VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Peter Náčin

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

DETEKCE ÚTOKU SLOWDROP
SLOWDROP ATTACK DETECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Peter Náčin**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Marek Sikora**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Peter Náčin ***ID:*** 188963
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Detekce útoku SlowDrop

POKYNY PRO VYPRACOVÁNÍ:

Útok SlowDrop je jeden z nejnovějších tzv. pomalých DoS útoků. Hlavní charakteristikou tohoto útoku je velmi věrné napodobení legitimního uživatele s pomalým internetovým připojením. Útok nevykazuje žádnou výraznější signaturu, proto je jeho detekce poměrně komplikovaná.
Úkol této diplomové práce je podrobně nastudovat charakteristiku útoku SlowDrop, navrhnout a vytvořit laboratorní sít s legitimními uživateli, otestovat účinnost útoku, navrhnout a implementovat metodiku detekce útoku s co nejvyšší možnou přesností detekce. Implementovaný detektor bude podrobně popsán a jeho funkčnost ověřena v laboratorní síti.

DOPORUČENÁ LITERATURA:

[1] CAMBIASO, Enrico, Giovanni CHIOLA a Maurizio AIELLO. Introducing the SlowDrop Attack. Computer Networks [online]. 2019, 150, 234-249 [cit. 2019-04-18]. DOI: 10.1016/j.comnet.2019.01.007. ISSN 13891286. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1389128619300210

[2] MAZÁNEK, Pavel. Modelování a detekce útoku SlowDrop. Brno, Rok, 75 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Marek Sikora

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Marek Sikora

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Diplomová práca je zameraná na detekciu slow DoS útoku pomenovaného SlowDrop. Útok sa snaží napodobniť legitimného užívateľa s pomalým internetovým pripojením a nevykazuje žiadnu výraznú signatúru, preto je útok náročné detekovať. Diplomová práca vychádza z práce Ing. Mazánka, v ktorej bol vytvorený skript SlowDrop útoku. V teoretickej rovine je popísaná problematika DoS útokov vo všeobecnosti, ale aj konkrétne. Ďalej sú v práci navrhnuté metódy riešenia problematiky detekcie SlowDrop útoku. Metódy sú následne detailne opísané a odskúšané v simulačnom prostredí. Praktická časť opisuje analýzu dát, detekciu pomocou signatúr, detekciu anomálií pomocou neurónových sietí a detekčný skript. Vo všetkých praktických častiach sú detailne popísané použité technológie a postupy riešení. Takisto je uvedená konkrétna implementácia riešenia a dosiahnuté výsledky. Na záver sú jednotlivé výsledky zhodnotené, porovnávané jednotlivo, ale aj medzi sebou. Zo získaných výsledkov vyplýva, že útok je detekovateľný pomocou neurónovej siete a vytvoreného detekčného skriptu.

KLÍČOVÁ SLOVA

detekcia anomálií, DDoS, DoS, IDS, neurónová sieť, SDA, signatúry, SlowDrop útok, skript

ABSTRACT

The diploma thesis is focused on the detection of a slow DoS attack named SlowDrop. The attack tries to imitate a legitimate person with a slow internet connection and does not show a new strong signature, so the attack is difficult to detect. The diploma thesis is based on the work of Ing. Mazanek in which the SlowDrop attack script was created. At the theoretical level, the issue of DoS attacks is described in general, but also in particular. Furthermore, the work develops methods for solving the problem of SlowDrop attack detection. The methods are then defined in detail and tested in a simulation environment. The practical part describes data analysis, signature detection, anomaly detection using neural networks and a detection script. In all practical parts, the used technologies and solution procedures are described in detail. The specific implementation of the solution and the achieved results are also presented. Finally, the individual results are evaluated, compared individually, but also among themselves. The obtained results show that the attack is detectable using a neural network and by created detection script.

KEYWORDS

anomaly detection, DDoS, DoS, IDS, neural network, script, SDA, signatures, SlowDrop attack

NÁČIN, Peter. *Detekce útoku SlowDrop*. Brno, 2030, 83 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Marek Sikora,

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Detekce útoku SlowDrop“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Marku Sikorovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cieľom diplomovej práce bolo detekovať SlowDrop[1] útok. Pre splnenie tohoto cieľu bolo požadované splniť niekoľko bodov. Na začiatku bolo potrebné definovať a popísať všetky dôležité časti práce z teoretickej stránky. Vysvetlené sú podrobnosti a kategórie týkajúce sa Denial of Service útokov vo všeobecnosti, ale následne je pohľad zameraný konkrétne na SlowDrop útok. Pozornosť je venovaná návrhu a postupom riešení pri detekcii útoku.

Nevyhnutnou častou je kapitola, ktorá zahŕňa predchádzajúce riešenie, na ktorom je táto diplomová práca postavená. Sú v nej uvedené výsledky ku ktorým sa autor[7] dopracoval a taktiež informácie o vytvorenom skripte, ktorý útok iniciuje.

Praktická časť obsahuje niekoľko kategórií, medzi ktoré patria:

- vytvorenie simulačného prostredia,
- otestovanie skriptu,
- software Suricata,
- neurónová sieť,
- detekčný skript.

Po naštudovaní a pochopení teoretickej sféry tejto problematiky bolo nevyhnutné pripraviť testovaciu sieť obsahujúcu položky, ktoré budú kompatibilné s poskytnutým skriptom.

Prvým bodom praktickej časti je detekcia pomocou programu Suricata. Primárnym cieľom tohoto programu je detekcia potenciálnych hrozieb. Suricata obsahuje súbor pravidiel, podľa ktorých upozorňuje na hrozby a útoky. Práca sa detailnejšie venuje aj tejto problematike, tak ako aj vyskúšaniu jednotlivých pravidiel a ich modifikácii za účelom detekcie SlowDrop útoku.

Druhým bodom praktickej časti je popis neurónových sietí v teoretickej rovine. Z praktického hľadiska prebehla hlbšia analýza sledovaných dát, ktoré útok produkuje a vyhľadanie signifikantných znakov SlowDrop útoku, podľa ktorých bude možné vymodelovať neurónovú sieť. Sieť bude následne implementovaná, otestovaná a budú detailne popísané jednotlivé kroky od zberu dát, cez učenie, až po získanie výsledkov a ich následnú interpretáciu.

V neposlednom rade bude vytvorený a popísaný detekčný skript naprogramovaný v jazyku Python, ktorý bude mať za úlohu odhalenie útočníka a zobrazenie jeho identity. Taktiež v záverečnej časti budú uvedené návrhy, ako sa v prípade útoku brániť a ako odstrániť škodlivé zariadenia zo siete.

Na záver práce sa zhodnotí úspešnosť na stanovených cieľov a budú zhrnuté celkové výsledky diplomovej práce.

1 Kybernetické hrozby

V novodobom svete je internet každodennou súčasťou bežného života. Človek trávi na internete mnoho času, či už sa jedná o zábavu, alebo prácu. V dnešnej dobe je potrebné resp. nutné internet z času na čas použiť. Pre mnohých, ktorí ho používajú na dennej báze, môže internet poskytovať aj zdroj príjmu. Čím ďalej viac jednotlivcov a takisto aj spoločností si zakladá nové webové stránky, ktoré sa z rôznych dôvodov môžu stať cieľom útoku iných užívateľov, ktorí nemajú kladné úmysly. Dôvody môžu byť rôzne, ako napríklad znemožnenie prístupu na server za účelom zníženia príjmov daného vlastníka, alebo získaním osobných a citlivých informácií, či už o klientoch resp. o užívateľoch serveru. Výsledkom tohoto procesu v čase vzniklo viacero rôznych útokov na dané servery. Taktiež bola vyvinutá aj detekcia a obranné mechanizmy, ktoré majú dané hrozby zablokovať, alebo aspoň minimalizovať ich účinnosť.

Keďže prostredie informačných technológií sa rozvíja rapídnou rýchlosťou je potrebné sa týmto útokom venovať a rozpoznať ich hrozby. Podobne ako rôzne zdroje, práce a dokumentácie je táto práca zameraná na útoky typu DoS (Denial of Service).

1.1 Útoky typu DoS a ich kategorizácia

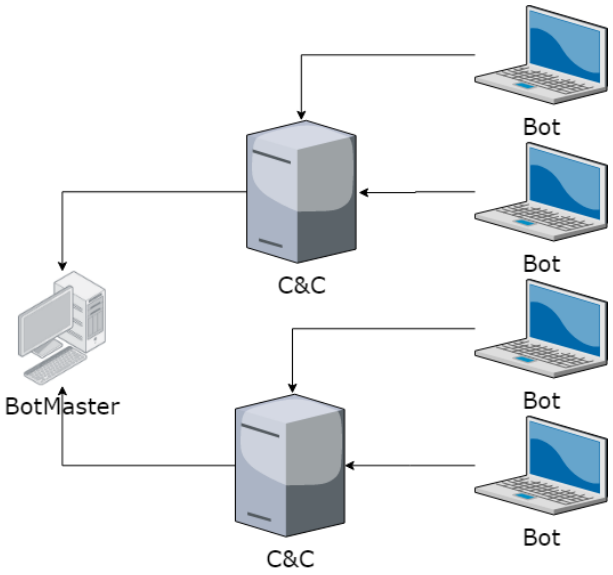
Dve hlavné kategórie týkajúce sa DoS útokov sú všeobecne známe ako DDoS (Distributed Denial of Serivce) alebo DoS.

Definícia DDoS[2] alebo celým názvom Distributed Denial of Serivce je podľa [3]: *"A DDoS attack is an attempt by an attacker to create so much traffic or congestion to a target application or an internet application that it impedes the traffic flow for normal visitors."*

Táto definícia je preložená do slovenčiny slovami: *"DDoS útok je pokus útočníka o vytvorenie takého prenosu, preťaženia cieľovej aplikácie, alebo internetovej aplikácie, ktoré obmedzujú dátový tok normálnych návštevníkov."*

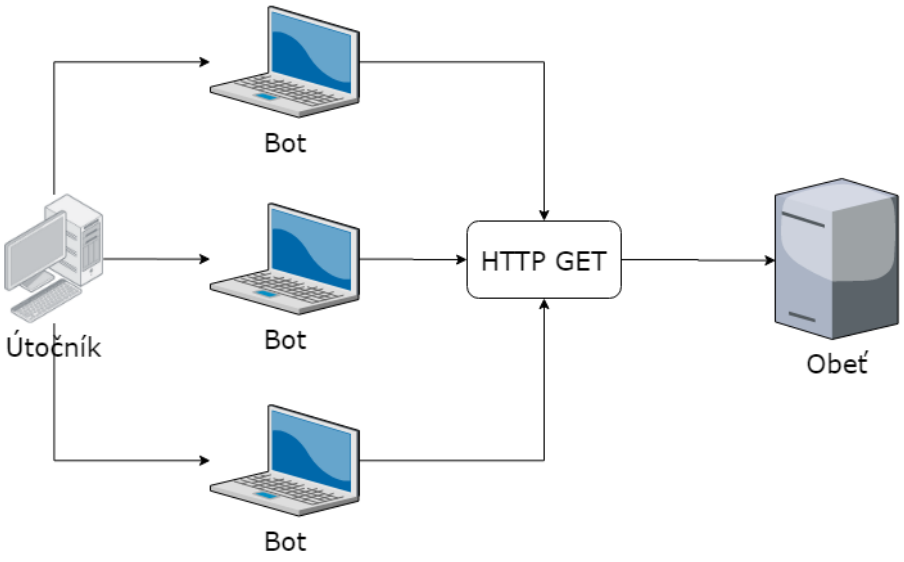
Z definície je vidieť, že DDoS útok za pomoci viacerých počítačov generuje škodlivý sieťový prenos, ktorý je zasielaný na stranu obete s cieľom obmedziť jej funkčnosť. DDoS útok môže byť cielený na využitie zraniteľnosti servera, a to takým spôsobom, že sa zameriava na posielanie špecifických paketov, ktoré majú za úlohu blokovanie servera. Druhou metódou je priame zahltenie servera posielaním nadmerného množstva požiadaviek, následkom ktorého je vyčerpanie zdrojov servera. Obe metódy vedú k obmedzeniu sieťovej prevádzky na serveri [2].

Pre efektívny beh a riadenie DDoS 1.1 útoku je potrebné vytvoriť Botnet [4][5][6] alebo sieť botov. Táto sieť sa skladá z 2 častí. Prvou časťou botnetu sú riadiace zariadenia. Odbornejšie sa nazývajú ako *CnC* servery. Pod touto skratkou sa skrýva Com-



Obr. 1.1: DDoS útok

mand and Control server, preložené ako server velenia a riadenia. Tieto stroje slúžia ako vedúce body, ktoré majú riadiacu schopnosť ovládať podriadené počítače nazývajúce sa boti. Útočník vyberie a nastaví riadiace stroje tak, aby boli schopné ovládať botov, cez ktorých je útok zahltenia vykonávaný. Už spomínaní boti, alebo inak pomenovaní aj zombies, reprezentujú koncové zariadenia, ktoré majú za úlohu zamaskovať skutočného útočníka a zahltiť cieľovú obeť. Distribuovaná varianta útoku DoS môže pod sebou skrývať desiatky a v niektorých prípadoch až tisíce botov, ktorí sú na server vypustení. Celková sieť môže vyzerať podobne ako na obrázku 1.2[5].



,

Obr. 1.2: Botnet štruktúra

Druhou kategóriou sú o tzv. nedistribuovaných Denial of Service útokov, alebo iba DoS. V tomto prípade sa útočník zameriava na používanie iba jedného počítača, z ktorého je útok iniciovaný. Tieto typy útoku mávajú nižšiu efektivitu, keďže DDoS dokáže pomocou viacerých zdrojov zablokovať sieť oveľa väčším spôsobom.

Nedistribuovaný typ je jednoduchšie detekovateľný ochrannými mechanizmami serveru. Tento fakt je zrejmý, pretože tok dát, ktoré sú považované ako škodlivé majú určitú spoločnú charakteristiku. Táto charakteristika je samozrejme skutočnosť, že pochádzajú z jedného stroja a preto je veľmi náročne vyprodukovať veľké škodlivé množstvo, ktoré by zahltilo server a na druhej strane, aby bolo od seba odlišiteľné. Z tohoto plynie, že DDoS útoky majú vyššiu efektivitu a nižšiu detekovateľnosť.

Ing. Mazánek [7] v jeho práci o Modelování a detekci útoku SlowDrop rozdeľuje DoS útoky podľa spôsobu útoku na:

- Záplavové
- Zneužívajúce zraniteľnosť

Záplavový typ útokov má jednoduchý cieľ a to zahltiť obeť vybranú útočníkom. Napriek tomu, že existuje viacero druhov týchto útokov v tejto práci sú uvedené iba niektoré ako napríklad tieto štyri:

- *ICMP Flood* - Internet Control Message Protocol Flood útok
- *DNS Flood* - Domain Name System Flood útok
- *DHCP Starvation* - Dynamic Host Control Protocol Starvation útok
- *SYN Flood*

Jednoduchým variantom je *ICMP Flood*. Tento útok neustále preposiela na server správu *echo request* a tým pádom je server nútený odpovedať správu *echo reply*. Touto jednoduchou formou dochádza k zahlteniu a následnému zablokovaniu serveru.

Druhým spomenutým je *DNS Flood*. Tento typ útoku sa zameriava iba na DNS servery. V podstate fungovanie spočíva v tom istom ako ICMP Flood, akurát sa jedná o DNS požiadavku. Útok je iniciovaný zasielaním veľkého množstva požiadavkou na DNS server, ktorý následne nestíha odpovedať.

Predposledným útokom z tejto kategórie je *DHCP Starvation*, alebo DHCP vyhladovanie. DHCP reprezentuje protokol, ktorý prideľuje adresy novým zariadeniam v sieti. V tomto prípade je využitá obmedzenosť adresového priestoru, ktorý útočník zneužije tým, že neustále posiela požiadavky a snaží sa tak vyčerpať priestor v sieti pre nové zariadenia.

Popis rozdelenia je zakončený popisom *SYN Flood* [8], ktorý bol následne vybratý na otestovanie funkčnosti softwaru Suricata [9] popísaného v kapitole 6. SYN Flood sa zameriava na spojenie pomocou 3-way handshaku protokolu TCP(Trasmission Control Protocol), kde predlžuje spojenie a snaží sa oddialiť situáciu, aby bolo spo-

jenie prerušené. Následne sa snaží zahltiť všetky spojenia, aby ostatní užívatelia nevytvorili spojenia so serverom.

Mimo záplavových útokov môžeme hovoriť aj o útokoch sústrediacich sa na zraniteľnosť. Do tejto kategórie spadá aj SlowDrop útok ako taký. Táto kategória je vo všeobecnosti menej známa, pretože nájsť konkrétne zraniteľnosti systému je obtiažnejšie ako spustenie DoS útoku, na ktorý sú v prvom rade potrebné hlavne zdroje. Mazánek túto kategóriu opisuje slovami: *"Mezi tyto útoky spadají mimo jiné zejména tzv. pomalé neboli slow DoS útoky, mezi které spadá i útok SlowDrop."* [7]

1.2 Pomalé DoS útoky

SDA [10][11], alebo celým názvom Slow DoS Attacks, je typ útokov, ktorý používa malý dátový tok zo strany útočníka. V porovnaní s bežnými útokmi tento typ nevyžaduje veľkú šírku pásma a tak je náročné ich zmierniť. Následkom tejto skutočnosti je, že útok vygeneruje dátový prenos, ktorý je náročne odlíšiť od bežného dátového toku a tak tieto typy útokov dokážu byť neodhalené po dlhú dobu.

Vďaka ich nenáročnosti na zdroje a prostriedky, sú nízke a pomalé útoky zväčša spúšťané ako nedistrubovaná varianta DoS útoku a tým pádom nie je potrebné zabezpečiť vytváranie botnetu. Medzi najznámejšie útoky tohoto typu patrí Slowloris a R.U.D.Y, kde prvý spomenutý bude vyskúšaný pre testovanie servera.

Útoky tohoto typu sú smerované na web server. Majú za cieľ spojiť všetky vlákna s pomalými požiadavkami a tak zabrániť legitímnym užívateľom sa pripojiť ku serveru. Prenos dát je pomalý, ale nie až tak pomalý, aby došlo k zatvoreniu spojenia zo strany servera vďaka vypršaniu časového limitu stanoveného serverom. Útok zväčša používa hlavičky Hypertext Transfer Protocol, alebo skrátene iba HTTP a súčasne aj TCP prenos. V závislosti na útoku sa zasahuje buď do HTTP alebo TCP hlavičky, prípadne do oboch.

SlowLoris útok sa najprv pripojí k obeti a posiela nekompletné HTTP GET hlavičky, týmto spôsobom obsadí jednotlivé vlákna a server ponecháva otvorené spojenie s očakávaním príjmu ďalších častí hlavičiek. Druhým spomínaným je R.U.D.Y alebo celý názvom ARE-YOU-DEAD-YET?. Tento typ útoku generuje požiadavky HTTP POST. Server obdrží informácie o tom, koľko dát môže očakávať a následne ich posiela veľmi pomalým tempom. Týmto princípom server udržuje spojenie, pretože očakáva prichádzajúce dáta.

Proti útokom je možné sa brániť dvoma spôsobmi. Prvým je aktualizácia dostupnosti serveru. Zvýšením dostupnosti a tak vytvorením viacerých možných spojení dokáže server udržiavať pripojenie viacerých užívateľov a tak sťažiť útočníkovi schopnosť zahltiť server.

Druhý možným riešením je rezervná ochrana založená na proxy, ktorá zredukuje účinok nízkych a pomalých útokov, predtým ako sa vôbec dostanú na cieľový server.

1.3 SlowDrop

Problematika SlowDrop útoku [1] je pomerne novou záležitosťou vo svete informačných technológií. Po pochopení a rozdelení DoS útokov je nutné zaradiť a špecifikovať kritériá, ktoré charakterizujú SlowDrop útok, ktorému sa práca podrobnejšie venuje z viacerých uhlov. Útok SlowDrop využíva protokol spoľahlivého prenosu TCP a snaží sa simulovať užívateľa s nekvalitným internetovým pripojením. Vďaka tomu, že útok simuluje bežného užívateľa, ktorý nevykazuje známky podozrivej aktivity, myšlienka SlowDrop útoku je veľmi podobná Slow Next útoku, ktorý posiela legitimnú požiadavku a dostane naspäť štandardnú odpoveď a tým pádom môže zneužiť *next parameter*. [1]

Ak by sme analyzovali trvalé spojenie, klient môže využívať rovnaký komunikačný kanál pre viaceré po sebe idúce žiadosti. Tým pádom *next parameter* opisuje čas, ktorý prebehol medzi koncom jednej a začiatkom druhej požiadavky rovnakého spojenia. V tomto čase útočník neposiela žiadne ďalšie dáta a týmto spôsobom sa zvýši hodnota pre *next parameter* oproti normálnemu chodu. Týmto spôsobom je dosiahnuté, že obeť bude zablokovaná a ďalší klienti nebudú mať možnosť komunikovať so serverom.

Rovnaký koncept je zavedený pre SlowDrop útoku, ktorý zahŕňa klientov s nespoľahlivým spojením. V tomto prípade sa nejedná o *next parameter*, ktorý je zneužitý. SlowDrop útok používa rozdielny a to parameter, ktorý reprezentuje čas potrebný na odpoveď poslanú ku klientovi. Myšlienka za SlowDrop útokom je taká, že opakovane požaduje jeden zdroj zo serveru, ak je to možné jedná sa o veľký fragment dát.

Pre účely diplomovej práce je použitý nedistribuovaný variant DoS útoku. Pochopiteľne pre vyššiu efektivitu útoku a aplikovanie drastickejších následkov na server, by bolo možné použiť distribuovanú variantu útoku, pre ktorý by bolo potrebné oveľa väčšie množstvo zdrojov. Špecifickou vlastnosťou tohoto typu útoku je aj jeho náročná detekovateľnosť.

Ako môžu webové služby zistiť nízky a pomalý útok? Techniky detekcie rýchlosti používané na identifikáciu a zastavenie tradičných útokov DDoS sa nezachytia pri nízkom a pomalom útoku, pretože vyzerajú ako bežná prevádzka. Najlepším riešením pri ich detekcii je starostlivé sledovanie a zaznamenávanie využívania zdrojov servera v kombinácii s analýzou správania. Porovnajte prenos a správanie používateľov v normálnych časoch s prenosom a správaním používateľov počas obdobia potenciálneho útoku.

Ak servery pracujú pomaly alebo zlyhajú a existuje podozrenie na nízky a pomalý útok, jedným znakom takéhoto útoku je, že bežné procesy používateľov trvajú oveľa dlhšie. Ak akcia používateľa (napríklad vyplnenie formulára) zvyčajne trvá niekoľko sekúnd, ale namiesto toho trvá minúty, alebo hodiny a zaberá oveľa viac zdrojov servera, ako je bežné, príčinou môže byť slabý a pomalý útok.

1.3.1 Efekty

Prvým aspektom útoku SlowDrop je efekt útoku. Efekt sa venuje následkom, ktoré má útok spôsobiť. Analýzou spojenie je zistené, že toto spojenie pretrváva po celú dobu. SlowDrop útok spôsobuje zahadzovanie veľkého množstva paketov, ktoré prichádzajú od serveru. Server tak rozumie tomuto stavu ako komunikačnému problému a preto je spojenie predlžované. Vyvolaný stav má za následok, že server sa stáva zraniteľný na dlhšie obdobie, keďže zdroje sú vyčerpané vďaka útoku. Táto špecifická charakteristika je zdielaná s ďalšími typmi DoS útokov. V tomto prípade je dosiahnuté rozšírenie časovania okupovaných zdrojov, ale aj zníženie pásma, ktoré je potrebné pre vykonanie prenosu. Tieto dôvody vedú k tomu, že server udržuje spojenie až do kým nie je prenos dokončený. Tento mechanizmus je hlavnou myšlienkou ako SlowDrop útok funguje a môže viesť k extrémne dlhým spojenia, kde v praxi môže server ukončiť spojenie s klientom.

1.3.2 Držanie spojenia

Útočník obsaďuje všetky dostupné naslúchajúce vlákna serveru a tie drží obsadené a to spôsobí, že ostatní užívatelia se nemôžu pripojiť. Po nadviazaní TCP spojenia cez 3-way handshake útočník požiada o dáta HTTP requestom a až potom odpovede na tento požiadavok zahadzuje. Server je takto donútený nedoručené dáta posielať znovu. Týmto mechanizmom bude celé spojenie predlžované a bude viesť k obmedzeniu funkčnosti pri obsluhovaní legitimných klientov.

1.3.3 Zahadzovanie paketov

SlowDrop útok simuluje nespoľahlivých klientov, t.j. nespoľahlivé pripojenie. Aby bolo možné správne simulovať takéto prostredie, klient by nemal prijímať zahodené ACK segmenty, ktoré potvrdzujú príjem dát zo serveru. Príjem ACK segmentov ho môže prinútiť k tomu, aby odosielal súvisiace ACK segmenty obeti (nesprávne), pričom špecifikuje správny príjem dát.

Bolo by možné selektívne zahodiť pakety ACK smerujúce na obeť s tým, aby sa obeť mylne domnievala, že odoslané pakety nedorazili na miesto určenia. Toto správanie by však správne nesimulovalo nespoľahlivé pripojenie. Preto je potrebný

presnejší prístup. Všeobecne je možné za účelom zahodenia paketov skôr, ako ich obdrží klientská aplikácia, použiť dva možné prístupy:

- zahodiť pakety skôr, ako sa dostanú do cieľa, napríklad prostredníctvom vhodných sieťových zariadení (napríklad firewall, alebo klepnutie v sieti), alebo
- zahodiť pakety pre cieľového hostiteľa skôr, ako ich príjme aplikácia.

1.3.4 Limity

Poslednou podsekciou v opise jednotlivých aspektov SlowDrop útoku je časť, ktorá opisuje konkrétne limity, ktoré sprevádzajú tento útok. SlowDrop útok je limitovaný dvoma podstatnými vecami. Prvým problémom je veľkosť zdrojov. Keďže je zneužitá fragmentácia transportnej vrstvy, útok zlyhá v prípade obsahovo malých odpovedí. V prípade, že útočník chce zneužiť fragmentáciu, veľkosť požadovaných zdrojov musí byť vyššia ako dáta, ktoré sú v vynesené paketom obsahujú rovnakú veľkosť ako je maximálna veľkosť segmentu.

Druhým limitom je použitie SlowDrop útoku iba pre špecifický protokol transportnej vrstvy, ktorým je *TCP*. Keďže *UDP*(User Datagram Protocol) poskytuje nespoľahlivé spojenie, tým pádom server nepodporuje retransmisiu paketov.

2 Prechádzajúce riešenia

V poradí druhá kapitola zhrnie dôležité informácie o diplomovej práci [7] z roku 2020, na ktorú bude táto práca nadväzovať. V spomínanej práci je opísaná problematika kybernetických útokov so špecifikáciou na SlowDrop útok. Zameriava sa na jednotlivé typy DoS útokov, ciele útoku, spôsob útoku a typy spojenia v teoretickej rovine. Následne sa venuje podrobnejšiemu popisu SlowDrop útoku ako takému s prípravou útoku a testovacieho prostredia. Celkovým výsledkom práce je skript naprogramovaný v jazyku Python 3.8, ktorý iniciuje útok, ktorému je venovaná podkapitola 2.1.

2.1 SlowDrop Skript

Poskytnutý skript je použitý výhradne na testovanie v lokálnom prostredí pre účely diplomovej práce. Skript spoločne s ďalšími útokmi typu DoS, ako sú napríklad SlowLoris a SYN flood, boli použité za účelom testu na serveri, aby sa overil správny beh a funkcionalita oboch, tak ako aj servera, tak aj útoku.

Všetky informácie ako pracovať so skriptom sú uvedené v README, ktorý je priložený ku skriptu. Na druhej strane verím, že je potrebné niektoré informácie uviesť pre lepšie pochopenie a objasnenie toho ako skript funguje.

Opis skriptu sa skladá z 3 časti:

- Serverová časť
- Útočníkova časť
- Použitie a príklady

Prvým segmentom je serverová strana, ktorá pokrýva 3 body. Najprv je vhodné vybrať server nachádzajúci sa na lokálnej sieti, aby nedošlo k nedorozumeniam, ak by sme mali potrebu testovať DoS útok vo verejnom internete. Pre optimálne použitie je treba sa zamerať na veľké súbory nachádzajúce sa na serveri a posielať požiadavku na ich stiahnutie pomocou HTTP GET požiadavky. V poslednom bode je potrebné a užitočné celú prevádzku na siete monitorovať a následne analyzovať.

Útočník, ktorý má k dispozícii skript, ktorý je vhodne spustiteľný na operačnom systéme Linux. Tento stroj musí mať prístup k nástroju *iptables*. Taktiež dôležitou časťou je interpreter Pythonu 3.8.x, ktorý je potrebný na spustenie skriptu, ktorý útok vyvolá. V neposlednej rade je nutné mať spojenie medzi oboma stranami, útočníkom a serverom.

Spustenie behu prebieha otvorením terminálu v prostredí Linux a premiestnením sa do priečinku, ktorý obsahuje skript. Pre vypísanie nápovedy je možné skript použiť s parametrom *-h* a to konkrétne *python3 slowdrop_mazanek.py -h*. Skript spustí škodlivý prenos a vypnutie útoku je možné pomocou kombinácie kláves CTRL+C.

Príklad použitia za účelom útoku môže byť napríklad:

python3 slowdrop_mazanek.py -url 'http://1.2.3.4/picture.jpg'

Defaultné nastavenia obsahujú tieto hodnoty:

- HTTP GET request je generovaný 20 vláknami (-t 20).
- Generovanie náhodných rozostupov medzi vláknami (-tgd 3).
- Čakacia doba 30 sekúnd v rámci jednotlivého vlákna pred generovaním nasledujúceho HTTP GET requestu (-gn 30).
- Zahadzovacie percento je nastavené na 40 (-d 0.40).

Všetky informácie uvedené o skripte sú uvedené v práci [7] za pomoci README súboru.

3 Simulačné prostredie

Táto kapitola pokrýva opis celého prostredia, ktoré je vytvorené za účelom simulácie a testovania diplomovej práce. Prostredie je pochopiteľne čo najviac priblížené k diplomovej práci Ing. Mazánka [7] a to hlavne z dôvodu kompatibility. Z tohoto dôvodu je opis tejto časti nevyhnutný a je treba sa jej venovať detailnejšie a bližšie ju opísať. Primárnym cieľom celej práce je simulácie a následná detekcia SlowDrop útoku. Aby tento cieľ mohol byť dosiahnutý, bolo potrebné vytvoriť vhodné simulačné prostredie, ktoré poskytuje možnosť testovania útoku v rôznych scenároch a variáciách. Nižšie v kapitole budú podrobnejšie vysvetlené jednotlivé aspekty prostredia, ktoré majú vplyv na výsledky testovania tak ako aj na výsledky celej práce.

Simulačné prostredie bolo vytvorené ako lokálna sieť, ktorá sa skladá z 3 hlavných prvkov, ktorými sú:

- Server
- Klient (Užívateľ) - skupina
- Útočník - jeden, alebo viac

Server poskytuje klientom možnosť interakcie za bežných podmienok, to znamená posielanie *GET requestu* webovej stránky. Do tohto procesu zasahuje nenápadne útočník, ktorý sa snaží pomocou SlowDrop útoku obmedziť funkčnosť servera a tak zablokovať prístup pre legitimného užívateľa.

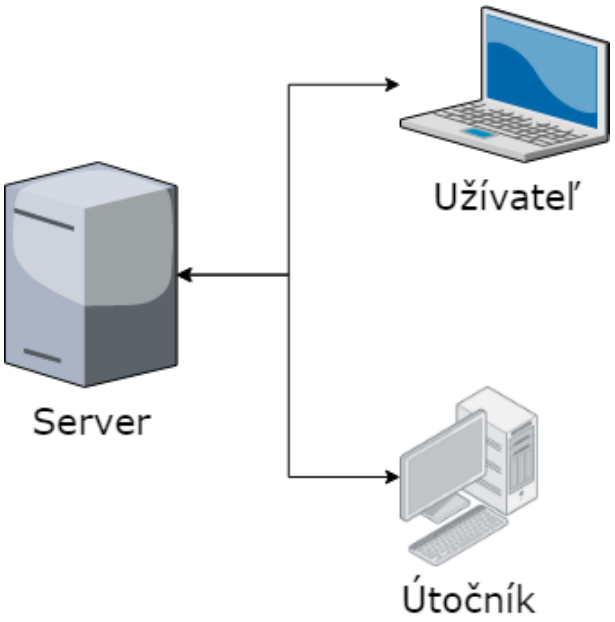
Vytvorené prostredie je prehľadnejšie zobrazené na obrázku 3.1, kde je vidieť, ktoré položky sa v prostredí nachádzajú. Z obrázku je ľahko a jednoznačne pochopiteľné, že útočník sa snaží infliltrovať medzi legitimného užívateľa a narušiť tak bežný chod servera.

3.1 Server

Ako už bolo spomenuté prvou a esenciálnou časťou celého prostredia je server. Server sám o sebe obsahuje viac funkcií, ktoré súvisia s touto prácou. Prvou funkciou je vytvorenie a nastavenie webového serveru ako takého, ktoré dáva následnú možnosť pre pripojenie klientov, ktorí môžu komunikovať so serverom a tváriť sa ako legitimní užívatelia, ktorých v simulácii aj predstavujú. Vďaka tomuto istému dôvodu je možné na server aj zaútočiť a tak otestovať, či prostredie funguje správne a taktiež otestovať silu útoku, ktorá je aplikovaná na server.

Server je v neposlednom rade vytvorený ako súčasť umelej siete, na ktorej bude prebiehať celkové testovanie a realizácia práce.

Zdroje potrebné pre beh serveru nie sú náročné. Pre domáce prostredie je zvolený stolový počítač, ktorý je popísaný nižšie v tejto kapitole. Na druhej strane nemôžu



Obr. 3.1: Simulačné prostredie

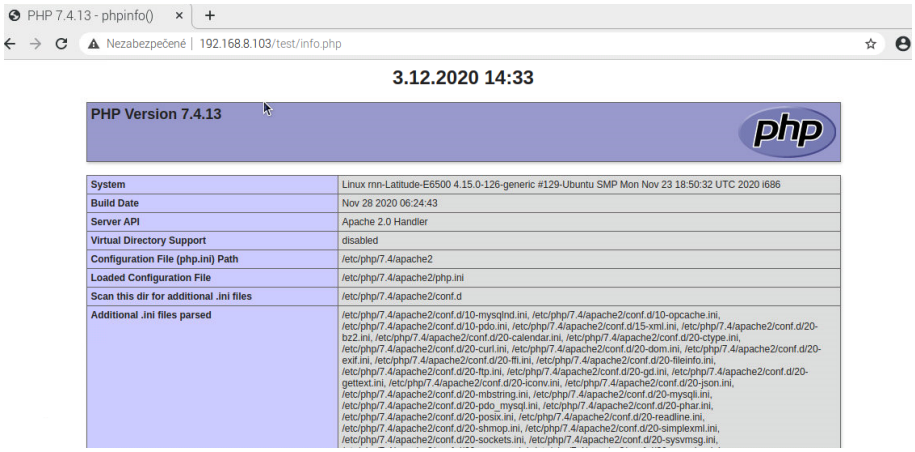
byť zdroje minimálne, pretože systém musí simulovať prevádzku obsahujúcu legitímnych užívateľov, preto serverový stroj musí byť schopný obslúžiť niekoľko desiatok klientov za bežnej prevádzky. Taktiež spojenie s klientmi musí byť stabilné a udržateľné. V ideálnom prípade by serverová časť bežala separátne na veľmi výkonnom počítači, ktorí by bol schopní obslúžiť nespočetné množstvo klientov a vytvoril tak bežnú sieť, ktorá je v dnešnej dobe očakávaným štandardom.

Prvou časťou hardwaru, ktorý server poskytuje je pamäť RAM. Pamäť RAM karta poskytuje 2GB pamäte. Toto množstvo by mohlo byť potenciálne rozšírené po prípadnej kúpe ďalšej pamäte RAM, aby mohol byť výkon serveru zvýšený. Nasledujúcou častou je procesor, ktorého celá názov je Intel Core 2 Duo CPU P8400 @ 2.26GHz x 2. Grafická karta je špecifikovaná ako Quadro NVS 160M/PCIe/SSE2. S týmito parametrami dokázal server obslúžiť viac ako 30 klientov bez jediného problému.

Server beží v domácom prostredí ako stolový počítač, na ktorom je nainštalovaný operačný systém Linux/Ubuntu 18.04.5 LTS. Špecificky sa jedná o 32 bitovú verziu tohoto systému s desktopovým prostredím GNOME 3.28.2

Na už spomínaný server bol nainštalovaný webový server, ktorý beží pod softwarom Apache/2.4.29(Ubuntu). Jedná sa o verziu serveru 5.7.32-0ubuntu0.18.04.1 s verziou protokolu 10. Znaková sada servera je nastavená ako UTF-8 Unicode (utf8). Keďže sa jedná o webový server je potrebné, aby bola zavedená aj databáza, v ktorej sa budú nachádzať jednotlivé súbory. Databázový klient bežiaci na serveri je libmysql - mysqlnd 7.4 s podporou programovacieho jazyka PHP 7.4.

Už spomínané súbory reprezentujú jednotlivé stránky, kde vo väčšej časti sa jedná o súbory typu php, css, html. Prehľadnejšia ukážka je zobrazená na obrázku 3.2, kde je vidieť webové prostredie na ktoré sú legitimní užívatelia pripojení.



Obr. 3.2: Jednoduché webové rozhranie

3.2 Klient

Pre vhodnú simuláciu celého SlowDrop útoku bolo potrebné namodelovať sieť. Táto sieť potrebuje obsahovať niekoľko legitimných užívateľov, ktorí sa správajú vo webovom prostredí normálne a nevykazujú žiadne známky potenciálneho útoku, aby nebolo jednoduché odhaliť útočníka.

Bolo zvolené domáce prostredie s obmedzenými zdrojmi. Z toho dôvodu je forma klienta reprezentovaná pomocou virtuálneho stroja, ktorý je pustený cez program Oracle VM VirtualBox. Pre prípravu siete bolo najprv potrebné vytvoriť virtuálny nástroj, ktorý beží pod operačným systémom Linux/Ubuntu [12]. Konkrétne sa jedná o voľne dostupný software verzie 20.04.1 Ubuntu. Z počiatku bolo treba vytvoriť dostatočný počet virtuálnych strojov na sieti. pripojený na webový server, kde každých pár sekúnd posiela GET request pre získanie informácií o webe. Týmto spôsobom je simulované prostredie pre legitímnych užívateľov pohybujúcich sa na serveri a je tak vytvorená možnosť začleniť medzi nich útočníka, o ktorom bude viac povedané v ďalšej podsekcii.

3.3 Útočník

Poslednou časťou v opise jednotlivých aspektov prostredia je útočník. Keďže sa v našom prípade jedná o nedistribuovaný DoS útok, tak útočník bude napádať server

iba z jedného počítača. Útok je reprezentovaný pustením skriptu, ktorý štartuje SlowDrop útok. Vďaka množstvu klientov sa dokáže útočník infiltrovať a tváriť sa ako bežný užívateľ, aby dokázal zahaliť svoj zámer a mohol zaútočiť na vybranú obeť.

4 Analýza dat

Esenciálnou častou práce pred implementáciou bola analýza dat. Analýza mala za úlohu zistiť na aké dáta sa zamerať. Skutočná identita dát spojená so SlowDrop útokom bola zisťovaná pomocou analýzy vo programe Wireshark.

Pre dosiahnutie čo najpresnejších výsledkov kybernetickej analýzy bolo spustených niekoľko variánt sieťového prenosu. Prvý scenár obsahoval DOS útok iba s jedným útočníkom a 25 bežnými klientmi na sieti. Pakety samotné neobsahovali žiadne špecifické dáta, podľa ktorých by bol útok rozoznateľný, avšak jednotlivá sekvencia pri útoku vykazovala zvýšený počet paketov obsahujúcich TCP Retransmission, ďalej uvádzané v tejto práci ako indikácia.

Druhým scenárom bolo sledovať bežnú komunikáciu bez útočníka a porovnať tak jednotlivé *pcap* súbory. V druhom prípade sa daná indikácia vyskytla iba v jednotkách, prípadne desiatkach na celý 30 minútový záznam. Táto skutočnosť viedla k tretiemu scenáru a indikovala hlbší pohľad na danú problematiku a výskyt tejto indikácie.

Posledným scenárom pre analýzu bolo spustenie viacerých útokov na jedno zariadenie. Tieto útoky bežali kontinuálne jeden vedľa druhého a výskyt TCP Retransmission výrazne narástol až na stovky za minútu podľa intenzity útoku a počtu útočiacich zariadení.

Charakteristiky a hlbší popis so zachytenými dátami budú uvedené v nasledujúcej kapitole. Taktiež bude pohľad zameraný na prostriedky a metódy, ktorými bol útok sledovaný a následne analyzovaný.

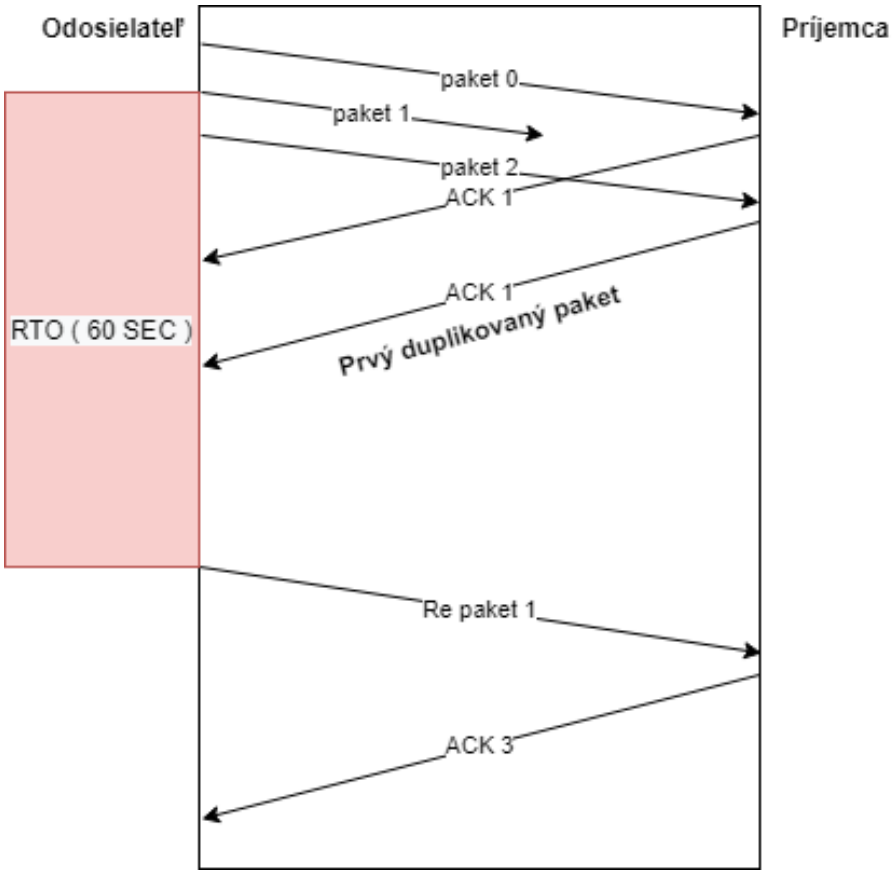
4.1 TCP Retransmission

Po analyzovaní dát bol rozpoznaný najvýraznejší znak, ktorý útok charakterizoval. Týmto znakom je správa s popisom TCP Retransmission, ktorej bude venovaná táto kapitola a jej detailnejší popis a dôvod výskytu.

Podľa oficiálnej dokumentácie [27] TCP používa retransmission timer, aby zaistil doručenie dát v prípade absencie spätnej väzby od vzdialeného prijímača. Tento časovač sa nazýva Retransmission timeout(RTO). SlowDrop útok sa pomocou RTO snaží udržať čo najdlhšie spojenie. Časovač je podľa dokumentácie nastavený na maximálne 60 sekúnd. V jednoduchosti povedané TCP Retransmission paket je taký packet, ktorý nebol potvrdený v časovom limite.

Inými slovami povedané, že časovač reprezentuje dobu 60 sekúnd, počas ktorých ak nepríde k obdržaniu potvrdenia o prijatí paketu(ACK), spojenie je ukončené. Tento jav a teda aj priebeh SlowDrop útoku neuzaviera TCP spojenia, vďaka tomuto

javu je vygenerovaná TCP Retransmission, podľa ktorej je možné následne útok v určitej miere detekovať. Jav TCP Retransmission je možné vidieť na obrázku 4.1.



Obr. 4.1: TCP Retransmission spojenie

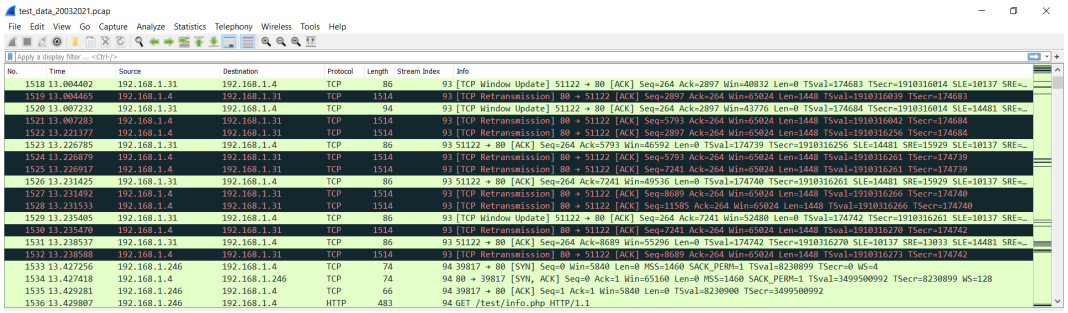
Táto charakteristika sa vyskytla v každom zaznamenanom *pcap* súbore, pre ktorý bol útok sledovaný. TCP Retransmission sa môže vyskytnúť aj v prípade klienta so slabým internetovým pripojením. Avšak v prípade väčšieho počtu útočníkov je hrozba ľahko rozpoznateľná a keďže sa jedná o cielený útok, je možné odsledovať vzor, alebo opakovateľnosť, ktorá útok charakterizuje.

Daná sieťová komunikácia, na ktorej útok sledujeme obsahuje výrazne vyšší výskyt TCP Retransmission tak, ako môžeme vidieť na obrázku 4.2

Z obrázku je jasne vidieť, že počas útoku je výskyt tejto správy v komunikácii útočníka násobne vyšší ako pre bežného užívateľa. Na obrázku je zobrazená iba niekoľko sekundová komunikácia z celého 30 minútového monitorovania.

4.2 Wireshark

Pre hĺbkovú analýzy dát som sa rozhodol použiť nástroj Wireshark. Je to široko používaný analyzátor na zachytávanie dátového toku. Poskytuje mnohé funkcie, ktoré



Obr. 4.2: Sekvencia TCP Retransmission čas útoku

pomáhajú sieťovému monitorovaniu. Nie len monitorovanie, ale tento software poskytuje omnoho viac pre detailnú analýzu zachytených dát.

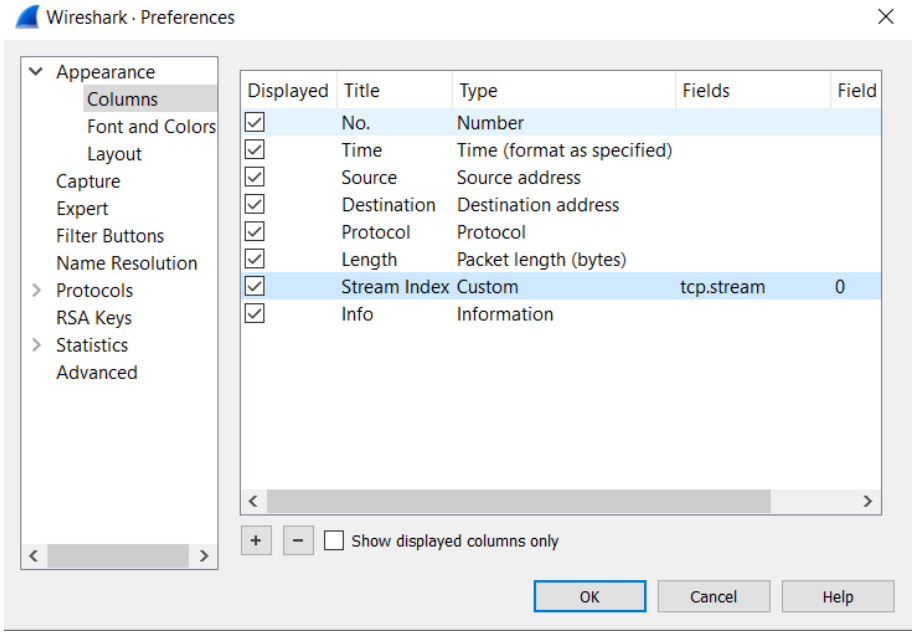
Po monitorovaní dátového toku sú informácie o jednotlivých paketoch zobrazené a je možné ich filtrovať podľa potreby užívateľa. Avšak dôležitou súčasťou je interpretácia dát a ich zobrazenie v jednoducho pochopiteľných formách. Wireshark v sebe obsahuje nespočetné množstvo už vytvorených funkcií, ktoré majú za úlohu dáta zobraziť pomocou grafov a štatistík. Tieto nástroje, ktoré sú implementované vo Wiresharku pomáhajú vývoju nových aplikácií v mnohých sieťových projektoch, taktiež ako aj v tejto diplomovej práci.

Wireshark bol kľúčovým nástrojom, cez ktorý boli dáta zachytené a následne analyzované. Pre podrobnú a efektívnu analýzu je potrebné prejsť veľké množstvo sieťou prenesených dát. Aby užívateľ vykonávajúci analýzu nemusel prechádzať paket po pakete, kde by táto činnosť mohla zabrať hodiny až dni pri masívnych vzorkách, je vhodné použiť jednotlivé filtre.

Wireshark umožňuje manuálne vytváranie jednotlivých filtrov k dátovému prenosu, ktoré užívateľ potrebuje pre efektívnu analýzu. Ak by sa uvažovalo nad scenárom, kde je cieľom nájsť komunikáciu smerujúcu na server, ktorý by mal adresu 192.168.1.123, stačilo by použiť jednoduchý filter *ip.dst==192.168.1.123*. Druhým príkladom môže byť *tcp.stream eq 93*. Tento filter má na starosti zobrazenie TCP spojenia s číslom 93 a vypustiť tak ostatnú komunikáciu. Tento konkrétny filter je v práci použitý a zobrazený na obrázku 4.8.

Pred samotným začatím analýzy je vhodné pridať stĺpec stream index, ktorý reprezentuje jednotlivé TCP spojenia vyskytujúce sa v dátovom toku. Túto informáciu o pakete je možné pridať cez možnosť Edit-Prefences alebo klávesovou skratkou Ctrl+Shift+P a pridaním nasledujúceho riadku v menu Column ako je vidieť na obrázku 4.3. Tento pridaný stĺpec je taktiež zahrnutý v exportovanom súbore, ktorý je potrebné pre ďalší postup.

Mimo excelentného zachytenia sieťovej komunikácie je možné o sieťovej komunikácia zaznamenať aj detailnejšie dáta, ktoré sa týkajú jednotlivých paketov. Jedná



Obr. 4.3: WireShark Edit-Preferences

sa napríklad o IP adresy, MAC adresy, flag-y, časové značky a mnohé ďalšie. Wireshark taktiež poskytuje pred nastavené zachytávania štatistík o dátovom toku ako napríklad dĺžka paketov, veľkosť paketov a mnohé ďalšie. Vo forme prehľadných grafov je možné zobraziť množstvo prenesených paketov v čase, prípade pridať filter pre zobrazovanie dát obsahujúce konkrétne adresy, flag-y či chyby. Wireshark poskytuje zobrazenie prednastavených grafov ako napríklad I/O graf, Flow Graf či TCP Stream Graf.

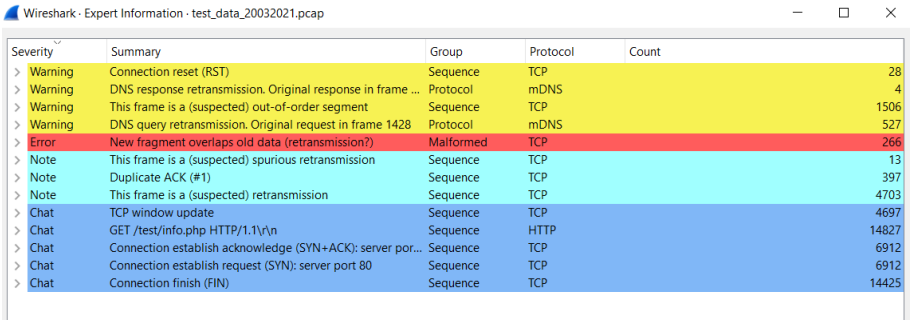
Ďalšou funkciou Wiresharku je Expertná analýza. Táto rozdeľuje dáta podľa ich závažnosti. Túto možnosť je možné nájsť v menu Analyze a následne Expert Information. Tento typ analýzy rozdelí dáta podľa závažnosti do 4 skupín:

- Chat
- Note
- Warning
- Error

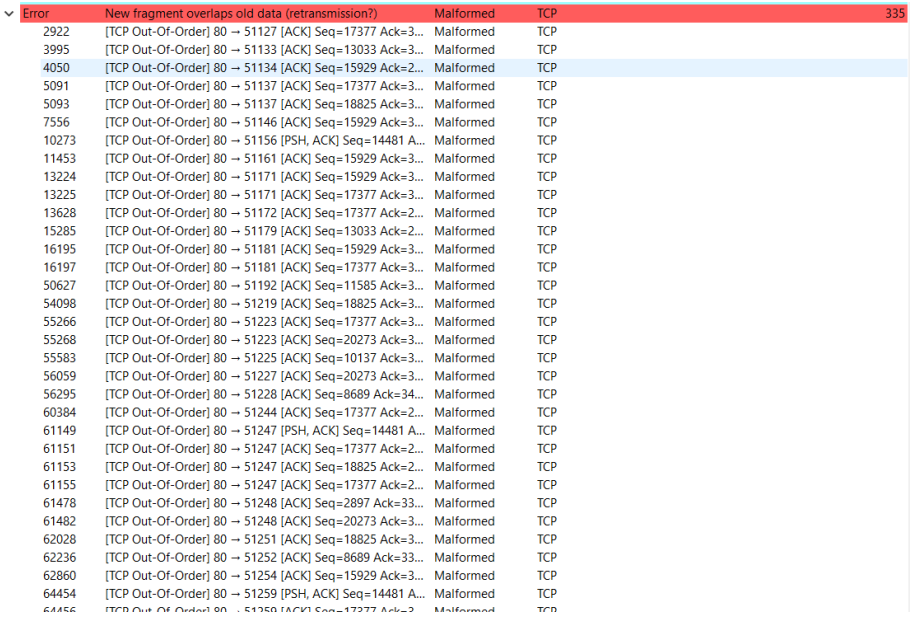
Obrázok 4.4, ukazuje rôzne typy správ, ktoré sa zobrazili v analyzovanom súbore.

Ako je možné vyčítať z obrázku prvé dva typy Chat a Note správy nie sú závažné a vo väčšine prípadov majú informačný účel. Tretím typom je Warning, ktorý upozorňuje na skutočnosť, že je potrebné zvýšiť pozornosť na tieto pakety, pretože niečo nie je úplne v poriadku. Posledným a najpodstatnejším typom pre účel práce boli správy typu Error, ktoré sú zobrazené na obrázku 4.5.

Nasledujúcim nástrojom na vykreslenie grafu a tak prehľadné zobrazenie dát,



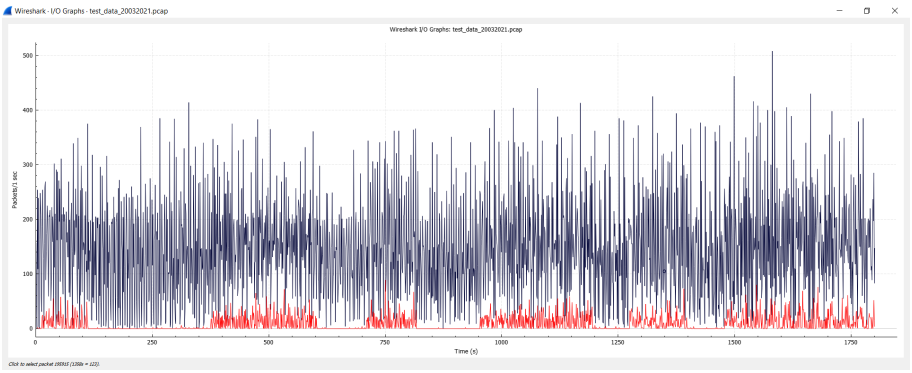
Obr. 4.4: Expertná analýza vo Wiresharku



Obr. 4.5: Expertná analýza chýb vo Wiresharku

ktoré markantne pomáha pri analýze je I/O graf, ktorý je zobraziteľný cez menu Statistics. Ako z názvu plynie jedná sa o zobrazenie grafom, pri ktorom je umožnené pridanie množiny špecifických dát do zobrazenia. V prípade obrázku 4.6 sa jedná o 2 množiny dát. Os X zobrazuje počet prenesených paketov, zatiaľ čo os Y reprezentuje 30 minútový časový interval v ktorom boli zaznamenávané dáta.

Graf teda zobrazuje počet prenesených paketov v čase. Prvá a to tmavo modrá množina dát zobrazená na grafe je množina všetkých prenesených paketov v čase, zatiaľ čo druhá červená skupina dát ukazuje počet TCP Retransmission chýb v dátovom toku.



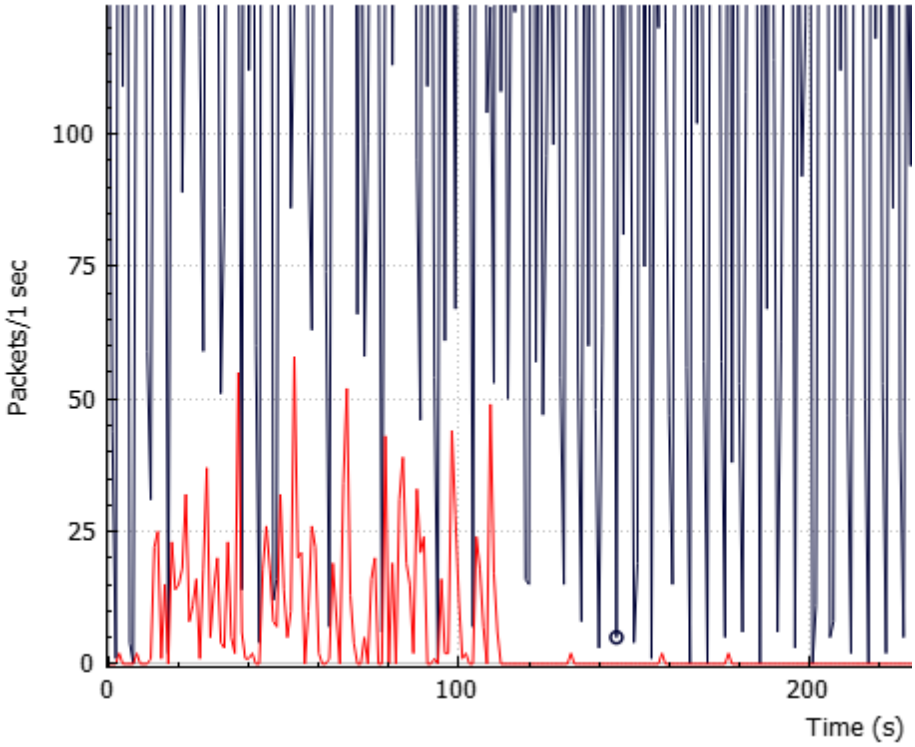
Obr. 4.6: I/O graf, 30 minútový segment

Z obrázku 4.6 vidieť, že útoky boli prerušovane spúšťané a počet chýb počas útoku a mimo útoku je jednoznačne rozlišiteľný. Počet paketov sa pohybuje približne od počtu 200 a dosahuje v niektorých prípadoch až 400 či 500 za jednotku času. Na druhej strane počet chýb v dátovom prenose bez útoku sa pohybuje maximálne v počte jednotiek.

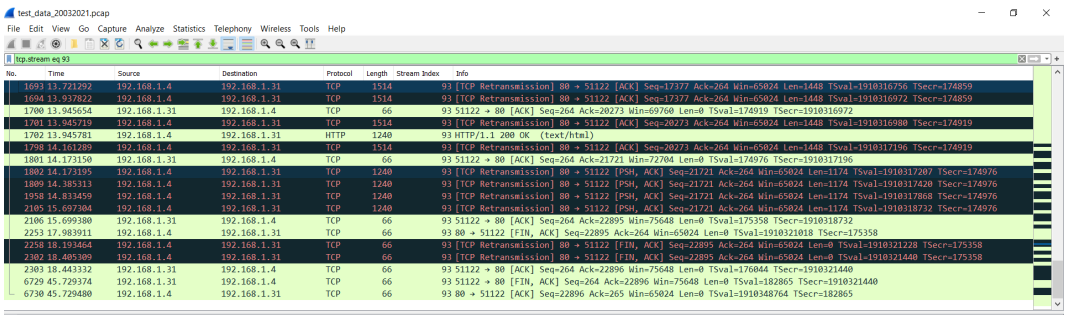
Zobrazenie 4.7 ukazuje bližší pohľad na vybraný krátky segment zo začiatku monitorovania. Jedná sa približne o 4 minútovú vzorku, z ktorej vidieť, že po začatí jediného útoku počet TCP Retransmission chýb vyskočí na hodnoty 20-25 za jednotku času. V niektorých prípadoch sa počet chybných paketov vyšplhá až cez hodnotu 50. Po 2 minútach je útok zastavený a chyby sa okamžite takmer minimalizujú.

Ďalší obrázok 4.8, ukazuje TCP spojenie 93, ktoré je odfiltrované pomocou tcp.stream eq 93, ktorý zobrazí iba pakety zaradené do TCP spojenia 93. Toto TCP spojenie jednoznačne reprezentuje SlowDrop útoku, kde predposledný paket a jeden pred ním majú rozdiel 30 sekúnd. Obrázok jasne odráža chovanie útočníka, ktorý sa snaží udržovať spojenie až do jednej minúty ako je uvedené v oficiálnej dokumentácii.

TCP Spojenie 93 je jedným z mnohých sledovaných spojení, na ktorých prebehla dátová analýza, ktorá viedla k praktickej realizácii a implementácii skriptu, ktorý už samotnú detekciu vykonáva.



Obr. 4.7: I/O graf, 4 minútový segment



Obr. 4.8: Wireshark, TCP spojenie 93

5 Detekcia útoku

Po teoretickom opísaní SlowDrop [1] útoku, objasnení si predchádzajúceho riešenia zahrňujúceho opis skriptu a hlbšej analýze dát, viedlo k motivácii vytvoriť túto diplomovú prácu. Ing. Mazánek mal za hlavný cieľ v práci vytvoriť SlowDrop skript. V tejto práci sa cieľ mení a je ním detekcia tohoto útoku.

Po úspešnom vytvorení lokálnej testovacej siete a odskúšaní prvých útokov ako napríklad SlowLoris za pomoci slowhttptestu [13] [14] a SYN flood [8], ktorými bolo zistené, že sieť funguje správne môžme prejsť na ďalšiu časť. V tejto kapitole je dôležité hovoriť o detekcii útokov. Keďže sa útočník tvári ako legitimný užívateľ, nie je jednoduché útok rozpoznať a tak sa voči nemu brániť. Špecifické pre SlowDrop útok je nízka detekovateľnosť.

Navrhované riešenia sú založené na troch mechanizmoch [15]. Prvým mechanizmom sú princípy detekcie signatúr. Mechanizmus spočíva v tom, že je neustále monitorovaný a analyzovaný prenos dát na sieti a následne prebieha sken týchto dát. Efektivita detekcie jednotlivých útokov spočíva v aktuálnosti databáze obsahujúcej jednotlivé signatúry. Tým pádom je potrebné zabezpečiť častú aktualizáciu týchto dát aby bolo možné odhaliť rôzne druhy útokov. Výhodou je, že známe útoky sú jednoducho detekovateľné bez problému, kde na druhej strane nové útoky nemusia byť zistené pokiaľ nie sú nové signatúry pridané do databázy.

Princíp detekcie signatúr prechádza binárne alebo textové vzory, ktoré sú vopred nastavené v signatúrach. Software monitorujúci dátový prenos hľadá vopred stanovené postupnosti, ktoré sú považované ako útočný dátový tok a ak nájde zhodu na útok upozorní a dátový tok zahodí. S počtom pribúdajúcich pravidiel a vyhľadávaných postupností sa prehľadávanie dátového toku značne spomalí, keďže je software nútený vyskúšať väčšie množstvo signatúr.

Taktiež tento mechanizmus poskytuje vytváranie vlastných pravidiel a vlastnej databázy týchto pravidiel. Odskúšanie, modifikácia a vytvorenie týchto pravidiel bude taktiež náplňou tejto práce.

Druhým postupom v riešení detekcie SlowDrop útoku je mechanizmus detekcie anomálií pomocou strojového učenia. Tento mechanizmus slúži na monitorovanie neštandardného dátového prenosu na sieti. Jedná sa konkrétne s monitorovanie dátového doku, ktorý prekračuje prahové hodnoty. Podľa [16] prahová hodnota je "Prahová hodnota je taková hodnota, která představuje určitou mez, jejíž překročením nebo vychýlením se, dojde k nějakému jevu."

Prekročením tejto hodnoty je dátový tok považovaný za neštandardný. Nevýhodou tohoto mechanizmu je náročnosť vhodne nastaviť prahové hodnoty tak, aby dokázali monitorovať sieť a rozpoznať potenciálny útok bez falošných poplachov.

Za jeden z najefektívnejších algoritmov strojového učenia sú považované neuró-

nové siete. Pre dosiahnutie vysokej detekcie SlowDrop útoku je potrebné stanoviť si dáta, s ktorými bude neurónová sieť pracovať. V kapitole 7 budú stanovené a opísané jednotlivé parametre sieťového prenosu, ktoré majú signifikantnú rolu v rozpoznaní útoku. Množiny dát, ktoré je potrebné nazbierať pre vytvorenie neurónovej siete sú rozdelené na dve časti. Dáta zahrňujúce obyčajné chovanie legitimných užívateľov na sieti a dáta, ktoré obsahujú aj útočníka pripojeného v sieti, ktorý iniciuje útočný dátový tok. Dáta budú analyzované Wiresharkom [17], ktorý je monitorovací program slúžiaci na analýzu dátového toku na sieti.

Tretím krokom bolo vytvorenie detekčného skriptu napísaného v jazyku Python. Tento skript má za úlohu detekovať útočníka a uviesť jeho IP adresu a tak odhaliť škodlivé zariadenie na sieti. Útok sa primárne sústredí na jednotlivú sekvenciu chybových paketov, ale aj na frekvenciu ich opakovania.

Ako už bolo spomenuté primárnym cieľom práce je detekcia SlowDrop útoku. Avšak pre splnenie toho bodu boli z počiatku zvolené metódy zahrňujúce programy, ktoré detekujú signatúry, vytvorenie neurónovej siete s účelom detekcie útoku v dátovom toku a vytvorenie detekčného skriptu. Tieto tri spôsoby budú hlavným aspektom praktickej časti práce.

- Detekcia signatúr pomocou programu Suricata
- Detekcia anomálií pomocou neurónovej siete
- Detekcia pomocou detekčného skriptu

6 Detekcia signatúr

Suricata [9] je voľne dostupný software, ktorého hlavným cieľom je detekcia hrozieb a útokov, ktoré prenikajú do systému. Taktiež ma za úlohu prevenciu voči týmto nástrahám internetu. Suricata bola vyvinutá Open Security Foundation firmou, ktorá spustila beta verziu na konci roku 2009 a následne prvé vydanie bolo následne vypustené o rok neskôr v Júly 2010. Pred inštaláciou bol najnovší software Suricata verzie 6 stiahnutý a následne nainštalovaný na server. S pomocou oficiálnej dokumentácie bola Suricata spustená a následnej otestovaná. Prvotný test funkcionality prebehol pomocou dvoch prístupov.

V prvom kroku testovania správneho behu Suricaty bol stiahnutý SlowLoris útok. Po úspešnej inštalácii bol útok spustený a Suricata ho jednoznačne rozpoznala.

Pre uistenie sa bolo ďalším krokom zavedenie rozličného testu, ktorý by mal byť jednoznačne identifikovateľní pomocou Suricata. V poradí druhý útok bol prevedený pomocou frameworku MSFConsole, ktorý reprezentuje spojenie slov MetaSploit Framework [28].

MetaSploit Framework má v sebe implementované rôzne typy útokov a framework je zameraný na využitie rôznych útokov, ktoré predstavujú hrozbu pre systém. Pri testovaní sa jednalo konkrétne o SYN flood útok, bol prevedený pomocou príkazu cez msfconsole po nastavení payloadu a Suricata ho okamžite a jednoznačne rozoznala. Útok bol Suricatou vyhodnotený ako: *"ET DOS Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST"*. Z popisu je viditeľné, že Suricata správne vyhodnotila SYN flood útok, kde vypísala na konzolu, že sa jedná o DoS útok.

Dve položky, ktoré treba nastaviť sú *RHOST* a *RPORT*. Prvou spomenutou je IP adresa cieľa na ktorý je útok poslaný, kde následne *RPORT* reprezentuje port, ktorý je potrebný uviesť pred začiatkom útoku. Zaujímavou položkou v zobrazenom payload-e na obrázku 6.1 je *SHOST*. Táto položka reprezentuje "Spoofable source address", alebo inak povedané adresu, ktorá bude uvedená ako útočiaca. Takýmto spôsobom sa útočník môže schovať a tváriť sa ako niekto iný a tak obmedziť jeho detekovateľnosť systémom.

```
msf auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   NUM                         no        Number of SYNs to send (else unlimited)
   RHOST                       yes       The target address
   RPORT      80               yes       The target port
   SHOST                       no        The spoofable source address (else randomizes)
   SNAPLEN    65535            yes       The number of bytes to capture
   SPORT                       no        The source port (else randomizes)
   TIMEOUT    500              yes       The number of seconds to wait for new data
```

Obr. 6.1: MSF Syn Flood payload

Napriek faktu, že Suricata je voľne dostupný software, jej využitie môže pomôcť serveru s detekciou a ochranou proti napadnutiu z inej strany. Detekcia je poskytnutá vo forme pravidiel, ktoré sú modifikovateľné a tým pádom aplikovateľné na rôzne scenáre, ktoré môžu vzniknúť.

6.1 Signatúry

Jednou z najpodstatnejších častí Suricata z pohľadu detekovania DoS SlowDrop útoku v našej práci sú tzv. Signatúry. Tie môžu byť jednoduchšie preložiteľné ako pravidlá, ktorými Suricata skenuje prevádzku na sieti a vracia tak informácie o tom, či sa jedná o útok, alebo inú hrozbu na sieti. Software ako taký poskytuje základnú sadu pravidiel, s ktorými Suricata dokáže pracovať priamo po inštalácii softwaru.

Oficiálne dokumentácia Suricata nám dáva podrobnejší pohľad na to ako s pravidlami pracovať a ako rozumieť jednotlivým častiam týchto pravidiel v prípade, že chcem vytvoriť vlastné.

Každé pravidlo, ktoré Suricata používa sa skladá z 3 častí:

- Akcia
- Hlavička
- Možnosti pravidla

Pre lepšie pochopenie môžme uviesť príklad, na ktorom budú jednoznačne vysvetlené jednotlivé časti pravidla. Ako príklad bude uvažovaná signatúra z oficiálnej dokumentácie [9]:

*"drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +..)"; flow:established,to_server; flowbits:isset, is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]3,/i"; reference:url, doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)"*

Signatúra slúži k monitorovaniu TCP toku, ktorý prebieha na domácej sieti. Dátový tok je smerovaný na server vo vonkajšej sieti a sleduje tok za účelov rozpoznania potenciálneho trojského koňa. Pri úspešnom zachytení takéhoto škodlivého dátového toku pakety zahodí. Pravidlo je postupne opísané detailnejšie a je popísané, kam ktorá časť patrí, do ktorej časti a prípadne, aké sú ďalšie alternatívy pre úpravu jednotlivých signatúr.

Prvou častou je "drop", ktorá je považovaná za akciu. Akcie platné pre Suricatu sú:

- alert - vygeneruje upozornenie
- pass - zastaví ďalšie prehľadávanie paketov
- drop - zahodí paket a vygeneruje upozornenie

- reject - pošle odosielateľovi RST/ICMP paket o tom, že paket nedorazil
- rejectsrc - rovnako ako reject
- rejectdst - pošle príjemcovi RST/ICMP paket o tom, že paket nedorazil
- rejectboth - pošle obom stranám RST/ICMP paket o tom, že paket nedorazil

Ďalšou časťou je *"tcp $HOME'_NET any -> $EXTERNAL_NET any"*, ktorá je považovaná za hlavičku. Prvá časť hlavičky v našom príklade ako *"tcp"* je veľmi jednoducho zrozumiteľná. Jedná sa samozrejme o protokol, kde sa dá uvažovať o 4 základných možnostiach ako sú:

- TCP
- UDP
- ICMP
- IP

Ďalšou častou po protokole je zdroj a cieľ. V našom príklade je to: *$HOME'_NET* a *$EXTERNAL_NET*. Prvým v poradí je zdroj, ktorý ide zapísať v rôznych formátoch. Zahrnuté sú skupiny, jednotlivé adresy a taktiež je možnosť zapísať adresy, ktoré sú vylúčené pomocou znaku !. Taktiež je možné použiť skupiny IP, alebo jednoducho zablokovať skupinu IP adries. Pre vytvorenie skupiny adries je použité [].

Ak je potrebné pustiť prevádzku na všetky IP adresy okrem 1.2.3.4, toto opatrenie je zabezpečené pomocou príkazu !1.2.3.4 a Suricata vyhodnotí pravidlo tak, že nepustí prevádzku, ktorá je spojená s IP adresou 1.2.3.4.

Ak sa má jednať o skupinu adries je potrebné použiť napríklad [1.2.3.4, 11.12.13.14, !21.22.23.24], toto pravidlo hovorí, že sú akceptované IP adresy 1.2.3.4, 11.12.13.14 a je vylúčená adresa 21.22.23.24.

Tiež je potrebné uviesť príklad pre rozsah IP adries. Kde [1.1.1.0:1.1.1.100] udáva rozsah od adresy 1.1.1.0 do 1.1.1.100.

Obidva typy tak ako IPv4 aj IPv6 je podporovaná a použiteľná pri vytváraní vlastných pravidiel. Po IP adresách logicky nadväzujú porty, kde v našom vyššie uvedenom príklad je použité slovíčko *any*, ktoré reprezentuje možnosť ľubovolného portu. Pri portoch sa používa rovnaká syntax zápisu, kde znak ! vylučuje port, [] reprezentuje skupinu portov a znak : signalizuje, že sa jedná o rozsah.

Poslednou časťou hlavičky je direction, teda smer. Sú len tri možnosti ako uviesť smer a to <-, -> alebo <>. Symbol <> reprezentuje obojsmernú prevádzku na sieti. Týmto spôsobom určené, o ktorý smer sieťovej prevádzky sa jedná.

Poslednou časťou signatúr sú možnosti pravidiel. Medzi možnosti môže patriť veľké množstvo variant, ktoré môžu byť špecifikované pre konkrétnu signatúru. Jednotlivé časti, ktoré sa špecifikujú na konkrétne veci sa nazývajú *KeyWords*, alebo kľúčové slová. Všetky, ktoré Suricata podporuje sa nachádzajú v dokumentácii oficiálneho programu. Pre hlbšie objasnenie sa bude pozornosť venovaná tým, ktoré sú uvedené v príklade vyššie.

Prvou hlavnou skupinou kľúčových slov sú *Meta KeyWords*. Do tejto kategórie spadá napríklad aj *message* , ktorá je použitá v príklade. Konkrétne *msg:"ET TROJAN Likely Bot Nick in IRC (USA +..)"* je vlastne správa, ktorá je vypísaná na konzolu v textovej forme a indikuje informácie o signatúre a možné upozornenie.

V kategórií Meta KeyWords sa nachádzajú aj ďalšie časti, ktoré je vhodné spomenúť, jedná sa konkrétne o *sid* a *rev*. *Sid* reprezentuje ID pridelené signatúre, tým pádom je jasne určená každá signatúra, keďže ID musí byt unikátne a nesmie sa zhodovať s iným. S *sid* súvisí *rev*, kde *rev* je skratka pre revision, ktorá opisuje "verziu"signatúry. Zrozumiteľnejšie je pochopiteľný *rev* takým spôsobom, že v prípade ak je signatúra upravená, číslo *rev* sa zvyšuje na o jedno vyššie. V príklad je použité sid:2008124, to znamená žiadna iná signatúra nie je identifikovateľná podľa tohto čísla a rev:2 hovorí o tom, že signatúra bola upravená raz.

Ďalšia časť Meta KeyWord, ktorá sa nachádza v príklade je classtype, v ktorom je daná informácia o klasifikácii pravidla. Jednoduchým príkladom je *classtype:trojan-activity*, kde je jasné, že sa jedná o útok typu trojan.

Poslednou časťou Meta Keywords je referencia. Tá ukazuje, kde sa nachádza informácia o signatúre a akým problémom sa zaoberá.

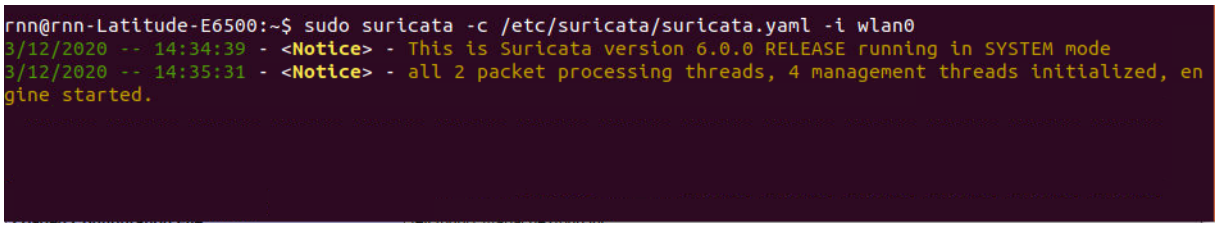
Ako sme si už spomenuli pravidlá Suricaty rozumejú veľkému množstvu kľúčových slov. Po prvej hlavnej kategórii sme si opísali časti Meta KeyWords a môžme sa postupne presunúť na ďalšie špecifickejšie kategórie. Uvedený príklad spomína kľúčové slovo Flow.

Do tejto kategórie spadajú napríklad*flow:established, to_server; flowbits:isset, is_proto_irc*. Prvá časť obsahujúca *flow:* signalizuje ustanovené spojenie medzi serverom a klientom, kde prevádzka je smerovaná práve na server. V druhej časti je zapnutý tok bitov, keďže Suricata ako taká túto možnosť podporuje.

V tejto časti boli prebraté jednotlivé kategórie kľúčových slov, ktoré obsahoval uvedený príklad. Celkový počet kategórii je omnoho vyšší, ale pre účely tejto práce ich nie je potrebné spomínať. Pre prípadnú potrebu je možné naštudovať detailnejšie ďalšie časti v oficiálnej dokumentácii Suricaty [9].

6.2 Testovanie signatúr

Signifikantnou časťou celej práce bolo použitie programu Suricata na detekciu DoS SlowDrop útoku. Na začiatku testovanie bola Suricata nainštalovaná na server, ktorý beží na už detailne popísanom serveri v kapitole 3.1. Po spustení Suricaty pomocou príkazu *sudo suricata -c /etc/suricata/suricata.yaml -i wlan0*. Po potvrdení daného príkazu začne software monitorovať čo sa sieti deje a zobrazí sa nasledujúca správa o úspešnom behu Suricaty, ktorá je viditeľná na obrázku 6.2. Z tohoto dôvodu je potrebné z iného stroju iniciovať útok a sledovať ako sa Suricata bude správať.



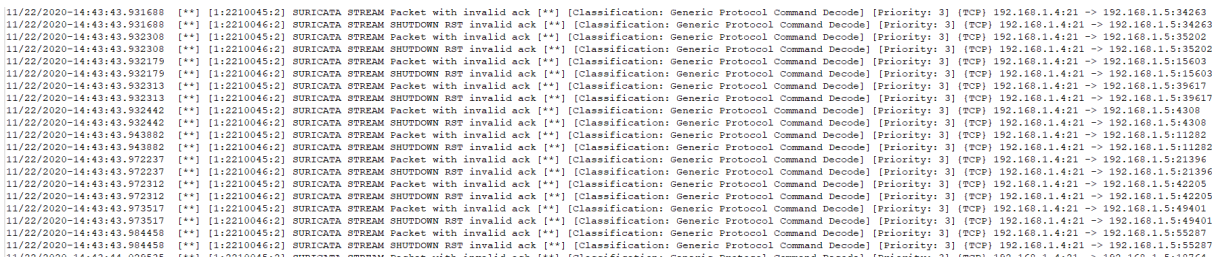
Obr. 6.2: Zapnutie Suricaty

Testovanie bolo zahájené použitím štandardizovaných signatúr, ktoré Suricata ponúka. Z predpokladu, že SlowDrop útok má minimálnu detekovateľnosť sme usúdili, že bude potrebné vyskúšať viaceré útoky, aby sme boli presvedčení o správnom fungovaní Suricaty na serveri.

Suricata ponúka dva typy signatúr:

- Štandardizované signatúry
- Umelo vytvorené signatúry

Prvými spomenutými sú signatúry štandardizované, ktoré Suricata ponúka v inštalačnou balíčku a je ich možné použiť priamo po inštalácii softwaru. S minimálnou úpravou default nastavení bol otestovaný SlowLoris a SYN flood útok. Obidva útoky boli okamžite detekované, ako je možno vidieť na obrázku 6.3. V log súbore je zobrazené okamžité a neustále oznamovanie o útoku. Poslednou a to najdôležitejšou častou bolo spustenie SlowDrop útoku a sledovanie Suricaty. Ako bolo očakávané, SlowDrop útok nebol rozpoznaný štandardizovanými signatúrami. Avšak neúspešný výsledok nám dáva priestor pre ďalšie testovanie, ktoré bude spočívať v upravení a vytváraní signatúr.



Obr. 6.3: Výpis Suricaty po útoku SYN Flood

6.2.1 Výsledky

Z opisu SlowDrop [1] vieme, že idea skrývajúca sa za útokom je opakované posielanie požiadavkov na server. Tieto požiadavky sú smerované na špecifický zdroj, ktorý by mal byť väčšieho dátového formátu a tak vyvolať fragmentáciu, ktorú bude

útočník neustále vyžadovať a zahadzovať obdržané pakety. Vďaka TCP hlavičke sa dokážeme zamerať na fragmentáciu a to pomocou Flag-s, kde je možné určiť, či sa má jednať o fragmentáciu alebo nie.

Flag-s v TCP hlavičke obsahujú 3 bity. Prvý bit je rezervovaný. Druhý bit reprezentuje fragment-ovanie ak je hodnota bitu 1. Ak táto situácia nastane paket obsahuje *Flag-s: 0x4000, Don't fragment*. Posledným spomínaným bitom je fragmentácia, kde v prípade kladného bitu Flag-s vyzerá ako *Flag-s:0x2000, More Fragment-s*.

Fragment-ovanie nám ešte dáva ďalšiu špecifikáciu, ktorá môže byť zahrnutá v signatúre. Touto špecifikáciou je Fragment offset. Táto položka má hodnotu 0, ak ku fragmentácii nedochádza. Týmto spôsobom je možné hľadať všetky iné pakety v ktorých ku fragmentácii došlo.

Pre vytvorenie niekoľkých signatúr boli použité hlavne špecifikácie z oblasti fragmentácie. Jedná sa konkrétne o R - rezervovaný bit, D - fragmentácia, M - viac fragmentov. Pre účely testovania bola použitá posledná možnosť s monitorovaním prichádzajúcich fragmentov.

Ďalšou častou signatúry je fragoffset. Táto časť určuje počiatočnú pozíciu dát vo vzťahu k originálnemu paketu. Prvý fragment má frag offset nastavené ako 0, keďže sa jedná o počiatočný paket a dáta začínajú na prvej pozícii. V testovaných signatúrach bolo cieľom nájsť fragmentované pakety a preto bolo použitá možnosť *fragoffset:[!|<|>] <number>*. V signatúrach bolo konkrétne použitá časť *frag-offset: !0*, ktorá špecifikuje, že sa jedná o offset, ktorý je iný od nuly. Pre testovanie fragmentácie boli použité 3 signatúry:

1. Prvou uvedenou je signatúra, ktorá monitoruje dátový tok zo serveru na klienta za účelom hľadania paketov, ktoré signalizujú prichádzajúcu fragmentáciu.

alert tcp $HOME_NET any -> any any (msg:"Potential SlowDrop detected"; fragbits: M+; flow:to_client, established;)

2. Druhú signatúru v poradí charakterizuje zameranie sa na offset ako taký, kde je očakávaným výsledkom zobrazenie fragmentovaných paketov.

alert tcp $HOME_NET any -> any any (msg:"Potential SlowDrop detected"; fragoffset: !0; flow:to_client, established)

3. Posledná spomenutá obsahuje kombináciu prvých dvoch, to znamená fragmentáciu, ale aj offset rozdielny od 0.

alert tcp $HOME_NET any -> any any (msg:"Potential SlowDrop detected"; fragbits: M+; fragoffset: !0; flow:to_client, established)

Ani jedna z troch uvedených signatúr neprišla k úspešnej detekcii SlowDrop útoku a to z dôvodu, že aj legitimný užívateľ, môže požadovať rovnaké dáta bez

zámeru blokovania komunikácie. Ak táto situácia nastane systém signalizuje útok, ktorý je ale v skutočnosti falošným poplachom označujúcim legitimného užívateľa ako útočníka. Taktiež žiadna z štandardných signatúr, ktoré poskytuje software Suricata nebola schopná SlowDrop útok detekovať a dodať tak pozitívne výsledky vedúce k odhaleniu útoku a útočníka.

Suricata nebola schopná detekovať útok ako taký a nie je vhodný nástroj pre detekciu tohoto útoku a to z dôvodov, že hlavnou charakteristikou je frekvencia jednotlivých TCP Retransmission správ. Suricata dokáže upozorniť na jednotlivé chyby, ale nie ich vyskytujúcu sa frekvenciu. Suricata by bola schopná odhaliť jednotlivé TCP Retransmission správy, ale zároveň tak dávať planý poplach, keďže sa nemusí jednať o útok, ale iba užívateľa so slabým internetovým spojením. V konečnom dôsledku takýto podrobný zápis pravidiel Suricata neumožňuje a preto nebola úplne vhodným nástrojom pre realizáciu riešenia v tejto diplomovej práci.

7 Detekcia anomálií

Vhodnou metódou pre detekciu anomálií v sieťovej prevádzke je použitie neurónových sietí [16][29][30][31]. Detekcia anomálií je založená na získaní a porovnaní dvoch skupín dát. Jedná sa konkrétne o dáta zozbierané z bežného sieťového prenosu a o dáta zo sieťového prenosu, na ktorom prebieha útok. Kľúčovým bodom je dáta porovnávať a získať tak charakteristiky, ktoré hovoria o rozdielnostiach medzi jednotlivými vzorkami. Vzorky budú následne slúžiť ako vstupné dáta, podľa ktorých sa neurónová sieť bude učiť rozoznať bežnú komunikáciu od útoku.

Neurónové siete je potrebné najprv teoreticky popísať a zadefinovať. Problematike neurónových sietí sa venujú aj práce [32][33]. Z predchádzajúcich dvoch kapitolách som odvodil, že pre získanie lepších výsledkov je potrebné hlbšie sa venovať neurónovým sieťam a to nie len v teoretickej rovine, ale aj praktickej rovine. Praktická časť diplomovej práce bude zameraná na implementáciu neurónovej siete, ktorej bude viac venovaná pozornosť v neskoršej fáze práce. Predtým ako sa bude opísaná implementácia a výsledky, ktoré neurónová sieť priniesla, bude objasnené, čo je to neurónová sieť a návrh na jej vytvorenie.

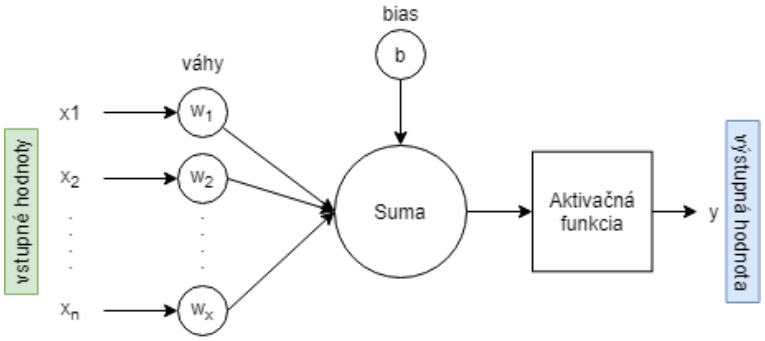
7.1 Popis neurónovej siete

Podstatnou časťou neurónovej siete je neurón ako taký. Neurón sám o sebe v prvom rade reprezentuje a odkazuje na biologický formát, ktorý sa nachádza v ľudskom tele. Pre nás je ale podstatnejší neurón, ktorému je pridaný prívlastok umelý. Autor práce [16] popisuje umelý neurón ako základnú stavebnú jednotku celej neurónovej siete. Neuróny sú medzi sebou pospájané a pomocou signálov medzi sebou komunikujú. Týmto spôsobom je vytvorená celá sieť.

V našom prípade je potrebné si objasniť jednotlivé premenné, ktoré hrajú rolu v neurónovej sieti. Konkrétne sa jedná o vstupné a výstupné parametre, na ktoré nadviaže aktivačná funkcia. Model umelého neurónu je zobrazený na obrázku 7.1.

V ľavej časti obrázku viditeľné, že premenná x_1,x_n reprezentuje vstup. Hodnoty, ktoré môže táto premenná nadobudnúť sa pohybuje v množine reálnych čísel R. Následne do procesu vstupuje w_1,w_n, ktorá reprezentuje váhu. Váha slúži na ovplyvnenie vstupnej hodnoty, ktorá je ňou vynásobená. Σ je funkcia suma, ktorá vyjadruje súčet všetkých súčinov medzi vstupnou hodnotou a váhou.

Premenná **G** je považovaná za výsledok predchádzajúcej sumy. Tento výstup následne pokračuje a je použitý ako vstup do tzv. Aktivačnej funkcie ϕ. Dôležité je spomenúť a vysvetliť aj slovné spojenie aktivačná funkcia. Toto spojenie opisuje funkciu, ktorá spracováva výstupné hodnoty G a následne posiela výsledok



Obr. 7.1: Štruktúra umelého neurónu

ďalej do neurónovej siete. Týmto spôsobom sú jednotlivé neuróny pospájané a tak je možné iniciovať komunikáciu v celej neurónovej sieti.

Pre pochopenie ako aktivačná funkcia funguje je vhodné spomenúť ďalšie dve premenné ktorými sú **P** a o_j. Prvou uvedenou je P, ktorá je celý názvom pomenovaná ako Prahová hodnota. Voľnými slovami vyjadruje hodnotu od ktorej závisí, či je neurón aktivovaný alebo nie. Odbornejšie povedané, prahová hodnota porovnáva hodnotu z výstupu Sumy, čiže hodnotu G porovnáva s prahovou hodnotou. Ak je **G** vyššie ako prahová hodnota umelý neurón je aktivovaný. V prípade aktivácie sa jedná o poslednú uvedenú premennú a to o_j. Táto premenná reprezentuje hodnotu výstupu celého neurónu, ktorá následne pokračuje na vstup ďalšieho. V prípade že hodnota **G** je nižšia ako prahová hodnota nastane inhibícia neurónu, alebo jednoducho povedané, nestane sa nič.

7.2 Realizácia neurónovej siete

Po ukončení testovania použitím Suricata s pomocou signatúr a následnej dátovej analýze, ktorá poskytla hlbší pohľad na dáta a ich štruktúru sa prešlo na samotnú realizáciu neurónovej siete. Predtým ako budú uvedené informácie k jednotlivým častiam neurónovej siete a ich detailnejší popis je potrebné si objasniť niektoré esenciálne pojmy týkajúce sa neurónových sietí.

Neurónové siete sú v dnešnej dobe hlavným detekčným nástrojom pri vyhľadávaní anomálií v sieťovom toku. Preto táto metóda bola navrhnutá ako jedna z častí celkového riešenia. Získanie a následné parsovanie dat s opisom jednotlivých softwarových technológií a použitie knižníc, ktoré boli použité pri tvorení a testovaní siete budú opísané v nasledujúcich kapitolách. Po otestovaní a získaní výsledok z neurónovej siete bol navrhnutý a vytvorený Python skript s účelom detekcie útočníkov v dátovom toku.

V tejto kapitole bude obsiahnutý popis jednotlivých technológii a taktiež ich použitie v diplomovej práci. Pohľad bude smerovaný aj na dáta a opísaný bude celkový proces ktorým si dáta prejdú od zachytenia cez spracovanie až po ich výslednú formu, ktorá bude prezentovaná ako výsledky rozpoznania anomálii v sieti.

Na konci kapitoly je pozornosť taktiež venovaná samostatnému skriptu, ktorý má za úlohu detekovať útočníka ako takého a uviesť jeho IP adresu. Druhé z uvedených riešení vedie k návrhu a teoretickým zmienkam o tom, ako daný útok riešiť a brániť sa voči nemu. Taktiež je spomenutá filtrácia a blokovanie IP adries po získaní výsledkov z detekčného skriptu.

7.2.1 Technológie

Pred vybudovaním neurónovej siete je potrebné si stanoviť programovací jazyk na ktorom bude sieť postavená. Možností bolo niekoľko ako napríklad: Python, Java, Julia, Haskell či Lisp.

Z pohľadu vedúceho programovacieho jazyka vo svete umelej inteligencie a neurónových sietí sa hovorí o Pythone. Hlavným dôvodov sú knižnice ako NumPy, Pandas, Pybrain a SciPy, ktoré pomáhajúce urýchliť vývoj jednotlivých projektov. V konečnom dôsledku z dôvodu rozšírenia neurónových sietí a framework-ov postavených práve na Pythone bolo zvolené toto riešenie pre realizáciu a implementáciu neurónovej siete v tejto diplomovej práci.

Ďalším bodom bolo nájsť vhodné knižnice pracujúce s neurónovými sieťami. Na internetovom trhu je ľahko dohľadateľných niekoľko voľne dostupných software medzi ktoré patrí napríklad [25]:

- TensorFlow
- Keras
- PyTorch
- Theano
- DL4J
- Caffe
- Chainer
- Microsoft CNTK

Z mnohých knižníc boli vybraté práve TensorFlow a Keras. Ku obom knižnicám budú uvedené podrobnejšie informácie.

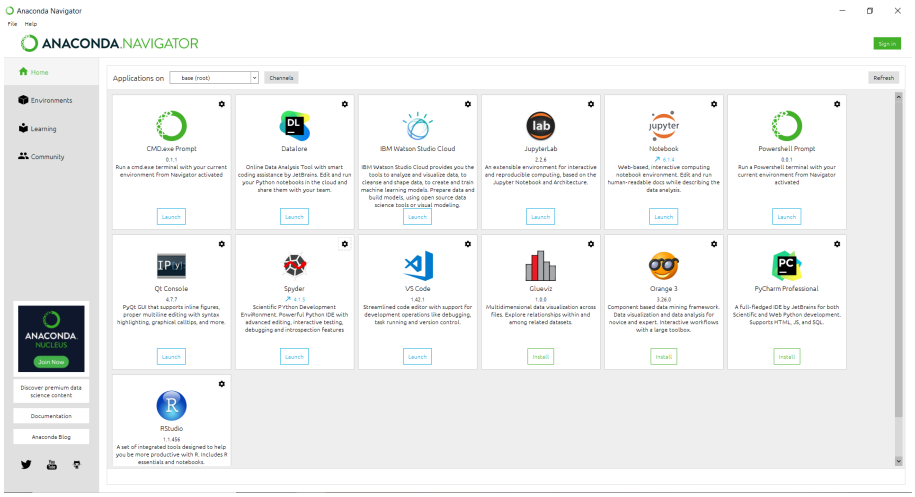
Prvým spomenutým je Tensorflow [21]. TensorFlow bol vyvinutý v Google a podporuje jazyky Python alebo R. Výhodou TensorFlow je, že dokáže zobraziť grafy dátového toku, ktoré su spracovávané. Tento bod je dôležitý, pretože pri vytváraní siete je možné sledovať ako dáta prúdia cez neurónovú sieť. TensorFlow taktiež obsahuje časť zvanú TensorBoard, ktorá sa zameriava na vizualizáciu dát. Taktiež je možné

použiť vizualizačné balíčky, ktoré poskytuje Python alebo R. TensorFlow knižnica ako taká poskytuje mnoho prostriedkov pre jednoduché budovanie projektov spojených so strojovým učením a podporuje tak budovanie experimentov pre výskum v tejto oblasti.

Druhou použitou knižnicou je Keras [22], ktorý sa stal jednou z najrýchlejšie rastúcich knižníc, ktorá sa zaoberá neurónovými sieťami. Keras je napísaný v programovacom jazyku Python a dôležitými znakom je, že dokáže pracovať nad niektorými knižnicami ako napríklad TensorFlow. Keras používajú spoločnosti ako Microsoft Research, NASA, Netflix, alebo Cern. Ďalšími vlastnosťami Kerasu sú napríklad prehľadné užívateľské rozhranie, v ktorom je možné sledovať prípadné chyby v programe, alebo jednoduché pridávania nových modulov, ktoré pomáhajú pri pokročilom výskume.

Obe spomenuté knižnice s voľne stiahnuteľné a pracujú spoločne vo frameworku Anaconda [18], ktorý je všeobecne známy a rozšírený so svete Pythonu. Anacodna je voľne dostupný software postavený na programovacom jazyku Python, ktorý obsahuje niekoľko modulov špecializujúcich sa na rôzne účely. Pre realizáciu neurónovej siete bol najzaujímavejším modelom Jupyter Notebook a Spyder3, ktoré reprezentujú prehľadné užívateľské rozhrania s možnosťou vytvárania skriptu a následného testovania.

Obrázok 7.2 ukazuje jednotlivé moduly, ktoré sú implementované v súčasnej verzii Anacondy. Pre ich používanie je nutné modul v prvom rade stiahnuť. Prvotné stiahnutie a inštalácia automaticky zahrňuje všetky knižnice, ktoré sú nevyhnutné pre beh daného modulu.

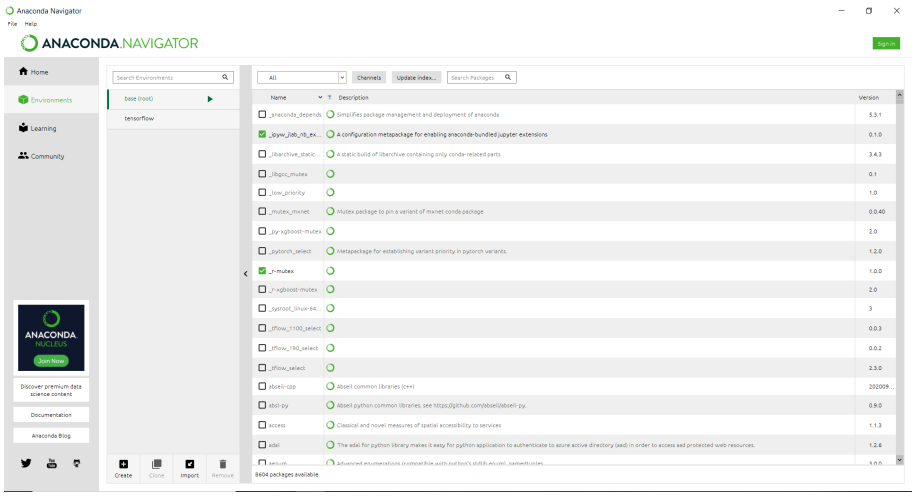


Obr. 7.2: Užívateľské rozhranie frameworku Anaconda

Užívateľské prostredie Anacondy poskytuje taktiež jednoduché stiahnutie vyžadovaných knižníc s následným importovaním pre jednotlivé projekty. Mimo štan-

dardných knižníc bežne používaných pri programovaní ako *sys* pre systémové funkcie, výstupy a parsovanie jednotlivých argumentov, alebo knižnica *csv* pre prácu s týmto typom súborov bola taktiež použitá knižnica timedelta, ktorá dokáže spracovávať časové formáty premenných a robiť nad nimi operácie ako sčítania, odčítanie času a podobné. Timedelta knižnica bola použitá pri vytváraní detekčného skriptu.

Na obrázku 7.3 je vidno prehľadné a jednoduché užívateľské rozhranie pre sťahovanie jednotlivých knižníc, ktoré sú spojené s konkrétnym projektom vytvoreným v Anaconde.

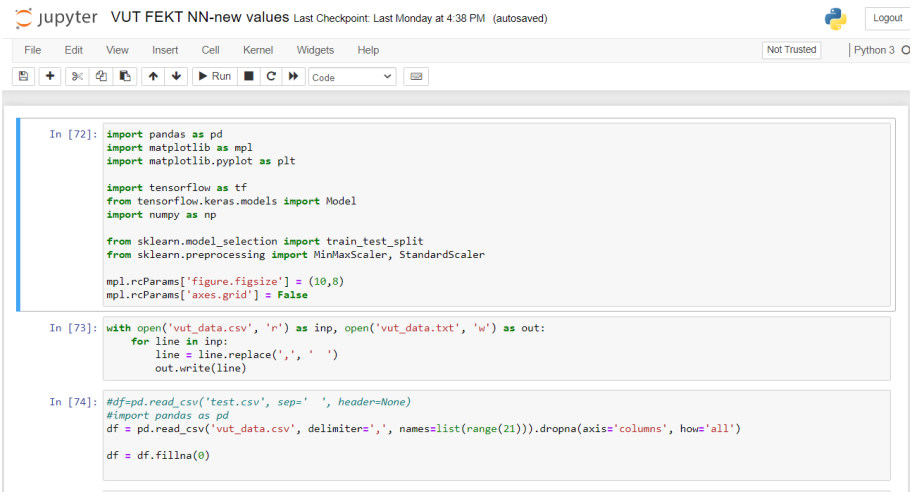


Obr. 7.3: Užívateľské rozhranie frameworku Anaconda Environments

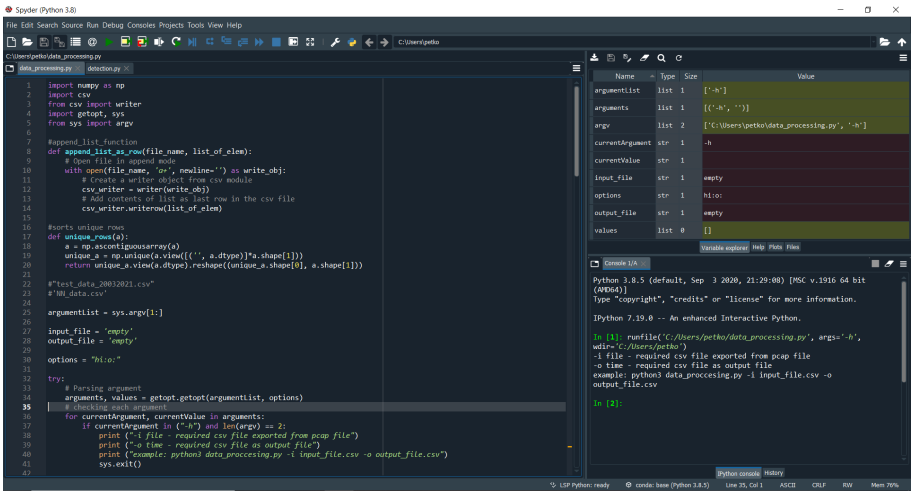
Modul, v ktorom prebehla implementácia neurónovej siete sa volá Jupyter Notebook [19]. Inými názvami Jupyter, alebo JupyterLab je webovo orientované prostredie pre vývoj. JupyterLab pomáha konfigurovať užívateľské rozhranie pre podporu široko zameraný projektov v oblasti vedy, vedeckých výpočtov a strojového učenia.

Pri samotnej implementácii programu, ktorá prebieha v užívateľskom rozhraní Jupyteru je potrebné zavolať stiahnuté knižnice pomocou príkazu *import knižnica* ako je možné vidieť na vrchu obrázku 7.10

Posledným použitým modulom z Anacondy je Spyder 3[20]. Spyder 3 je voľné dostupné prostredie pre vedecký vývoj napísané v Pythone. Modul je vytvorené pre vedcov, inžinierov a d8tový analytikov. Poskytuje kombináciu pokročilého editovania, analýzy, debuggovania a profilovania komplexného vývojového nástroja. Taktiež obsahuje vedecký balík na prieskum dát, interaktívne vykonávanie, hĺbkovú kontrolu a vizualizačné schopnosti. V tomto užívateľskom rozhraní bol naprogramovaný detekčný skript a parsovanie jednotlivých dát z *csv* a *pcap* súborov.



Obr. 7.4: Uživateľské rozhranie Jupyter Notebooku



Obr. 7.5: Užívateľské rozhranie Spyderu 3

7.2.2 Príprava vstupných dát

Táto kapitola sa bude zaoberať spracovaním dat na vstupné údaje a taktiež celkovým procesom, ktorým si dáta prejdú od počiatku monitorovania až po ich výsledné zobrazenie a interpretáciu.

Pred začiatkom monitorovania a získavania dat pre neurónovú sieť bolo potrebné si stanoviť veľkosť a typ dátovej sady, ktorá bude následne spracovaná. Aby bol dátová sada dostatočne obšírna na lokálnu sieť, časový interval bol stanovený na niekoľko minút. Podrobnejšie detaily o jednotlivých útokoch je možné vyčítať z tabuľky 7.1.

	Klienti/Útočníci	Prenesené pakety	TCP spojenia
30 min. lokálna vzorka	20/1	268000	9000
11 min. lokálna vzorka	5/5	29000	770
13 min. lokálna vzorka	25/5	654000	50370
5 min. verejná vzorka	2000/0	791000	22000
5 min. verejná vzorka	220/0	14000	480

Tab. 7.1: Použité Dátové sady

Z tabuľky je vidno, že prvé 3 vzorky obsahujú jedného, alebo viac útočníkov. Všetky tieto vzorky boli získané pomocou monitorovania lokálnej siete vytvorenej pre testovanie tohoto útoku.

Pre porovnanie časový interval prvého záznamu je stanovený na 30 minút, zatiaľ čo ďalšie dva lokálne záznamy majú dĺžku 11 a 13 minút. Prvý záznam oproti dvom ďalším reprezentuje DoS útok, kedy sa jedná iba o jediného útočníka. Na druhe strane pre nasledujúce dva záznamy bol ich počet zvýšený a tak celková škodlivá aktivita stúpla. Všetky 3 scenáre postupne menia jednotlivý počet klientov a útočníkov, aby tak bola zabezpečená väčšia variabilita a presnosť výsledných dát.

Posledné 2 vzorky sú použité z verejného repozitáru [23]. Obidva pcap súbory slúžia pre overenie správnosti výsledkov na lokálnej sieti. Keďže SlowDrop útok je relatívne nový, na internet sa nenachádzajú voľne dostupné pcap súbory, na ktorých by bolo možné celú funkcionalitu overiť. Keďže ani jeden zo stiahnutých súborov sa nešpecializuje a neobsahuje SlowDrop útok, nebolo ich tak možné použiť pri neurónovej sieti. Avšak na druhej strane obidve verejné vzorky boli následne použité pre overenie detekčného skriptu.

Po hlbšej analýze pomocou programu Wireshark, bol nájdený hlavný príznak, ktorý útok špecifikuje. Detekovateľná anomália sa objavuje vo forme TCP Retransmission paketov, ktoré nastávajú v prípade, že jednotlivý paket s potvrdzo-

vacím číslo ACK je zopakovaný najmenej 3 krát a strana klienta paket neprijíma. Týmto spôsobom je zistiteľná sekvencia jednotlivých hodnôt ACK pre dané pakety. Ak je postupnosť nemenná t.j. ostáva na jednej hodnote nastane TCP Retransmission. Touto cestou sú dáta z pcap súboru parsované a spracované pre použitie do neurónovej siete.

Pomocou spomínanej metódy je spracovaný celý súbor a každému TCP spojeniu priradená hodnota, ktorá reprezentuje normálne spojenie, alebo anomáliu. Bezchybné spojenie je reprezentované hodnotou 1, zatiaľ co anomália je reprezentovaná hodnotou 2. Neurónová sieť následne rozlíši, ktoré vzorky má považovať za bežné a ktoré za chybné.

Pred začatím spracovania dát a vytvorenia vstupných dát pre neurónovú sieť je potrebné si určite podmienky, ktoré je treba dodržať. Tieto podmienky sú dve.

Prvou podmienkou je dĺžka jednotlivých vzoriek. Tieto vzorky sú v tomto prípade jednotlivé TCP spojenia. Aby neurónová sieť dáta vôbec dokázala spracovať, všetky vzorky musia mať jednotnú dĺžku. Jednotlivé TCP spojenia sa zameriavajú na ACK hodnoty a ich sekvenciu pre dané spojenie.

Druhá podmienka nie je všeobecne nutná, ale je vhodné ju dodržať. Jedná sa o usporiadanie vstupných dat. V ideálnom prípadne pre efektívnejšie učenie neurónovej siete je vhodné dodržať hodnoty vstupných dát od 0 do 1.

Aby neurónová sieť dokázala dáta spracovať a následne sa ich naučiť s čo najmenšou stratovosťou a chybou je potrebné obecne dáta štandardizovať. Tento krok prebehol vo 2 úrovniach.

Pre splnenie prvej podmienky je potrebné nájsť vhodnú dĺžku dát, ktorá bude obsahovať čo najviac potrebných informácii, podľa ktorých je anomália rozlíšiteľná. Za predpokladu, že nie všetky spojenia majú rovnakú dĺžku je potrebné nájsť metódu, ktorá tento výsledok dosiahne.

Navrhnutou metódou pre riešenie daného problému a dodržania prvej podmienky, ktorá ma zabezpečiť rovnakú dĺžku každej vstupnej vzorky do neurónovej siete bola nulová výplň. Inými slovami povedané, všetky TCP spojenia, ktoré nedosiahli dĺžku 20, boli dodatočne doplnené nulovými hodnotami. Tento proces je nutné dodržať z dôvodu, že sieť potrebuje rovnakú dĺžku vstupných dát pre každú jednu vzorku.

V sledovanom dátovom sete, ktorý je použitý pre učenie siete malo najdlhšie TCP spojenia zo zaznamenaných dát dĺžku 20, čiže sekvenciu 20 hodnôt ACK, pred ukončením daného spojenia. Výplň je nutná, aby neurónová sieť dokázala dáta čítať, pretože niektoré spojenia sa vyskytujú v kratšom formáte.

Kratší formát môže vzniknúť napríklad z dôvodov samotného útoku, kedy útočník zatvára jednotlivé spojenia za účelov zahltenia serveru. Taktiež či už bežný užívateľ, alebo útočník, obaja môžu spojenie ukončiť predčasne z rôznych dôvodov. Dáta

v takejto forme stále nie sú považované za anomáliu týkajúcu sa DDOS SlowDrop útoku a treba uchovať rovnakú štruktúru a dĺžku dat.

Po prvom spracovaní vznikne csv súbor, ktorému je podľa duplikovateľnosti jednotlivých ACK hodnôt priradená hodnota. V prípade, že ACK hodnota sa neopakuje viac ako 3 krát a tak nesplňuje podmienku o TCP Retransmission indikácii, danej vzorke je priradená hodnota 1, ktorá reprezentuje TCP spojenie bez indikácie. V opačnom prípade ak sa duplikované hodnoty vyskytnú viac ako 3 krát, daná vzorka je označená hodnotou 2 a jedná sa tak o anomáliu. Obe tieto hodnoty sú následne použité pri rozdeľovaní dát do dvoch množín.

Druhá podmienka, ktorá bola spomenutá skoršie v práci je škálovanie dát zo vstupných dat na hodnoty v interval <0,1>. Tento môže prebehnúť hneď následne po dodržaní prvej podmienky o dĺžke dát, ale môže byť oddialený až do samotného učenia.

Druhá podmienka na dáta je aplikovaná až po prvom spracovaní a to pomocou funkcie, ktorá má na starosti zabezpečiť škálovanie dát z pôvodných hodnôt do hodnôt na intervale <0,1>. Tento krok je dôležitý z pohľadu efektívnosti učenia neurónovej siete. Vo všeobecnosti sa neurónové siete učia lepšie ak sú vstupné hodnoty v tomto intervale. Volanie funkcie je zobrazené na obrázku 7.6

```
scaler = MinMaxScaler()
data_scaled = scaler.fit(train_data)
```

Obr. 7.6: Funkcia na škálovanie vstupných dát

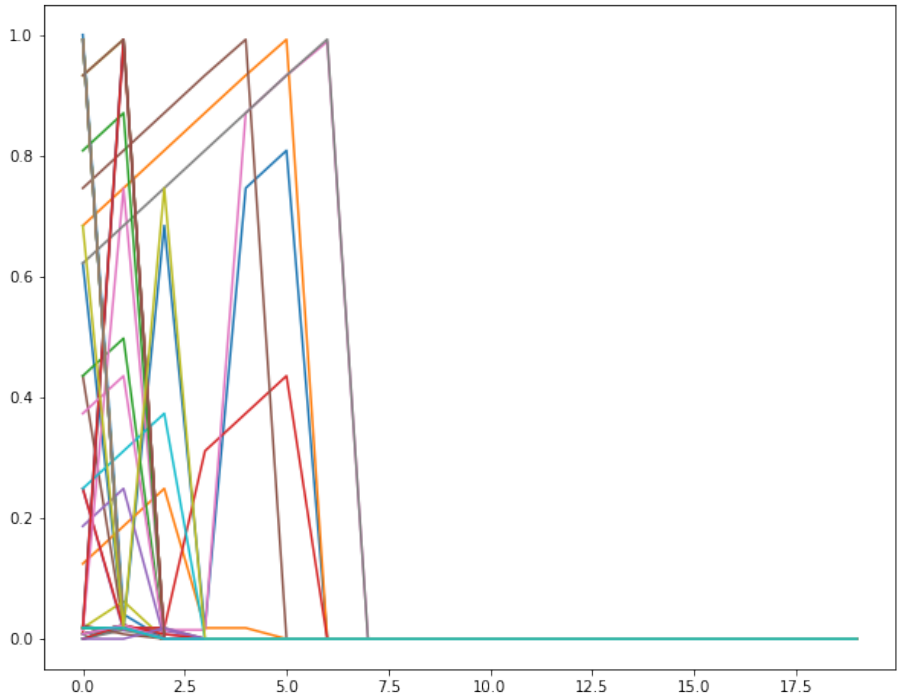
Po škálovaní dát, je vhodné zobraziť niekoľko vzoriek pre názornú ukážku ako dáta vyzerajú. Obrázok 7.7 ukazuje vstupné dáta pre normálnu trénovaciu množinu.

Naopak druhý obrázok 7.8 ukazuje dáta obsahujúce anomáliu, ktoré nadobúdajú rôznorodejšie hodnoty oproti prvému snímku.

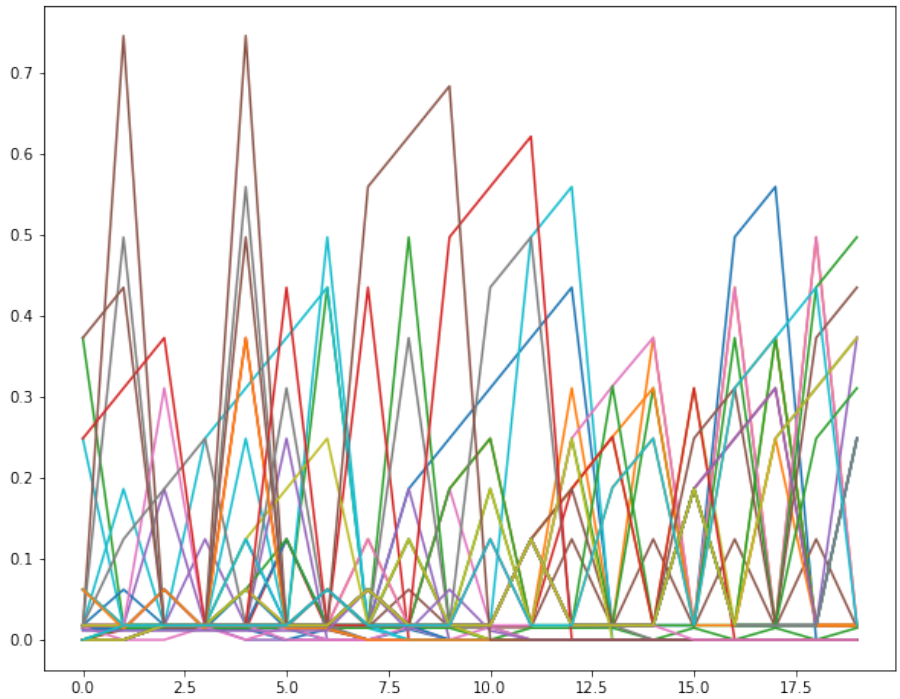
Po jednoduchom porovnaní oboch obrázkov je jednoznačne rozlíšiteľné, že dáta obsahujúce anomáliu majú vyššiu variabilitu a jednoznačne sa odlišujú od normálnych dát.

7.3 Model a implementácia

Táto podkapitola je najrozsiahlejšia v celej práci a opisuje celkové riešenie pomocou neurónovej siete. Na začiatku popisuje model siete a následne po poradí sa venuje jednotlivým častiam siete, ktoré sú uvedené aj s následným vysvetlením a zobrazením kódu alebo výsledok pre jednoduchšie pochopenie.



Obr. 7.7: Reprezentácia normálnych vstupných dát v grafe



Obr. 7.8: Reprezentácia vstupných dát s anomáliou v grafe

Autoencoder [26] sa používa ako technika učenia bez dozoru. Pri návrhu architektúry neurónovej siete je použité úzke miesto ako posledná vrstva encoderu, ktoré sa nazýva "bottleneck". Keďže vstupné dáta sú závislé jedny od druhých, encoder s koncom v bottlenecku sa používa na komprimáciu dát a tak zvýraznenie významnej charakteristicky pôvodného vstupu. Pretože autoencoder sa učí ako komprimovať dáta na základe jednotlivých atribútov (korelácie medzi vektormi) zistených počas tréningu, sú tieto modely schopné rekonštruovať dáta, ktoré sú podobné tým, ktoré boli pozorované počas učenia. Následne opačným princípom sa dáta vracajú do pôvodnej formy a dĺžky. Autoencoder sa používa pri aplikáciach s týmto zameraním:

- Detekcia anomálií
- Odšumenie údajov - obrázky, zvuk
- Maľba obrazu
- Získavanie informácií

Konkrétny model použitého autoencoderu v tejto diplomovej práci je zobrazený na obrázku D.1. Čísla na obrázku reprezentujú postupnosť jednotlivých vrstiev od vstupu až po výstup a tým pádom rekonštrukciu dát.

Motivácia, prečo bolo zvolené riešenie práve s autoencoderom je taká, že tento model neurónovej siete udržuje rovnakú štruktúru a dĺžku vstupných a výstupných dát.

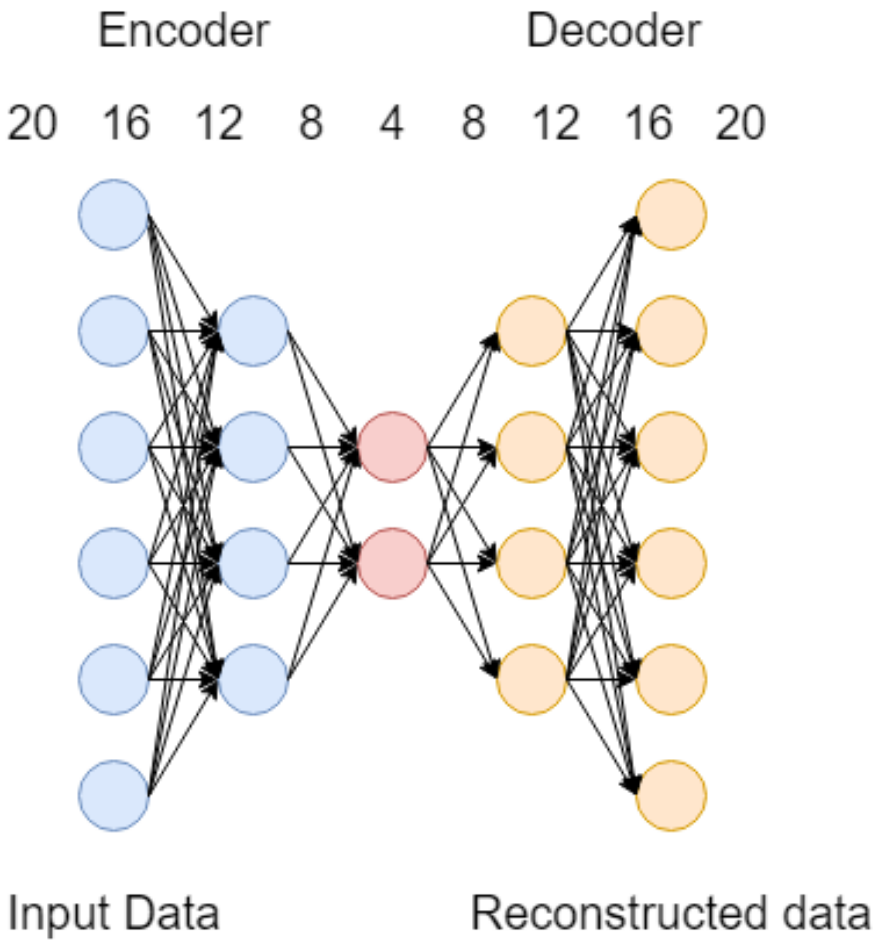
Dôvod použitia neurónovej siete s autoencoderom je ten, aby bola zabezpečená rovnaká štruktúra a dĺžka dat po dokončení učenia a následnej rekonštrukcii. Tento typ dáva možnosť porovnať jednotlivú štruktúru vzoriek pred a po učení a vypočítať tzv. Reconstruction Error(RE), ktorému bude viac venovaná pozornosť na konci kapitoly.

Ako bolo uvedené v tabuľke 7.1, jednotlivé vzorky, s ktorými neurónová sieť bude pracovať sú uvedené v nasledujúcich tabuľkách 7.2, 7.3, 7.4, kde je vidieť počet jednotlivých TCP spojení, ktoré sa bude sieť snažiť rozpoznať.

	Celkovo	Testovacia vzorka
Normálne	30602	24491
Anomálie	6695	6111

Tab. 7.2: Dátová sada vzorky 1

Pre vysvetlenie princípu budú použité informácie z lokálnej vzorky 1 uvedenej v tabuľke 7.2. Celková vzorka dát, na ktorej prebehne učenie neurónovej siete obsahovala viac ako 37000 vzoriek z toho viac ako 30000 normálnych a viac ako 6500 s anomáliou. Z celkovej sady bolo vybratých 20% ako trénovacia množina, ktorá je následne použitá pre učenie. Poradie jednotlivých vzoriek nehrá rolu keďže



Obr. 7.9: Model siete - Autoencoder

	Celkovo	Testovacia vzorka
Normálne	2355	483
Anomálie	959	180

Tab. 7.3: Dátová sada vzorky 2

	Celkovo	Testovacia vzorka
Normálne	95024	19016
Anomálie	831	155

Tab. 7.4: Dátová sada vzorky 3

pred učením bola použitá funkcia Shuffle, ktorá jednotlivé vzorky náhodne rozhádže a tým zabezpečí, že sieť sa nesústredí na poradie vzoriek ale na ich obsah. V prípade, že by Shuffle nebol použitý, mohlo byť nastať, že poradie vstupných vzoriek jednotlivej dátovej sady by malo určitú postupne, kde by sa sieť naučila rozlišovať anomáliu podľa postupnosti jednotlivých vzoriek. Pre všetky použité vzorky budú na záver uvedené jednotlivé výsledky.

Ako už bolo spomenuté neurónová sieť rekonštruovala dáta pomocou autoencoderu. Dáta boli upravené a bola im nastavená fixná dĺžka 20 to znamená že prvá vstupná vrstva modelu obsahuje 20 vstupných prvkov. Keďže účelom autoencoderu je kompresia dát, následujúce vrstvy obsahujú menší a menší počet vstupných prvkov a to 16, 12, 8 a 4.

Decoder má na dáta opačný účinok, nejedná sa o zmenšovanie, ale o zväčšenie dat do jej pôvodnej formy. Jednotlivé vrstvy v decoderi obsahujú rovnaké vrstvy, ale v opačnom poradí. Po bottlenecku, vrstve o 4 prvkoch nasleduje vrstva s 8, 12, 16 a nakoniec 20 prvkami. Tento proces má za úlohu sa sústrediť len na najpodstatnejšie informácie charakterizujúce vzorky. Posledná vrstva v decoderi je považovaná za celkový výstup autoencoderu a dokáže rekonštruovať dáta do rovnakého formátu a dĺžky ako boli do nej vložené.

Pred spustením samotného učenia je vhodné spomenúť taktiež termín CallBack funkcia. Tento typ funkcií je spojený so samotným učením neurónovej siete a slúži na prípadné zastavenie učenia ak nastane uvedená podmienka vo funkcii. Jednotlivé CallBack funkcie sa starajú o vedenie a kontrolu samotného procesu učenia. CallBack funkcií je možné použiť niekoľko a zároveň nie sú povinným prvkom pri učení neurónovej siete. V práci bola použitá iba jediná CallBack funkcia zobrazená na obrázku 7.11

```
class AutoEncoder(Model):
  def __init__(self):
    super(AutoEncoder, self).__init__()
    self.encoder = tf.keras.Sequential([
      tf.keras.layers.Dense(20, activation="relu"),
      tf.keras.layers.Dense(16, activation="relu"),
      tf.keras.layers.Dense(12, activation="relu"),
      tf.keras.layers.Dense(8, activation="relu"),
      tf.keras.layers.Dense(4, activation="relu"),])

    self.decoder = tf.keras.Sequential([
      tf.keras.layers.Dense(8, activation="relu"),
      tf.keras.layers.Dense(12, activation="relu"),
      tf.keras.layers.Dense(16, activation="relu"),
      tf.keras.layers.Dense(20, activation="sigmoid")])
```

Obr. 7.10: Zdrojový kód autoencoderu

```
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                  patience=3,
                                                  mode='min')
```

Obr. 7.11: CallBack funkcia

Táto funkcia má za úlohu zastaviť učenie v prípade, že val_loss sa nezvyšuje 3 epochy po sebe. Dôvodom toho Callbacku je vyhnutie sa jednej z dvoch situácii, ktorá sa nazýva overfitting a underfitting.

Premenná val_loss [35] je hodnota nákladovej funkcie pre overenia dát, Ak sa hodnota znižuje, znamená to, že sieť sa učí dobre. V opačnom prípade, ak sa val_loss hodnota navyšuje nastáva tzv. overfitting.

"K overfittingu dôjde, keď sa model naučí podrobnosti a šum v tréningových dátach do tej miery, že to negatívne ovplyvní výkonnosť modelu na nových dátach. To znamená, že šum alebo náhodné výkyvy v tréningových dátach model zachytáva a učí sa ich ako koncepty. Problém je v tom, že tieto koncepty sa nevzťahujú na nové údaje a negatívne ovplyvňujú schopnosť generalizovať modely."[24]

"Underfitting sa týka modelu, ktorý nedokáže modelovať údaje o tréningu ani zovšeobecniť ich na nové údaje. Underfitting model strojového učenia vo výstroji nie je vhodný model a bude zrejmé, že bude mať slabý výkon z tréningových údajov. O underfittingu sa často nediskutuje, pretože je ľahké ho zistiť pri dobrej metrike výkonu. Nápravou je ísť ďalej a vyskúšať alternatívne algoritmy strojového učenia. Napriek tomu poskytuje dobrý kontrast s problémom overfittingu."[24]

Po vytvorení Callback funkcií, určenia trénovacej dátovej sady a určenia počtu epóch, ktoré majú v učení prebehnúť je učenie spustené. Pre uvedenie prehľadnej-

šieho volania CallBack funkcie je nutné rozobrať obrázok 7.12. Na obrázku je viditeľné, že val_loss sa nezmenšil pod danú hodnotu 3 epochy. CallBack funkcia ukončí učenie siete z dôvodu, aby nedošlo k overfittingu.

```
192/192 [==============================] - 0s 2ms/step - loss: 0.0107 - val_loss: 0.0199
Epoch 31/50
192/192 [==============================] - 0s 2ms/step - loss: 0.0107 - val_loss: 0.0196
Epoch 32/50
192/192 [==============================] - 0s 2ms/step - loss: 0.0107 - val_loss: 0.0195
Epoch 33/50
192/192 [==============================] - 1s 3ms/step - loss: 0.0106 - val_loss: 0.0196
Epoch 34/50
192/192 [==============================] - 1s 4ms/step - loss: 0.0105 - val_loss: 0.0198
Epoch 35/50
192/192 [==============================] - 1s 4ms/step - loss: 0.0105 - val_loss: 0.0195
```

Obr. 7.12: Aktivácia CallBack funkcie

V druhom prípade pri inej dátovej sade je vidieť na obrázku 7.13, že k danej podmienke neprišlo, a tak CallBack funkcia neprerušila učenie, ktoré prebehlo až do 50-tej epochy.

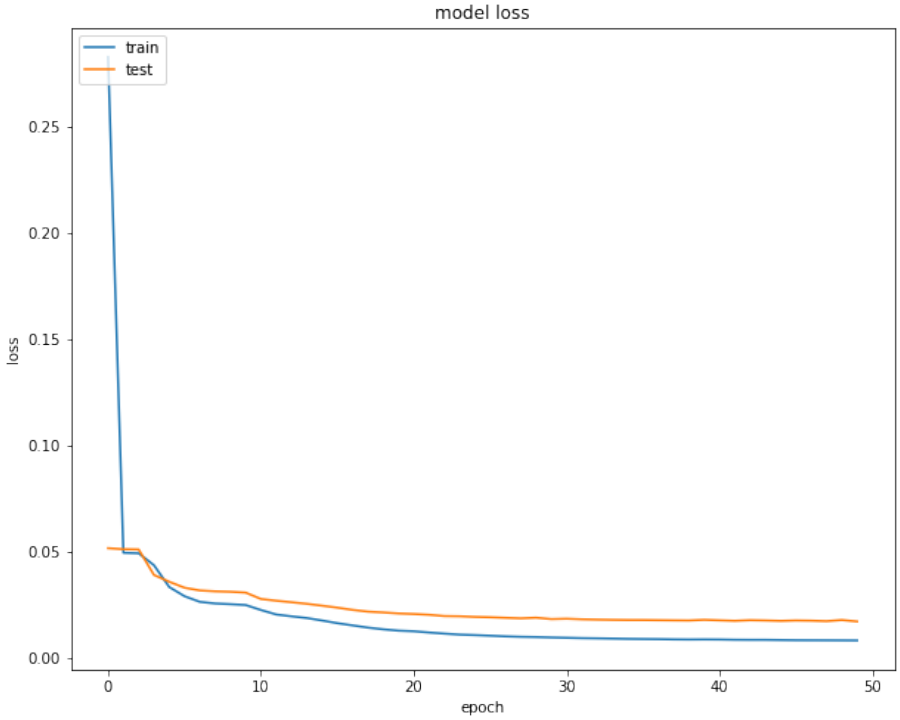
```
Epoch 47/50
192/192 [==============================] - 1s 3ms/step - loss: 0.0076 - val_loss: 0.0168
Epoch 48/50
192/192 [==============================] - 1s 3ms/step - loss: 0.0076 - val_loss: 0.0166
Epoch 49/50
192/192 [==============================] - 1s 4ms/step - loss: 0.0076 - val_loss: 0.0166
Epoch 50/50
192/192 [==============================] - 1s 4ms/step - loss: 0.0076 - val_loss: 0.0167
```

Obr. 7.13: Kompletné dokončenie učenia

Počas procesu učenia boli sledované 2 meniace sa hodnoty, ktoré majú dôležitú úlohu pri neurónových sieťach. Z obrázku 7.13 je vidieť jednotlivé sledované hodnoty, ktorými sú loss a val_loss. Ako už bol vysvetlený význam val_loss a jeho sledovanie vyššie, je taktiež potrebné uviesť dôležitosť parametru loss.

Loss je hodnota, ktorú sa neurónová sieť počas tréningu snaží znížiť. Čím nižšia hodnota loss, tým je presnejšia následná predikcia siete. Na obrázku 7.14 sú zobrazené hodnoty stratovosti neurónovej siete pri učení. Po prvý epochách učenia sa stratovosť pohybuje pod 5% a blíži sa k 0. Táto hodnota je následne použitá pri rekonštrukcii jednotlivých vzoriek a jej hodnota je reprezentovaná vo forme rekonštrukčnej chyby.

Po tom ako sa sieť naučí rozlišovať bežnú proti škodlivej vzorke je potrebné hovoriť o spôsobe ako prezentovať výsledky tejto siete. Ako bolo spomenuté pri opise modelu, výsledkom funkcie autoencoderu je rekonštrukcia dát do rovnakej formy ako bola na vstupe. Táto rekonštrukcia dáva možnosť zobraziť dáta a porovnať tak dve vzorky jednu pred učením a druhú po učení vo forme rekonštrukcie. Porovnanie týchto dvoch kriviek je zobrazené na obrázku 7.15. Chyba rekonštrukcie oproti originálu sa nazýva Reconstruction error.



Obr. 7.14: Graf stratovosti

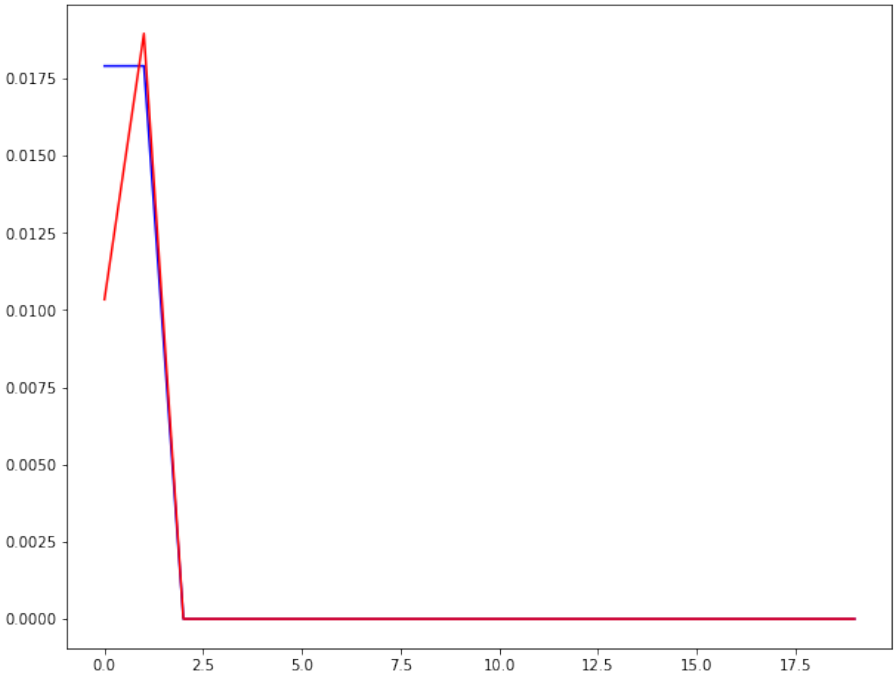
Reconstruction error(RE) je jedným z vedúcich výsledkov z neurónovej siete a je na ňom postavené určenie úspešnosti neurónovej siete. Po učení je sieť schopná rekonštruovať jednotlivé vzorky a následne ich porovná s originálnou vzorkou, ktorá obsahuje hodnoty pred učením. Chybe z rekonštrukcie oproti originálu rozumieme ako RE. Hodnota je vyjadrená z intervale <0,1> ako percentuálna chyba autoencoderu, ktorý rekonštrukciu vytvára.

Pre porovnanie je vhodné upriet pozornosť na obrázok 7.15 a 7.16. Na oboch obrázkoch sú dve krivky. Modrá reprezentuje originálne dáta, ktoré vstupujú do neurónovej siete pred samotným učením. Druhá, červená krivka ukazuje rekonštruované dáta, ktoré vytvoril autoencoder. RE je následne vypočítaný percentuálneho rozdielu rekonštrukcie oproti normálnej vzorke. V prípade, že by boli obe krivky identické RE by mala hodnotu 0.

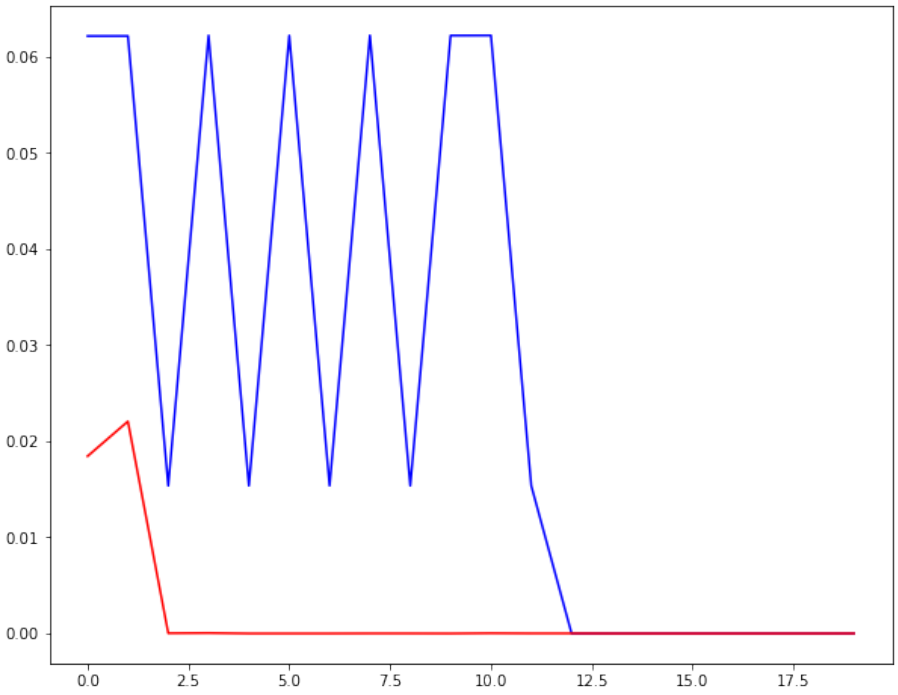
Obrázok 7.15 náhodnú vzorku z množiny bez anomálie, z ktorej je jednoznačne vidieť, že rekonštrukcia vo vysokej miere kopíruje originálnu vzorku. Takto je jasne viditeľné, že RE sa bude pohybovať v hodnotách blízkych 0.

Na druhej strane vzorka zobrazená na obrázku 7.16 opisuje anomáliu, kedy je viditeľné, že rekonštrukcia nie je blízko k tomu aby bola identická s originálom. Preto pri výpočte RE bude hodnota oveľa vyššia ako v prvom prípade.

Pre prehľadnejšie a jednoznačnejšie zobrazenie sú použité obrázky 7.17 a 7.18, kde prvý spomenutý ukazuje 50 normálnych vzoriek rekonštrukcie a druhý obrázok

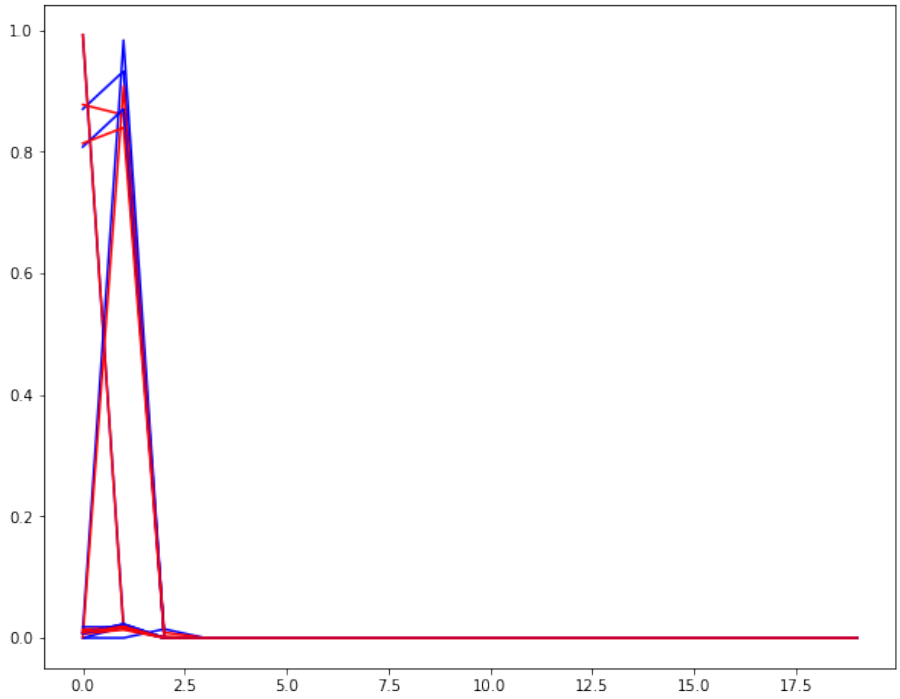


Obr. 7.15: Rekonštrukcia 1 normálnej vzorky



Obr. 7.16: Rekonštrukcia 1 vzorky s anomáliou

50 vzoriek rekonštrukcie s anomáliou.



Obr. 7.17: Rekonštrukcia 50 normálnych vzoriek

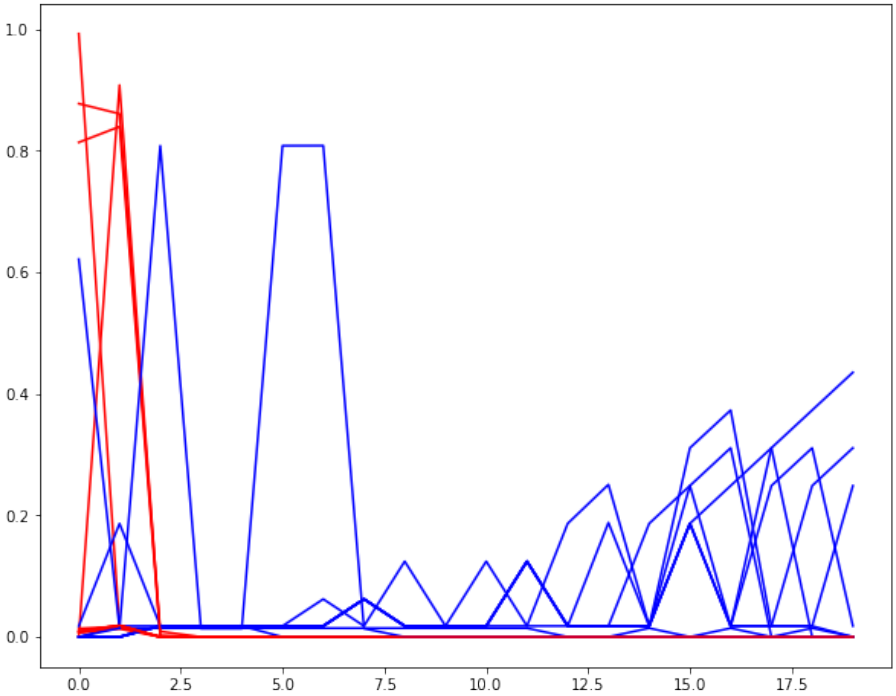
Ako je z obrázkov zrejmé, RE pri bezchybných vzorkách je minimálny a pre väčšinou vzoriek sa pohybuje okolo 1%. Po porovnaní oboch zobrazení, druhý obrázok obsahujúci vzorky s anomáliami vykazuje jednoznačne vyššie hodnoty RE. Jednotlivé hodnoty RE sú neskôr použité na určenie výsledkov a tak vypočítanie presnosti neurónovej siete a jej schopnosti určiť, či dané TCP spojenie je anomália, alebo nie.

Pre lepšie pochopenie a vhodnú reprezentáciu stratovosti siete pomocou RE, je vhodné vytvoriť graf, ktorý zobrazuje všetky RE testovacej sady v jednom celku. Tento graf je viditeľný na obrázku 7.17. Väčšina hodnôt sa pohybuje okolo blízko 0, to znamená, že normálne hodnoty bez anomálie sa sieť dokázala naučiť a dokáže túto množinu rozpoznať z veľkou percentuálnou úspešnosťou.

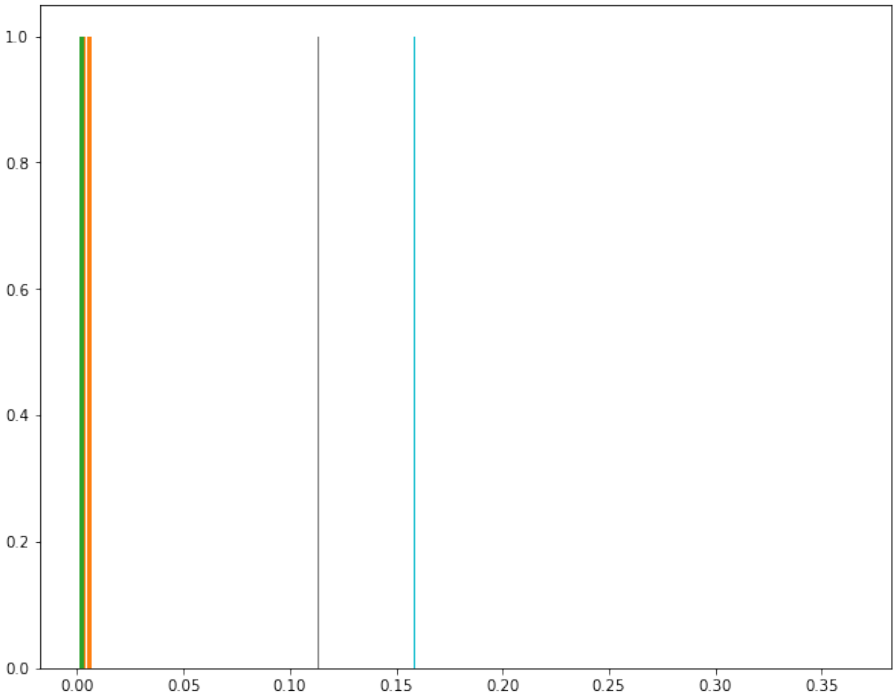
Avšak na obrázku 7.18, ktorý zobrazuje množinu obsahujúcu anomáliu je zreteľne vidieť, že RE v priemere nadobúda oveľa vyššie hodnoty ako pri rekonštrukcii normálnych vzoriek.

Obidva grafy dávajú možnosť porovnania jednotlivých dátových setov, ale primárne dávajú možnosť na výpočet tzv. thresholdu. Threshold hodnota je vypočítaná pomocou vzorca zobrazeného na obrázku 7.21. Tento vzorec udáva hodnotu medzi jednotlivými dátovými sadami podľa ktorej sú následnej rozdelené do dvoch množín.

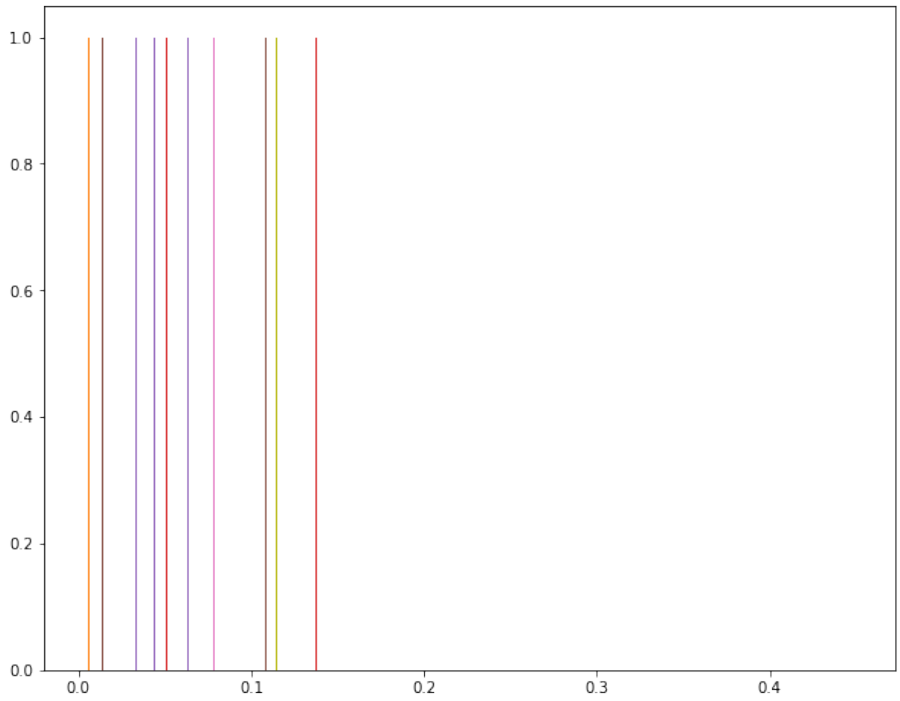
Po získaní hodnoty thresholdu je vhodné spojiť obe rekonštruované množiny do jedného grafu a zobraziť ich spoločne s hodnotou thresholdu. Obrázok 7.22 zob-



Obr. 7.18: Rekonštrukcia 50 vzoriek s anomáliou



Obr. 7.19: Graf RE pre všetky vzorky z normálnej množiny



Obr. 7.20: Graf RE pre všetky vzorky z množiny s anomáliou

```
threshold = np.mean(train_loss) + 2*np.std(train_loss)
```

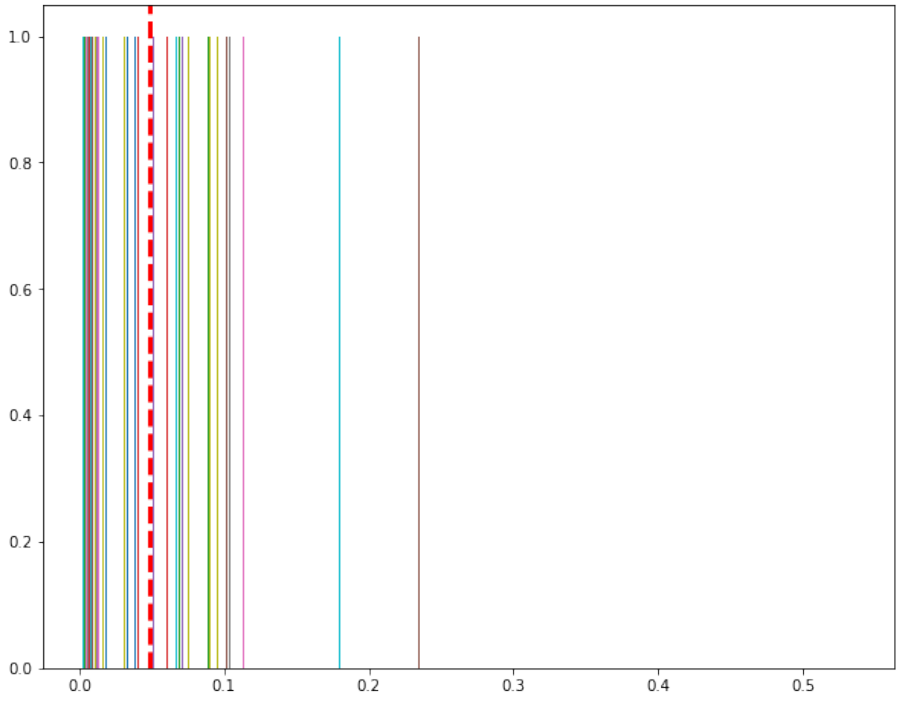
```
threshold
```

```
0.04828959282445906
```

Obr. 7.21: Vzorec pre výpočet thresholdu

razuje túto skutočnosť, kde červená pásikovaná čiara na hodnote 0.48 reprezentuje threshold a rozdeľuje tak graf na dve časti.

Jednotlivé vzorky RE, ktoré majú nižšiu hodnotu ako threshold sú priradené do jednej množiny a naopak hodnoty vyššie od thresholdu do druhej skupiny ako je možné vidieť na obrázku 7.22.



Obr. 7.22: Graf všetkých vzoriek RE s thresholdom

Porovnanie skutočnej hodnoty a rekonštruovanej hodnoty prebehne pre celý dátový set a je tak určená percentuálna úspešnosť ako neurónová sieť dokáže rozlíšiť jednotlivé vzorky, ktoré reprezentujú TCP spojenia.

Obrázky 7.23 a 7.24, porovnávajú testovaciu dátovú sadu s touto sadou po rekonštrukcii autoencoderu. Prvý spomenutý obrázok ukazuje normálne dáta bez anomálie, kde celkový počet vzoriek je 6111. Pomocou rekonštrukcie a hodnoty RE v porovnaní s thresholdom(hodnoty menšie ako threshold), sieť dokázala správne odhadnúť 5771 vzoriek, tým pádom je jej úspešnosť určená na 94,43%.

V druhom prípade sa jedná o dáta obsahujúce anomáliu, kde testovacia sada obsahovala 1349 vzoriek, z ktorých rovnakým princípom (RE hodnoty vyššie ako treshold) sieť určila 662 vzoriek ako anomáliu. Jej úspešnosť je určená na 49.07%.

Výsledky všetkých testov budú podrobnejšie popísané v kapitole9. Podrobnejšie budú opísané aj jednotlivé dôvody, prečo výsledky nadobúdajú dané hodnoty a čo je ich dôvodom. Taktiež sú následne uvedené návrhy ako postupovať a ďalej pracovať so získanými informáciami a dátami.

```
print(tf.math.count_nonzero(preds))
print(preds.shape)

tf.math.count_nonzero(preds) / preds.shape
```

```
tf.Tensor(5771, shape=(), dtype=int64)
(6111,)

<tf.Tensor: shape=(1,), dtype=float64, numpy=array([0.94436262])>
```

Obr. 7.23: Výsledky rozpoznania bežnej komunikácie

```
print(tf.math.count_nonzero(preds_a))
print(preds_a.shape)

tf.math.count_nonzero(preds_a) / preds_a.shape
```

```
tf.Tensor(662, shape=(), dtype=int64)
(1349,)

<tf.Tensor: shape=(1,), dtype=float64, numpy=array([0.49073388])>
```

Obr. 7.24: Výsledky rozpoznania komunikácie s anomáliou

8 Detekčný skript

Poslednou časťou realizácie riešenia bolo navrhnutie a vytvorenie detekčného skriptu, ktorý odhalí útočníka a zobrazí jeho IP adresu. Vývojový diagram skriptu je uvedený ako príloha D. Skript sa nezaoberá zahadzovaním packetov, alebo následným blokovaním komunikácie s útočiacou stranou. Filtrovanie komunikácie a zahadzovanie packetov zo škodlivého zdroju je nad rámec tejto práce.

Vytvorený skript sa skladá z dvoch častí:

- Filtrovanie dát
- Spracovanie dát

Po analýze dát a získaní podrobnejších informácii o tom, ako sa útok sa chová bolo potrebné dáta filtrovať na mieru. Pre tento účel bol použitý Tshark ako modul softwaru Wiresharku. Tshark umožňuje zachytiť z reálneho prenosu na sieti, alebo čítať pcap súbory z zachytených súborov z minulosti. Hlavnou výhodou je, že dokáže vypísať tieto dáta do rôzny foriem a zapisovať ich do súborov. Jednoduchý príklad ako by mohol vyzerať príkaz volajúci tshark.

tshark -i wlan0 -w capture-output.pcap

Tento príkaz zahŕňa iba 2 dodatočné argumenty, ktorými sú *-i interface* a *-w file*. Prvý spomenutý jednoznačne určuje o ktoré sieťové zariadenia sa má jednať a druhý zaobstaráva písanie do súboru.

Na začiatku detekčného skriptu sú parsované argumenty pre ideálne spustenie skriptu, kde v prípadne chybných vstupných hodnôt je užívateľ navedený na spustenie *-h* argumentu, ktorý mu vypíše dodatočné informácie, kde je vysvetlené ako skript správne spustiť.

Ako názorná ukážka je uvedené spustenie skript vo dvoch formách. Jediným povinným argumentom skriptu je argument *-p* file.pcap, ktorý je nutný pre beh skriptu. Tento argument určuje vstupný súbor, ktorý má byť spracovaný.

python3 detection.py -p input_pcap_file.pcap -t 2 -a 30

Avšak pri spustení je možné použiť 2 voliteľné argumenty, ktorými sú *-t time* a *-a time*. Prvý uvedený *-t* reprezentuje časový interval v sekundách medzi jednotlivými TCP spojeniami, ktorý má byť považovaný za útok. Druhým uvedeným argumentom je *-a*, ktorý je taktiež reprezentovaný časovým intervalom v sekundách medzi jednotlivými útokmi. Inými slovami povedané po koľkých sekundách sa má počítadlo vynulovať, aby dva po sebe nadväzujúce TCP chyby neboli vyhlásené ako útok.

Druhým uvedeným príkladom je spustenie skriptu bez argumentu *-t time* a *-a*

time. Skript má štandardne nastavené hodnoty týchto premenných na 1 sekundu v prípade *-t* a 10 sekúnd v prípade *-a*.

python3 detection.py -p input_pcap_file.pcap

Obidve premenné majú nastavenú štandardnú hodnotu z dôvodu, že v prípade DOS útoku tieto nie je sieť zahltená až v takom množstve, ako by bola pri DDOS útoku a tým pádom jednotlivé časové intervaly medzi útokmi a jednotlivými chybnými TCP spojeniami sú vyššie. V prípade viacerých útočníkov a masívnejšieho útoku je vhodné obidva argumenty minimalizovať na kladné hodnoty blízke 0.

Po prvej časti je predaný vstupný súbor do bash skriptu, kde sa parsuje pcap súbor do formy dvojrozmerného pola. Hlavnou časťou tejto časti je volanie programu Tshark, ktorý spracováva vstupný pcap súbor a následne pomocou filtrov, vyberie len tie pakety, ktoré obsahujú TCP Retransmission indikáciu. Po získaní vyfiltrovaných dát pomocou Tsharku nám ostane namapované pole, ktoré jednu ku jednej spája čas a IP adresu klienta, ktorý obdržal paket obsahujúci TCP retransmission error. Výstup z bashu je predaný naspäť Python skriptu vo forme dvojrozmerného pola s ktorým Python pracuje ďalej.

Obrázok 8.1 ukazuje volanie programu tsharku v Bash skripte.

```
tshark -r $FILE -t ud -T fields -e _ws.col.Time  -e ip.dst -e tcp.dstport -e ip.src -e tcp.srcport
-e tcp.ack -e _ws.expert.group -e _ws.expert.message
-Y "(ip.src == 172.16.255.1 || ip.src == 192.168.3.131) && tcp.flags.ack == 1 &&
tcp.flags.push == 0 && tcp.flags.syn == 0 && tcp.flags.fin == 0 "
```

Obr. 8.1: Volanie Tsharku v bash skripte

Prvým uvedeným argumentom je *-r* File.. Tento argument zaisťuje, že sa jedná o čítanie a nie o živý dátový prenos. Z tohoto dôvodu argument potrebuje aj uviesť súbor, ktorý má spracovávať. Druhým v poradí je *-t* ud. Tento argument má na starosti uvedenie času v UTC formáte.

Nasledujúce dva argumenty *-T* a *-e* sú spoločne prepojené. *-T* zabezpečuje zobrazenie jednotlivých stĺpcov, ktoré *-e* následne menuje. V tomto konkrétnom prípade sa jedná o čas, IP adresu a port zdroja aj cieľa, hodnota ACK, expertnú správu jednotlivej kategórie a popis chyby.

Posledným argumentom je *-Y*, ktorý reprezentuje filter, ktorý chceme uviesť. Použitý príkaz obsahuje filter, ktorý sa skladá z dvoch hlavných častí. V prvej časti sú uvedené dve IP adresy, kde jedna z nich reprezentuje WAN a druhá LAN adresu serveru, na ktorý je útok vedený. Druhá časť filtru vyberá jednotlivé flag-y, ktoré je treba sledovať. Keďže sa jedná primárne o ACK hodnoty ako už bolo skoršie

spomenuté, je nutné nastaviť *tcp.flags.ack == 1* a hodnotu 0 pre všetky nepodstatné flagy.

Následné sú odstránené všetky nadbytočné chyby, ktoré neobsahujú TCP Retransmission a je tak vytvorené dvojrozmerné pole mapujúce čas a IP adresu zariadenia. Dvojrozmerné pole je následne zoradené podľa IP adries, ktoré obdržali danú chybu a sleduje sa čas medzi jednotlivými správami. Skript sleduje časové intervaly medzi TCP Retransmission správami, v prípade, že sa objaví sekvencia dvoch po sebe idúcich chýb a časový interval medzi nimi je menší ako 1 sekundy je oznámená správa vo forme varovania. Ak daná situácia nastane viac krát v 10 sekundách od poslednej detekcie, skript vyhlási, že sa jedná o útok.

Obe časové intervaly sú modifikovateľné cez argumenty pri spúštaní skriptu, kde 1 sekunda reprezentuje štandardnú hodnotu pre argument *-t* a 10 sekúnd ako štandardná hodnota pre argument *-a*.

Kľúčovým parametrom pre efektívne zachytenie útočníka je práve *-t*. Ako už bolo spomenuté tento parameter udáva frekvenciu medzi jednotlivými TCP Retransmission správami, ktoré charakterizujú útok, preto je potrebné uviesť podrobnejší popis.

Ako príklad je možné uvažovať nad situáciou, kedy sa jedná iba o DoS útok. V tomto prípade je útočník iba jeden a tým pádom frekvencia medzi jednotlivými chybami bude vyššia ako v prípade DDoS útoku. V prípade jedného útočníka je vhodné nastaviť sledovaný časový interval na mierne vyššiu hodnotu od 0, napríklad 1 sekunda.

Avšak ak by bolo uvažované s reálnou situáciou, je potrebné sa na útok pozerať ako DDoS, kde účelom je zahltenie serveru. Taktiež je potrebné uvažovať na scenárom bežnej siete, kde sa nachádza niekoľko desiatok, stoviek či až tisícom legitimných klientov. V prípade DDoS útoku bude server a aj komunikácia zahltená TCP Retransmission správami a tak ich frekvencia bude niekoľko násobne vyššia. Z tohoto dôvodu je vhodné znížiť veľkosť časového intervalu na hodnoty blízke 0, napríklad 1 sekunda. Druhý parameter *-a* z pohľadu dlhodobého zahltenia siete by mal nadobúdať hodnotu vo vyšších jednotkách, prípadne nižších desiatkach sekúnd.

Parameter *-a* slúži ako ochrana pre legitimných klientov so slabým spojení a to takým spôsobom, že ak prejde časový interval stanovený v danej premennej a nevyskytne sa zvýšenie správy TCP Retransmission, počítadlo chýb je obnovené na hodnotu 0. Inak povedané parameter *-a* udáva maximálny časový interval od posledného útoku, kedy je možné zaznamenať pokračovanie útoku.

Pre lepšie pochopenie je dobré uviesť jednoduchý príklad. Ak by bolo uvažované nad situáciou, kde útočník útok spustí a následne vypne, parameter *-a* reprezentuje časový interval, kedy skript prestane zaznamenávať jednotlivé TCP Retransmission správy ako chyby, až do kým sa ich výskyt znova nezvýši a nezačne opakovať vo väčšej škále. Hlavným dôvodom je to, aby legitimný užívateľ nebol prehlásený za útočníka.

Ak by útok trval 100 sekúnd(výskyt TCP Retransmission správy by bol zvýšený od 0 až po 100 sekúnd a po 100 sekundách sa výrazne zníži) a parameter *-a* by bol nastavený na hodnotu 10, od 110 sekundy prestane intenzívne sledovať komunikáciu a TCP Retransmission správy v malej škále bude ignorovať. Od 110 sekundy skript čaká na masívny výskyt a opakovanie daných správ v krátkom časovom intervale udaným parametrom *-t*.

V tabuľke 8.1 sú zobrazené štandardné a odporúčané hodnoty podľa typu útoku.

	Štandardné	DoS	DDoS
-t sekundy	1	3	1
-a sekundy	10	10 a viac	10 a viac

Tab. 8.1: Odporúčané hodnoty argumentov pri spúštaní skriptu

V prípade zachytenia útoku je vypísané upozornenie na štandardný konzolový výstup. S upozornením je uvedená aj IP adresa útočníka. Pre jednoduchšie zobrazenie výstupných dát programovací jazyk Python umožňuje priamo vypisovanie do súboru pomocou znaku > pri spúštaní skriptu ako je uvedené nižšie.
python3 detection.py -p input_pcap_file.pcap > output.txt

Výsledky detekčného skriptu, ako aj výsledky softwaru Suricata a úspešnosť neurónovej siete budú prebraté v ďalšej kapitole, ktorá bude zároveň záverečnou kapitolou celej práce.

9 Výsledky detekcie

Posledná kapitola popisuje a porovnáva výsledky získané z jednotlivých častí práce, tak ako aj porovnanie jednotlivých dátových vzoriek z pohľadu neurónovej siete ale aj detekčného skriptu. Po uvedení výsledkov sú v teoretickej rovine navrhnuté základné riešenia ako sa proti útoku brániť.

Neurónová sieť spracovávala niekoľko súborov obsahujúcich sieťovú komunikáciu, ktorá v sebe niesla škodlivé dáta spôsobené SlowDrop útokom. Ako bolo popísané v podkapitole 7.2.2, sieť pracovala s dátovými sadami uvedenými v tabuľke 7.1. Tieto dáta po celom procese spracovania a učenia vykazujú výsledky prezentované v tabuľke 9.1

	Rozpoznanie normálneho spojenia	Rozpoznanie spojenia s anomáliou
30 min. lokálna vzorka	94,43%	49,07%
11 min. lokálna vzorka	93,37%	9,44%
13 min. lokálna vzorka	98,77%	36,12%

Tab. 9.1: Výsledky neurónovej siete

Z obrázku vidieť, že všetky lokálne vzorky vykazujú vysokú rozpoznateľnosť pre vzorky bez anomálie, ktorá sa má presnosť takmer 99% a v dvoch prípadoch 94,43% a 93,37%. V opačnom prípade pre vzorky s anomáliou sa hodnota pohybuje pod 50%, konkrétne 49,07%, 9,44% a 36,12%.

Celkové výsledky sú v danej miere skreslené nulovou výplňou, ktorá zabezpečuje rovnakú dĺžku jednotlivých vzoriek a je tak nutná pre beh neurónovej siete. Druhým faktorom, ktorý má dopad na presnosť je aj nevyvážený pomer bežných a škodlivých spojení. Táto situácia vzniká z dôvodu, že na sieti sa vyskytujú aj legitimní užívatelia a taktiež aj útočník sa snaží tváriť ako bežný užívateľ. Z tohoto dôvodu nie všetká jeho komunikácia je považovaná za škodlivú a tak narastá počet legitimných dát voči škodlivým.

Výsledky sa nemusia javiť úplne v najideálnejšom svetle. Avšak, ak by sa do úvahy zobal predpoklad vysokej správnosti detekcie normálnych vzoriek(v priemere 95.52%) je možné uvažovať nad princípom detekcie pomocou logiky doplnkového prvku. Za predpokladu, že sú dátové sady rovnaké a chyba je v priemere 4.48% je možné vzorky z množiny označenej ako anomália prehlásiť ako anomália.

Pre lepšie pochopenie je vhodné uviesť príklad. Ak by celková dátová sada obsahovala 20000 vzoriek a z toho 10000 normálnych vzoriek a 10000 škodlivých vzoriek, sieť by v priemere vyhlásila 95.52% správnych z normálnej množiny konkrétne 9552

vzoriek a tým pádom by bolo celú druhú množinu prehlásiť za škodlivú s maximálnou chybou 4.48% alebo teda 448 vzorkami.

Verejné vzorky z repozitáru [23] neboli v tomto prípade zahrnuté z dôvodu, že sieť by nemala k dispozícii žiadne anomálie pre detekciu. V ideálnom prípade by bolo treba neurónovú sieť otestovať na viacerých masívnych vzorkách z globálnych sietí, kde by sa vyskytovali prvky tohoto útoku. Avšak keďže je útok relatívne nový takéto verejne dostupné pcap súbory nie sú k dispozícii. Taktiež momentálne pandemická situácia vo svete neumožňuje zaobstaranie zdrojov a prostriedkov pre vytvorenie a vedenie takejto testovacej siete.

Avšak neurónová sieť s autoencoderom má aj svoje slabé stránky. To čo sa môže javiť spočiatku ako silnou stránkou, má aj svoje nedostatky. Jedná sa konkrétne o rovnakú štruktúru dát, ktoré sieť zrekonštruuje. Táto rekonštrukcia má rovnaký formát ako vstupné dáta a tým pádom sa jedná o hľadanie sekvencie opakujúcich sa dát, ktoré sú vyhodnotené ako anomália. Nevýhodou v tomto smere je, že výstupné dáta síce dokážu rozoznať bežnú komunikáciu na sieti, ale nehovoria nič o pôvode dát a o tom, kto ich posiela.

Ďalšou nevýhodou pri riešení problému tohoto typu je jednotnosť a jednotná forma vstupných dát. Keďže SlowDrop útok udržuje jednotlivé TCP spojenia a tak nasilu predlžuje ich životnosť, tieto spojenia oproti bežnej komunikácii majú rozdielnu dĺžku. Taktiež tento útok s dopredu určenou pravdepodobnosťou zatvára vytvorené TCP spojenia. Často krát tak nastane situácia, že ich dĺžka sa zmenší. Z týchto dvoch dôvodov bolo nutné použiť výplň, ktorá v určitej miere skresľuje finálne výsledky detekcie neurónovej siete, hlavne v prípade dát obsahujúcich anomáliu.

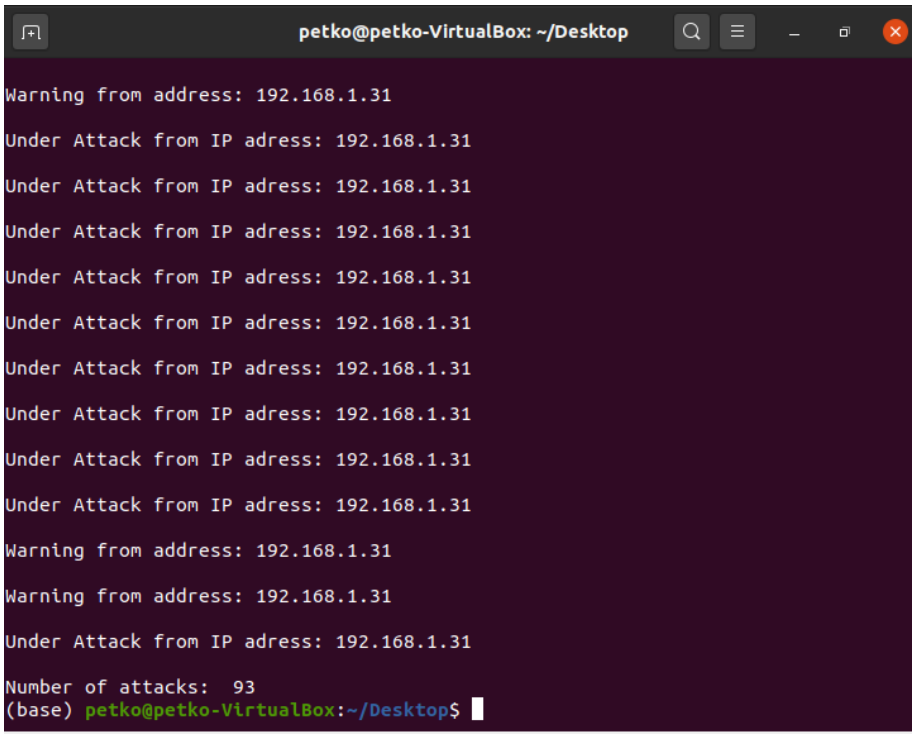
Pre potrebu presnej detekcie jednotlivých útočníkov bol taktiež vyvinutý skript, ktorý pracuje s IP adresami a časovými intervalmi v ktorých sa TCP Retransmission vyskytuje ako charakterizujúci prvok útoku.

Skript bol odskúšaný na lokálnych vzorkách, tak ako aj na verejne dostupných vzorkách. Ako testovacie dáta pre detekčný skript boli použité lokálne vzorky, s ktorými pracovala aj neurónová sieť. Dôvod bol taký, že práve tieto vzorky obsahovali škodlivý dátový tok spôsobený SlowDrop útokom.

Počiatočné testovanie prebehlo na lokálnych vzorkách ako je vidieť na obrázku 9.1. Vďaka SlowDrop útoku je frekvencia TCP Retransmission správ zvýšená v krátkych časových intervaloch a tak je detekčný skript schopný nájsť prebiehajúci útok.

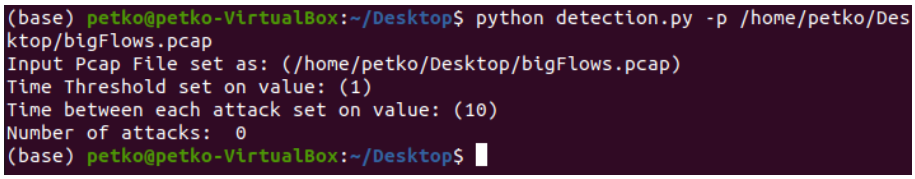
Testovanie prebehlo aj na verejne dostupných pcap súboroch stiahnutých z voľne dostupného repozitáru [23]. Ani jeden z voľne dostupných pcap súborov neobsahoval SlowDrop útok, a tak bolo umožnené testovanie aj na skutočnej vzorke dát. Obrázok 9.2 ukazuje výsledky testovania pre stiahnutý súbor bigFlows.pcap pre ktorý platia nasledujúce informácie:

- Veľkosť: 368 MB



Obr. 9.1: Výsledky detekčného skriptu, vzorka so SlowDrop útokom

- Počet paketov: 791615
- Spojení: 40656
- Priemerná veľkosť paketu: 449 bytov
- Dĺžka monitorovania: 5 minút
- Počet aplikácií: 12



Obr. 9.2: Výsledky detekčného skriptu, vzorka bez SlowDrop útoku

Z dôvodu, že súbor obsahuje množstvo TCP Retransmission správ v malých rozmeroch, kedy sa jedná o separované správy od jednotlivých klientov. Tým pádom sa nejedná o útok, pretože sa v tomto súbore nevyskytuje kontinuálna sekvencia jednotlivých TCP Retransmission správ, ktoré by sa objavovali v krátkom časovom intervale. Toto vedie k záveru, že skript neupozorňuje na legitimného užívateľa so slabším spojením. Preto je tento skript považovaný za úspešný a splniť svoj účel detekcie útočníka.

Výsledky vedú k záveru, že je potrebné škodlivý dátový tok filtrov a prípadne blokovať útočiace zariadenia. Táto problematika je nad rámec diplomovej práce, aj napriek tomu budú v krátkosti spomenuté návrhy ako problém riesiť.

9.1 Filtrovanie komunikácie

Posledná podkapitola sa venuje filtrovaniu komunikácie, ktorá nie je zahrnutá v skripte, ale z pohľadu praktického použitia je dôležité ju spomenúť. V prípade reálneho riešenia pre masívny DDOS útok je potrebné jednotlivé adresy okamžite filtrovať a zamedziť tak zvyšujúcej sa hrozbe a zahlteniu siete.

Keďže celá práca bola zameraná hlavne na detekciu SlowDrop útoku, výsledky sú rozdelené na dve časti:

- Rozpoznanie normálneho vs. škodlivého dátového toku
- Zoznam IP adries útočníkov

Oba tieto prístupy vedú k myšlienke "Čo urobiť s útočníkom, ak na nejakého narazíme?". Obecne najideálnejším prístupom by bolo monitorovať sieť a v menších časových intervaloch ako napríklad 5-30 sekúnd zaznamenávať dáta, ktoré sú následne použité pre monitorovanie siete a prípadnú detekciu útočníka. Jednotlivé pcap súbory poskytnuté Wiresharkom by následne smerovali do neurónovej siete, alebo detekčného skriptu. Týmto spôsobom by bolo možné kontinuálne sledovať dátový prenos na sieti. V prípade, že by sa medzi jednotlivými TCP spojeniami vyskytol útočník, jeho IP adresa bude po prebehnutí skriptu známa.

Výpis jednotlivých IP adries dáva možnosť sa proti útoku brániť. Jedným z možných riešení je filtrovanie, alebo blokovanie pomocou firewallu. Firewall poskytuje možnosť nastavenia pravidiel pre danú IP adresu, respektíve komunikáciu a tak zamedziť prechod škodlivej komunikácie.

Už konkrétna implementácia skriptu či nastavenia iných softwarov, ktoré by zabezpečovali vyriešenie tohoto problému vo forme zahadzovania, filtrovania paketov, alebo prípadnej blokácie jednotlivých IP adries nie je zahrnutá v práci.

Závěr

Cieľom diplomovej práce bolo teoreticky popísať problematiku DoS útoku s názvom SlowDrop. Druhý cieľ bol stanovený ako praktický pohľad na detekciu SlowDrop útoku, kde po teoretickom poňatí bolo treba vytvoriť simulačné prostredie a otestovať ho.

Teoretické informácie pochádzali primárne z práce autorov útoku [1]. Počiatočná časť práce sa zaoberá DoS a DDoS problematiku ako takou. Tento typ útokov je popísaný vo všeobecnosti a následne je pozornosť venovaná SlowDrop útoku ako takému. Prebraté sú jednotlivé efekty, držanie spojenia, zahadzovanie paketov a limity útoku.

Druhou časťou teoretickej časti bol hlbší pohľad do simulačného prostredia. Toto prostredie bolo navrhnuté na lokálnej sieti a bolo na ňom útok otestovaný. Taktiež v simulačnom prostredí prebehla celková analýza dát a tieto dáta boli použité neskôr ako vstupy do skriptov vytvorených v praktickej časti.

Záverečnou časťou teoretického segmentu boli informácie z predchádzajúcej diplomovej práce napísanej Ing. Mazánkom[7]. V tejto časti bol opísaný vytvorený skript SlowDrop útoku, ktorý bol použitý v práci.

Po rozobraní danej problematiky v teoretickej rovine sa práca zamerala na praktický pohľad. V prvej praktickej časti je pozornosť smerovaná na analýzu dát, ktoré boli získané zo SlowDrop útoku pomocou softwaru Wireshark. Otestované boli jednotlivé scenáre obsahujúce útok, ale aj komunikácia bez útoku, aby bolo možné porovnať dve dátové sady. Analýza smerovala k nájdeniu charakteristických znakov útoku, ktoré ďalej viedli k vytvoreniu vstupných dát pre jednotlivé skripty a neurónovú sieť. Primárnou charakteristikou útoku je indikácia TCP Retransmission, ktorej je venovaná celá podkapitola s podrobnejších opisom a podmienkami, ktoré musia byť splnené, aby sa táto indikácia vyskytla.

Praktická časť pokračovala testovaním detekčných signatúr pomocou softwaru Suricata. Jednotlivá štruktúra a formát zápisu konkrétnych signatúr boli v práci detailne popísané a taktiež uvedené použité príklady signatúr s účelom detekcie SlowDrop útoku. Pomocou signatúr sa útok nepodarilo detekovať a to z dôvodu, že software Suricata neumožňuje vytvorenie signatúr, ktoré by dokázali rozoznať opakujúcu sa frekvenciu TCP správ v dátovom toku.

Treťou a zároveň najrozsiahlejšou časťou diplomovej práce je kapitola o neurónových sieťach. Táto kapitola nadväzuje na výsledky detekcie signatúr pomocou Suricaty a výstupy z analýzy dát, ktoré následne spracováva a používa ich ako vstupné dáta pre neurónovú sieť. V kapitole o neurónových sieťach je popísané spracovanie jednotlivých dát a celkový ich prechod cez neurónovú sieť. Detail a pozornosť je venovaná každej časti tohoto procesu s detailnejších popisom a prehľadným zobrazením pomocou obrázkov. Sieť je navrhnutá pomocou autoencoderu, ktorý má za úlohu

rekonštrukciu samotných vzoriek a dáva tak možnosť pre porovnávanie originálneho a zrekonštruovaného vzorku. Celkový proces rekonštrukcie a následného porovnávania patrí medzi výsledky neurónovej siete. Presnosť detekcie bežného dátového toku dosahovala od 93,37% až do 98,77&. V prípade spojení s anomáliou sa presnosť detekcie pohybovala od 9,44% až do 49,07%. Rozdieľnosť nameraných hodnôt je spôsobená meniacim sa počtom legitimných klientov a útočníkov. Pre reálne použitie by bolo sieť potrebné otestovať na verejných vzorkách, ktoré by obsahovali SlowDrop útok.

Poslednou časťou praktickej časti je opis skriptu, ktorý má za úlohu detekciu útočníka ako takého. Skript spracováva výstupný pcap súbor zo softwaru Wireshark, ktorý následne pomocou softwaru Tshark filtruje a vyberá tak iba podstatné dáta potrebné pre detekciu útočníka SlowDrop útoku. Skript sa zameriava už konkrétnejšie na frekvenciu opakovaní jednotlivých TCP chýb v sieťovej komunikácii. V prípade zaznamenania zvyšujúceho sa výskytu TCP Retransmission chyby v určenom časovom intervale, skript sleduje a zaznamenáva IP adresy klientov, u ktorých sa tento problém objavuje. Tento zoznam obsahujúci IP adresy je následne aj výstupom skriptu a dokáže pomôcť pri filtrovaní komunikácie s týmito zariadeniami. Detekcia útoku pomocou skriptu bola úspešná a zobrazuje IP adresu útočníka. Skript bol otestovaný aj na verejných vzorkách, kde aj v prípade výskytu TCP Retransmission paketov na útok neupozornil, pretože tieto vzorky útok neobsahovali. Z toho dôvodu je vhodnou metódou pre detekciu útoku.

Po uzavretí praktickej časti obsahujúcej analýzu dát, detekciu signatúr, neurónovú sieť a detekčný skript sú jednotlivé výsledky zhrnuté a porovnané medzi sebou. Kapitola uvádza taktiež percentuálnu úspešnosť detekcie neurónovej siete. Zo získaných výsledkov sú odvodené výhody a nevýhody jednotlivých software-ov a metód použitých pri riešení tejto problematiky. Výsledky detekčného skriptu z lokálnej, ale aj verejnej vzorky sú zobrazené na obrázkoch.

Záverečná časť práce nadväzuje na získané výsledky a upiera pohľad na dané riešenia spojené s problematikou. V krátkej forme sú uvedené jednotlivé návrhy a riešenia, ktoré pomáhajú obeti brániť sa v prípade útoku.

Z troch použitých metód sa útok nepodarilo zaznamenať iba pomocou signatúr. Metóda neurónovej siete a detekčného skriptu útok zaznamenala, aj keď v rozdielnych formách. Neurónová sieť dokázala rozlíšiť bežný od škodlivého dátového toku v niektorých prípadoch až 98,77% pre bežné dáta a 49,07% pre anomálie. Avšak na druhej strane detekčný skript dokázal upozorniť na útok, aj s uvedením IP adresy útočníka.

Pre zvýšenie presnosti oboch metód by bolo potrebné zaobstarať masívnu vzorku z globálnych sietí. Táto vzorky by mala obsahovať SlowDrop útok, ale zároveň by bolo potrebné, aby sa výskyt spojení s anomáliou v dátovom toku nachádzal vo väč-

šej škále. Možnosťou by bolo upravovať vstupné parametre ako napríklad dĺžka TCP spojenia. V prípade rozdielnej dĺžky by bolo potrebné upraviť aj model neurónovej siete a jeho jednotlivé vrstvy.

Pravidelným monitorovaním siete pomocou neurónovej siete je možné zachytiť útok a následne pomocou detekčného skriptu upozorniť na útočníka. V reálnej situácii by bolo potrebné monitorovať sieť v krátkych časových intervaloch napr. 30 sekúnd. Ako ďalší vývoj práce by bolo vhodné obe metódy prepojiť. Získané dáta z monitorovania následne použiť ako vstupné dáta do tohoto systému, ktorému by bolo vhodné v konečnom dôsledku implementovať filtrovanie škodlivej komunikácie.

Literatura

[1] CAMBIASO, Enrico, Giovanni CHIOLA a Maurizio AIELLO, 2019. *Introducing the SlowDrop Attack* Computer Networks [online]. Amsterdam, Elsevier, 26 February 2019. DOI: https://doi.org/10.1016/j.comnet.2019.01.007. ISSN 13891286. Dostupné z URL: <https://linkinghub.elsevier.com/retrieve/pii/S1389128619300210>.

[2] CloudFlare: *What is a DDoS attack?* [online]. Dostupné z URL: <https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/>.

[3] SUMNER, Ryan, 2019.: *What is a DDoS Attack?* [online]. Dostupné z URL: <https://www.ibm.com/cloud/blog/video-what-is-a-ddos-attack>.

[4] ANEXIO Inc.: *What's A DDoS Attack & How Does It Work?* [online]. Dostupné z URL: <https://www.anexio.com/whats-ddos-attack-work/>.

[5] VORMAYR, Gernot, Tanja ZSEBY, Joachim FABINI, 2013: *Botnet Communication Patterns. IEEE Xplore* [online]. 19(4), 2768 - 2796. DOI: 10.1109/COMST.2017.2749442. ISSN 1553-877X Dostupné z URL: <https://ieeexplore.ieee.org/document/8026031/>.

[6] EC-Council|Blog *BOTNETS AND THEIR TYPES* [online]. 2020 Dostupné z URL: <https://blog.eccouncil.org/botnets-and-their-types/>.

[7] MAZÁNEK, Pavel: *Modelování a detekce útoku SlowDrop* [online]. Brno, 2020. Dostupné z URL: <www.https://www.vutbr.cz/studenti/zav-prace/detail/125969>. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Marek Sikora.

[8] BOGDANOSKI, Mitko, Tomisla,v SHUMINOSKO Aleksandar RISTESKI 2013.: *Analysis of the SYN flood DoS attack* [online]. DOI: 10.5815/ijcnis.2013.08.01 Dostupné z URL: <https://www.researchgate.net/publication/237076791_Analysis_of_the_SYN_flood_DoS_attack>.

[9] Canonical Ltd: *Suricata* [online]. Dostupné z URL: <https://suricata-ids.org/>.

[10] CAMBIASO, Enrico, Gianluca PAPALEO, Giovanni CHIOLA a Maurizio AIELLO, 2013. *Slow DoS attacks: definition and categorisation.* Internatio- nal Journal of Trust Management in Computing and Communications [online]. October 2013. DOI: 10.1504/IJTMCC.2013.056440. ISSN 2048-8378. Dostupné z URL: <http://www.inderscience.com/offer.php?id=56440>.

[11] CloudFlare: *What is a low and slow attack?* [online]. Dostupné z URL: <https://www.cloudflare.com/learning/ddos/ddos-low-and-slow-attack/>.

[12] OISF: *Ubuntu* [online]. Dostupné z URL: <https://ubuntu.com/>.

[13] DHANAPAL, A, P. NITHYANANDAM 2019.: *The Slow Http Distributed Denial of Service Attack Detection Incloud* [online]. DOI 10.12694/scpe.v20i2.1501. ISSN 1895-1767. Dostupné z URL: <https://www.researchgate.net/publication/332829479_The_Slow_HTTP_Distributed_Denial_of_Service_Attack_Detection_in_Cloud>.

[14] SHEKYAN, 2011: *SlowHTTPTest: SlowHTTPTest Package Description. KALI TOOLS* [online]. Dostupné z URL: <https://tools.kali.org/stress-testing/slowhttptest>.

[15] KLIMEŠ, Jan: *Filtrace útoků na odepření služeb* [online]. Brno, 2019. Dostupné z URL: <https://www.vutbr.cz/studenti/zav-prace/detail/118197>. Bakalárska práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Tomáš Gerlich.

[16] STRAKOŠ, Jan: *Detekce anomálií pomocí neuronových sítí* [online]. Brno, 2019. Dostupné z URL: <https://www.vutbr.cz/studenti/zav-prace/detail/118101>. Bakalárska práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Petr Blažek.

[17] Wireshark : *Wireshark* [online]. Dostupné z URL: <https://www.wireshark.org/>.

[18] Anaconda : *Anaconda* [online]. Dostupné z URL: <https://www.anaconda.com/>.

[19] Jupyter : *Jupyter* [online]. Dostupné z URL: <https://jupyter.org/>.

[20] Spyder : *Spyder* [online]. Dostupné z URL: <https://www.spyder-ide.org/>.

[21] Tensorflow : *Tensorflow* [online]. Dostupné z URL: <https://www.tensorflow.org/>.

[22] Keras : *Keras* [online]. Dostupné z URL: <https://keras.io/>.

[23] Tcpreplay : *Sample Captures* [online]. Dostupné z URL: <https://tcpreplay.appneta.com/wiki/captures.html>.

[24] BROWNLEE, Jason : *Overfitting and Underfitting With Machine Learning Algorithms* [online]. 2016 Dostupné z URL: <https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/>.

[25] Simplilearn : *Top 10 Deep Learning Frameworks You Should Know in 2021* [online]. Dostupné z URL: <https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-frameworks/>

[26] Towards data sience : *Applied Deep Learning - Part 3: Autoencoders* [online]. Dostupné z URL: <https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798>

[27] IETF : *Computing TCP's Retransmission Timer* [online]. Dostupné z URL: <https://tools.ietf.org/html/rfc6298>

[28] Offensive Security: *MetaSploit Unleashed* [online]. Dostupné z URL: <https://www.offensive-security.com/metasploit-unleashed/>

[29] JELÍNEK, Michael: *Parametrizace síťových útoků* [online]. Brno, 2019. Dostupné z URL: <https://www.vutbr.cz/studenti/zav-prace/detail/118096>. Bakalárska práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Petr Blažek.

[30] AHIRE, Jayesh Bapu, 2018: *The Artificial Neural Networks Handbook: Part 4* [online]. Dostupné z URL: <https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e>.

[31] CHALUPNÍK, Vitalij: *Biologické algoritmy (4) – Neuronové sítě.* [online]. Dostupné z URL: <https://www.root.cz/clanky/biologicke-algoritmy-4-neuronove-site/>.

[32] BAŽÍk, Martin: *Oprimalizace hlubokých neurónových síti* [online]. Brno, 2018. Dostupné z URL: <https://www.vutbr.cz/studenti/zav-prace/detail/114839>. Bakalárska práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačových systémov. Vedoucí práce Prof. Ing. Lukáš Sekanina, Ph.D.

[33] LUKÁČ, Michal: *Hlboké neurónové siete pre spracovanie multimédií* [online]. Brno, 2016. Dostupné z URL: <https://is.muni.cz/th/r1pur/thesis.pdf>.

[34] MATISKO, Maroš: *Neuronové sítě pro detekci anomálií v síťové komunikaci* [online]. Brno, 2018. Dostupné z URL: <https://www.vutbr.cz/studenti/zav-prace/detail/110194>. Bakalárska práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Petr Blažek.

[35] Data Sience : *Keras difference beetween val_loss and loss during training* [online]. Dostupné z URL: <https://datascience.stackexchange.com/questions/25267/keras-difference-beetween-val-loss-and-loss-during-training>.

Seznam symbolů, veličin a zkratek

DoS	Denial of Service
DDoS	Distributed Denial of Service
MSF	MetaSploit Framework
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
ICMP	Internet Control Message Protocol
DHCP	Dynamic Host Control Protocol
DNS	Domain Name System
CnC	Command and Control
RE	Reconstruction Error

Seznam příloh

A Použitie neurónovej siete

Prerekvizity pre použitie **neurónovej siete**:

- Python 3
- Framework Anaconda
  - kniženice keras, tensorflow
  - modul Jupyter Notebook

Po stiahnutí a nainštalovaní frameworku Anaconda je potrebné importovať prostredie *environment.yml*, ktoré v sebe obsahuje všetky použité knižnice. Následne je potrebné spustiť modul Jupyter Notebook a vytvoriť nový projekt v jazyku Python a otvoriť a *neural_network.py*. Dáta je najprv potrebné spracovať skriptom *data_processing.py*. Skript bol implementovaný a otestovaný v operačnom systéme Linux.

Príklad použitia skriptu na parsovanie dát pre neurónovú sieť:
python3 data_processing.py -i input_file.csv -o output_file.csv

Povinné argumenty: *-i* file.csv *-o* file.csv

- vypísanie informácií ako spustiť program *-h*
- vstupný csv súbor ako výstup z wiresharku/tsharku *-i* file.csv
- výstupný csv súbor následne použitý pre neurónovú sieť *-o* file.csv

Vytvorený *csv* súbor je potrebné uviesť ako vstupný súbor na začiatku skriptu miesto defaultne nastaveného súboru *vut_data.csv*.

B Použitie detekčného skriptu

Prerekvizity pre použitie **detekčného skriptu**:

- Python 3

Skript bol implementovaný a otestovaný v operačnom systéme Linux.

Príklad použitia skriptu na detekciu útočníkov:
python3 detection.py -p input_pcap_file.pcap
python3 detection.py -p input_pcap_file.pcap -t 2 -a 15

Povinné argumenty: *-p* file.pcap

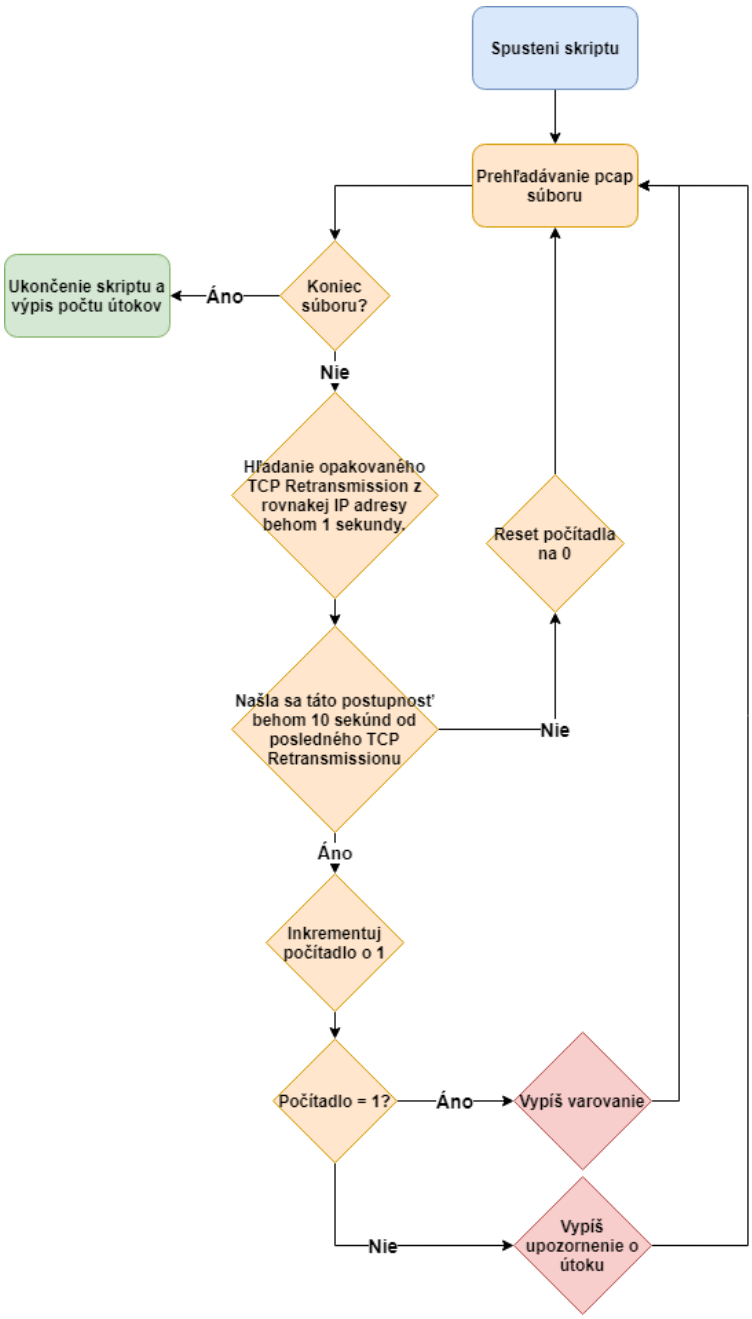
- vypísanie informácií ako spustiť program *-h*
- vstupný pcap súbor ako výstup z wiresharku/tsharku *-p* file.pcap
- časový interval medzi chybnými packetmi -t seconds (default *-t* 1)
- časový interval medzi jednotlivými útokmi -a seconds (default *-a* 10)

So zvyšujúcou sa intenzitou útoku a väčším počtom útočníkov sú odporúčané kladné hodnoty blízke 0, ako napríklad *-t* 1 *-a* 3.

C Priložené súbory

/ ..... koreňový adresár zipovej zložky priložených súborov
data_processing.py .... súbor skriptu spracovania dát pre neurónovú sieť v jazyku Python
detection.py ..... súbor skriptu detekcie útoku v jazyku Python
script.sh ..... pomocný skript v jazyku bash
neural_network.py ..... súbor skriptu neurónovej siete v jazyku Python
environment.yml ..... prostredie z frameworku Anaconda
testdata.pcap ..... lokálna vzorka z Wiresharku
testdata2.pcap ..... lokálne vzorka 2 z Wiresharku
README.md ..... textový súbor s informáciami o skriptoch

D Vývojový diagram detekčného skriptu



Obr. D.1: Vývojový diagram detekčného skriptu