľké. Všetky sa odvíjajú od algoritmu ID3.

Algoritmus ID3

ID3 je najznámejším klasifikačným algoritmom pre generovanie rozhodovacích stromov metódou zhora nadol. Jeho výsledkom je rozhodovací strom, ktorý predstavuje klasifikačný model a používa sa na predikciu. Trénovacie dáta sa skladajú z príkladov, kde každý príklad obsahuje konečné množstvo atribútov a je zaradený do jednej z výsledných tried. Nelistové uzly stromu slúžia na rozhodovanie, predstavujú jeden z atribútov a každá vetva, ktorá vedie z tohoto uzlu predstavuje jednu možnú hodnotu tohto atribútu. Listové uzly predstavujú výslednú hodnotu predikcie.

1. Koreňový uzol reprezentuje všetky trénovacie príklady
2. Pokiaľ tieto príklady patria do rovnakej triedy, tento uzol bude označený tou triedou a stane sa listovým uzlom.
3. Inak bude pomocou metódy informačného zisku vybraný atribút, ktorý najlepšie rozdelí príklady do jednotlivých tried. Zvolený atribút bude slúžiť buď na testovanie alebo sa stane triedou. Všetky hodnoty, ktoré nadobúdajú atribúty, musia byť diskrétne, ak sa nájde spojitá veličina, je potrebné ju diskretizovať.
4. Z príslušného uzla bude vychádzať počet vetiev odpovedajúci počtu hodnôt atribútu, ktoré tento uzol nadobúda a príklady budú pridelené do jednotlivých vetiev.
5. Rovnakým spôsobom spôsobom pokračuje po každom uzle. V prípade, že sa atribút použije na testovanie, nesmie byť znovu použitý pre rozhodovanie v tejto časti stromu.
6. Tento postup skončí v momente, kedy je splnená jedna z nasledujúcich podmienok:
   (a) Všetky príklady daného uzla patria do rovnakej triedy.
   (b) Všetky atribúty, podľa ktorých by mohlo dôjsť k ďalšiemu deleniu už boli použité po ceste do tohto uzlu. Uzol sa označí za listový a za jeho triedu sa vyberie tá, ktorá á najpočetnejšej zastúpenie v príkladoch tohto uzla.
   (c) Pokiaľ už nie sú žiadne ďalšie príklady patriace k tejto vetve. Trieda listového uzla bude vybratá z najpočetnejšie zastúpenej triedy v príkladoch.

Výber testovacieho atribútu

V každom uzle stromu, ktorý nie je listový a nie je ani koreňom musí byť atribút, na základe ktorého sa bude strom ďalej vetviť. Tento atribút je potrené vybrať správne, aby čo najlepšie dokázal rozdeliť príklady. Štatistická metóda informačný zisk je vhodným meradlom vhodnosti atribútu. Meria totiž, ako dobre budú atribúty rozdelené. Na opísanie informačného zisku je potrebné najprv definovať entropiu. Ak pravidlá $x_1, x_2, ..., x_n$ sú možné s pravdepodobnosťami $p(x_1), p(x_2), ..., p(x_n)$ a tieto vytvárajú úplný súbor pravdepodobností v uzle S:

$$\sum_{j=1}^{n} p(x_j) = 1$$

potom entropiu možno vyjadriť ako:

$$H(S) = -\sum_{j=1}^{n} p(T_j) log_2 p(T_j)$$

kde $T_1, T_2, ..., T_n$ predstavujú jednotlivé triedy. Z tohto možno odvodiť nasledujúce:

- Ak uzol obsahuje len príklady jednej triedy, entropia je nulová.
- Všetky listové uzly majú entropiu nula.
- Každý nelistový uzol ma nenulovú entropiu.
- Pokiaľ sa počet príkladov v uzle dá rovnomerne rozdeliť do tried, tak entropia bude mať hodnotu 1.

Aby bol strom čo najmenší, je potrebné aby entropia klesala čo najrýchlejšie. Celková entropia v uzle S s použitím atribútu A a hodnôt, ktoré nadobúda $a_0, a_1, ...a_n$ s sa dá vyjadriť ako:

$$H(S, A) = -\sum_{j=1}^{n} p(x_j) H(S_j)$$

kde $p(x_j)$ predstavuje pravdepodobnosť výskytu hodnoty atribútu.
Informácia získaná použitím atribútu A v uzle S je informačný zisk, s výpočtom:

$$I(A) = H(S) - H(S, A)$$

V každom uzle je treba vypočítať informačný zisk pre každý nepoužitý atribút. Vybratým bude ten atribút, ktorý má najväčší informačný zisk.

Na stránke sa uvádza, že [7] scikit-learn využíva optimalizovanú verziu algoritmu CART. Avšak, scikit-learn implementácia zatiaľ nepodporuje kategorické premenné. Preto budú ešte uvedené rozdiely medzi jednotlivými algoritmami a vysvetlený gini index. Entropia a gini index vyjadrujú tú istú vec, nečistotu.

Vlastnosť	ID3	C4.5	CART
Typ dát	Diskrétne hodnoty	Diskrétne a spojité hodnoty	Spojité a nominálne atribúty
Posilňovanie	Nepodporuje	Nepodporuje	Podporuje
Orezávanie	Nie	Orezávanie pred	Orezávanie po
Chýbajúce hodnoty	Nezvládne	Zvládne	Zvládne
Používaný vzorec	Informačná entropia a informačný zisk	Pomer rozdelenia a zisku	Gini index

Tabuľka 5.1: Rozdiely medzi algoritmami rozhodovacieho stromu, prevzaté z [1].

Gini index

Gini index funguje podobne ako entropia, jeho výpočet je však strojovo rýchlejší, nakoľko nepoužíva logaritmus. Gini index nemá rozsah <0;1>, tak ako entropia ale <0;0,5>. Gini index má nasledujúci vzorec:

$$G(A) = 1 - \sum_{1}^{j} p_j^2$$

kde p_j predstavuje pravdepodobnosť výskytu j-tej triedy v atribúte A. Výpočet informačného zisku je nahradený váženým priemerom Gini indexov pre jednotlivé atribúty a vybraný bude ten s najmenšou hodnotou.

Kapitola 6

Návrh a implementácia mitigácie DDoS útokov

Táto kapitola opisuje návrh a implementáciu algoritmu, ktorý som si zvolil na tvorbu pravidiel pre mitigáciu DDoS útokov. Cieľom tejto práce je navrhnúť algoritmus, ktorý by bol schopný nájsť vzor v útočiacej prevádzke, dostatočne špecifický na to, aby bolo možné jeho aplikovaním zmierniť aktuálne prebiehajúci útok, ale zároveň neobmedziť legitímnych používateľov. Mnou navrhovaný algoritmus je koncipovaný ako reakcia na útok s dodatočnou analýzou dát. Úlohou mnou navrhnutého algoritmu nie je útok detekovať, nejde teda o detekciu. Algoritmus očakáva, že bude spustený detekčným nástrojom v momente kedy prebieha útok a budú mu predané potrebné dáta. Po konzultácií s vedúcim tejto práce bolo upresnené, že výsledné pravidlá majú byť vo formáte berkeley packet filter (BPF), ktorý sa používa na filtráciu paketov v Linuxe. Štruktúra BPF a všetky poznatky potrené pre jeho použitie v tejto práci boli opísané v kapitole 4. Najprv bude opísaný výber metódy a vysvetlené dôvody výberu tejto metódy. Ďalej bude opísaný návrh fungovania modelu a jeho následná implementácia. Pri implementácií budú vysvetlené jednotlivé technológie, ktoré budú použité. Pri návrhu metódy sa dával dôraz na to, aby výsledný model nevykazoval vysokú mieru falošne pozitívnych označení, pretože toto by útočníkovi napomáhalo k dosiahnutiu jeho cieľa a optimalizácie, aby výpočet netrval príliš dlho a bolo možné proti útoku zasiahnuť čo najskôr.

6.1 Výber metódy

Hlavné tri predpoklady, na ktorých je postavená táto práca sú:

- **Prvým predpokladom** je, že sa aktuálne prebiehajúci útok odlišuje určitými znakmi od od legitímnej prevádzky. Pokiaľ teda porovnáme prevádzku, o ktorej vieme, že obsahuje iba legitímne dáta a prevádzku, ktorá v určitej miere predstavuje útok, musia sa navzájom líčiť.
- **Druhým predpokladom** je, že jednotlivé pakety v útočiacej prevádzke vykazujú do určitej miery vzájomnú podobnosť. Tou môže byť napríklad určitá dĺžka paketu. Tento predpoklad sa ukázal ako pravdivý, pri vytváraní datasetov použitých na experimenty a testovanie z voľne dostupných vzoriek DDoS útokov.
- **Tretím predpokladom** je, že v čase útoku väčšina prevádzky na sieti, patrí práve útoku. Predpokladáme teda, že pokiaľ zachytíme prevádzku v momente, kedy žiadny detekčný mechanizmus nehlási útok, takmer celá táto prevádzka sa dá považovať za

legitímnu a môže obsahovať iba veľmi malú časť útoku, pravdepodobne nie volumetrického. Naopak, v momente kedy je detekovaný útok predpokladáme, že väčšina zachytenej prevádzky bude patriť práve tomuto útoku. To vyplýva aj z faktu, že infraštruktúra siete a konečné zariadenia sú budované takým spôsobom, aby v čase bežnej prevádzky bolo množstvo prevádzky na sieti hlboko pod hranicou toho, čo dokážu zniesť. A že útočník keď si chce byť istý, aby vyčerpal zdroje tejto siete alebo koncového bodu, tak musí zostrojiť útok, ktorý je schopný sám tieto zdroje vyčerpať. Pretože ak by napríklad dokázal zahltiť iba polovičnú šírku pásma, jeho cieľ by nebol splnený, nakoľko je služba väčšine užívateľov stále dostupná.

Za stratégiu pre tvorbu mitigačných pravidiel som zvolil strojové učenie, konkrétne metódu rozhodovacích stromov. Toto rozhodnutie bolo ovplyvnené najmä nasledujúcimi faktormi:

- Rozhodovací strom má jednoduchú štruktúru, ktorú je možné previesť do podoby pravidiel ako je napríklad BPF. Kde je možné jednotlivé časti pravidla navzájom spájať logickými výrazmi. Model ktorý vznikne po natrénovaní je teda možné previesť do tohto formátu. Pri niektorých iných metódach strojového učenia ako napríklad neurónové siete by toto nebolo možné. Tieto pravidlá je možné uplatniť v už existujúcich zariadeniach určených na mitigáciu.
- Štruktúra rozhodovacieho stromu je ľahko čitateľná pre ľudí. Administrátor, ktorý disponuje informáciami, ktoré algoritmus nemá, môže vygenerované pravidlo ľubovoľne upravovať a rozširovať ho. Prípadne je možné toto pravidlo zlúčiť s pravidlom rovnakého formátu vygenerovaného iným nástrojom.
- Rozhodovací strom patrí medzi algoritmy, v ktorých nie je potrebné škálovať dáta(takzvaný data scaling). S dátami je teda možné pracovať v ich originálnych hodnotách, bez nutnosti ich upravovania pred a po tvorbe modelu. Vďaka tomu je možné predísť prípadným nepresnostiam vznikajúcim pri spätnom prevode a strane výpočetných zdrojov, ktoré súvisia so škálovaním dát.
- Podľa [2] algoritmus rozhodovacieho stromu výrazne nezaostáva za ostatnými dostupnými algoritmami, čo sa týka presnosti, aj napriek jeho jednoduchosti.

6.2 Tvorba modelu

Nakoľko sa jedná o rozhodovací strom ide o učenie s učiteľom. Algoritmu je teda potrebné dodať ohodnotenú dátovú sadu. Táto dátová sada sa bude skladať z dvoch častí:

1. Prevádzka zachytená v čase, kedy nebol detekovaný žiadny útok a je teda nepravdepodobné, že by sa v nej nachádzala nejaká útočiaca prevádzka. Celá táto sada bude označená ako legitímna prevádzka. Je dôležité, aby táto sada pravdivo reprezentovala legitímnu prevádzku, preto navrhujem dva kroky, ktoré môžu zlepšiť jej výpovednú hodnotu.
   (a) Prvým je mať pripravené rôzne sady odpovedajúce konkrétnemu dňu v týždni a hodine dňa, keďže prevádzka na sieti sa v jednotlivé dni môže líšiť. Príkladom môže byť online komunikácia so službami, hovory v práci alebo škole, čo prebieha prevažne v pracovné dni, pravidelné online vysielania a iné.
   (b) Druhým je zabezpečiť, aby boli v každej sade dostatočne zastúpené jednotlivé protokoly. Pri záchyte je potrebné odchytiť vzorku dát, ktorá je väčšia než tá,

ktorú plánujeme používať a pri tvorbe výslednej dbať na to, aby sa do nej dostalo dostatočné množstvo paketov s jednotlivými protokolmi. Tento krok je dôležitý hlavne pre málo používané protokoly, ktoré útočníci využívajú na útočenie, nakoľko nemusíme mať dosť dát na to, aby sme správne odlíšili útok.

2. Prevádzka zachytená v čase útoku. Všetká prevádzka zachytená v čase útoku bude označená ako útočiaca prevádzka. Je teda zrejmé, že sa v nej budú nachádzať časti legitímnej prevádzky. No ako bolo spomenuté skôr v kapitole 6.1, predpokladá sa, že väčšina tejto prevádzky bude útočiaca. Taktiež ako bolo spomenuté v spomínanej kapitole, predpokladáme, že útočiaca prevádzka vykazuje určitú mieru podobnosti, vďaka čomu vo výslednej stromovej štruktúre budú listy, do ktorých bude patriť väčšina útočiacej prevádzky, oveľa objemnejšie a bude ich teda možné odlíšiť od falošne pozitívnych označení.

Tieto sady budú dodané vo forme PCAP súborov a na ich spracovanie bude použitá knižnica dpkt[1]. Použité budú informácie z hlavičiek sieťovej a transportnej vrstvy, no algoritmus je možné použiť aj na ďalšie vrstvy. V tabuľke 6.1 je prehľad položiek z hlavičiek IPv4, UDP a TCP protokolu, ktoré budú spracované a použité na strojové učenie. Protokoly UDP a TCP boli zvolené na základe faktu, že sú najpopulárnejšou voľbou útočníkov. Zdrojový kód výsledného programu je prispôsobený možnosti pridávania ďalších protokolov nezávisle od vrstvy, na ktorej sa nachádzajú. Podľa dokumentácie BPF [14] nie je prístup k jednotlivým položkám IPv6 hlavičky implementovaný, preto nebudem tento protokol brať do úvahy.

Protokol	**Položky**
IPv4	Type of Service, Total Lengtht, Identification, Dont fragment, More Fragment, Fragment Offset, Time to Live, Header Checksum, Source Address
TCP	Source Port, Destination Port, Sequence Number, Acknowledgment Number, Data offset, Flags, Window, Checksum, Urgent Pointer
UDP	Source Port, Destination Port, Length, Checksum

Tabuľka 6.1: Prehľad analyzovaných položiek

Na trénovanie modelu bude použitá knižnica scikit-learn[2], ktorá ponúka množstvo algoritmov strojového učenia a taktiež možnosti ich úpravy skrze parametre funkcií týchto algoritmov. Konkrétne použijeme DecisionTreeClassifier[3](DTC), ktorý implementuje rozhodovací strom. Pri prvotných testoch bez použitia akýchkoľvek parametrov tohto klasifikátora nastali situácie, kedy bolo množstvo výsledných listov stromu v stovkách a výsledné pravidlo obsahovalo viac ako milión znakov. To je prvým dôvodom, prečo musíme rast stromu obmedziť. Druhým je, že veľký výsledný model znamená aj dlhé výsledné pravidlo. Toto pravidlo sa bude používať pri filtrovaní všetkých paketov, čo znamená, že jeho zložitosť musí byť v prijateľnej miere. Tretím dôvodom na obmedzenie rastu je fakt, že pretrénovaný model môže viesť k horším výsledkom. Toto sa taktiež potvrdilo pri prvotných testoch a dôvodom tohto javu je, že prevádzka označená ako útočiaca obsahuje aj malý podiel legitímnej prevádzky. Avšak, keďže útočiaca prevádzka vykazuje určitý vzor, tieto pakety skončia v niekoľkých veľkých listoch. Zvyšné pakety, ktoré sú v skutočnosti legitímne, majú priestor

[1] https://dpkt.readthedocs.io/en/latest/
[2] https://scikit-learn.org/stable/
[3] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

na to, aby boli z uzlov, v ktorých sú spoločne s legitímnymi, ďalej triedené, až sa hodnota listu, v ktorom skončia, zmení na útok, aj keď to nie je pravda. Preto zamedzenie rastu stromu môže pomôcť k zmenšeniu miery falošne pozitívnych označení tým, že uzly aktuálne označené ako legitímne nedostanú ďalšiu šancu sa deliť. Týmto problémom sa ďalej zaoberá nasledujúca sekcia.

V dokumentácii DTC[4] sa nachádzajú štyri parametre, ktorými je možné v našom prípade obmedziť rast stromu:

1. **min_samples_leaf : int or float, default=1** - Minimálne množstvo vzoriek, ktoré musí byť v lisotvom uzle.
2. **max_depth : int, default=Non** - Maximálna hĺbka stromu.
3. **min_samples_split : int or float, default=2** - Minimálne množstvo vzoriek potrebné na rozdelenie uzla.
4. **max_leaf_nodes : int, default=None** - Maximálne množstvo listov stromu.

Týmto parametrom sa budem viac venovať v kapitole 7 kde sa pokúsim na základe experimentov zistiť, ktorý z týchto parametrov bude mať najlepšie výsledky.

Pre trénovanie som zvolil postup, v ktorom sa model trénuje z kombinácie IP hlavičky a jedného ďalšieho protokolu. Nakoľko má tento prístup väčšiu vyjadrovaciu silu, ako trénovať model nad jednotlivými protokolmi, keďže niektoré protokoly majú len veľmi málo položiek v hlavičke. Je intuitívne, že napríklad na základe kombinácie dĺžky paketu, ip adresy a cieľového portu bude model presnejší, ako keby sa trénoval samostatne nad každým protokolom a následne by boli možnosti zreťazené. Keby sa model trénoval raz, bolo by potrebné, aby atribúty modelu boli všetky položky z tabuľky 6.1.V tomto prípade by to avšak znamenalo, že vždy budú prázdne buď položky UDP alebo TCP hlavičky a pri implementácii ďalších protokolov by sa počet chýbajúcich atribútov zvyšoval. Aby bolo možné sa tomuto problému vyhnúť, model bude vykonávať trénovanie vždy osobitne nad každou kombináciou IP a jednej ďalšej hlavičky (aktuálne UDP a TCP). Vygenerované pravidlá budú následne spojené do jedného.

6.3 Tvorba pravidiel

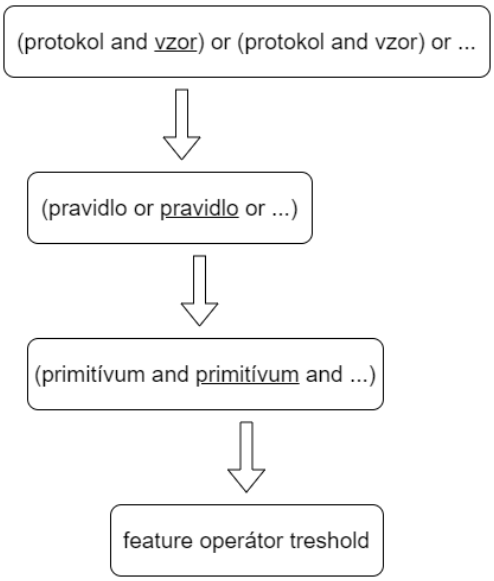
Okrem pravidiel BPF, je program schopný generovať pravidlá vo formáte wireshark-filter[5] používanom v programe wireshark na filtráciu po záchyte. Toto rozšírenie bolo pridané pre výskumné účely. V tabuľkách 6.2, 6.3 a 6.4 sú uvedené hodnoty položiek v hlavičkách protokolov vo formáte BPF a wireshark-filter. V okamihu keď je model vo svojej finálnej podobe, bude prevedený na formát BPF. DTC model obsahuje štruktúru *tree_*[6], v ktorej sa nachádza výsledná stromová štruktúra. Položky štruktúry, ktoré využívam v mojom programe sú nasledovné (pri každej ide o pole):

- **children_left[i]** - id ľavého potomka uzla alebo -1 ak sa jedná o list
- **children_right[i]** - id pravého potomka uzla i alebo -1 ak sa jedná o list
- **feature[i]** - atribút použitý na delenie uzla i
- **treshold[i]** - prahová hodnota v uzle i
- **value[i]** - pole, vyjadruje počet vzorkov spadajúcich do jednotlivých tried v uzle i

[4] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
[5] https://www.wireshark.org/docs/man-pages/wireshark-filter.html
[6] https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html

- **impurity[i]** - nečistota v uzle i

Nasleduje krátky opis zdrojového kódu programu. Funkcia DFS vykonáva prehľadávanie do hĺbky. Na začiatku vytvorí pole feature_name_bpf, ktoré je kópiou poľa feature avšak namiesto číselných hodnôt obsahuje formát týchto atribútov v BPF. Vo funkcií DFS sa vytvára a napĺňa slovník child_parent, vďaka ktorému poznáme rodiča každého uzlu. V momente kedy DFC natrafí na list, ktorý je zároveň označený ako útočný, zavolá funkciu printPathBPF, ktorá predchádza strom od tohoto uzla smerom ku koreňu. Toto je možné vďaka slovníku child_parent. Pre každý nelistový uzol vytvára primitívum, pozostávajúce z feature (atribútu), operátora a treshold (číselná hodnota). Hodnota operátora je v prípade, že ide o ľavého potomka <= inak >. Tieto primitíva sú spojené do jedného pravidla pomocou and. Všetky takto vytvorené pravidlá sú nakoniec spojené do jedného vzoru pomocou or. Funkcia printPathWireshark vykonáva rovnakú funkciu, ale pre formát wireshark-filter. Tento celý postup prebehne pre každý model. Z každého modelu máme teda jeden vzor výsledného filtra. Ku každému z týchto vzorov je následne pridaný protokol, pre ktorý sa má aplikovať pomocou and a následné sú tieto vzory spojené dokopy pomocou or. Pre lepšie znázornenie vyššie spomenutého slúži obrázok 6.1.



Obr. 6.1: Schéma tvorby pravidiel

	BPF	wireshark
Type of Service	ip[1:1]	ip.tos
Total Length	ip[2:2]	ip.len
Identification	ip[4:2]	ip.id
Dont fragment	ip[6:1] & 0x40	ip.flags.df
More Fragment	ip[6:1] & 0x20	ip.flags.mf
Fragment Offset	ip[6:2] & 0x1fff	ip.frag_offset
Time to Live	ip[8:1]	ip.ttl
Header Checksum	ip[10:2]	ip.checksum
Source Address	ip[12:4]	ip.src

Tabuľka 6.2: Formát položiek v IP hlavičke

	BPF	wireshark
Source Port	tcp[0:2]	tcp.srcport
Destination Port	tcp[2:2]	tcp.dstport
Sequence Number	tcp[4:4]	tcp.seq
Acknowledgment Number	tcp[8:4]	tcp.ack_raw
Data offset	tcp[12:1] & 0xF0	tcp.hdr_len
Flags	tcp[13:1]	tcp.flags
Window	tcp[14:2]	tcp.window_size_value
Checksum	tcp[16:2]	tcp.checksum
Urgent Pointer	tcp[18:2]	tcp.urgent_pointer

Tabuľka 6.3: Formát položiek v TCP hlavičke

	BPF	wireshark
Source Port	udp[0:2]	udp.srcport
Destination Port	udp[2:2]	udp.dstport
Length	udp[4:2]	udp.length
Checksum	udp[6:2]	udp.checksum

Tabuľka 6.4: Formát položiek v UDP hlavičke

6.4 Vylepšenie modelu

Dôraz pri vyhodnocovaní funkčnosti mitigačného pravidla je v tejto práci kladený na malú mieru falošne pozitívnych označení (FP). Je teda žiadúce, aby aj pre prípadnú stratu skutočne pozitívnych (TP) sa nám podarilo dostať FP na minimálne možné hodnoty. Najpopulárnejšou metódou na orezávanie stromu po vytvorení modelu je metóda Cost complexity pruning[7]. Táto metóda opakovane vytvára model vždy s inou hodnotou ccp_alphan, a následne je zvolený strom s najlepšími výsledkami. Toto je avšak časovo veľmi náročná operácia a vzhľadom na to, že sa snažíme o algoritmus, ktorý by bol schopný zasiahnuť voči útokom v čo najkratšom čase, nie je pre nás vhodná. Taktiež pracuje s predpokladom, že dátová sada je správne ohodnotená, čo ale v našom prípade neplatí, keďže niektorá prevádzka

[7] https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html#sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py

označená ako útočiaca je v skutočnosti legitímna. Z tohoto dôvodu som navrhol vylepšenie na úpravu výsledného stromu. Vylepšenie je v zdrojovom kóde programu implementované vo funkcií edit_tree a vykonáva sa ešte pred prevodom na BPF.

Pri generovaní výsledného filtra, ako je opísané v 6.3, nás zaujímajú iba listové uzly označené ako útok. Pre každý z týchto listov bude vygenerované pravidlo od tohto listu ku koreňu. Pri predpokladoch, že sa útočné pakety na seba navzájom podobajú, a že sa zároveň líšia od legitímnej prevádzky, môžeme usúdiť nasledovné. Väčšina útočiacich paketov skončí v listových uzloch spolu. Všetky listové uzly označené ako útočiace by sa teda dali rozdeliť nasledovne: prvá skupina budú tie, ktoré budú obsahovať veľa útočiacich paketov a budú vykazovať nízku mieru nečistoty; druhá skupina budú tie, v ktorých pôjde pravdepodobne o falošne pozitívne označenia a budú mať vyššiu mieru nečistoty. To znamená, že pri správne zvolenej prahovej hodnote nečistoty je možné znížiť mieru falošne pozitívnych označení tým, že listy prekračujúce túto hodnotu budú označené ako legitímne.
Presne to sa deje vo funkcii edit_tree. Funkcia prechádza listy a na základe tejto hodnoty mení listy označené ako útok na legitímne.

Kapitola 7

Experimenty a testovanie

V tejto kapitole sa venujem experimentom vykonaným na programe. Cieľom experimentov bolo zistiť, nakoľko dokáže program rozpoznať útočiacu prevádzku od legitímnej v dátach čo najviac podobným reálnej prevádzke a aké sú správne nastavenia na obmedzenie rastu stromu a jeho upravovanie. Cieľom teda nie je iba označiť útok, ale snažiť sa minimalizovať obmedzovanie legitímnych používateľov. V kapitole 7.2 bude opísaný postup, ktorý som zvolil na vykonanie experimentov a ako budú vyhodnocované. Kapitola 7.1 opisuje použité dátové sady a ich úpravu. Za ňou nasledujú výsledky jednotlivých experimentov a následne ich zhrnutie.

7.1 Datasety

Pre správne vyhodnotenie bolo potrebné najprv získať dáta čo najviac odrážajúce skutočnosť. Legitímna prevádzka bola zachytená pri 100 Gbps na chrbtovej sieti medzi Rakúskou a Českou výskumnou a vzdelávacou sieťou (ACONET resp. CESNET). Záchyt obsahuje 1 227 000 paketov s celkovou veľkosťou 1,1GB. Z tejto sady bolo použité iba množstvo potrebné pre účely nasledujúcich testov. Jedná sa o dataset LEGIT a obsahuje približne 76 800 paketov z čoho je 40% TCP. Pakety boli vybrané tak, aby mal dataset dostatočnú rôznorodosť. Ďalej bol z tohto záchytu vytvorený aj dataset určený na miešanie. Ide o dataset MIX s približne 57 500 paketmi z čoho je 52% TCP. Viac o datasete MIX v kapitole 7.2.

Použité dáta útoku sú z dvoch zdrojov. Prvým je Canadian Institute for Cybersecurity a ich dataset DDoS Evaluation Dataset (CIC-DDoS2019) [5]. Tento dataset obsahuje niekoľko rôznych útokov. Pre naše účely z nich boli vybrané SYN flood, UDP flood, DNS amplifikácia a NTP amplifikácia. Druhým zdrojom sú útoky vygenerované z voľne dostupných nástrojov na generovanie DDoS útoku. Útok bol vykonaný na lokálnej sieti pomocou dvoch zariadení, bol zachytený programom Wireshark a následne vyfiltrovaný. Použité boli nástroje:

- LOIC veľmi známa aplikácia vďaka jej grafickému rozhraniu, naprogramovaná v C# a určená pre Windows. Ovláda sa jednoducho, užívateľ zadá cieľovú adresu, port prípadne správu, ktorá má byť v paketoch útoku. Je tu možnosť zvoliť protokol a veľkosť útoku. Zvolený bol protokol UDP.
- HULK Python skript, ktorého aktuálna verzia podporuje rôzne typy útokov. Umožňuje zmeniť vlastnosti útoku zásahom do zdrojového kódu. Útok prebiehal na TCP protokole a boli ponechané pôvodné nastavenia.
- Torshammer je jednoduchý python skript, ktorý generuje útoky bez potrebnej ďalšej konfigurácie.

Všetky spomenuté sady útokov, generované pomocou voľne dostupných nástrojov aj z Canadian Institute for Cybersecurity na útok používajú buď jednu alebo len pár IP adries. Keďže v navrhnutom programe sa používa aj zdrojová IP adresa, bola prevedená úprava týchto datasetov a adresy boli zmenené na náhodné.

Aby sme čo najviac simulovali reálne prostredie, získané datasety spojíme a vytvoríme multivektorové útoky. V tabuľke 7.1 sú uvedené jednotlivé datasety a v tabuľke 7.2 ich pospájané verzie, ktoré boli použité na testovanie. Okrem toho boli vykonané aj testy nad jednotlivými útokmi, tie avšak nie sú bližšie opísané v tejto práci, nakoľko boli ich výsledky veľmi dobré. Okrem veľmi dobrých výsledkov nepriniesli žiadne iné výsledky, ktoré by boli prínosné a bolo potrebné ich ďalej skúmať.

Názov	**Počet paketov**	**Protokol**
DNS	4000	UDP
LOIC	4000	UDP
NTP	4000	UDP
UDP	4000	UDP
TORSHAMMER	4000	TCP
SYN	4000	TCP
HULK	4000	TCP

Tabuľka 7.1: Datasety

Názov	**Obsahuje datasety**
SYNDNS	SYN,DNS
ALLUDP	DNS, LOIC, NTP, UDP
ALLTCP	TORSHAMMER, SYN, HULK
ALL	DNS, LOIC, NTP, UDP,TORSHAMMER, SYN, HULK

Tabuľka 7.2: Nakombinované datasety

7.2 Postup experimentov

Na to aby sme vedeli výsledky jednotlivých experimentov navzájom porovnávať, potrebujeme predstaviť hodnotu(skóre), ktorou by sme ich mohli opísať. Keďže cieľom nie je iba označiť útočiacu prevádzku, ale aj ju dostatočne odlíšiť od legitímnej, je potrebné brať do úvahy všetky prípady ktoré môžu nastať a nimi sú:

1. **skutočne pozitívne** (z ang. True positive, ďalej už len TP) - Útočiaci paket označený ako pozitívny(teda útočiaci)
2. **skutočne negatívne** (z ang. True negative, ďalej už len TN) - Legitímny paket označený ako negatívny(teda legitímny)
3. **falošne pozitívne** (z ang. False positive, ďalej už len FP) - Legitímny paket označený ako pozitívny
4. **falošne negatívne** (z ang. False negative, ďalej už len FN) - Útočiaci paket označený ako legitímny

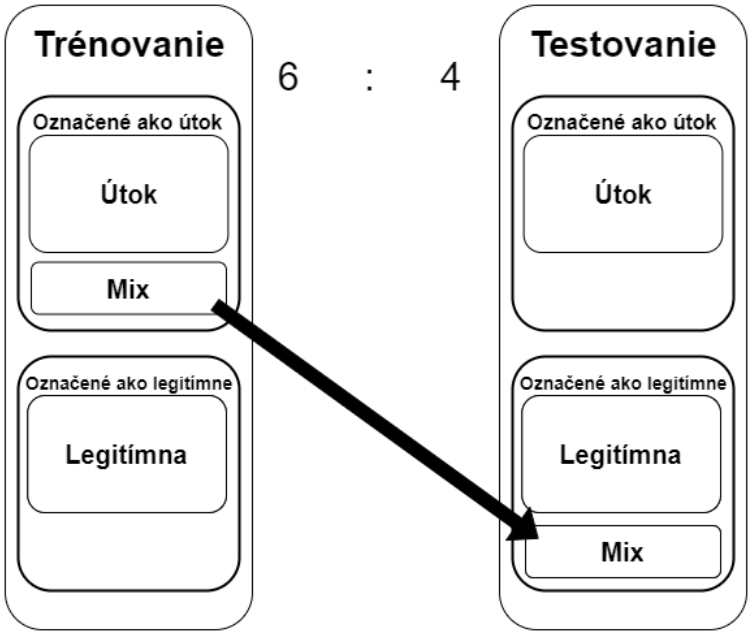
Presnosť stromového modelu sa zvykne označovať pomocou skóre, ktoré je celkovým priemerom úspešnosti, teda priemerom hodnôt TP,TN,FP a FN (FP a FN sú odčítané od

hodnoty 100). Alebo sa používa f1 skóre, ktoré je váženým priemerom týchto hodnôt. Tieto metódy pre tento program avšak nie sú vhodné. To z toho dôvodu, že na všetky prípady, ktoré môžu nastať sa kladie rovnaký dôraz, avšak našim cieľom je mať čo najnižšiu možnú hodnotu FP, pri relatívne vysokej TP. Nie sú pre nás avšak rovnako dôležité. Keby boli hodnoty nasledujúce TP=100, FP=10 tak skóre je rovnaké ako v prípade TP=90, FP=0. Avšak druhý prípad je pre nás lepším riešením. Preto budem hodnote FP prideľovať väčšiu váhu a vzorec, ktorým budem skóre počítať bude nasledovný:

$$score = \frac{\frac{1}{3} * (TP + TN + 100 - FN) + 3 * (100 - FP)}{4}$$

Aby bolo možné simulovať reálne prostredie a zároveň bolo možné vyhodnotiť účinnosť programu, navrhol som nasledujúci postup.

Pre každý test budú spracované 3 súbory. Jeden obsahuje útok, druhý legitímnu prevádzku a tretí je mix. Súbor mix obsahuje legitímnu prevádzku. Dataset LEGIT je použitý v každom teste celý. Najprv sú nazbierané dáta rozdelené v pomere 6:4 na dáta určené na trénovanie a na dáta určené na testovanie. K týmto dátam sú následne pridané dáta zo súboru mix, avšak k dátam určeným na trénovanie sú pridané ako dáta útoku a k dátam určeným na testovanie ako dáta legitímnej prevádzky. Týmto simulujeme reálnu situáciu. Znázornené to je na obrázku 7.1. Dát zo súboru mix bude vždy k obom sadám také množstvo, aby predstavovali 30% útočiacej prevádzky.



Obr. 7.1: Postup pri testovaní

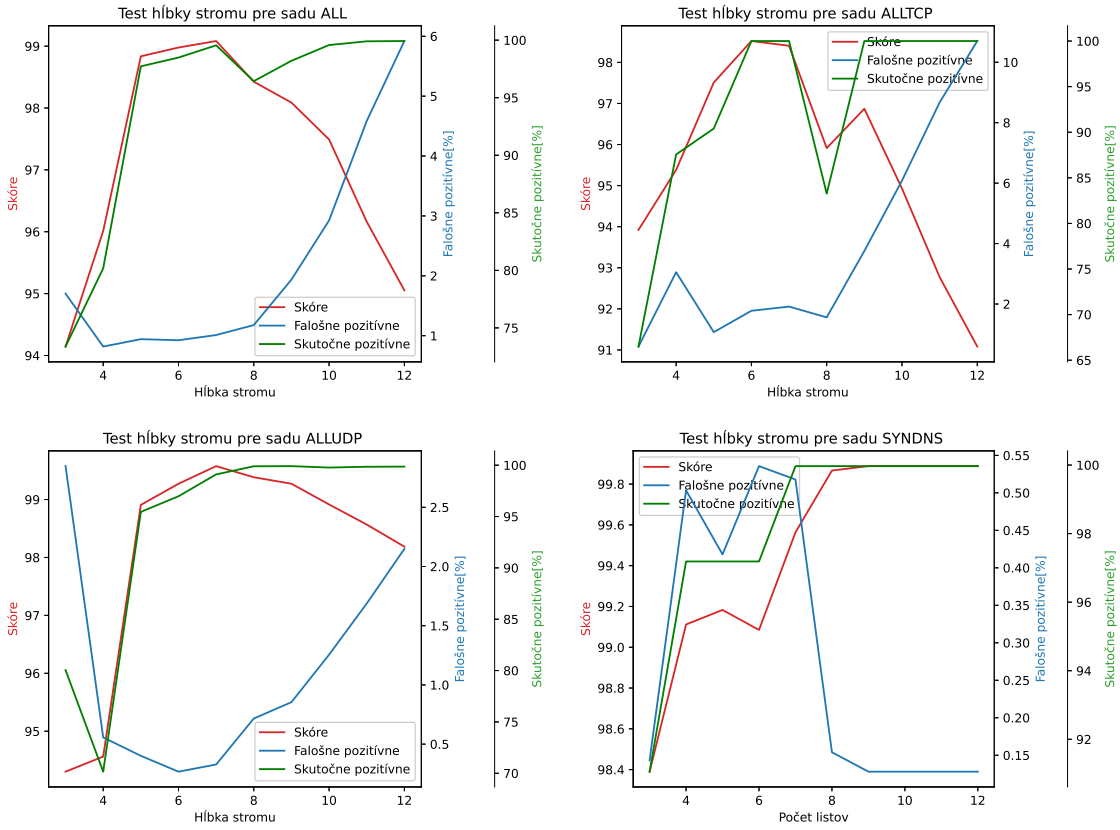
Ako bolo spomenuté v kapitole 6.2, rast stromu je treba obmedziť a máme k dispozícii 4 parametre, ktorými to môžeme dosiahnuť. Sú nimi nasledovné:

1. **max_depth**
2. **max_leaf_nodes**
3. **min_samples_leaf**
4. **min_samples_split**

Pre každú z týchto funkcií vykonáme súbor testov, ktoré budú prebiehať nasledovne. Po spracovaní súborov a rozdelení dát, ako je spomenuté vyššie, sa vytvorí model vždy pre inú hodnotu práve testovaného vstupného parametru. Tento model sa potom avšak upraví sedem krát, vždy nanovo podľa inej hodnoty nečistoty. Tieto hodnoty boli zvolené po ručnej analýze dát, sú nimi: 0.007, 0.01, 0.025, 0.05, 0.1, 0.15, 0.3. Pre každú z týchto úprav sa spraví vyhodnotenie a vyberie sa z nich tá s najlepším skóre, ktorí je počítané podľa vzorca pre skóre. Toto robíme preto, lebo sa snažíme nájsť nie len najlepší vstupný parameter, ale taktiež vhodnú hodnotu nečistoty. Predpokladom pre túto hodnotu je, že pri väčšom strome je potrebné na správnu úpravu zvoliť nižšiu hodnotu. Toto sa pri testovaní ukázalo ako pravdivé len čiastočne.

7.3 Jednotlivé experimenty

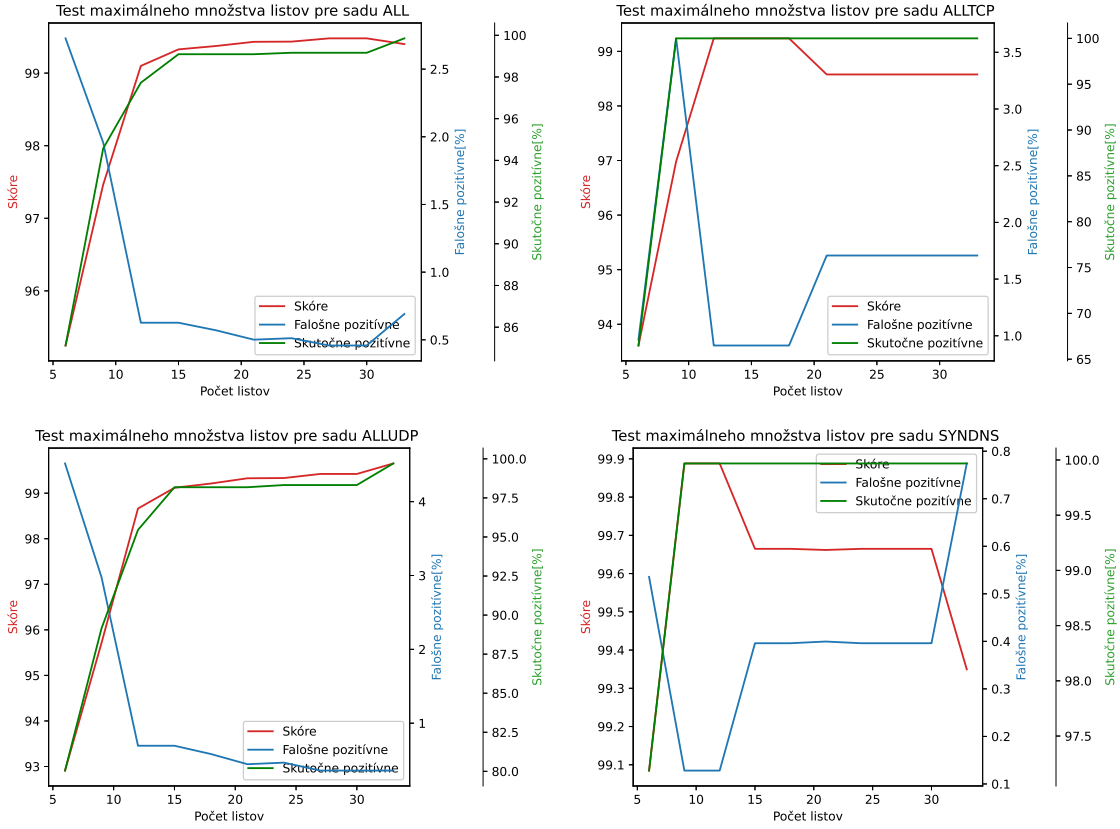
V tejto sekcii budú postupne znázornené experimenty pre všetky 4 parametre spomenuté vyššie, pre každý z nich budú zobrazené 4 grafy, každý z nich znázorňuje jeden dataset. V experimente na obrázku 7.2 sa hodnoty na osi x inkrementujú hodnotou jedna. Na obrázku 7.3 hodnotou 3 a na 7.4 a 7.5 hodnotou 0.5.



Obr. 7.2: Experimenty pre parameter max_depth

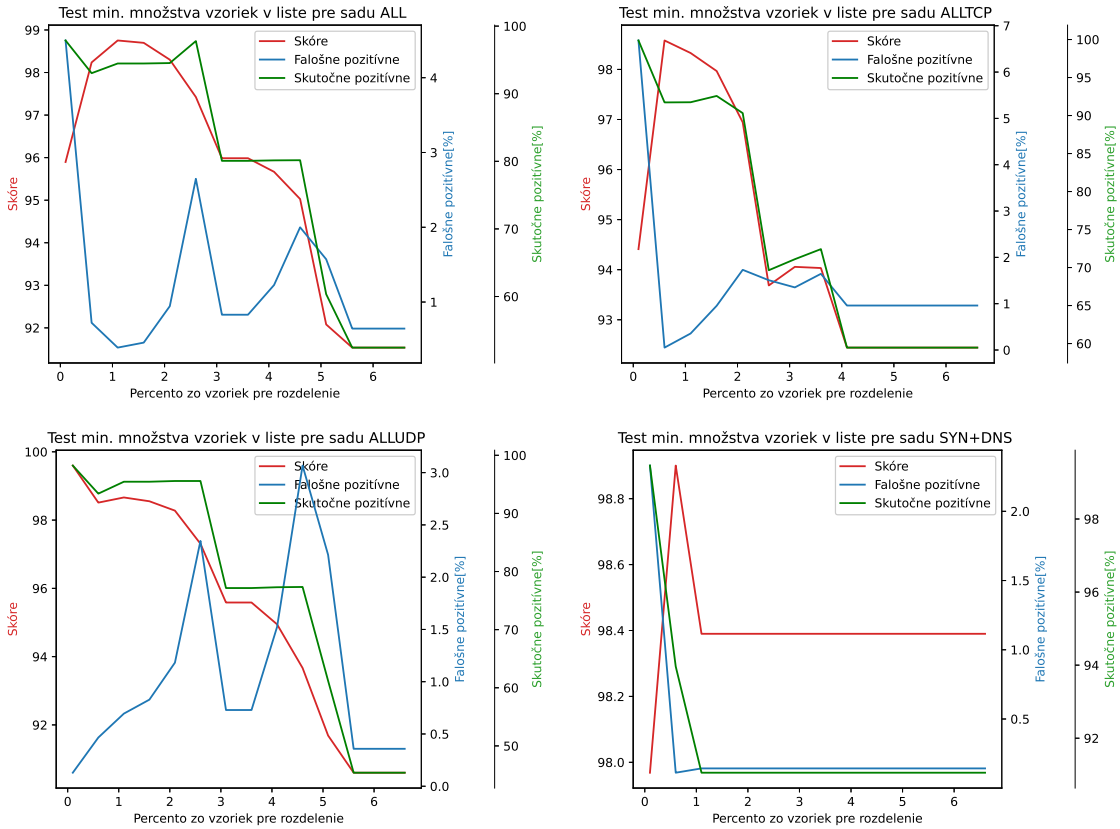
Pri použití hĺbky stromu na obmedzenie rastu môžeme pozorovať, že s väčším počtom stromov sa zvyšuje miera TP, ktorá je avšak uspokojivá v každom prípade už od piatich stromov. Avšak miera FP sa so zvyšujúcim počtom stromov zvyšuje, s výnimkou SYNDNS, kde ide o najjednoduchší útok. Vďaka tomu ho je ľahšie rozpoznať a strom sa ďalším zväčšova-

ním čistí od útočiacich paketov a používa na to veľmi malé hodnoty nečistoty. U zložitejších útokov by sa dalo za ideálne považovať 5 alebo 6 stromov, kedy je miera FP najnižšia a TP je už tiež takmer na svojom maxime. Miera nečistoty je v tomto prípade málo uniformná, ale najviac sa blíži stredu, takže okolo hodnoty 0.05.



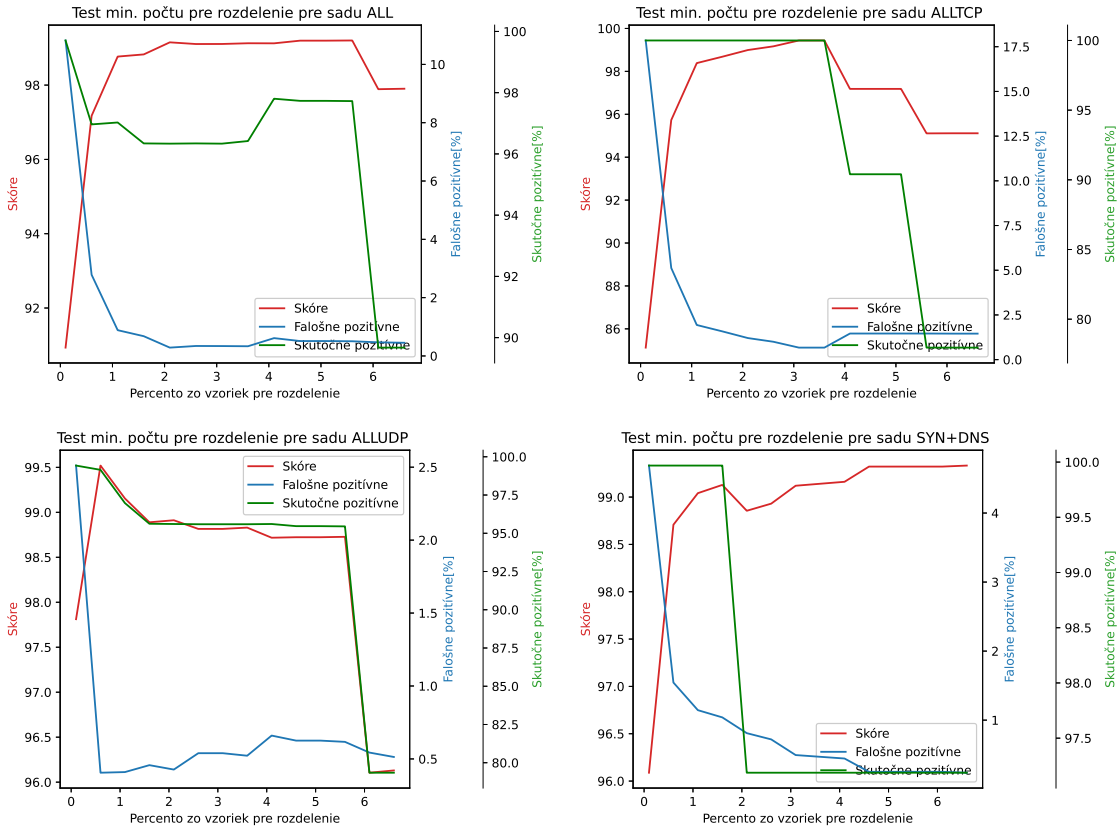
Obr. 7.3: Experimenty pre parameter max_leaf_nodes

V tomto teste sledujeme opačný trend oproti predchádzajúcemu. V SYNDNS sa miera FP zo zvyšujúcim počtom listov zvyšuje a v ostatných znižuje. 30 listov avšak odpovedá približne hĺbke stromu 5, takže v skutočnosti sa jedná o podobný trend pri všetkých datasetoch a rozdiely oproti hĺbke stromu sú minimálne. Jedinou zmenou je ALLTCP, kde sa nám podarilo dosiahnuť celkovo lepšie výsledky. Navyše vo všetkých datasetoch platil trend, že najlepšie výsledky boli dosiahnuté s hodnotou nečistoty 0.05 a 0.007 pri SYN a taktiež platil trend, že sa s pribúdajúcimi listami toto číslo znižovalo.



Obr. 7.4: Experimenty pre parameter min_samples_leaf

Výsledky testov tejto metódy sú najmenej uspokojivé, sú príliš málo uniformné. Hodnoty nečistôt sa skoro vôbec nemenili, išlo prevažne o hodnotu 0,3. V hodnotách okolo 1 možno vidieť najlepšie výsledky porovnateľné s predchádzajúcimi testami.



Obr. 7.5: Experimenty pre parameter min_samples_split

V tomto teste možno vidieť, že sa miera FP drží na minimálnych hodnotách takmer po celý čas a zároveň je miera nečistoty nastavená na 0.3 pri zložitejších útokoch a na 0.007 pri SYNDNS. Ale vo vyšších hodnotách SYNDNS nadobúda 0.1 a zároveň tam vykazuje najlepšie výsledky. V prevažnej väčšine testov. Aj napriek klesajúcim hodnotám TP sa stále jedná o dobré čísla. Možno povedať, že tento parameter sa javí ako najlepšia voľba, pokiaľ nehodláme riskovať. Pretože ak nastavíme hodnotu na 3, tak aj pri odchýlke vzhľadom na typ útoku si môžeme byť istý, že neobmedzíme legitímnych užívateľov.

Priebežné zhodnotenie

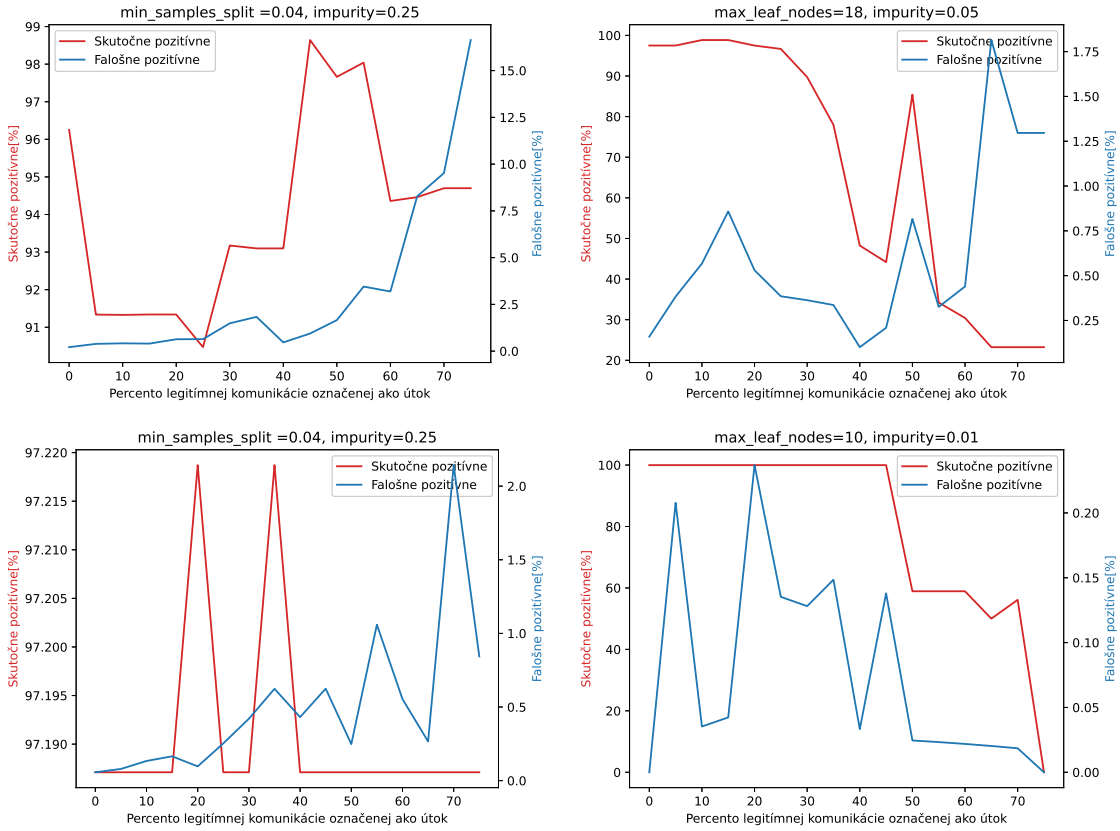
Z vykonaných experimentov možno usúdiť, že aj napriek 30% podielu legitímne prevádzky v útoku, bolo možné všetkými parametrami možné obmedziť rast stromu a zároveň zachovať dobré výsledky. Pozitívnym je taktiež zistenie, že je možné dosiahnuť uspokojivé výsledky na veľkom rozsahu zadaných parametrov ako aj nečistoty. Grafy nie sú dokonale uniformné a často sa správajú opačne než by sme predpokladali. To je avšak pochopiteľné, keď si uvedomíme tieto tri veci:

1. Jedná sa o stromovú štruktúru, ktorá pri každom ďalšom rozdelenom uzle vykazuje iné správanie, ako bolo to predchádzajúce, tento uzol sa môže rozdeliť pre nás výhodným spôsobom, ale aj naopak.
2. Jedná sa o malé odchýlky často iba v desatinách, maximálne jednotkách.
3. Grafy sú robené na malom rozsahu hodnôt, čo je vyhovujúce, lebo väčší tu nemá zmysel. Lenže kvôli tomu graf nevieme uhladiť.

4. DTC sa riadni presne danými pravidlami bez prvku náhody. a preto je jedno koľko krát by sme strom generovali s rovnakými vstupnými hodnotami, vždy by sme dostali rovnaký model. Preto nemá zmysel priemerovať testy a snažiť sa o uhladenie (zmysel by to malo napríklad pri Random Forest).
5. Svoj podiel na tom má aj to, že strom upravujeme, čo sa v niektorom stave stromu oplatí viac ako v inom a teda nám to výrazne zmení hodnoty, ktoré potom nezapadajú do očakávaného vývoja.

Možno usúdiť, že úplne najlepšie v týchto experimentoch dopadli parametre max_leaf_nodes a min_samples_split a každý z nich z iným využitím. max_leaf_nodes s hodnotou 18 a nečistotou 0.05 sa javí ako najlepšia voľba. Nie len kvôli hodnotám TP a FP, ale aj uniformite dát. Od týchto nastavení môžeme pri multivektorovom útoku očakávať dobré výsledky. Pri jednoduchom type útoku, je potrebné znížiť nečistotu na 0.01 alebo ešte nižšie a držať sa v nižších číslach listov, okolo 10. min_samples_split s nastavením 4 percent (resp. 0,04, pozri dokumentáciu[1]) a nečistotou 0,25(mierne posunuté smerom nadol oproti 0,3) je stred, v ktorom máme zo všetkých parametrov najväčšiu istotu, že budeme mať dobré výsledky ako pri jednoduchých tak aj zložitejších útokoch.

Tieto hodnoty teraz použijem na najväčší a najmenší dataset, teda ALL a SYNDNS. V prvok riadku sa nachádzajú testy pre ALL v druhom pre SYNDNS. Cieľom testu bude zistiť, koľko legitímnej prevádzky dokážu zvládnuť zmiešanej s útokom.



Obr. 7.6: Experimenty pre parameter min_samples_split

[1] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

7.4 Zhodnotenie výsledkov

Hodnoty, ktoré boli získané v prvej fáze testov a aplikované v druhej, doručili prekvapivé výsledky. Parameter max_leaf_nodes si držal stále nízku mieru FP aj vo vysokých číslach legitímnej prímesy. Pokles TP je veľký, ale tento algoritmus nepredpokladá fungovať v situáciach ako je 40% legitímnej prímesi a do tej hodnoty si drží stále dobré čísla TP. Parameter min_samples_split ukázal, že pri vyšších číslach legitímnej prímesi nie je použiteľný, nakoľko vykazuje veľké množstvo FP a malé TP. Z výsledkov experimentov je ale vhodné usúdiť, že algoritmus spĺňa svoj pôvodný účel.

Kapitola 8

Záver

Cieľom tejto práce bolo navrhnúť algoritmus pre blokovanie paketov patriacich DDoS útoku. Pre tento účel bolo potrebné dostatočne naštudovať teoretický základ tejto problematiky. Kapitola 2 sa zaoberala DDoS útokmi, ich priebehom, typmi a motiváciou k nim. Taktiež bol vysvetlený botnet a jeho úloha v rámci útoku. V kapitole 3 boli vysvetlené existujúce princípy mitigácie DDoS útokov. Boli vysvetlené rôzne prístupy k tomuto problému, konkrétne metódy a ich postupy. Ako sa ukázalo, žiadna z metód nie je dokonalá a s implementáciou každej z nich prichádzajú aj nevýhody. V kapitole 4 bol opísaný formát filtru BPF, s ktorým bolo potrebné sa oboznámiť kvôli následnej implementácii. Ako základ pre tvorbu pravidiel bol zvolený algoritmus rozhodovacieho stromu, ten bol podrobne opísaný v kapitole 5. Nasledovalo jadro práce, ktorým bol návrh, implementácia a následné experimenty. Boli objasnené dôvody výberu algoritmu rozhodovacieho stromu, iné časti programu a technológie použité na ich implementáciu. Pri návrhu algoritmu boli spomenuté zásady pre správny zber dát, a teda fungovanie celého programu. Bol kladený dôraz na rýchlosť výpočtu a nízku mieru falošne pozitívnych označení paketov. Pre správne fungovanie boli predstavené tri predpoklady, ktorými sú: útok sa odlišuje od bežnej prevádzky, útočiaca prevádzka vykazuje určitú mieru podobnosti, a že v čase útoku je väčšina prevádzky útočiacej. Na základe týchto troch predpokladov sa podarilo implementovať program v podobe skriptu v jazyku Python, ktorý dokáže nájsť vzor v paketoch prebiehajúceho útoku s vysokou mierou presnosti. Pokiaľ sú zvolené správne nastavenia na zamedzenie rastu stromu a na jeho úpravu. Z toho dôvodu boli experimenty smerované tak, aby bolo možné zistiť, ktoré nastavenia sú najideálnejšie. Experimentami na nazbieraných dátach DDoS útokov sa podarilo navrhnúť optimálne nastavenia.

Program je pripravený na ďalšie rozširovanie a bolo do neho taktiež implementované generovanie pravidiel do formátu wireshark-filter a export výsledného modelu stromu vo vizuálnej podobe. Pre ďalšie výskumné účely. Do budúcnosti sa naskytá možnosť tento program implementovať priamo do zariadenia DDoS Protector a vykonať testy na skutočných útokoch.

Literatúra

[1] Abedinya, A. *Survey of the Decision Trees Algorithms (CART, C4.5, ID3)* [online]. 2019 [cit. 2021-03-19]. Dostupné z: https://medium.com/@abedinia.aydin/survey-of-the-decision-trees-algorithms-cart-c4-5-id3-97df842831cd.

[2] Baumann, P., Hochbaum, D. a Yang, Y. A comparative study of the leading machine learning techniques and two new optimization algorithms. *European Journal of Operational research* [online]. 2019, Vyd. 272, Č. 3, [cit. 2021-04-02]. Dostupné z: https://www.sciencedirect.com/science/article/abs/pii/S0377221718306143.

[3] Burget, L. *Strojové učení a rozpoznávání* [online]. VUT, Brno [cit. 2021-04-07]. Dostupné z: https://www.fit.vutbr.cz/study/courses/SUR/public/prednasky/01_uvod/SUR_uvod.pdf.

[4] Cheng, C., Kung, H. a Tan, K.-S. *Use of Spectral Analysis in Defense Against DoS Attacks* [online]. Harvard University, 2002 [cit. 2021-03-27]. Dostupné z: https://homepages.laas.fr/owe/METROSEC/DOC/2002-globecom-cheng-kung-tan.pdf.

[5] *DDoS Evaluation Dataset (CIC-DDoS2019)* [online]. [cit. 2021-04-11]. Dostupné z: https://www.unb.ca/cic/datasets/ddos-2019.html.

[6] *DDoS Protector* [online]. CESNET [cit. 2021-04-12]. Dostupné z: https://www.liberouter.org/technologies/ddos-protector/.

[7] *Decision Trees* [online]. [cit. 2021-04-11]. Dostupné z: https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart.

[8] *GAMING – You Can't Solo Security* [online]. Akamai, 2020 [cit. 2021-03-29]. Dostupné z: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/soti-security-gaming-you-cant-solo-security-report-2020.pdf.

[9] *HTTP Flood Attack* [online]. [cit. 2021-04-11]. Dostupné z: https://www.cloudflare.com/en-gb/learning/ddos/http-flood-ddos-attack/.

[10] Kerekesová, K. a Malaťáková, M. *ID3* [online]. TUKE, 2011 [cit. 2021-04-06]. Dostupné z: http://oz.koncz.sk/attachments/article/155/ID3_Kerekesova_Malatakova.pdf.

[11] Kim, Y., Lau Cheong, W., Chuah, C. a Chao, H. PacketScore: a statistics-based packet filtering scheme against distributed denial-of-service attacks. *IEEE Transactions on Dependable and Secure Computing* [online]. IEEE. 2006, [cit. 2021-03-29]. Dostupné z: https://ieeexplore.ieee.org/document/1632008.

[12] Kupreev, O., Badovskaya, E. a Gutnikov, A. *DDoS attacks in Q4 2020* [online]. 2021 [cit. 2021-03-26]. Dostupné z: https://securelist.com/ddos-attacks-in-q4-2020/100650/.

[13] Mahjabin, T., Xiao, Y., Sun, G. a Jiang, W. *A survey of distributed denial-of-service attack, prevention, and mitigation techniques* [online]. 2017 [cit. 2021-05-03]. Dostupné z: https://journals.sagepub.com/doi/pdf/10.1177/1550147717741463.

[14] McCanne, S. a Jacobson, V. *The BSD packet filter: A new architecture for user-level packet capture.* [online]. Lawrence Berkeley Laboratory, 1992 [cit. 2021-04-07]. Dostupné z: http://www.tcpdump.org/papers/bpf-usenix93.pdf.

[15] Nilsson, N. *Introduction to Machine Learning* [online]. Stanford University, 1998 [cit. 2021-04-11]. Dostupné z: https://ai.stanford.edu/~nilsson/MLBOOK.pdf.

[16] Patrikakis, C., Masikos, M. a Zouraraki, O. Distributed Denial of Service Attacks. *The Internet Protocol Journal* [online]. 2004, Vyd. 7, Č. 4, [cit. 2021-03-15]. Dostupné z: https://web.archive.org/web/20190826143507/https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-30/dos-attacks.html.

[17] Seralathan, Y., Oh, T. a Jadhav, S. e. a. IoT security vulnerability: A case study of a Web camera. *20th International Conference on Advanced Communication Technology* [online]. IEEE. 2018, [cit. 2021-04-03]. Dostupné z: https://ieeexplore.ieee.org/document/8323685.

[18] Walkowski, D. *What Is a Distributed Denial-of-Service Attack?* [online]. 2019 [cit. 2021-03-18]. Dostupné z: https://www.f5.com/labs/articles/education/what-is-a-distributed-denial-of-service-attack-.

[19] *What is a DDoS attack?* [online]. [cit. 2021-04-09]. Dostupné z: https://www.cloudflare.com/en-gb/learning/ddos/what-is-a-ddos-attack/.

[20] *What is the Mirai Botnet?* [online]. [cit. 2021-03-28]. Dostupné z: https://www.cloudflare.com/en-gb/learning/ddos/glossary/mirai-botnet/.

[21] *What is the OSI Model?* [online]. [cit. 2021-04-08]. Dostupné z: https://www.cloudflare.com/en-gb/learning/ddos/glossary/open-systems-interconnection-model-osi/.