VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCIA ANOMÁLIÍ PRE IDS SYSTÉMY

DETECTION OF ANOMALIES FOR IDS SYSTEMS

BAKALÁRSKA PRÁCA
BACHELOR'S THESIS

AUTOR PRÁCE **JOHANN ADAM GAWRON**
AUTHOR

VEDÚCI PRÁCE **Mgr. Ing. PAVEL OČENÁŠEK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

148443

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Gawron Johann**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Detekce anomálií IDS systémy**
Kategorie:	Počítačové sítě
Akademický rok:	2022/23

Zadání:

1. Seznamte se s principy analýzy anomálií v prostředí systémů pro detekci síťového průniku (Network Intrusion Detection System).
2. Analyzujte požadavky na systém umožňující analýzu toků a událostí v rámci systémů IDS/IPS s otevřeným zdrojovým kódem.
3. Navrhněte systém pro detekci anomálií dle instrukcí vedoucího práce, který bude založen na umělé inteligenci.
4. Navržený systém implementujte.
5. Implementovaný systém ověřte na vhodně zvolených reálných datech.
6. Diskutujte získané výsledky a možnosti dalšího rozšíření.

Literatura:

- Kurose, J. F. Computer networking: A top-down approach. Pearson, Essex, 2017, ISBN 978-1-292-15359-9.
- Stallings, W. Network security essentials: Applications and standards. Hoboken, 2016, ISBN 978-0-13-452733-8.
- Bishop, M. Computer security: Art & Science. Addison-Wesley, Boston, 2003, ISBN 0-201-44099-7.
- Ahmed, M., Mahmood Naser, A., Hu, J. A survey of network anomaly detection techniques. Journal of network and computer applications. Elsevier, 2016, 60(C), s. 19-31. ISSN 1084-8045.
- Buczak, A., Guven, E.. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications surveys and tutorials. IEEE, 2016, 18(2), s. 1153-1176.

Při obhajobě semestrální části projektu je požadováno:
Body 1 - 3 zadání.

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

Vedoucí práce:	**Očenášek Pavel, Mgr. Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	17.5.2023
Datum schválení:	28.10.2022

Abstrakt

Cieľom tejto práce je zoznámiť sa s problematikou detekcie anomálií na sieťovej prevádzke pomocou umelej inteligencie. Navrhnúť a následne implementovať metodiku na vytvorenie klasifikátoru anomálií na profile sieťovej komunikácie. Klasifikačná metóda by mala byť schopná čo najefektívnejšie a najpresnejšie identifikovať anomálie na sieťovej prevádzke tak, aby sa predošlo generácii nesprávnych výstupov. Pri rešeršé problematiky boli preskúmané systémy IDS, rôzne typy útokov a prístupy k detekcii a klasifikácii anomálií. Pri vyhodnocovaní efektívnosti boli preskúmané a použité viaceré štandartné metódy na vyjadrenie kvality klasifikátorov.

Abstract

The goal of this thesis is to familiarize myself, and the reader, with the issues surrounding anomaly detection in network traffic using artificial inteligence. To propose and subsequently implement a methodology for creating an anomaly classifier for network communication profiles. The classification method should be able to efficiently and accurately identify anomalies in network traffic to avoid generating false outputs. During the research of the issue, IDS systems, various types of attacks, and approaches to anomaly detection and classification were examined. In evaluating the effectiveness, several standard methods were examined and used to express the quality of classifiers.

Kľúčové slová

Strojové, Učenie, Klasifikátor, Klasifikačné, Metódy, Anomálie, Sieť, Sieťová, Komunikácia, Provoz, Prevádzka, IDS, Systém, Detekcia, Anomália, Útok, XGBoost.

Keywords

Machine, Learning, ML, Classificator, Ćlassification, Method, Anomalies, Anomaly, Network, Intrusion, Detection, IDS, System, Communication, Traffic, XGBoost.

Citácia

GAWRON, Johann Adam. *Detekcia Anomálií pre IDS systémy.* Brno, 2023. Bakalárska práca. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedúci práce Mgr. Ing. Pavel Očenášek, Ph.D.

Detekcia Anomálií pre IDS systémy

Prehlásenie

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

.....
Johann Adam Gawron
17. mája 2023

Poďakovanie

Týmto by som chcel poďakovať vedúcemu práce Mgr. Ing. Pavlovi Očenáškovi Ph.D. za vedenie práce, pedagogickú pomoc a spätnú väzbu.

Ďalej by som chcel poďakovať odbornému konzultantovi Petrovi Chmelařovi za odbornú pomoc, ľudský prístup a kvalitné konzultácie, ktoré mi pomohli zvládnuť problematiku práce.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

S neustále rýchlejšie narastajúcou závislosťou na počítačových sieťach, v rôznych aspektoch modernej spoločnosti, sa stáva ochrana týchto systémov čoraz väčšou nutnosťou. To však, ako mnohé iné problémy, nie je riešené centrálne a systematicky, ale skôr ad hoc. V dnešnej dobe je všeobecne nasadzované stále väčšie množstvo nástrojov na správu sietí, ktoré samotné rastú v objeme. Taktiež je možné sledovať trend, kde sa takmer na každý vektor útoku tvorí nový nástroj a nasadzuje sa viac a viac rôznych agentov na koncové body. Tým veľmi narastá komplexita prostredí a nároky na ich správu. Pre kyberbezpečnostných expertov sa neustále stáva čoraz väčším problémom zahltenie veľkým množstvom bezpečnostných incidentov. To môže, v horšom prípade, viesť k tomu, že v dôsledku únavy z incidentov môžu byť skutočné hrozby prehliadnuté, pretože si ich zodpovedná osoba v záplave falošných pozitív nevšimne. Koreň tohto fenoménu, cudzojazyčne taktiež nazývaný `alert fatigue`, vieme dohľadať už na samom začiatku, a to pri generovaní incidentov rôznymi bezpečnostnými nástrojmi.

Jedným z nástrojov používaných na ochranu sietí sú práve systémy detekcie útokov IDS[1]. Spolu so systémami na prevenciu útokov IPS[2] tvoria chrbtovú kosť bezpečnostnému prvku NDR[3], ktorý zodpovedá za rýchlu, spoľahlivú detekciu a reakciu na hrozby v sieti. IDS je zodpovedné za detekciu týchto hrozieb. So zvyšujúcou sa komplexitou sietí a útokov však narastá aj potreba čoraz viac prepracovaných detekčných metód.

Cieľom tejto práce je hlbšie porozumenie problematike ochrany sietí pomocou nástrojov sieťovej detekcie útokov. Presnejšie oblastiam, ako automatizovaná generácia anomálií, princípy ich analýzy, tvorby bezpečnostných incidentov a zefektívneniu týchto procesov. Dosiahnuť toho chcem formou experimentácie s trénovaním modulu s využitím výstupov z open-source platformy Suricata[4]. Tento modul bude postavený na algoritmoch strojového učenia, ktoré budú v prijímať dátové toky zo siete a vyhodnocovať ich abnormalitu.

Na začiatku mojej práce vysvetlím podrobnejšie IDS systémy, ako fungujú, a ako sa implementujú v rámci sietí. Následne priblížim problematiku anomálií a klasifikačných metód. Taktiež stručne popíšem existujúce riešenia. Na základe zistených výsledkov navrhnem a implementujem klasifikátor pomocou algoritmov strojového učenia, overím jeho validitu a vyhodnotím či je dostatočne efektívny, aby po jeho implementácií neprispieval ku fenoménu `alert fatigue`. Nakoniec zhodnotím výhody a nevýhody môjho riešenia a navrhnem možnosti jeho vylepšenia.

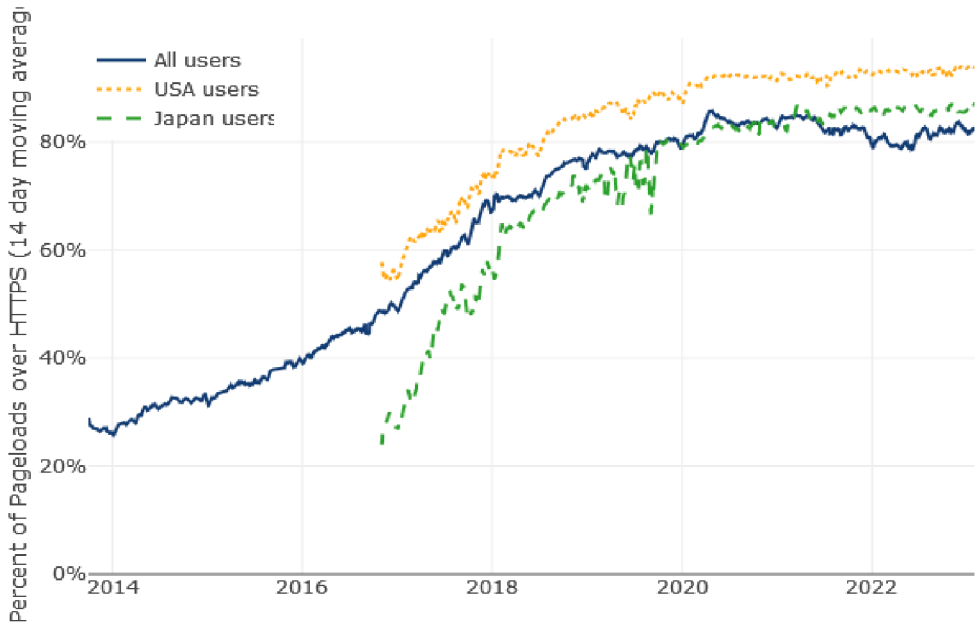
[1] IDS (Intrusion Detection System) - Systémy na detekciu útokov
[2] IPS (Intrusion Prevention Systems) - Systémy na prevenciu útokov
[3] NDR (Network Detection and Response) - Sieťová detekcia a reakcia
[4] Suricata https://suricata.io

Kapitola 2

Systémy na detekciu útokov - IDS

Systémy na detekciu útokov (ďalej IDS) sú bezpečnostné nástroje, často implementované na dedikovaných serveroch, ktoré monitorujú sieťovú prevádzku a aktíva v sieti[8].

Systémy IDS sú dôležitou súčasťou bezpečnosti sietí, pretože poskytujú rýchlu detekciu aktivít útočníkov v sieti. Skorým upovedomením správcu systému napomáhajú včasnej reakcii a minimalizácii škody, ktorú útočník môže napáchať. Najväčším problémom pre IDS systémy je náročnosť ich konfigurácie a správy. Aby fungovali efektívne tak je nutné zaistiť periodické aktualizácie a validáciu výstupov, inak sa môže stať, že budú systémy generovať falošne pozitívne alebo falošne negatívne incidenty.



Obr. 2.1: Percento načítaných stránok cez HTTPS[3]

Na obrázku 2.1 môžeme vidieť percento načítaných stránok pomocou TLS/SSL[1] protokolov. Toto je paradoxne jedným z najväčších problémov nie len pre útoční-

[1]TLS/SSL (Transport Layer Security/Secure Sockets Layer) je kryptografický protokol, ktorý zabezpečuje bezpečnú komunikáciu medzi klientom a serverom na internete.

kov ale aj pre bezpečnostných inžinierov. Ak sa totiž podarí útočníkovi inicializovať šifrované spojenie do siete, bez dešiforavania TLS prevádzky nie je možné štandardnými metódami detegovať útok. Až 82% všetkých užívateľov používa TLS šifru na HTTP[2] služby.

V nasledujúcich častiach tejto kapitoly stručne popíšem hlavné typy IDS systémov. Začnem, tradične dôležitejšími, rozdielmi medzi hostiteľskými a sieťovými IDS spolu s demonštračnou ukážkou zapojenia zložitejšieho systému a následne porovnám IDS založené na signatúrach a na anomáliách.

2.1 Sieťové a Hostiteľské IDS

Systémy IDS sa primárne delia na sieťové IDS (NIDS), a Hostiteľské (HIDS).

NIDS sa primárne implementujú vo forme sónd, do ktorých je pomocou SPAN[3]/TAP[4] rozhraní zrkadlená sieťová prevádzka, alebo nasadením priamo na linku (režim in-line) komunikácie zbierajú a vyhodnocujú dátové toky. Takto zozbierané informácie sa vo forme tokov (flows), alebo iných proprietárnych formátov obohatených kolekcií metadát, zasielajú ďalej na prvok kolektoru, ktorý slúži na stredne až dlhodobú úschovu určenú pre potreby retrospektívnej analýzy. V prípade, že sa analýzou v reálnom čase odhalí podozrivá činnosť na sieti, je do centrálnej konzole zodpovednej za systém zasielaný vygenerovaný incident, ktorý obsahuje metadáta spolu s informáciou o kategorizácii podozrivého správania.

HIDS monitorujú, často vo forme softwarových agentov, aktivitu na jednotlivých koncových bodoch v sieti. Môže sa jednať o rôzne aktíva od osobných či firemných počítačov, cez servery až po rôzne IoT[5] zariadenia či operačné technológie (napríklad ovládanie ventilu plynovodov). Dáta získané týmto spôsobom taktiež putujú do kolektoru a do manažment konzole.

Administrátori siete alebo bezpečnostní analytici potom cez GUI[6] môžu pristupovať ku centrálnej konzole systému a prehľadávať záznamy a vygenerované incidenty. V prípade, že má systém implementovaný aj kolektor, poskytuje často práve konzola možnosť jednoduchého prehľadávania jeho dát. Túto konzolu je možno ďalej prepojiť so systémami na zber informácií a udalostí SIEM[7].

2.1.1 Základný popis architektúry

V dnešnej dobe je časté, že sa IDS systémy kombinujú so systémami IPS a vznikajú NDR platformy. Tieto zohrávajú veľmi dôležitú úlohu pri ochrane siete a sú chrbticou robustnejších SOAR[8] systémov. To už je však vysokoúrovňový prístup, ktorý nepatrí do rozsahu tejto práce.

[2]HTTP (Hypertext Transfer Protocol) je protokol, ktorý sa používa na komunikáciu medzi webovým prehliadačom a webovým serverom.

[3]SPAN (Switched Port Analyzer) - dedikované rozhranie na prepínači, ktoré slúži na zrkadlenie prevádzky

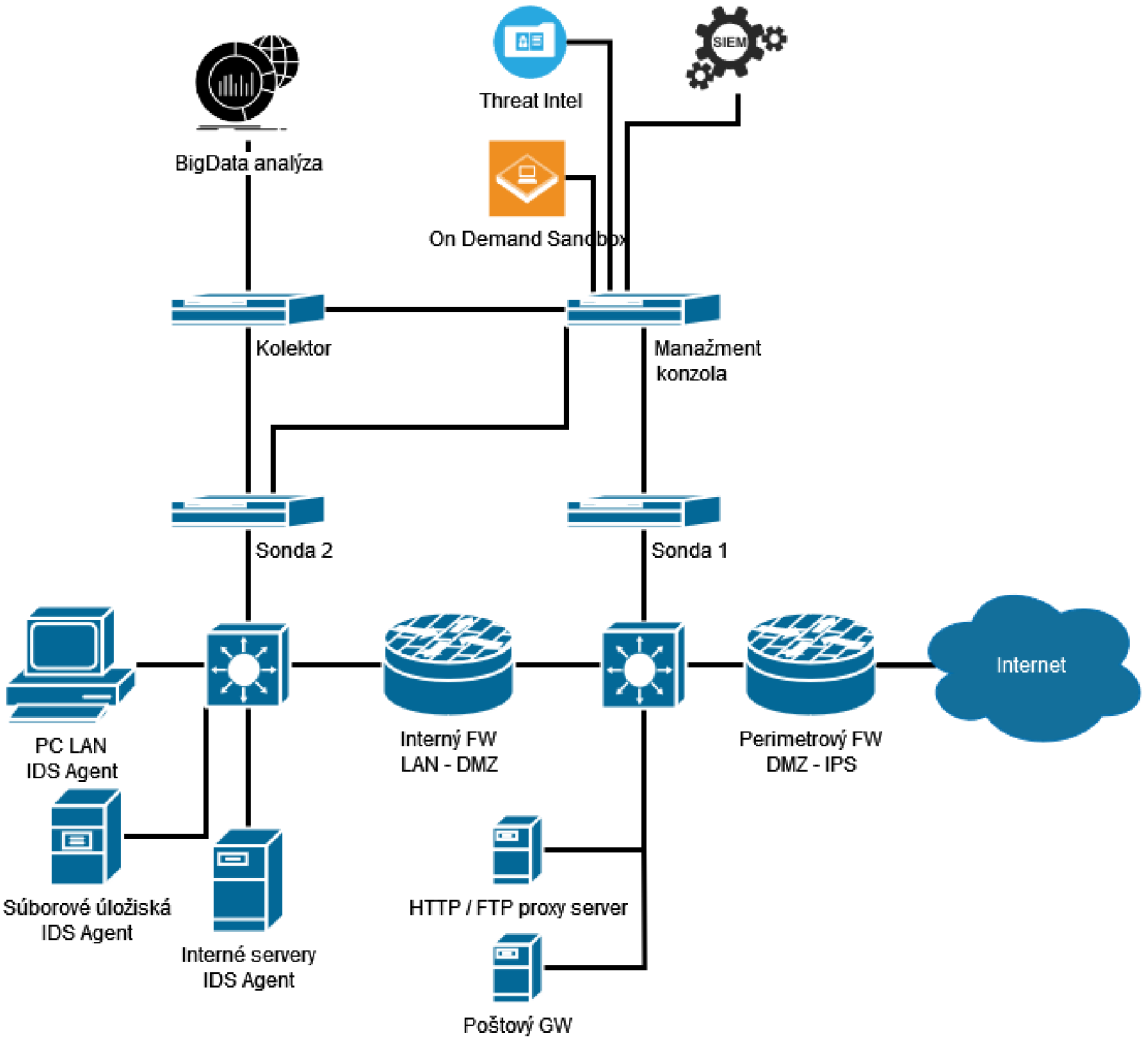
[4]Test Access Point - modernejšia alternatíva rozhrania SPAN

[5]IoT (Internet of Things) je súbor vzájomne prepojených fyzických zariadení, ktoré majú schopnosť komunikovať a vymieňať si dáta pomocou internetu.

[6]GUI (Graphical User Interface) - grafické užívateľské rozhranie

[7]SIEM (Security Information and Event Management) je bezpečnostný systém, ktorý slúži na monitorovanie a správu bezpečnostných udalostí a informácií.

[8]SOAR (Security Orchestration, Automation and Response) - systémy, ktorými sa implementuje centrálny prístup ku zabezpečeniu siete od zberu incidentov až po automatizáciu ich riešenia



Obr. 2.2: Sieťová architektúra nasadenia IDS systému

Na obrázku 2.2 môžeme vidieť vcelku robustné no štandardné nasadenie IDS systému v režime mimo linky - iba počúva. Je nutné poznamenať, že samostatne nasadzované IDS systémy slúžia len na poskytnutie viditeľnosti do systému. Vyššie znázornené zapojenie by bolo optimálnejšie pre kombinované, hybridné IDS + IPS systémy, ktoré okrem viditeľnosti poskytujú aj veľkú mieru kontroly nad sieťou a s tým spojené reakčné schopnosti (napr. automatizovanie aktualizácie pravidiel Firewall (FW) alebo pre in-line zapojenie možnosť vynútiť ukončenie TCP[9] komunikácie zaslaním reset (RST) paketu). Manažment konzola a Kolektor sú v tejto schéme rozdelené inštancie, môže sa jednať o virtualizované stroje, alebo dedikovaný hardware.

V schéme je znázornené zapojenie dvoch sond. Sonda 1, ktorá sa nachádza v DMZ[10] zóne, kde sa nachádzajú viac otvorené servery, ktoré vyžadujú vyššiu úroveň prístupnosti do a z Internetu. Množstvo vygenerovanej sieťovej prevádzky tu bude značne vyššie, kvôli tomu, že sa nachádza pred interným FW. Je napojená na Manažment konzolu, do ktorej zasiela prípadné bezpečnostné incidenty. Sonda 2 predstavuje senzor v internej sieti, kde

[9]TCP (Transmission Control Protocol) je spojovaný protokol, ktorý zabezpečuje spoľahlivý prenos dát

[10]DMZ (Demilitarized Zone) predstavuje izolovanú zónu, ktorá sa nachádza medzi internetom a vnútornou sieťou organizácie.

analyzuje menšie množstvo kritickejšej komunikácie. Preto je napojená aj na kolektor, kam zasiela incidenty vždy, keď ich posiela aj do manažment konzole. Týmto zapojením sa docielil dôležitý krok ku optimalizácii IDS systému, ktorý umožňuje incidentom zo sondy 1 dávať nižšiu prioritu z dôvodu, že incidenty vygenerované v DMZ majú vyššiu pravdepodobnosť byť falošne pozitívne.

Prvky v internej sieti ako PC LAN, ktorý predstavuje sieť počítačov, majú na sebe nainštalovaných agentov, ktorí komunikujú s centrálnou konzolou a poskytujú informácie o bezpečnostných incidentoch a obohacujú kontextové dáta z pohľadu koncových bodov. Nasadením HIDS agentov sme schopný bez zasahovania do siete získať dôležitý vektor pohľadu na, už koncovým bodom dešifrovanú, komunikáciu. Týmto sa čiastočne vyhýbame nutnosti využívať technológie ako MITM [11] TLS inšpekcia[13] a iné, na systémové zdroje náročné, dešifrovanie TLS prevádzky.

2.1.2 Všeobecné zhrnutie

Sieťové IDS

Väčšina komerčných projektov patrí medzi sieťové IDS. Tieto systémy detegujú útoky zberom a analýzou paketov. Ako bolo na vyššie uvedenej ukážke znázornené, tieto systémy často pozostávajú zo súpravy senzorov, ktoré sú rozmiestnené na kritických bodoch v sieti, ktoré treba monitorovať. Lokálne analyzujú prevádzku a prípadné identifikované bezpečnostné incidenty posielajú do centrálnej konzole.

Výhody NIDS

- Pár dobre implementovaných senzorov vie pokryť celú sieť.
- Nasadenie nových senzorov vo väčšine prípadov predstavuje minimálny zásah do existujúcej siete. Väčšina senzorov je pasívnym prvkom, na ktorý sa zrkadlí prevádzka cez SPAN/TAP rozhranie.
- Veľmi jednoducho zabezpečiteľné a takmer neviditeľné pre útočníkov.

Nevýhody NIDS

- Problémy pri špičkách v prevádzke, nie vždy stačí výkon strojov na spracovanie veľkého množstva dát – môže spôsobiť veľkú dieru v obrane, keď je preťažený sieť.
- Komplikácie pri sieťach, ktoré používajú veľké množstvo prepínačov na vytvorenie veľkého množstva menších lokálnych sietí.
- Bez zložitých prídavných modulov nedokážu pracovať so šifrovanou komunikáciou.
- Nevedia zistiť výsledok útoku, s istotou vedia len oznámiť začiatok útoku.

Hostiteľské IDS

Zbierajú dáta z individuálnych aktív v sieti. Táto pozícia umožňuje spoľahlivo a s vysokou presnosťou analyzovať aktivity. Využívajú najmä auditné logy operačných systémov (aj z úrovne kernelu) a iné systémové logy. Systémové logy sú jednoduchšie na spracovanie vďaka ich menším veľkostiam. Mnohé HIDS pracujú s centralizovaným manažmentom, čím sa analytikom umožňuje z jednej konzole kontrolovať mnohé sieťové aktíva.

[11] Man in the middle - človek uprostred. Jedná sa o typ útoku, kedy útočník odpočúva komunikáciu medzi dvoma stranami a môže sa ju pokúsiť narušiť alebo zneužiť.

Výhody HIDS

- Vďaka viditeľnosti priamo do systémových logov sú schopné detegovať aj hrozby, ktoré NIDS nevie rozpoznať.
- Keďže sa nachádzajú na koncových bodoch, tak vidia dáta komunikáciu predtým, ako sú zašifrované, a potom ako ich systém dešifruje.
- Nezáleží im na topológii celkovej siete.
- Sú schopné vidieť na všetky systémové udalosti, od premenovania súboru, cez zmeny v registroch až po nerovnosti v poradí procesov, a teda vedia detegovať útoky softvérového pôvodu.

Nevýhody HIDS

- Zložitejšie na manažovanie (nadnárodné firmy s tisíckami zamestnancov) a ich implementácia na koncových zariadeniach prirodzene znamená, že využívajú ich systémové prostriedky a uberajú tak na výkone daných zariadení.
- Sú náchylné na cielené útoky zo strany útočníka, ktorému po deaktivácii agenta už nič nestojí v ceste.
- Nie sú optimálne na detegovanie sieťových skenov alebo iných prieskumných útokov.
- Pri analýze auditných logov dochádza kvôli ich zložitosti ku zvýšeným požiadavkám na systémové prostriedky.

2.2 IDS založené na detekciách zneužitia

Na detegovanie útokov systémami IDS existujú dva primárne prístupy analýzy. Prvý je detekcia zneužitia. Tá pozostáva z detegovania známych preddefinovaných vzorcov udalostí, ktoré popisujú útok. Tieto vzorce udalostí sú taktiež známe ako **signatúry**. Preto sa niekedy celý tento prístup nazýva aj detekcie signatúr. Časť komerčných riešení využíva izomorfnú špecifikáciu signatúr k útokom. Existuje však aj sofistikovanejšia metóda analýzy, tzv. `state-based` alebo analýza založená na stavoch, ktorá dokáže identifikovať pomocou jednej signatúry celú skupinu útokov.

Výhody

- Veľmi efektívne a presné, generujú minimum falošných incidentov.
- Presne identifikujú typ útoku, čo urýchľuje reakciu a nápravu systému.
- Jednoducho pochopiteľné aj pre bezpečnosť neznalých členov tímu.

Nevýhody

- Databáze signatúr musia byť konštantne aktualizované, inak systém nie je schopný rozpoznať nové typy útokov. Tieto aktualizácie sú náročné pre air-gapped[12] siete.
- Pri menej sofistikovaných riešeniach môže dôjsť k jednoduchému obídeniu detekcie aj malou variáciou udalostí známeho útoku. Stavová analýza tento problém plne mitiguje.

2.3 IDS založené na anomáliách

Druhým prístupom detekcie útokov je práve detekcia anomálií. Práca [15] sa tejto problematike venuje a nasledujúca časť z nej vychádza. Tento prístup je taktiež známy ako heuristický alebo založený na správaní. Zameriava sa na identifikovanie abnormálneho správania sa komunikácie v počítačových sieťach alebo procesov na koncových zariadeniach. Fungujú na predpoklade, že útoky sa charakteristicky odlišujú od normálnej, legitímnej aktivity, a teda sú detegovateľné systémami, ktoré sa na tieto odchýlky zameriavajú. Tohoto sa štandardne dosiahne vytvorením profilov z nazbieraných holistických, historických dát z obdobia normálnej prevádzky. Tieto profily predstavujú normálne správanie sa užívateľov, koncových bodov alebo počítačových sietí.

Tento prístup pozostáva z 3 nasledujúcich krokov:

- Parametrizácia - tvorenie profilov
- Trénovanie - vytvorenie modelu na základe kľúčových charakteristík pozorovaného systému
- Detekcia - porovnávanei modelu s živými dátami

Výhody

- Dobre vytvorené moduly dokážu identifikovať neznáme a dokonca aj zero-day[13] útoky, vďaka ich schopnosti detegovať symptómy útoku.

[12] Air-gapped sieť je fyzicky oddelená sieť od vonkajšieho sveta, čo znamená, že nie je pripojená k žiadnej externej sieti, ako je internet.

[13] Zero-day útok označuje situáciu, keď sa zneužije zraniteľnosť, ktorá je doposiaľ neznáma prevádzkovateľom softvéru.

- Vedia vytvoriť základ pre automatizáciu tvorby zero-day signatúr. Tie sa potom môžu použiť na detekčné modely založené na zneužití.

Nevýhody

- Sú najčastejším zdrojom veľkého množstva falošne pozitívnych incidentov a prispievajú tým ku zahlteniu bezpečnostných expertov.
- Musia byť vytvorené a nastavené pomocou veľkého množstva trénovacích dát aby boli schopné správne identifikovať normálne správanie systémov.

Nakoľko nie každá anomália môže byť detegovaná len porovnávaním s normou, je nutné umožniť nepretržité vylepšovanie a modifikáciu tejto normy, čo je možné robiť manuálne alebo automaticky. Odvíja sa od toho kategorizácia modelov na detekciu anomálií na nasledujúce metódy:

Kategorizácia modelov

- Štatistické parametrické modely – založené na definovaní parametrov normy, ako napríklad prietok dát, počet paketov na protokol, alebo definovaním prahu jednotlivým atribútom správania systému, napr. prístupov k súborom alebo neúspešných pokusov o prihlásenie
- Štatistické neparametrické modely – podobné parametrickým, ale štatistické dáta sa berú z historických hodnôt
- Expertné modely – podobné neparametrickým modelom, kľúčovým rozdielom je, že norma je definovaná naskriptovanými pravidlami a nie numerickými hodnotami (napr. YARA rules[2])
- Modely strojového učenia – založené na trénovaní algoritmov pomocou trénovacích súborov dát, tvorením klasifikátorov, ktorý potom generuje pravdepodobnosť anomálií na živých dátach

2.4 Členenie anomálií

V predchádzajúcom bode som sa venoval systémom IDS založených na anomáliach. Pre potreby tejto práce je však potrebné bližšie definovať nie len, čo anomália je, ale aj aké typy poznáme.

Za anomáliu, špecificky v kontexte tejto práce, považujeme vzory dát, ktoré identifikujeme na základe ich abnormality v relácii k charakteristike legitímnej sieťovej aktivity. Týmto spôsobom sme schopný dynamicky reagovať na širokú škálu útokov, no nejedná sa o najpresnejšiu metodiku a jej efektivita je úmerná množstvu dostupných, holistických dát. Ako som už vyššie uvádzal, metóda detekcie anomálií je veľmi silným nástrojom, a síce preto, že vie odhaliť široké spektrum útokov, a to aj `zero-day` útoky, no bez dostatočného ladenia nedosahuje vysokú mieru presnosti a často je zdrojom značnej časti `false positives`[14].

Podľa práce [4] môžme anomálie členiť na nasledujúce kategórie:

[14] false positives - falošné pozitíva - nesprávne označkované dáta.

- **Bodová anomália**

  Bodovou anomáliou nazývame takú inštanciu dát, kde sa dáta značne vychyľujú od charakteristiky súboru dát. Príkladom môže byť nadmerná spotreba paliva v kontexte: Človek má dlhodobo priemernú dennú spotrebu 5 litrov za deň. Ak zaznamenáme, že jeden náhodný deň je spotreba 50 litrov, jedná sa o bodovú anomáliu.

  Príkladom z kyberbezpečnosti je jednorázovo zvýšený počet nesprávnych pokusov o prihlásenie pomocou SSH[15], čo by mohlo indikovať pokus o bruteforce[16].

- **Kontextová anomália**

  Keď sa dáta správajú abnormálne v konkrétnom kontexte, nazýva to kontextovou, alebo podmienenou, anomáliou. Pre príklad: Výdavky na kreditnej karte počas sviatočných období, napríklad Vianoc alebo Nového roka, sú obvykle vyššie ako počas zvyšku roka. Aj keď v kontexte jednotlivých mesiacov sa jedná o anomáliu, ak oddialime rozsah dát na roky, môžeme pozorovať, že daná anomália, je vlastne opakujúcou sa normou pre dané mesiace. Naopak, dáta, ktoré ukazujú na vysoké výdavky na mieste, kde nemajú na základe historických dát dôvod byť, môžeme vyhodnotiť ako anomáliu.

  Príkladom z kyberbezpečnosti by mohol byť zachytený Portscan[17] mimo dobu plánovaného skenu pre účely manažmentu zraniteľností. Môže sa jednať o anomáliu, ak nemáme v systéme zalogovaný dôvod na tento neplánovaný sken siete.

- **Kolektívna anomália**

  Keď sa kolekcia podobných dátových inštancií správa anomálne vzhľadom na celý dataset jedná sa o kolektívnu anomáliu. Napríklad v prípade výstupu z ľudského elektrokardiogramu (EKG) naznačuje existencia nízkych hodnôt po dlhú dobu výskyt výnimočného fenoménu, ktorý zodpovedá abnormálnemu predčasnému sťahovaniu[18], zatiaľ čo samotná jedna nízka hodnota sa nepovažuje za anomáliu.

  Príklad z oblasti kyberbezpečnosti by mohol byť následovný: Koncový bod zamestnanca firmy v pracovných hodinách zasiela DNS[18] dotazy v náhodných intervaloch, no ak systém zaznamená malý rozptyl intervalu, s dlhším intervalom medzi jednotlivými dotazmi a s konzistentne väčšou veľkosťou packetov (DNS tunneling väčšinou kóduje dáta ako subdomény) môže sa jednať o exfiltráciu[19] dát pomocou DNS tunelu[29].

2.5 Modely strojového učenia

Keďže práve modely strojového učenia sú stredobodom tejto práce, budem sa im v nasledujúcej sekcii bližšie venovať. Jedná sa o asi najoptimálnejšiu formu detekcie anomálií. Je to práve preto, že táto metóda umožňuje modely neustále učiť a vyvíjať sa. Toto všetko je možné robiť automaticky, iteratívne a aj počas živého chodu modulu. Nevýhodou je nutnosť predpripraviť veľké množstvo označkovaných dát, ktoré poslúžia na úvodné trénovanie

[15] SSH (Secure Shell) jedná sa o kryptografický protokol na vzdialené riadenie a bezpečnú komunikáciu.

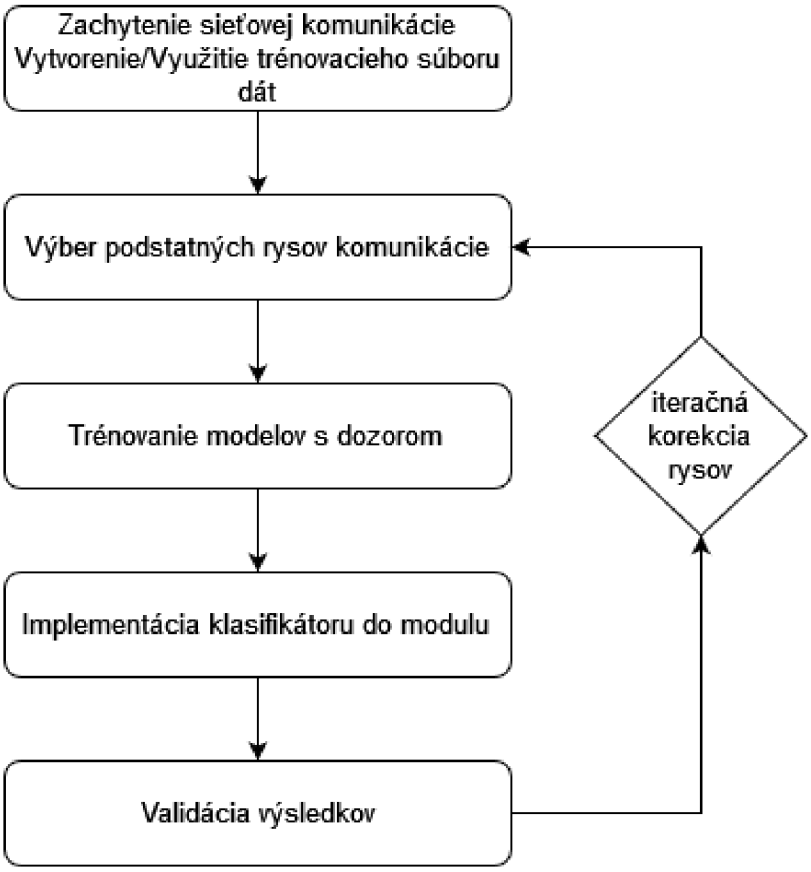
[16] Brute-force je automatizovaná metóda útoku na zabezpečený systém alebo heslo, pri ktorej sa vyskúšajú všetky možné kombinácie.

[17] Portscan je technika používaná na skenovanie sietí s cieľom identifikovať aktívne sieťové porty na danom systéme. Porty sú číselné adresy, ktoré systém používa na identifikáciu a poskytovanie služieb.

[18] DNS (Domain Name System) je systém na uchovávanie doménových mien

[19] Exfiltrácia znamená neoprávnené odcudzenie dát z chráneného systému

modelu a vytvorenie váh klasifikátoru. Najväčšou limitáciou tohoto prístupu je relatívne vysoká náročnosť na systémové prostriedky.



Obr. 2.3: Model klasifikácie sieťovej prevádzky [27]

Na obrázku 2.3 je popísaný model klasifikácie sieťovej prevádzky. Tento všeobecný model sa dá uplatniť na spektrum rôznych schém strojového učenia. Príkladom môžu byť Bayesovské siete, Neurálne siete, Markovské modely, Fuzzy logic modely a iné.

Po vytvorení trénovacieho súboru dát sa iteračne vyberajú charakteristické rysy komunikácie. Toto bude pri implementácií najväčším bodom experimentácie, pretože nie všetky označkované rysy musia napomáhať presnosti klasifikátoru. Tento model zobrazuje základ metodiky pri experimentovaní s rôznymi algoritmami strojového učenia a ich efektívnosti pri detegovaní anomálií.

Kapitola 3

Parametrizácia a Klasifikácia

Na úvod kapitoly priblížim princípy klasifikačných metód. Následne popíšem metodiku evaluácie daných metód za použitia označkovaného súboru dát. Počas výskumu vhodných kandidátov budem krížovo referencovať viaceré štúdie [6, 29], ktoré podobné experimenty už vykonávali.

3.1 Klasifikačné metódy

Klasifikácia je proces triedenia dát. Na začiatok si musíme vytvoriť presne definované triedy, do ktorých budeme dáta triediť. Cieľom práce je korektne odlíšiť abnormálne správanie siete od normy. Zo základu sa teda zaoberáme binárnou klasifikáciou. Avšak tejto definície sa práca dlho držať nebude.

3.1.1 Výber modelu

Pri začiatku tejto práce som prešiel široké spektrum klasifikačných techník od úplne triviálnych rozhodovacích stromov až po moderné rafinované knižnice. Práve týmto sofistikovaným knižniciam strojového učenia budem venovať väčšinu pozornosti pri implementačnej časti práce. Stručne predstavím rôzne modely, o ktorých som sa učil.

Support Vector Machine

Algoritmus s dozorom. Môže sa použiť, ako na klasifikáčné, tak na regresné úlohy. Vytvára `hyperlane`[1] na oddelenie rozdielnych tried dát v súbore. Maximalizuje vzdialenosť medzi jednotlivými triedami, zatiaľ čo minimalizuje chybovosť klasifikácie príkladov.

Relatívne rýchlejšie vycvičiteľné a tvoria menej nadmernej nákladovej záťaže, takže majú v praxi dobrý výkon. Dobre generalizujú z tréningových dát na nové vstupy. Vhodné na klasifikáciu textu, rozpoznávanie obrazu a pod.

Neurónové siete

Algoritmus navrhnutý tak, aby dokázal rozpoznávať zložité vzory a dáta, a tak vykonávať rôzne úlohy, ako napríklad klasifikáciu, predikciu, rozpoznávanie obrazu a reči.

[1] Hyperplane (Hyperrovina) sa označuje ako rozhodovacia hranica, ktorá sa používa na rozdelenie dátových bodov do rôznych tried.

Pozostávajú z umelých neurónov, ktoré sú ich základnými stavebnými blokmi. Tieto neuróny sú spojené v rôznych vrstvách, kde každý neurón prijíma vstupné hodnoty, vykonáva výpočet a generuje výstup. Tieto spojenia medzi neurónmi sa nazývajú váhy a slúžia na priraďovanie dôležitosti jednotlivým vstupným hodnotám.

Trénovanie pozostáva z procesu, kde sa váhy medzi neurónmi prispôsobujú na základe trénovacej sady dát. Jedná sa o proces nazývaný učenie siete. Počas učenia sa sieť snaží minimalizovať chybu medzi jej výstupom a korektnými očakávanými výstupmi. Keď je sieť dobre vytrénovaná, môže byť schopná generalizovať a aplikovať naučené vzorce a poznatky na nové vstupné dáta.

Neurónové siete, založené na využití hlbokého učenia (deep learning), sa stali veľmi účinným nástrojom v mnohých oblastiach, ako je spracovanie obrazu, rozpoznávanie reči či prírodného jazyka alebo odporúčacie systémy a mnoho ďalších. Jedná sa o podoblasť strojového učenia, ktorá sa zameriava na trénovanie umelej neurónovej siete, aby sa naučila a robila predikcie alebo rozhodnutia bez explicitného programovania. Jeho cieľom je napodobniť spôsob, akým ľudský mozog spracováva informácie pomocou viacerých vrstiev prepojených neurónových buniek.

Rozhodovacie Stromy

Taktiež známe ako `Decision Trees` sú vo svojej podstate relatívne triviálnou technikou na klasifikačné a regresné úlohy. Fungujú na princípe hierarchického rozdeľovania súboru dát na čo najmenšie podsúbory až do momentu, kedy každý z týchto podsúborov obsahuje len inštancie dát s veľmi podobnými vlastnosťami. Inak nazývané aj listy, tieto podsúbory sú dostupné za cestou tvorenou sériou binárnych podmienok vo forme uzlov. Vo finále technika vytvorí binárny strom podsúborov, kde každý uzol obsahuje podmienku pre parametre a vetvy predstavujúce pravdepodobnosť, s ktorou daný parameter patrí do tried.

Konštrukcia rozhodovacieho stromu na jednotlivé podsúbory zahŕňa rekurzívne rozdeľovanie dát na základe vybraných vlastností. V každom vnútornom uzle algoritmus vyberie najlepšiu vlastnosť pre rozdelenie dát s cieľom maximalizovať oddelenie medzi rôznymi triedami alebo minimalizovať varianciu v každom rozdelení. Tento proces pokračuje, kým nie je dosiahnuté určené kritérium ukončenia, ako je dosiahnutie maximálnej hĺbky stromu alebo nedosahovanie ďalšieho zlepšenia prediktívnej presnosti.

Čo je vynikajúcou vlastnosťou tejto techniky je jej jednoduchosť interpretácie a robustnosť voči veľkým súborom dát. Tým je myslené, že presnosť predikcií neklesá s nárastom objemu dát.

Rozhodovacie pravidlá reprezentované štruktúrou stromu možno ľahko pochopiť a vizualizovať, čo umožňuje získať pochopenie rozhodovacieho procesu. Okrem toho rozhodovacie stromy dokážu pracovať s kategorickými aj numerickými vlastnosťami a sú odolné voči chýbajúcim hodnotám.

Jednoznačne najsilnejšou stránkou tejto techniky je jej evolúcia do `Gradient Boosted Trees`. To je metóda, kde sa stromy trénujú iteračne. Každú iteráciu sa pôvodný strom dopĺňa o novovytvorený a ich úspešnosť sa hodnotí oproti predom zvolenému kritériu. Do tejto kategórie spadá aj algoritmus **XGBoost**

3.2 XGBoost vs. Tensorflow

Ako bolo spomenuté, v priebehu výskumu klasifikačných algoritmov som dospel k záveru, že práve algoritmy XGBoost a Tensorflow Keras Deep learning budú optimálnym začiatkom na ceste ku kvalitnej a presnej detekcii anomálií.

Oba algoritmy sú široko využívané, so špecifickými vlastnosťami a silnými stránkami. XGBoost je tzv. `gradient boosting algorithm`. V stručnsoti sa jedná o algoritmus, ktorý využíva rozhodovacie stromy ako svoju bázu. Je vysoko optimalizovaný a jeho silami sú výkon a rýchlosť. Preto je vhodný na prácu s veľkými datasetmi.

Na druhú stranu Tensorflow je `deep learning framework`, alebo teda hlbokého učenia, ktorý umožňuje tvorbu a trénovanie komplexných neurálnych sietí.

Podľa autora článku [9] je XGBoost rýchlejší a dobre sa prispôsobuje na chýbajúce dáta. Avšak najlepšie pracuje na štruktúrované problémy, kde sa dajú vzťahy medzi dátmi a rysmy dobre popísať vo forme rozhodovacích stromov s klasifikačnými váhami. Tensorflow, naopak, je robustnejší a dobre pracuje aj s vysokou úrovňou abstrakcie, no je niekoľkonásobne pomalší. Taktiež viaceré zdroje ako napr. [25] uvádzajú, že XGBoost funguje lepšie *"z krabice"*, teda bez zbytočného konfiguračného času, zatiaľ čo pri Tensorflow je často nutné stráviť čas konfiguráciou hyperparametrov a ich jemným upravovaním aby sme sa dostali k optimálnym výsledkom.

3.3 Metodika evaluácie modelov

V tejto časti budem bližšie popisovať metodiku evaluácie modelov strojového učenia. Jedná sa o sadu metrík, vďaka ktorým sme schopný numericky vyjadriť a následne posúdiť kvalitu a výkonnosť jednotlivých modelov. Je dôležitá pre porovnanie rôznych algoritmov a techník strojového učenia, a taktiež na zistenie, ako dobre sa model dokáže generalizovať na nové neznáme dáta. Vychádzal som zo štúdie [16]

Základnými metrikami sú:

TP, TN, FP, FN

- **TP** `True Positive` skutočné pozitívum - označuje správne vyhodnotené pozitívum
- **TN** `True Negative` skutočné negatívum - označuje správne vyhodnotené negatívum
- **FP** `False Positive` falošné pozitívum - označuje nesprávne vyhodnotené pozitívum
- **FN** `False Negative` falošné negatívum - označuje nesprávne vyhodnotené negatívum

Úspešnosť

Pomer správne vyhodnotených a všetkých bodov v súbore.

$$U = \frac{TP + TN}{TP + TN + FP + FN}$$

Fall-out (FPR) skóre

`Fall-out` alebo `False Positive Rate` je pomer negatívnych bodov, ktoré sa chybne klasifikujú ako pozitíva s ohľadom na všetky negatívne body. Čím vyššie FPR, tým viac negatívnych dát bude zle klasifikovaných. (viac FP)

$$FPR = \frac{FP}{FP + TN}$$

Recall (TPR) skóre

`Recall` alebo `True Positive Rate`. Vyjadruje pomer medzi všetkými `True positives` a `True positives + False Negatives`, a teda nám znázorňuje koľko anomálií model nesprávne vyhodnotí ako legitímnu komunikáciu.

$$TPR = \frac{TP}{TP + FN}$$

Zo vzorca vyššie je možné usúdiť, že metrika je využívaná na určenie presnosti modelu. Pre náš systém bude vrcholne dôležité sa zamerať na čo najvyššiu maximalizáciu tejto metriky, nakoľko sa jedná o systém kde si nemôžeme dovoliť neidentifikovať hrozby v sieťovej komunikácii.

Precision skóre

Taktiež nazývaná presnosť reprezentuje pomer všetkých správne označených pozitívnych nálezov ku všetkým pozitívnym nálezom. Taktiež známe ako pozitívna prediktívna hodnota.

$$Precision = \frac{TP}{TP + FP}$$

Používa sa v prípadoch, kedy je od modelu vyžadované správne vyhodnotiť čo najviac nálezov pozitívne, a teda na minimalizáciu falošných pozitív. Dobrým príkladom na využitie skóre presnosti sú spam filtre. V našom prípade je dôležité na toto skóre dbať aby náš systém generoval čo možno najmenšie množstvo falošne pozitívnych nálezov, a tým nespôsoboval zahltenie odborníkov.

Vzťah Precision x Recall

Ďalej by som priblížil problematiku často antagonistického vzťahu, ktorý sa týka vyššie uvedených metrík. Ako sa píše v článku [17] problém je totiž v tom, že všeobecne zvyšovaním presnosti modelu znižujeme jeho recall a naopak. Model s vysokým skóre presnosti nebude generovať tak vysoké množstvo falošných pozitív, no môže sa stať, že kvôli tomu prehliadne prípad nelegálneho toku. Naopak modelu s vysokým recall skóre neunikne žiadna nelegálna komunikácia, no bude generovať väčšie množstvo falošných pozitív. Na balansovanie týchto dvoch metrík však existuje viacero prístupov.

Jednou z metrík na tento prípad použitia je práve metrika úspešnosti (Pomer správne vyhodnotených a všetkých bodov v súbore), avšak jej slabinou sú súbory dát, ktoré sú nevyvážené alebo skreslené. Ukážkový scenár tohoto problému je súbor dát, v ktorom by sme mali 95% dát jednej triedy a zvyšných 5% by predstavovalo všetky ostatné triedy (1

alebo viac). V tomto prípade by úspešnosť skoro vždy ukazovala vysoké hodnoty, nakoľko by správne odhadovala 95% súboru. Pre náš prípad je teda táto metrika zbytočná, pretože naše súbory dát budú obsahovať >95% legálnej komunikácie a len malé množstvo budú predstavovať reálne útoky.

F1 skóre

Na evaluáciu nevyvážených súborov dát máme viaceré metódy. V prvom rade sa jedná o F1 skóre. To reprezentuje skóre modelu, ako funkciu hodnôt presnosti a recall. Jedná sa o alternatívu úspešnosti, ktorá nepotrebuje poznať celkové množstvo výsledkov. Matematicky sa jedná o harmonický priemer skóre presnosti a recall.

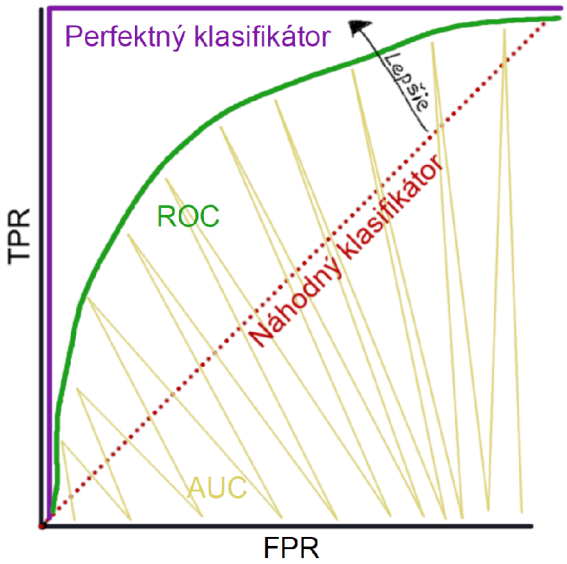
$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Pomocou F1 skóre sme schopný model optimalizovať buď na skóre presnosti alebo recall. Jej výsledkom je jedna hodnota, ktorá poukazuje na kvalitu výstupu z modelu.

AUROC

Ďalším dobrým prístupom na vyhodnotenie presnosti modelu je použiť **AUC ROC krivku**. Ako sa píše v článku [24] AUROC alebo teda Plocha pod krivkou ROC (z anglického Area Under the Receiver Operating Characteristic curve), je grafické zobrazenie efektivity binárneho klasifikačného modelu. Výsledkom je jedno číslo, ktoré meria celkový výkon klasifikačného modelu. Hodnota jedna značí perfektný klasifikátor, zatiaľ čo hodnota 0,5 značí náhodný klasifikátor. Tento prístup k vyhodnocovaniu efektivity modulu je lepší, ako samotná úspešnosť modelu, no môžme ju aplikovať len na binárnu klasifikáciu.

ROC `Receiver Operating Characteristic`, bližšie popísaná v príspevku [14] t.j. operačná / pravdepodobnostná krivka, ktorá ukazuje pomer medzi TPR a FPR pri rôznych konfiguráciách rozhodovacej úrovne. Napomáha pri určovaní optimálnej hranice.



Obr. 3.1: Znázornenie AUROC a ako ho interpretovať

Kapitola 4

Návrh riešenia

Táto kapitola sa venuje návrhom riešenia, ktoré bude riešiť detekciu anomálií v sieťovej komunikácii. Tento návrh je vypracovaný na základe získaných znalostí z predchádzajúcej kapitoly. Problémom bude nájsť prienik množinami rysov potrebných na detekciu rôznych typov útokov. Pri výbere rysov budem vychádzať z viacerých štúdií zaoberajúcich sa detekciou anomálií. Tieto štúdie sa často zameriavali na detekciu menších skupín navzájom podobných útokov (Bruteforce, DDoS[1], DoH[2] data exfiltration[3], atď.). Môžme teda usúdiť, že multiklasifikátor viacerých typov útokov bude potrebovať robustnejšiu sadu rysov aby sa v presnosti vyrovnal viacerým menším klasifikátorom.

Počas fáze experimentácie sa budem venovať trénovaniu a testovaniu presnosti modelu. Forma nadobudne podobu iteratívneho nastavovania parametrov klasifikátorov, tvorbu a trénovanie modulu s využitím trénovacích dát a vyhodnotenie výsledkov tohoto pomyselného šprintu. Na základe vyhodnotenia sa v ďalšej iteráciu upravia parametre a tento cyklus sa bude opakovať dokým model nedosiahne dobré výsledky na základe metrík bližšie vysvetlených v sekcii 3.3.

4.1 Výber a kombinovanie klasifikačných metód

Vývojový návrh sa moc od 2.3 nezmenil. Pôvodne som síce pracoval len s iteráciou pri experimentovaní s modelom klasifikácie 1 typu útoku, ale rovnaký postup je možné uplatniť aj pri kombinovaní klasifikačných modelov do väčšieho modelu, či trénovaní multiklasifikátoru, ktorý by bol schopný identifikovať a rozlíšiť viaceré typy útokov bez značnej straty na presnosti a tým pádom bez výrazného navýšenia počtu generovaných falošných incidentov.

Nakoniec som sa rozhodol zamerať prácu na trénovanie a následné využitie multiklasifikačného modelu, ktorého výstupom nebude len binárny výstup `útok? áno : nie`. Cieľom bude teda vytvoriť model schopný nie len identifikovať útok, ale aj ho kategorizovať. Takýto výstup je bližší schopnostiam moderných detekčných a reakčných platforiem[4].

[1] DDoS (Distributed Denial of Service) Distribuovaný útok odopretia služby je typ útoku, ktorý sa snaží spôsobiť nedostupnosť služby alebo systému pre legitímnych používateľov tým, že zahlcuje cieľový systém veľkým množstvom nepretržitého sieťového premávky.

[2] DoH (Domain Name System over HTTPS) Systém pre prevod doménových mien cez HTTPS, je protokol, ktorý kombinuje dve dôležité technológie - DNS a HTTPS.

[3] Exfiltrácia dát cez DoH (Domain Name System over HTTPS) sa odohráva v kontexte využitia DoH protokolu na neoprávnený prenos dát z postihnutého systému. Pretože DoH komunikácia je šifrovaná a nepatrí do bežnej prevádzky siete, môže to umožniť útočníkovi preniesť citlivé údaje cez sieť bez detekcie.

[4] Téma IDS a IPS platforiem je popisovaná v kapitole 2, špecificky sa o moderných riešeniach bližšie píše v sekcii 2.1.1

Pôvodne som pracoval s ideou, trénovať paralelne niekoľko klasifikačných modulov založených na rôznych algoritmoch, no v konečnom dôsledku som od tejto idey odstúpil z dôvodu prudkého nárastu, najmä časovej, komplexity bez úmerného dopadu na výsledné hodnoty. Hlavnou príčinou však bol práve časový faktor, keďže trénovanie na väčších súboroch dát je časovo náročné ak trénujem jeden relatívne efektívny model, ako napr. XGBoost. Iné modely sú, ako bude v nasledujúcej sekcii priblížené, časovo náročnejšie, ako práve XGBoost, a teda čas narastá nelineárnou krivkou.

4.1.1 Finálny výber modelu

Na základe experimentov, ktoré som spracoval v jazyku Python 3.10 som vyhodnotil model `xgboost.XGBClassifier` a `tfdf.keras.GradientBoostedTreesModel`. Tieto experimenty prebiehali formou prototypovania extrémnych prípadov pre oba modely. Pôvodne som zamýšľal do testovania porovnávať model `TF Boosted trees` s XGBoost, avšak už pri prvotnom výskume do tejto problematiky som narazil na viacero článkov, ktoré odporúčali prácu priamo s kerasom. Experimenty som prevádzal na už spracovanom datasete s 80 rysami CIC-IDS2018. Pomocou knižníce pandas som tieto dáta načítal do štruktúry `pandas.DataFrame` a pracoval s nimi.

Na základe týchto testov som došiel k nasledujúcim záverom:

- Potvrdil som, že čo sa časovej náročnosti konfigurácie týka, je jednoznačne jednoduchšie pracovať s xgboostom. Zanedbávajúc čas konfigurácie počas inštalácie, ktorý bol veľmi podobný, som dospel k záveru, že bez nadmerného upravovania hyperparametrov je presnejší *'z krabice'* xgboost.
- Časová náročnosť trénovania modulu sa tiež zhoduje s všeobecným konsenzom odbornej komunity, kde pri práci s dátami v tabuľkách XGBoost je minimálne o 40% rýchlejší.

Metrika	XGBoost	TFDF Keras DL
AUC Skóre	0.689	0.644
Recall	0.724	0.603
Čas CPU	7m34s	11-14m

Výsledky zodpovedajú dátam z rôznych štúdií. Avšak čo najviac zaváţilo pri finálnom rozhodnutí bol fakt, že s necelými 30 minútami nastavovania hyperparametrov pre XGBoost klasifikátor som bol schopný dosiahnuť 0.999 AUC skóre na testovaciom datasete, zatiaľ čo Keras som nebol schopný za rovnaký čas dostať cez hranicu hodnoty 0.9 na AUC-ROC.

Jeden faktor, ktorý som neúmyselne pri týchto testoch zanedbal a mohol by spraviť rozdiel pre iné prípady využitia je ten, že vzhľadom na skutočnosť môjho predspracovaného datasetu som nemusel takmer vôbec riešiť parsovanie dát do formy, ktorá je pre dané algoritmy vhodná. Tento faktor si myslím, že je dostatočne dôležitý na to, aby som ho retrospektívne spomenul.

Pri dodatočnom výskume tohoto problému som našiel článok[26], kde autor rieši túto skutočnosť ako faktor. Došiel k záveru, že pri podobnom type datasetu, sú oba modely čo sa komplexity nutného kódu vyrovnané, pretože zatiaľ čo Keras je nutné konfigurovať formou zložitejšieho a dlhšieho kódu, tak XGBoost sa ľahšie konfiguruje, no je nutná väčšia réžia na úpravu dát do formy, ktorú XGBoost očakáva.

Autor v článku taktiež rozoberá faktor flexibility algoritmov. Podľa jeho výsledkov je flexibilita neurálnej siete jednoznačne víťazom, nakoľko XGBoost nemá schopnosť spracovať tabuľkové datasety s kontinuálnymi, kategorickými a voľne formátovanými textovými stĺpcami.

4.1.2 XGBoost

Na základe vyššie zhrnutých faktov, som sa v konečnom dôsledku rozhodol pre implementáciu klasifikačného modelu XGBoost. Je to algoritmus, ktorý je založený na rozhodovacích stromoch a slúži na riešenie ako regresných, tak aj klasifikačných úloh.

Dôležité je zdôrazniť, že najväčšou slabinou algoritmu je potreba mať na vstupe spracované dáta, ktoré sú v rovnakom formáte ako dáta, na ktorých bol model trénovaný. Výhodou však je, že do určitej miery takmer bezstratovo toleruje ak niektoré z daných dát chýbajú, a sú vyplnené nulovými hodnotami.

Logika XGBoost-u

Na základe prác *XGBoost: A Scalable Tree Boosting System* [12] a *Introduction to Boosted Trees* [11] priblížim samotný algoritmus a potom okrajovo vysvetlím ako funguje po odbornej stránke. Meno XGBoost znamená `Extreme Gradient Boosting`, voľne by sme to mohli preložiť ako: extrémne gradientové zosilňovanie. V princípe sa jedná o proces, ktorý, ako meno zosilňovanie napovedá, vytvára silný prediktívny model spojením viacerých slabších modelov. Špecificky sa v našom prípade jedná o gradientovú optimalizáciu na iteratívne zlepšovanie slabých modelov.

Táto gradientová optimalizácia využíva gradient (vektor parciálnych derivácií) chybovej funkcie vzhľadom na parametre modelu. Inými slovami sa jedná o súbor hodnôt, ktoré napomáhajú udať smer najrýchlejšieho nárastu (alebo klesania) funkcie. Cieľom gradientovej optimalizácie je nájsť optimálne hodnoty parametrov modelu, pri ktorých je chybová funkcia minimalizovaná.

Chybovú funkciu chceme minimalizovať, pretože je ukazovateľom nepresnosti modelu. Konkrétne, XGBoost vytvára jednoduché stromové modely, ktoré sa postupne vylepšujú práve pomocou gradientovej optimalizácie. vďaka nej sme v každej iterácii schopný minimalizovať chybu predikcie a priblížiť sa tak k čo najpresnejšiemu modelu.

XGBoost je teda optimálnou voľbou na náš prípad použitia. Je schopný spracovať veľmi efektívne veľké množstvo dát a zachovať si vysokú presnosť aj pre komplexné modely s veľkým množstvom parametrov.

Konceptuálne sa trénovanie modelu dá popísať ako proces, ktorého cieľom je vytvoriť na základe trénovacích dát X_i také rozhodovacie stromy, aby správne predikovali cieľové trénovacie hodnoty y_i. Ako vstup očakáva algoritmus súbor dát zoradený podľa zadefinovaných parametrov. Algoritmus potom iteračne zostavuje rozhodovacie stromy tak, aby čo najpresnejšie na základe parametrizovaného súboru dát odhadoval cieľové hodnoty.

Matematika za XGBoostom

Dôležitou charakteristikou objektívnej funkcie 4.1 je, že pozostáva z dvoch častí, stratovej funkcie a regularizačného člena:

$$obj = L(\theta) + \Omega(\theta) \tag{4.1}$$

Ak začneme odzadu, $\Omega(\theta)$ je regularizačný člen, ktorý slúži na penalizáciu komplexnosti modelu. Objektívna funkcia teda nadobúda vyššej hodnoty ak je model príliš komplexný. $L(\theta)$ je stratová funkcia počas trénovania. Stratová funkcia počas trénovania slúži na meranie schopnosti modelu predikovať dáta, ktoré sa nachádzajú v trénovacej množine. Bežnou voľbou pre L je stredná kvadratická chyba, ktorá je definovaná ako $L(\theta) = \sum_i (y_i - \hat{y}_i)^2$.

Matematicky vieme XGBoost model zapísať vo forme:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F} \tag{4.2}$$

Kde K je počet stromov, f je funkcia v priestore funkcií $\mathcal{F}$, ktorý predstavuje množinu všetkých možných klasifikačných a regresných stromov (CARTs). Cieľová funkcia, ktorá sa optimalizuje, je definovaná ako:

$$obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{4.3}$$

Parametre stromu, ktoré sa použijú, sú tie, ktoré môžu byť určené funkciami f_i, z ktorých každá obsahuje štruktúru stromu a hodnoty listov. Je považované za nepraktické naučiť sa všetky stromy naraz. Namiesto toho sa používa aditívna stratégia, kde sa postupne pridáva jeden nový strom za druhým. Predikčná hodnota v kroku t je označovaná ako $\hat{y}_i^{(t)}$. Potom dostaneme:

$$\begin{aligned} \hat{y}_i^{(0)} &= 0 \\ \hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\ &\dots \\ \hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \end{aligned} \tag{4.4}$$

V každom kroku je pridaný strom, ktorý optimalizuje objektívnu funkciu.

$$\text{obj}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{const} \tag{4.5}$$

4.2 Analýza dát

V rámci tohoto projektu pracujem s bezpečnostným IDS nástrojom Suricata. Jedná sa o open-source nástroj pre detekciu a prevenciu sieťových útokov. Je navrhnutý na analýzu sieťových packetov v reálnom čase a poskytuje schopnosti detekcie známych a neznámych hrozieb pomocou pravidiel a algoritmov.

Medzi bežné výstupy Suricaty môžme zaradiť:

- **Logy**

  Suricata môže generovať logy, ktoré zachytávajú informácie o sieťovej prevádzke, udalostiach a ďalších relevantných detailoch. Tieto logy môžu obsahovať údaje, ako sú hlavičky paketov, dáta v paketoch, informácie o tokoch a časových značkách. Logy môžu byť užitočné pre forenznú analýzu, vyšetrovanie incidentov alebo tvorbu správ.

- **EVE výstup**

  EVE (Extensible Event Format) je flexibilný a prispôsobiteľný výstupný formát poskytovaný Suricatou. Umožňuje definovať, ktoré polia a atribúty udalostí majú byť zahrnuté vo výstupe, čím poskytuje presnú kontrolu nad zachytenými informáciami. EVE výstup je vo formáte JSON [5], ktorý je štruktúrovaný a ľahko čitateľný pre človeka, ale stále ľahko strojovo spracovateľný. Obsahuje podrobné informácie o zistených udalostiach, tokoch a ďalších relevantných údajoch. Často sa používa na integráciu Suricaty s inými bezpečnostnými nástrojmi alebo pre ďalšiu analýzu.

- **PCAP súbory**

  Suricata môže ukladať súbory zachytávania paketov (PCAP), ktoré obsahujú sieťovú prevádzku, a jej metadáta. Tieto PCAP súbory môžu byť následne analyzované pomocou iných nástrojov pre podrobnejšiu investigáciu a forenznú analýzu.

- **Štatistické súbory**

  Suricata môže generovať štatistické výstupy, ktoré poskytujú náhľad na sieťovú prevádzku a jej vzory. Obsahuje informácie o počte paketov a bytov, objeme komunikácie, používaných protokoloch atď. Tieto štatistiky sú užitočné pre monitorovanie siete, no pre túto prácu nie sú zaujímavé.

Hlavnou funkciou Suricaty je generácia varovaní (z angl. alert), keď deteguje podozrivú alebo potenciálne závadnú aktivitu v sieťovej prevádzke. Tieto varovania poskytujú podrobné informácie o zistených udalostiach, vrátane typu útoku, zdrojovej a cieľovej IP[6] adresy, použitých protokoloch, časových značkách a ďalších relevantných atribútoch. Varovania sú zaznamenané v rôznych súboroch, ako sú vyššie spomenuté EVE JSON alebo logovacie súbory. Suricata ďalej umožňuje generáciu varovaní v súlade s rámcom (z angl. Framework[7]) MITRE ATT&CK[8]. Toto umožňuje mapovať odhalené udalosti na konkrétne techniky, taktiky alebo postupy útočníkov. Tým je možné urýchliť a skvalitniť proces vyhodnotenia udalostí, zvýšiť účinnosť monitorovania a reakcie na útoky a taktiež môže napomôcť odkryť vektory útokov a potenciálne slabiny v obrane.

[5] JSON - JavaScript Object Notation

[6] IP (Internet Protocol) je protokol používaný na smerovanie a doručovanie dátových paketov v počítačových sieťach

[7] Framework je sada nástrojov, knižníc, pravidiel a štruktúr, ktorá umožňuje vývoj aplikácií alebo riešenie určitého problému.

[8] Framework MITRE ATT&CK je celosvetovo uznávaná znalostná báza taktík, techník a postupov útočníkov - https://attack.mitre.org/matrices/enterprise/

4.2.1 Výber útokov

Vzhľadom na schopnosti Suricaty som sa v záujme prehĺbenia znalostí v rámci daných problematík rozhodol pre vytvorenie multiklasifikačného modelu. Cieľom práce sa teda stáva model, ktorý by bol schopný odhaliť širšie spektrum útokov. Ako som je spomenuté vyššie, Suricata je schopná rozpoznať a kategorizovať rôzne typy útokov, čo mi umožňuje spätne validovať detekcie môjho modulu s jej detekciami.

Dos a DDoS

Cieľom DoS útoku je zasiahnuť dostupnosť systému, čo znamená, že legitímni používatelia nie sú schopní normálne využívať poskytované služby. Slowloris je typ nástroja pre útoky typu denial of service (DoS), ktorý umožňuje jednému počítaču znefunkčniť webový server iného počítača s minimálnym využitím šírky pásma a minimálnymi vedľajšími účinkami na nepodstatné služby a porty.

DoS útoky môžu mať rôzne formy, vrátane distribuovaného DoS (DDoS), ktorý využíva viacero zariadení na vykonávanie útoku z rôznych zdrojov. Cieľom zostáva preťažiť systém kombináciou veľkého množstva požiadaviek alebo správ z viacerých zdrojov, čo robí obranu a detekciu útoku zložitejšou. Útočník môže využiť rôzne techniky, ako sú SYN flood, UDP flood, ICMP flood alebo HTTP flood, aby zahltil sieťové pripojenia, zdroje alebo aplikácie cieľového systému.

Ako základný typ útoku považujem práve DDoS útok, v dobe písania práce vzhľadom na geopolitickú situáciu môžme sledovať prudký nárast práve DDoS útokov na celom spektre kybernetického terénu. Jedným z článkov upozorňujúcich na túto skutočnosť je aj od firmy Netscout[9] [1].

Bruteforce útok

Jedná sa o metódu, ktorá sa využíva na získanie prístupových údajov k systému prostredníctvom systematického skúšania všetkých možných kombinácií hesiel alebo iných autentifikačných údajov. Útočník používa automatizovaný proces, ktorý postupne skúša rôzne kombinácie, až kým nenájde správne autentifikačné údaje. Na optimalizáciu sa používajú slovníky obsahujúce slová, ktorých kombinácia by mohla tvoriť heslo. Tento druh útoku je často časovo náročný a vyžaduje veľké množstvo výpočtových zdrojov. Existuje veľa nástrojov na vykonávanie bruteforce útokov a crackovanie hesiel, ako napríklad Hydra, Medusa, moduly Metasploit a skripty Nmap NSE. Existujú tiež niektoré nástroje ako hashcat a hashpump na crackovanie hesiel uložených vo forme hashu. Ale jedným z najkomplexnejších viacvláknových nástrojov je Patator.

Botnet

Botnety sa často využívajú na zneužívanie a šírenie škodlivého softvéru, ako sú DDoS útoky, spamovanie, krádež citlivých údajov, ťažba kryptomeny a iné kybernetické útoky. Botnety sa šíria prostredníctvom malvéru, ktorý infikuje zraniteľné počítače a vytvára tak sieť infikovaných zombíkov. V datasete CIC-IDS2018 sa využíva Zeus, ktorý je balíkom malvéru typu Trojan, ktorý beží na verziách operačného systému Microsoft Windows. Hoci sa dá použiť na vykonávanie mnohých škodlivých a trestných činností, často sa využíva na krádež

[9] Zaujímavou je ich real time mapa DDoS útokov https://horizon.netscout.com/

bankových údajov prostredníctvom zaznamenávania stlačení klávesnice a zhromažďovania formulárov. Je tiež používaný na inštaláciu ransomwaru Crypto-Locker.

Infiltrácia

Jedná sa o proces, pri ktorom sa útočník snaží získať neoprávnený prístup k cieľovej sieti. Tento postup často zahŕňa využitie zraniteľností v systémoch, aplikáciách alebo nedostatočnej bezpečnosti siete. Útočník môže využiť rôzne metódy, ako napríklad phishing, útoky na heslá, zraniteľnosti softvéru alebo sociálne inžinierstvo.

Najväčšie riziko infiltrácie je v praxi často slabá ochrana vnútornej siete obeti v smere západ-východ. Ak útočník získa prístup do takejto siete, tak infraštruktúra nie je navrhnutá aby zabránila laterálnemu pohybu a útočník získa plnú kontrolu v priebehu minút až hodín. Následky infiltrácie je veľmi ťažké odstrániť, nakoľko útočník mohol vytvoriť mnoho backdoor prístupov.

V rámci scenáru datasetu CIC-IDS2018 sa využíva zraniteľná aplikácia (napríklad Adobe Acrobat Reader 9). Najskôr obeť dostane závadný dokument cez e-mail. Potom, po úspešnom využití pomocou Metasploit frameworku, sa na počítači obete spustí backdoor. Potom je možné vykonávať rôzne útoky na sieť obete, vrátane skenovania IP adries, plného skenovania portov, enumerácií služieb pomocou Nmap atď.

Webové útoky

Cieľom týchto útokov je získať neoprávnený prístup k citlivým informáciám, poškodiť webovú aplikáciu alebo inak narušiť jej prevádzku. Na tento účel sa zneužívajú zraniteľnosti a slabé miesta vo webových aplikáciách.

Existuje viacero druhov webových útokov:

- SQL Injection: Útočník vkladá nebezpečné SQL príkazy do vstupných polí webovej aplikácie, s cieľom získať prístup k databáze alebo zmeniť jej obsah.
- Cross-Site Scripting (XSS): Útočník vkladá skripty do webovej stránky, ktoré sa vykonávajú na strane klienta. To umožňuje útočníkovi získať citlivé informácie od používateľov alebo modifikovať obsah stránky.
- Cross-Site Request Forgery (CSRF): Útočník vytvára falošné požiadavky na webovú aplikáciu s cieľom vykonať neoprávnené akcie v mene používateľa, napríklad zmeniť heslo alebo odoslať peniaze.
- Brute-force útoky: Útočník systematicky skúša rôzne kombinácie hesiel, až kým nenájde správne. Týmto spôsobom môže získať prístup k ochráneným častiam webovej aplikácie.

4.3 Výber (optimalizácia) rysov pre klasifikáciu

Počas výskumu dôležitých rysov pre vyššie definované typy útokov som narazil na viaceré mienkotvorné fakty. Cieľom tejto sekcie je vysvetliť proces výberu rysov.

Na základe viacerých štúdií, zaoberajúcich sa augmentovaním dát na zlepšenie kvality výsledkov modulov som narazil na všeobecne široko akceptovaný koncept, ktorý tvrdí, že väčšie množstvo kvalitných dát prináša niekoľko výhod:

- **Zlepšená generalizácia**

  S väčším a rozmanitejším množstvom dát sa modely strojového učenia môžu naučiť lepšie generalizovať a presnejšie predikovať na reálnych dátach.

- **Redukcia preučenia**[10]

  Väčší súbor dát pomáha zabrániť preučeniu, kde sa model príliš špecializuje na trénovacie dáta a presnosť na nových dátach je nízka. Viac dát poskytuje širšiu reprezentáciu priestoru problému a znižuje riziko preučenia.

- **Zvýšená zložitosť modelu**

  Väčšie súbory dát môžu podporovať zložitejšie modely bez rizika preučenia. To umožňuje modelom zachytiť komplexné vzorce a vzťahy v dátach, čo môže viesť k zlepšeniu výkonu.

- **Vylepšená reprezentácia príznakov**

  Väčšie množstvo dát môže pomôcť odhaliť jemné vzorce alebo korelácie, ktoré neboli zrejmé v menších súboroch dát. Týmto sa model môže naučiť rozpoznávať nuansové príznaky a zlepšiť kvalitu svojich prediktívnych schopností.

Empirickým testovaním som si túto skutočnosť potvrdil. Pri iteráciach testovania som narazil na problém, kde výstupy zo Suricaty, vrátane spracovaných výstupov ako EVE JSON, neboli dostatočne konzistentné a preto nie sú vhodné pre túto prácu. Ďalším výskumom do tejto oblasti som našiel, že tvorci môjho dátoveho súboru používajú aj vlastný parser paketov do sieťových tokov, ktorého výstupy sú viac vhodné a bez potreby nadmerného spracovania do podoby vhodnej pre XGBoost Klasifikátor. Problém s touto aplikáciou je zase fakt, že aj keď je open source, je písaný v Jave[11] a už niekoľko rokov je pôvodným autorom zanedbávaný. Tým pádom je jeho integrácia na programovej úrovni projektovo náročná. Navrhovanýnm riešením je vytvorenie prostredia, ktoré využíva Suricatu na zachytenie a uloženie paketov a sprostredkováva ich následné spracovanie CICFlowMeter-om. Jeho výstup totiž umožňuje klasifikáciu viacerých druhov útokov s vysokou presnosťou.

Preto som sa v rámci výskumu hlboko zameral na rôzne predrobené súbory dát, ktoré sú určené na trénovanie detekcie anomálií v internetovej komunikácií. Výsledkom tejto činnosti bolo rozhodnutie pre využitie označkovaných súborov dát na trénovanie, a časť validácie. Následne využijem možnosť vlastnoručne simulovaných dát na finálne zvalidovanie validácie. Rozhodol som sa tak preto, lebo na akademické účely stvorené, kvalitné datasety sú dostupné v takom objeme, že nie je dôvod strácať čas tvorením vlastného, robustného súboru dát.

4.4 Zhrnutie návrhu

Systém je implementovaný v jazyku Python 3 (verzia 3.10.0). Na vstupe očakáva predpripravené dáta, ktoré sú vo formáte súboru tokov a k jednotlivým tokom obsahuje potrebné rysy na klasifikáciu útoku modelom. Aby mohol systém fungovať aj v reálnom nasadení, musí byť možné pracovať aj s reálnou sieťovou komunikáciou. Štandardným formátom na

[10] Preučenie (overfitting) je stav, ktorý nastáva pri trénovaní modelu strojového učenia, kedy model je príliš špecificky prispôsobený trénovacím dátam a nedokáže sa dobre generalizovať na nové neznáme dáta.

[11] Java je vysokoúrovňový, programovací, platformovo nezávislý jazyk

zachytávanie komunikácie na sieti je PCAP. Pri navrhovaní systému som zvažoval spracovanie a prácu práve so samotnými paketmi v tomto formáte. Výhodou je, že všetky programy operujúce so sieťovou prevádzkou vedia s formátom PCAP pracovať. Najväčšou nevýhodou je však skutočnosť, že formát PCAP obsahuje veľké množstvo zbytočných informácií a bez ďalšieho spracovania by znižoval efektivitu trénovania a testovania. Najmä z pohľadu spomaľovania rýchlosti a znižovania presnosti redundantnými dátami, ktoré by zahmlievali súbor dát.

Na základe toho som vylúčil prácu s čisto dátami vo formáte PCAP. Zostáva teda možnosť tento formát osekať a dostať do stavu, kedy bude použiteľný pre účely práce. Keďže spracovanie formátu PCAP nepatrí medzi ciele tejto práce, využijem softvéru tretej strany, ktorý mi umožní dáta upraviť do lepšej formy.

Vybraným prístupom pre naplnenie cieľov tejto práce bude združovanie paketov do sieťových tokov. Pri výskume tejto problematiky som rýchlo narazil na program CICFlowmeter, historicky známy aj ako ISCXFlowMeter. Tento sieťový nástroj slúži na generáciu obojsmerných sieťových tokov a je koncipovaný špecificky ako nástroj pre tvorbu dátových súborov pre účely strojového učenia.

Výhodou je, že už zo základu spracováva toky do podoby, kde obsahujú všetky definované rysy v sekcii 4.2.1. Ako bolo definované, tento formát vygenerovaných tokov a k nim priradených rysov je optimálny pre potreby strojového učenia, nakoľko obsahuje 80 detailných príznakov o jednotlivých tokoch a umožňuje modelu vytvoriť zložitejší, no o to viac na nuansy vytrénovaný rozhodovací strom, vďaka ktorému budeme schopný dosiahnuť vysokú presnosť predikcií útokov.

Funkciou systému bude označiť jednotlivé toky hodnotami 0 až 14. Hodnota 0 znamená, že daný tok je legitímnou komunikáciou a môžme ho považovať za bezpečný. Hodnoty odlišné od 0 reprezentujú útoky a ich jednotlivé hodnoty slúžia na kategorizovanie typu útoku, ktorý systém vie rozpoznať.

Model bude trénovaný s pomocou jazyku Python 3 s využitím knižníc xgboost a sklearn. Vzhľadom na výskumné zameranie práce je dôležité spomenúť aj prostredie na prácu s modelom. Model budem trénovať a testovať v prostredí aplikácie Jupyter Notebook. Na prácu so strojovým učením je to optimálne prostredie, ktoré umožňuje prácu s kódom rozdeliť do samostatných buniek a atomicky upravovať jednotlivé premenné bez nutnosti reštartovania interpretu jazyka python. Toto umožňuje načítať dáta vopred a meniť len parametre trénovania modulu, čo značne eliminuje objem času potrebného na experimentáciu. Prostredie taktiež umožňuje výsledné dáta okamžite vizualizovať.

Samotný proces experimentácie je bližšie popísaný v sekcii 4. Z neho vyplýva, že bude prebiehať iteratívne za účelom nájdenia optimálnych parametrov pre trénovanie klasifikátoru. Ako bolo vysvetlené v sekcii 4.1.1, po vyčerpávajúcom výskume a experimentovaní som dospel k záveru, že práca sa bude zaoberať tvorbou klasifikátoru s pomocou algoritmu XGBoost. Ako je v úvode sekcie 4.1.2 spomenuté, xgboost potrebuje na vstupe správny formát dát. Toto bude docielené transformáciou dát s použitím CICFlowmetru. Vedľajším bonusom je, že tento nástroj generuje relatívne malé množstvo chýbajúcich hodnôt v dátach, s čím XGBoost vie efektívne pracovať. Výstup programu CICFlowmeter sú sieťové toky dát v CSV[12] formáte. Tieto toky vo formáte CSV sa následne spracujú pomocou python knižnice pandas. Vznikne tak dvojrozmerný súbor dát vo formáte `dataframe`, kde riadky predstavujú jednotlivé toky a stĺpce sú jednotlivé rysy príslušných tokov.

[12]CSV - z anglického Comma-Separated values, teda čiarkou oddelné hodnoty

Výstupom experimentácie budú správne zvolené parametre na trénovanie klasifikátoru, a taktiež s ich pomocou vytrénovaný model, ktorý bude schopný správne detegovať a klasifikovať anomálie na jednotlivých tokoch. Pre účely finálneho trénovania modelu použijem predpripravené, označkované súbory. Tieto súbory pochádzajú z datasetu CIC-IDS2017 a CIC-IDS2018. Tento súbor dát ďalej skript rozdelí na dva súbory, a to trénovací a testovací v pomere 9 ku 1. Pri stratifikácii berieme ohľad na čo najpresnejšiu reprezentáciu typov útokov (značiek) do súborov na trénovanie a testovanie, taktiež v pomere 9 ku 1. Teda v prípade existencie desiatich označených tokov ide 9 na trénovanie a 1 na následné testovanie.

S využitím trénovacieho súboru dát vytrénujeme model, ktorého validitu budeme testovať pomocou predpripraveného testovacieho súboru dát. Jeho úspešnosť vyhodnocujeme na základe rovnakých metrík, ktoré sme používali aj na vyhodnotenie vo fáze experimentácie. Špecificky sa jedná o AUROC a skóre F1, ktoré sú bližšie popísané v sekcii 4. Na základe výsledkov zisťujeme spoľahlivosť detekcie.

Následne je možné simulovať alebo získať ďalšie reálne dáta, ktoré sa pomocou fronty (pipeline) spracujú z formátu PCAP do podoby vyžadovaného tvaru tokov s rysmi v CSV a následne ich spracovať vytrénovaným modelom, ktorý bude generovať k jednotlivým tokom predikcie a klasifikovať ich do predurčených tried.

Vďaka vyššie popísanému postupu sa vytvorí model, ktorý je možno ďalej rozširovať, alebo optimalizovať. Môže sa buď kontinuálne trénovať stávajúci model vytvorením nových trénovacích a doplnením testovacích dát, alebo môžeme uplatniť celý spomenutý proces a trénovať ho od základu.

Kapitola 5

Implementácia

V nasledujúcej kapitole priblížim postup implementácie systému a jeho následnú validáciu s ňou spojené dosiahnuté výsledky. Podrobne vysvetlím, ako som vyberal dáta, ako sa spracovávajú pre potreby trénovania klasifikátoru a ako som pristupoval k výsledkom a ich vyhodnocovaniu. Budem sa držať návrhu implementácie a využijem jazyk Python 3 v prostredí Jupyter Notebook. Dáta budú transformované nástrojom CICFlowmeter.

5.1 Získanie a výber dát

Najdôležitejším faktorom pre strojové učenie sú kvalitné vstupné dáta. Primárne sa to týka trénovania klasifikačného modelu. Rovnako dôležitá je aj kvalita dát použitých na validáciu výsledkov. Pre dosiahnutie kvalitných dát som sa rozhodol držať sa odporúčaní a diverzifikovať súbory dát určené na tréning a testovanie modelu. Rôznorodosť zdrojov by mala pozitívne ovplyvniť, a teda znížiť skreslenie dát.

Už z podstaty sieťových tokov práca počíta s nevybalancovanými dátami, kde anomálie budú tvoriť len zlomok celkového objemu, a teda s tým musíme pri vyhodnocovaní výsledkov počítať. Dosiahneme toho využitím váženého priemeru keď budeme spracovávať výsledné predikcie.

Dáta na trénovanie sa primárne skladajú z označkovaných súborov sieťových tokov, ktoré obsahujú, ako normálnu, tak aj anomálnu komunikáciu.

5.1.1 Profil prevádzky

Aby bol model schopný správne detegovať anomálie je nutné modelu definovať legitímnej prevádzky na sieti. Následne je potrebné modelu predstaviť správne označené anomálne toky. Typy anomálií a ako sa dajú kontextuálne detegovať je bližšie popísané v sekcii 2.4. Na tento účel využívam primárne súbory dát CIC-IDS2017 a CIC-IDS2018 [28]. Tieto súbory boli vytvorené Kanadským Inštitútom pre Kybernetickú bezpečnosť[1] na University of New Brunswick.

- **CIC-IDS2017**:

  Jedná sa o veľkoobjemový súbor dát, ktorý zachytáva sieťovú prevádzku v reálnom a rozmanitom sieťovom prostredí. Obsahuje benígné (neškodné) aj anomálne dáta zo sieťovej prevádzky. Dataset pokrýva širokú škálu útokov založených na sieťovej

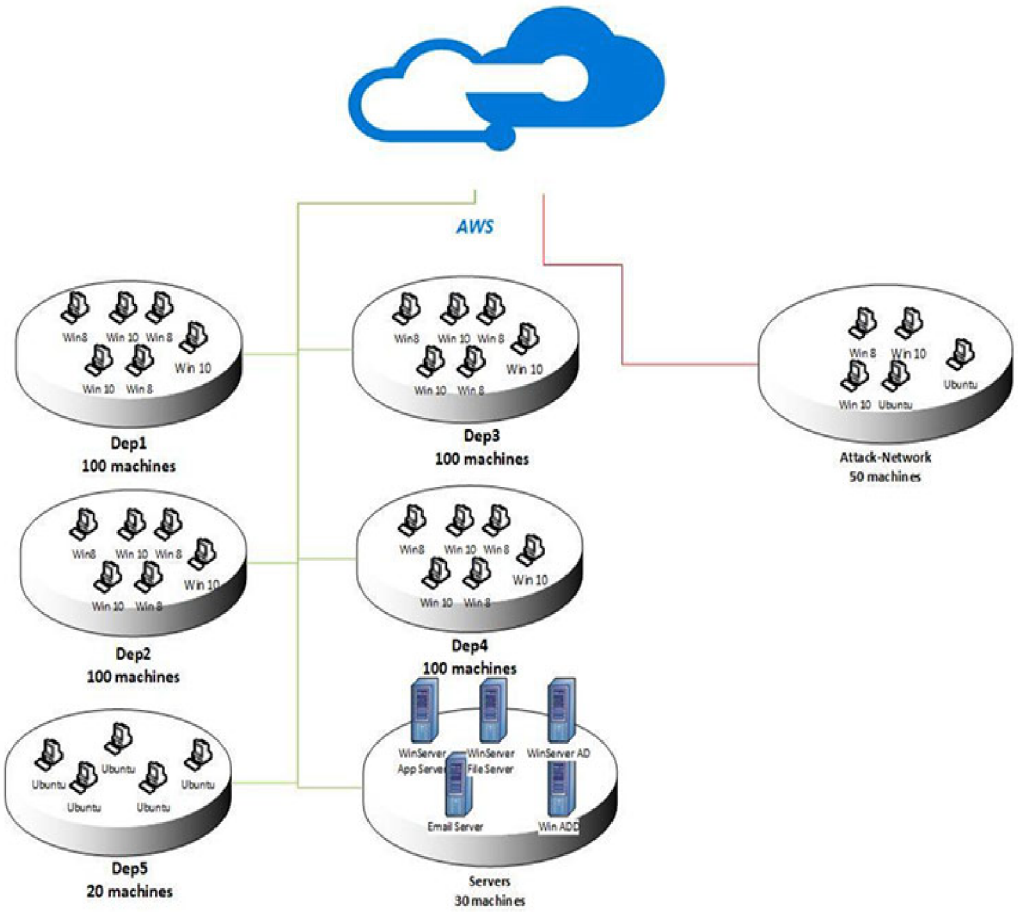
[1] anglicky Canadian Institute for Cybersecurity – CIC

komunikácii, vrátane útokov typu DoS (odmietnutie služby), DDoS (distribuované odmietnutie služby), vyzvedanie (Reconnaissance) a infiltrácia. Jedná sa o veľmi kvalitný zdroj na trénovanie a hodnotenie systémov na detekciu vniknutia do siete.

- **CIC-IDS2018**:

  Dataset CIC-IDS2018 je aktualizovanou, robustnejšiou verziou datasetu CIC-IDS2017. Obsahuje dodatočné vlastnosti, vylepšenia a presnejšie označenie dát z sieťovej prevádzky. Dataset zahŕňa nové scénare útokov, zachytáva sofistikovanejšie techniky útokov a poskytuje komplexnejšie reprezentácie sieťovej prevádzky aj vďaka navýšenému počtu simulovaných koncových bodov.

Oba datasety majú za cieľ poskytnúť realistické a reprezentatívne dáta zo sieťovej prevádzky, ktoré uľahčujú výskum a vývoj systémov IDS a na detekciu anomálií. Obsahujú mix sieťových protokolov, typov prevádzky a scénarov útokov, čo umožňuje analyzovať a modelovať rôzne typy sieťových útokov a hodnotiť účinnosť rôznych prístupov k detekcii intrúzie.



Obr. 5.1: Diagram topológie siete využitej k vytvoreniu datasetu CIC-IDS2018 https://www.unb.ca/cic/datasets/ids-2018.html

Ako je možné z obrázku 5.1 vidieť, jedná sa o veľmi robustný súbor dát. Jeho tvorba prebiehala v rámci niekoľkých týždňov každý deň prebiehala simulácia legitímnej prevádzky a

v určitých časových intervaloch boli simulované útoky. Narozdiel od roku 2017, kedy autori súborov dát pracovali desiatkami koncových bodov, v roku 2018 sa pracovalo so stovkami koncových bodov s rôznymi operačnými systémami a úlohami. Týmto bolo docielené vytvorenie realistických sieťových dát, ktoré sú vynikajúcou voľbou pre účely tejto práce. Dáta z roku 2017 som sa preto rozhodol použiť na záverečnú validáciu modelu a nie je nutné ich používať pri trénovaní.

Nasledujúca tabuľka obsahuje informácie o typoch útokov, ktoré boli v rámci tvorby datasetu CIC-IDS2018 simulované na vyššie spomenutej topológii (Obr. 5.1). Dáta sú vo forme označkovaných tokov s extrahovanými rysmi pomocou CICFlowMetru-V3. Sú rozdelené do súborov po dňoch. Každý deň boli simulované jeden alebo viacero druhov útokov.

Tabuľka 5.1: Obsah súboru dát CIC-IDS2018

Typ útoku	CSV súbor
FTP & SSH Bruteforce	Wednesday-14-02-2018_TrafficForML_CICFlowMeter.csv
GoldenEye & Slowloris DoS	Thursday-15-02-2018_TrafficForML_CICFlowMeter.csv
SlowHTTPtest & Hulk DoS	Friday-16-02-2018_TrafficForML_CICFlowMeter.csv
LOIC-HTTP/UDP DDoS	Tuesday-20-02-2018_TrafficForML_CICFlowMeter.csv
LOIC-UDP & HOIC DDoS	Wednesday-21-02-2018_TrafficForML_CICFlowMeter.csv
Web&XSS Bruteforce & SQL Injection	Thursday-22-02-2018_TrafficForML_CICFlowMeter.csv Friday-23-02-2018_TrafficForML_CICFlowMeter.csv
Infiltration	Wednesday-28-02-2018_TrafficForML_CICFlowMeter.csv Thursday-01-03-2018_TrafficForML_CICFlowMeter.csv
Botnet	Friday-02-03-2018_TrafficForML_CICFlowMeter.csv

Na trénovanie modelu boli využité práve tieto súbory. Na testovanie a validáciu modelu som použil aj dataset ToN-IoT [10, 5, 23, 22, 20, 19, 21, 7]. tento súbor dát obsahuje rôzne typy útokov a anomálnych aktivít, ktoré môžu ohroziť bezpečnosť IoT sietí. Medzi tieto útoky patria napríklad DoS útoky, Man-in-the-Middle útoky, Botnet útoky a rôzne typy skenovania a exploitačných pokusov.

5.2 Príprava dát a trénovanie modulu

V tejto sekcii sa budem venovať spracovaniu dát zo zachyteného súboru formátu PCAP na vstupný formát vhodný pre trénovanie a validáciu pomocou metódy XGBoost. Ako som priblížil v sekcii 4.3 základnú transformáciu dát a extrakciu rysov riešim externým programom CICFlowMeter. Jediná zmena ohľadom implementácie tohoto programu oproti návrhu je, že používam verziu 4, ktorá je implementovaná v jazyku Python a bola naposledy updatovaná v roku 2021. Toto značne uľahčuje jeho inštaláciu a implementáciu v procesovej fronte.

Na vstupe očakáva súbory vo forme PCAP a výstupom sú spracované pakety do formy sieťových tokov vo formáte CSV. Ďalej v Jupyter Notebooku načítame tieto súbory z definovanej zložky a pokračujeme v ich spracovaní. Jedinou otázkou v tomto bode zostáva, či je cieľom vytrénovať nový model, alebo využiť existujúci na klasifikáciu tokov.

5.2.1 Príprava dát

Akýkoľvek je náš cieľ, najskôr je nutné z súboru CSV odstrániť nepotrebné stĺpce, ako časovú značku, a oddeliť prípadné označkovanie tokov od rysov, ktoré ich kategorizujú.

Tohoto docielime postupným načítaním všetkých žiadaných CSV súborov pomocou knižnice `pandas`[2] do dátovej štruktúry `dataframe` vďaka ktorej sme schopný rýchlo a efektívne pracovať s celým konkatenovaným súborom dát.

Taktiež je dobré očistiť súbor a skonvertovať všetky chybné hodnoty, ako napríklad kladné a záporné nekonečná na NaN[3] a následne tieto hodnoty spolu s celými riadkami, ktoré ich obsahujú zmazať, aby sme zabránili zbytočnému skresľovaniu dát.

V prípade, že máme označkovaný súbor dát, kde sú k jednotlivým tokom priradené hodnoty značiace, či sa jedná o legitímnu komunikáciu alebo anomáliu, oddelíme tento stĺpec do vlastnej premennej. Po tomto oddelení je nutné manipulovať, ako s premennou `dataframe` tak s listom značiek spoločne a zachovať medzi nimi izomorfný vzťah. Taktiež pri zmene poradia prvkov v jednej premennej musíme rovnako zmeniť poradie aj v druhej.

5.2.2 Trénovanie modulu

Klasifikačný model XGBoost bol implementovaný pomocou knižnice scikit-learn. Pri trénovaní je nutné rozdeliť vstupný súbor dát na trénovací a testovací. Tohoto je dosiahnuté pomocou zabudovanej metódy `sklearn.model_selection.train_test_split`. Pri jej implementácii som využil pomer rozdelenia dát test:tréning 1:9. Táto funkcia rozdelí dané 2 premenné (dáta a značky) do 4 premenných:

- Trénovacie dáta
- Trénovacie značky
- Testovacie dáta
- Testovacie značky

Taktiež som využil vstavanej schopnosti funkcie `stratify=značky`, ktorá dáta stratifikuje vzhľadom na počet jednotlivých značiek dát aby som predišiel problému, že by sa dáta rozdelili bez reprezentácie v jednej z premenných.

Vzhľadom na veľkosť súborov dát, s ktorými som v rámci práce pracoval, bolo nutné v neskorších fázach práce (tvorba samotného robustného multiklasifikátoru) tieto dáta rozdeliť na menšie, rovnomerné časti a uložiť ich na disk. Následne som postupoval tak, že tieto rovnomerne rozdelené súbory som po jednom načítal a postupne nimi trénoval model.

Počas fáze experimentácie s jednotlivými útokmi a prvotných pokusoch o optimalizáciu hyperparametrov klasifikátoru nebolo nutné trénovacie dáta takto rozdeľovať a ukladať na disk. Vzhľadom na dôležitosť hyperparametrov pre vylepšenie presnosti a efektivity modelu sa im v nasledujúcej časti budem bližšie venovať. Samotná knižnica XGBoost, špecificky v mojom prípade scikit-learn API[4] nadstavba v podobe `xgboost.XGBClassifier`[5] má mnoho nastaviteľných hyperparametrov.

[2] https://pandas.pydata.org/

[3] NaN - z anglického Not a Number, značí, že hodnota premennej nie je číslo

[4] API - z anglického Application Programming Interface, je sada pravidiel a nástrojov, ktoré umožňujú programom a softvéru komunikovať a vzájomne sa integrovať.

[5] https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier

Tieto parametre sa v prípade použitia scikit-learn API nastavujú priamo objektu xgboost.XBGClassifier(parametre). Tento objekt následne umožňuje volať jeho funkciu `fit(trénovacie dáta, eval_set=testovacie dáta)` čím sa započne trénovanie modelu na súbore trénovacích dát. Parameter eval_set určuje súbor dát na kontrolu výkonu modelu počas jeho trénovania. V tomto prípadne je optimálne využitie parametru early_stopping_rounds, ktorý slúži na predčasné ukončenie trénovania modelu ak je zaznamenaný nezlepšujúci sa výkon modelu, čo môže signalizovať pretrénovanie.

5.2.3 Ladenie hyperparametrov

Na úvod tejto sekcie priblížim viaceré hyperparametre[6] klasifikátoru XGBoost. Následne stručne popíšem proces ich ladenia a ako som s nimi experimentoval a zhrniem moje zistenia a použitú konfiguráciu na model.

Ako bolo v závere predchádzajúcej sekcie písané, využívam nadstavbu scikit-learn API `xgboost.XGBClassifier` nad samotnou knižnicou XGBoost. Tá poskytuje, rovnako, ako originálna knižnica, možnosť nastaviť parametre v štruktúre `parameters`. Týmto spôsobom vieme algoritmu povedať ako sa má pri trénovaní správať a ovplyvniť tak jeho výsledky. Oproti originálnej knižnici je, v prípade tejto práce, v podstate jediným rozdielom možnosť priamo do parametrov nastaviť globálne parameter `eval_metric`, ktorým určíme metriku, ktorá udáva výkon modelu. Toto eliminuje potrebu túto metriku definovať pri každej inštancii trénovania. Bežne sa na regresné úlohy používa 'rmse' (Kvadratický koreň zo strednej kvadratickej chyby) alebo 'logloss' (Logaritmická strata) pre binárne klasifikačné problémy.

Keďže táto práca sa zaoberá nebinárnym multi klasifikačným problémom, tak som pre jej účely zvolil metriku 'auc' (Plocha pod ROC krivkou). Tradične sa tiež používa skôr v binárnych klasifikačných úlohách. Ale v rámci xgboost je s ňou možné pracovať aj pri multiklasifikačných problémoch a dosiahnuť výborných výsledkov. Metrika AUC je obzvlášť užitočná, keď je dôležité správne identifikovať obe triedy – pozitívnu a negatívnu. Zvyčajne sa používa v kritických oboroch ako medicína, finančný sektor alebo na biometrické úlohy.

Pomocou špecifikovanej evaluačnej metriky XGBoost optimalizuje parametre modelu počas trénovania s cieľom minimalizovať chybu alebo maximalizovať výkon, ako je naznačené vybranou evaluačnou metrikou.

- **tree_method**:

  Určuje metódu, ktorá sa použije na konštrukciu rozhodovacích stromov v algoritme XGBoost.

- **booster**:

  Určuje typ gradient boostingu použitý v algoritme.

- **n_estimators**:

  Určuje počet stromov v modeli.

- **early_stopping_rounds**:

  Určuje počet iterácií, po ktorých bude trénovanie zastavené, ak sa nezlepšuje výkon modelu.

- **objective**:

  Určuje stratovú funkciu, ktorá sa minimalizuje počas trénovania.

[6] https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier

- **num_class**:

  Určuje počet tried v klasifikačnom probléme.

- **learning_rate**:

  Určuje veľkosť kroku pri aktualizácii váh modelu.

- **max_depth**:

  Určuje maximálnu hĺbku každého rozhodovacieho stromu.

- **eval_metric**:

  Určuje metriku použitú na vyhodnocovanie výkonu modelu počas trénovania.

- **subsample:**

  Tento parameter určuje pomer (od 0 do 1) vzoriek, ktoré sa náhodne vyberú pri trénovaní každého stromu. Hodnota 1 znamená, že sa vyberú všetky vzorky, zatiaľ čo hodnota menšia ako 1 vyberá náhodný podiel vzoriek.

- **colsample_bytree:**

  Tento parameter určuje pomer (od 0 do 1) stĺpcov, ktoré sa náhodne vyberú pri vytváraní každého rozhodovacieho stromu. Hodnota 1 znamená, že sa vyberú všetky stĺpce, zatiaľ čo hodnota menšia ako 1 vyberá náhodný podiel stĺpcov.

- **reg_lambda:** Tento parameter predstavuje regularizačný člen pre L2 (ridge[7]) regularizáciu. Hodnota λ určuje silu regularizácie, pričom vyššia hodnota znamená silnejšiu regularizáciu.

- **reg_alpha:**

  Tento parameter predstavuje regularizačný člen pre L1 (LASSO[8]) regularizáciu. Hodnota α určuje silu regularizácie, pričom vyššia hodnota znamená silnejšiu regularizáciu. L1 regularizácia pomáha redukovať nevýznamné príznaky tým, že nastavuje ich váhy na nulu.

Pri procese ladenia hyperparametrov som vychádzal z rôznych štúdií, a jednoducho dostupných odporúčaní výskumníkov, ktorý často pracujú s XGBoostom. Na základe týchto znalostí som si na úvod navrhol rozsah parametrov pre funkciu[9] krížovej validácie, ktorá potom automaticky otestovala tieto rozsahy pre účely riešenia mojej problematiky.

Týmto spôsobom som vytvoril základnú líniu hyperparametrov a v priebehu iteračnej experimentácie som ich manuálne ladil, ak som videl priestor na optimalizáciu výkonu modelu. V následujúcej tabuľke 5.2 je možné vidieť porovnanie základnej línie a posledných hodnôt hyperparametrov použitých na tréning modelu.

Manuálne som počas finálnej fáze napríklad znížil `n_estimators` pretože komplexita 15 tried vyústila v tréning, ktorý sa bez pretrénovania bol schopný po malých inkrementoch zlepšovať aj 500 iterácii, no v rozsahu 125 až 175 som narazil na bod klesajúcich výnosov. Signálom bolo, že sa hodnota metriky auc zvyšovala na štvrtom až piatok desatinnom mieste každé 2-3 iterácie.

[7] Ridge regularizácia je technika v oblasti strojového učenia, ktorá sa používa na zmiernenie preučenia modelu.

[8] Lasso regularizácia je metóda používaná v strojovom učení na redukciu prebytočných parametrov vo vysoko-dimenzionálnych dátach.

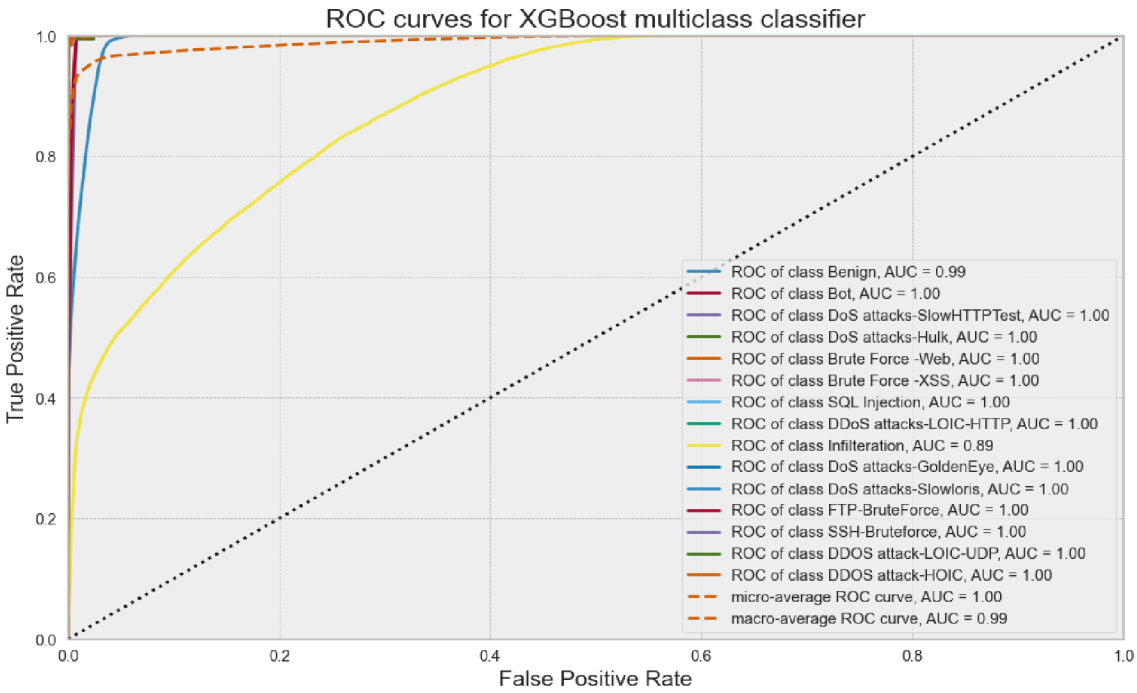
[9] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Tabuľka 5.2: Vybrané hyperparametre na základe ladenia

Parameter	Originálna Hodnota	Manuálne úpravy
tree_method	'gpu_hist'	'gpu_hist'
booster	'gbtree'	'gbtree'
n_estimators	250	150
early_stopping_rounds	30	10
objective	'multi_softmax'	'multi_softmax'
num_class	15	15
learning_rate	0.15	0.2
max_depth	8	7
eval_metric	'auc'	'auc'
subsample	0.8	1
colsample_bytree	0.8	1
reg_alpha	0	0.8
reg_lambda	1	0.8

5.3 Testovanie

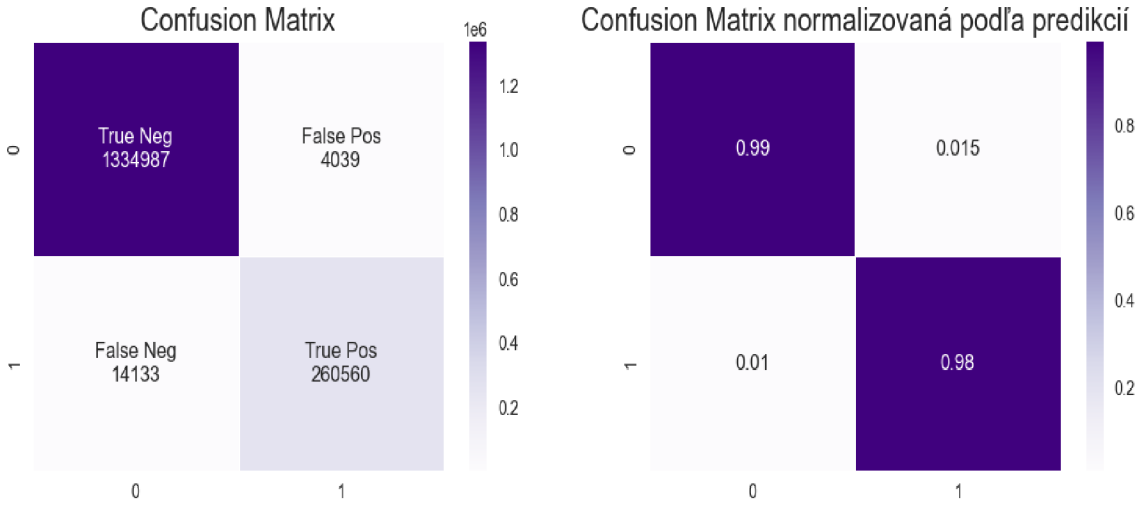
Testovanie prebiehalo vo viacerých fázach, výsledky ktorých budem prezentovať rôznymi formami výstupných dát. Prvou, najrobustnejšiou bolo využitie staršieho CIC-IDS2017 datasetu. Keďže obsahoval takmer rovnako spracované a označkované dáta na inej topológii jednalo sa o ideálne validačné dáta. Skombinoval som ich preto s testovacími dátami z datasetu CIC-IDS2018 a výsledkom je nasledujúci obrázok 5.2 všetkých AUROC kriviek jednotlivých útokov.



Obr. 5.2: Výsledky testovania modelu na klasifikáciu 15 tried.

AUROC krivka je zvyčajne používaná na vyhodnotenie výkonu binárnej klasifikácie, no vďaka knižnici yellowbrick[10] je možné spracovať výsledky multiklasifikátoru po jednotlivých triedach do jedného grafu. Limitáciou tejto funkcie je žiaľ slabá možnosť konfigurácie grafickej stránky výsledkov, a preto ju už viac nebudem používať na reprezentáciu výsledkov. Na grafe je však stále možné vidieť, že väčšina tried dosahuje hodnôt AUC >99% čo je veľmi pozitívny výsledok. Všetky krivky opisujú takmer dokonalý obdĺžnik čo značí vysokú presnosť modelu. Jedinou odľahlou triedou je infiltrácia. K tejto téme sa vrátim na záver sekcie kapitoly, kde sa podrobnejšie pozriem na samotné dáta a čo reprezentujú.

Pre bližší pohľad na túto fázu testovania budem používať teplotnú mapu implementovanú knižnicou Seaborn[11]. Nasledujúci obrázok 5.3 obsahuje dáta o štatistických veličinách modelu. Pre účely ilustrácie som na tieto matice spojil všetky výsledné dáta z klasifikácie 15 tried. Najdôležitejšie sú dáta na vedľajšiej diagonále a teda falošné pozitíva a falošné negatíva. Tie priamo určujú koľko tokov bolo zle kategorizovaných. Tieto čísla môžeme, spolu s anotáciami, vidieť na ľavej matici. Pre lepšiu vizualizáciu reálnych výsledkov je však vhodnejšie využiť normalizované hodnoty, ktoré vytvoria lepší obraz o našom nevybalancovanom súbore dát vo forme hodnôt z intevalu <0,1> kde 1 je perfektný klasifikátor.



Obr. 5.3: Zobrazenie štatistických veličín. Naľavo je matica s reálnymi hodnotami a anotáciou, napravo je matica normalizovaná podľa predikcií.

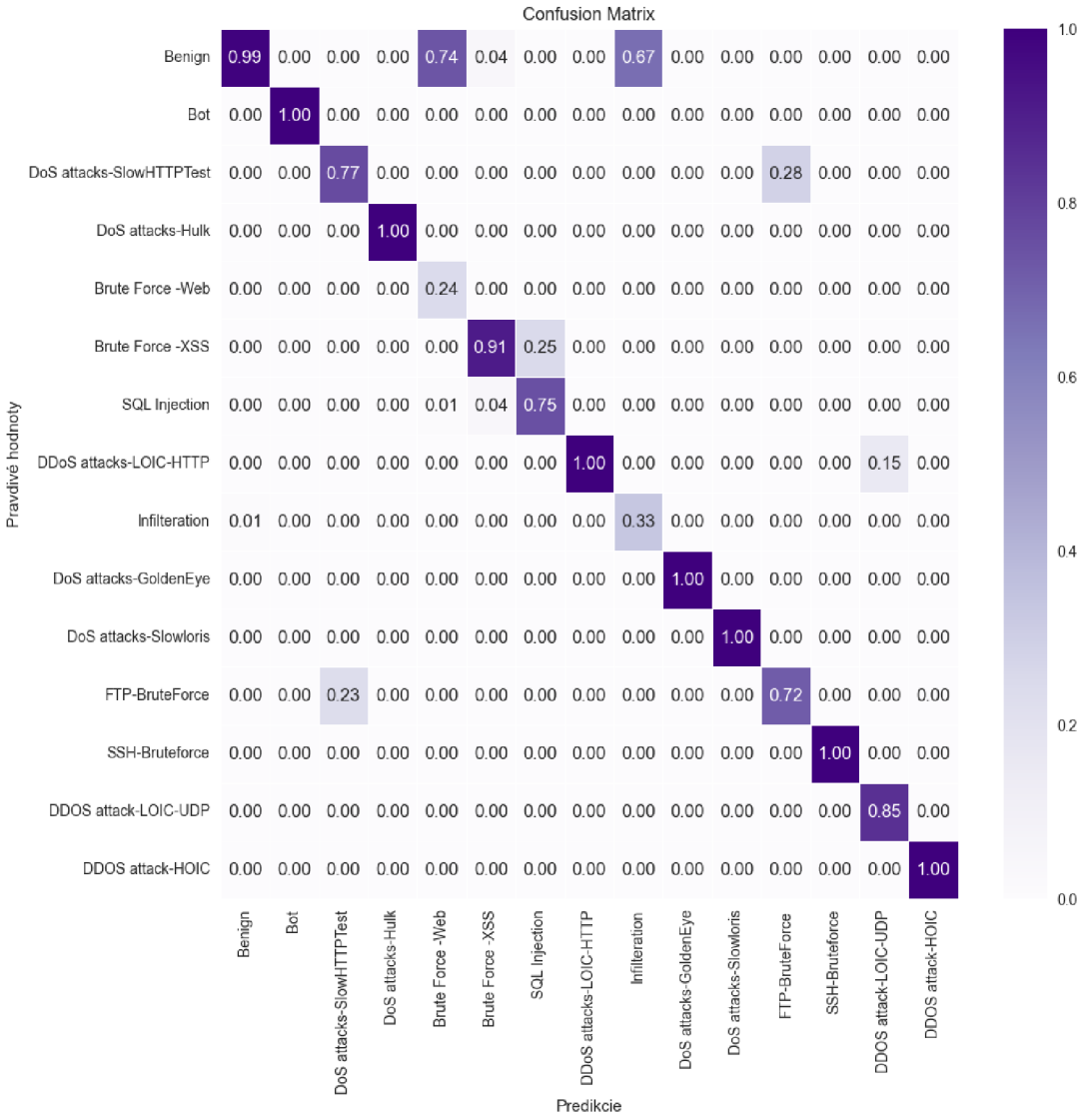
Z matíc je možno vyčítať, že náš model je vysoko presný, ale obsahuje alarmujúce množstvo falošne negatívne klasifikovaných tokov. Pre účely nasadenia modelu na rozpoznávanie útokov je toto horšia varianta, nakoľko to znamená, že veľa tokov, ktoré signalizujú útok sa nepodarilo detegovať a boli označené za legitímnu prevádzku. Ako je však možné vyčítať nižšie, z tabuľky 5.3, presnejšie podľa precision skóre, ktoré vyjadruje pomer všetkých správne označených pozitívnych nálezov ku všetkým pozitívnym nálezom, väčšina týchto tokov je výsledkom zlej klasifikácie útoku typu infiltrácia (Precision skóre pre tento typ útoku dosahuje len 0.33 pre 16 tisíc tokov, čo indikuje, že mu prislúcha takmer 12 tisíc z 14133 FN klasifikácií).

[10]Knižnica yellowbrick https://www.scikit-yb.org/en/latest/api/classifier/rocauc.html#multi-class-rocauc-curves

[11]Seaborn heatmap https://seaborn.pydata.org/generated/seaborn.heatmap.html

Falošne pozítívne klasifikované toky sú pre model, tvorený pre kritické prostredia lepšia, no stále neoptimálna varianta, keďže sú to práve FP klasifikácie, ktoré vytvárajú záplavu kyberbezpečnostných expertov a prispievajú k fenoménu `alert fatigue`.

Na nasledujúcej matici môžeme vidieť normalizované rozloženie presnosti klasifikácie na všetky modelom detegované typy útokov.



Obr. 5.4: Normalizované štatistické veličiny všetkých typov útokov

Matica je normalizovaná podľa predikcii, resp. stĺpcov. Všetky útoky, ktoré po hlavnej diagonále dosahujú hodnoty >0.99 by som nazval úspešným výsledkom modelu. Z matice sa dajú vyčítať rôzne anomálie, prvou je prípad prekrytia `DoS SlowHTTPTest` a `FTP-BruteForce`. Oba tieto útoky na hlavnej diagonále dosahujú hodnoty ~ 0.75 a zvyšných ~ 0.25 si premietajú navzájom, ako nesprávne klasifikované triedy, ale stále sú správne klasifikované ako anomália/útok. Toto by som označil ako čiastočne pozitívny výsledok, nakoľko model stále správne deteguje útok no nesprávne ho klasifikuje. Takmer rovnaký prípad je dvojica `SQL Injection` a `BruteForce -XSS`, s tým rozdielom, že v prípade druhého typu

útoku máme 4% tokov nesprávne klasifikovaných ako normálnu prevádzku. Týmto sa dostávame k druhým, tentokrát už menej pozitívnym, prípadom – tými sú `BruteForce-Web` a `Infiltration`. Tu vidíme, že chýbajúce hodnoty z hlavnej diagonály sa premietajú na benígnú prevádzku, a teda sú klasifikované ako legitímna komunikácia.

V nasledujúcej tabuľke 5.3, ktorá obsahuje dáta vygenerované pomocou funkcie `classification_report`[12], nájdeme hodnoty 3 dôležitých metrík používaných na vyhodnocovanie výkonu modelu. Z týchto hodnôt je možno získať bližší náhľad na nedokonalé hodnoty v matici 5.4.

Tabuľka 5.3: Výsledky z `Classification Report` v číslach

Trieda útoku	Precision	Recall	F1 skóre	# Tokov
Legitímne toky	0.99	1.0	0.99	1339026
Bot	1.0	1.0	1.0	28619
DoS attacks-SlowHTTPTest	0.77	0.51	0.61	13989
DoS attacks-Hulk	1.0	1.0	1.0	46191
Brute Force -Web	0.24	0.9	0.38	61
Brute Force -XSS	0.91	0.91	0.91	23
SQL Injection	0.75	0.33	0.46	9
DDoS attacks-LOIC-HTTP	1.0	1.0	1.0	57619
Infiltration	0.33	0.12	0.18	16064
DoS attacks-GoldenEye	1.0	1.0	1.0	4151
DoS attacks-Slowloris	1.0	1.0	1.0	1099
FTP-BruteForce	0.72	0.89	0.79	19335
SSH-Bruteforce	1.0	1.0	1.0	18759
DDOS attack-LOIC-UDP	0.85	0.76	0.8	173
DDOS attack-HOIC	1.0	1.0	1.0	68601
macro priemer	0.84	0.83	0.81	1613719
priemer	0.98	0.98	0.98	1613719

V tabuľke 5.3 je možné vidieť, že sme potvrdili v návrhu spomínané fakty, kde nedostatok dát vedie ku zníženej kvalite predikcií. Na túto skutočnosť som upozornil už pri obrázku 5.4, kde som tieto podpriemerné výsledky rozdelil do viacerých tried. Dáta z tabuľky podporujú hypotézu, že toky `DoS attacks-SlowHTTPTest` a `FTP-BruteForce` sa zamieňajú z dôvodu podobných hodnôt charakteristických rysov, nakoľko v našom súbore tokov majú dostatočnú reprezentáciu. Výnimkou ostávajú toky útokov typu infiltrácia, ktoré pravdepodobne nedosahujú vysokých hodnôt kvôli širokému spektru rozdielných techník, ktoré pod tento typ útoku spadajú. V trénovaciom datasete boli ako `Infiltration` označené všetky toky jednotlivých akcií vykonaných koncovými bodmi, ktoré boli monitorovateľné na sieti. Medzi inými sem patrí komunikácia Metasploitu[13] s cieľom vytvoriť C2[14] s koncovým bodom kam patrí vytvorenie backdooru[15], IP sken, sken všetkých portov, enumerácia služieb, komunikácia (vrátane exfiltrácie všetkých zistených dát) späť na C2 kontrolný bod atď.

[12] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

[13] Metasploit je populárny a výkonný rámec (framework) pre penetračné testovanie a využívanie zraniteľností.

[14] C2 (Command and Control), je termín používaný v kybernetickej bezpečnosti a týka komunikačného kanálu medzi útočníkom a infikovaným systémom (napríklad kompromitovaný koncový bod alebo botnet)

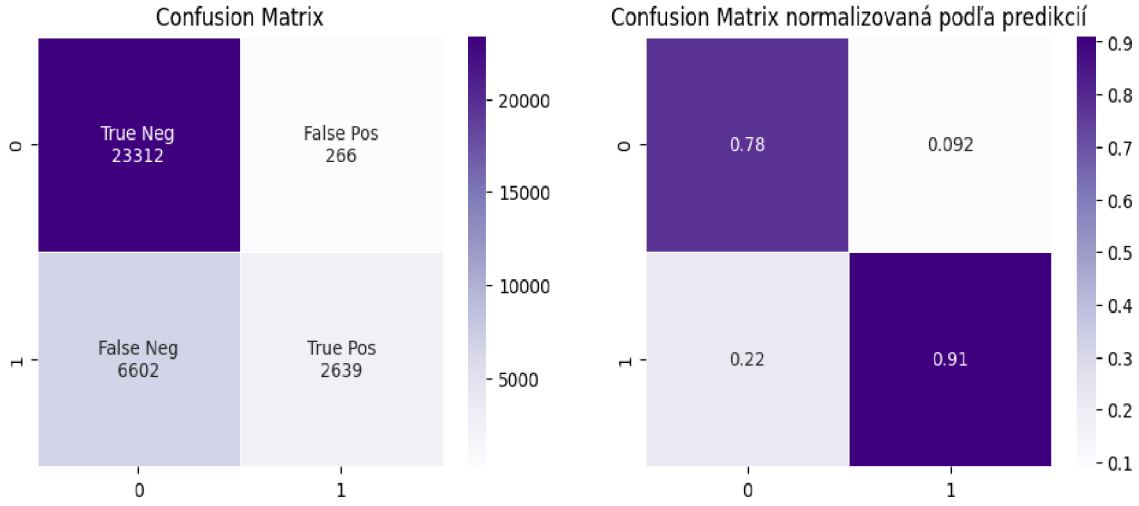
[15] Backdoor je bezpečnostná slabosť alebo malware, ktorý je úmyselne vytvorený s cieľom umožniť neoprávnený prístup a ovládanie tohto systému.

Pri bližšom prieskume počas 2. fáze testovania špecificky zameranej na klasifikáciu infiltrácie na súboroch dát som dospel k nasledujúcim výsledkom:

Tabuľka 5.4: Výsledky z `Classification Report` pre infiltráciu v číslach

Trieda útoku	Precision	Recall	F1 skóre	# Tokov
Benign	0.78	0.99	0.87	23578
Infiltration	0.91	0.29	0.43	9241
macro avg	0.84	0.64	0.65	32819
avg	0.82	0.79	0.75	32819

Z dát viditeľných v tabuľke 5.4 a na maticiach nižšie môžme vidieť, že aj pri špecializovanom modele na typ útoku infiltrácie z datasetu CIC-IDS2018 nedosahuje kvalitných výsledkov. Toto naznačuje 2 možné problémy. Prvým môže byť fakt, že vybrané hyperparametre a rysy nie sú vhodné na trénovanie modelu na ich detekcie. Vzhľadom na skutočnosť, že sa mi tieto hodnoty nepodarilo štatisticky významne zmeniť ani s použitím krížovej validácie viacerých parametrov sa žiaľ musím priklоniť k druhému vysvetleniu. Je totiž možné, že kvôli komplexite a rôznorodosti operácií, ktoré boli tvorcami súboru dát vykonané a následne zhrnuté do jedného vreca, ako Infiltrácia, došlo k vytvoreniu natoľko náhodného súboru rysov, že ich nie je možné správne rozlíšiť od profilu legitímnych sieťových tokov. Toto tvrdenie podopiera aj fakt, že z matíc na obrázku 5.5 môžeme vyčítať veľké množstvo falošne negatívne a naopak významne znížené množstvo správne negatívne vyhodnotených tokov. Táto skutočnosť naznačuje, že model nevie správne rozlíšiť normálnu komunikáciu od komunikácie útočník(ov). Naopak relatívne nízke, no samostatne nezanedbateľné, množstvo falošne pozitívnych klasifikácií v porovnaní s FN značí, že model nie je pretrénovaný a v prípade dostatku kvalitných rysov je schopný správne rozoznať tento typ útoku. Chyba teda bude pravdepodobne v skreslení trénovacích dát. Vzhľadom na relatívne objemný súbor dát sa s vysokou pravdepodobnosťou problém týka skreslenia spôsobeným zle definovanými dátami. Problém by mohlo vyriešiť rozdelenie značky infiltrácia na jej jednotlivé podmnožiny popísané bližšie v sekcii 4.2.1.



Obr. 5.5: Dvojica matíc zobrazujúca štatistické veličiny modelu na detekciu infiltrácie. Naľavo je matica s reálnymi hodnotami a popisom, napravo je matica normalizovaná podľa predikcií.

5.4 Vyhodnotenie

Klasifikácia anomálií nie je triviálnou úlohou. V každom kroku popísanom v tejto práci a zhrnutom v sekcii 4.4 je možné urobiť natoľko kritickú chybu, že jej následky negatívne ovplyvnia výsledný modul a jeho schopnosť správne klasifikovať útoky.

V tejto sekcii sa budem venovať záverečnému vyhodnoteniu práce a vyjadrím sa k jednému podstatnému bodu, ktorý v konečnom dôsledku nezohral takú úlohu, akú som pri výskume a navrhovaní očakával.

5.4.1 Rysy

Tým bodom sú práve rysy, resp. volatilita ich dôležitosti. V priebehu fáze experimentácie som narazil na skutočnosť, že v jednotlivých iteráciach trénovania modulu a ladení hyperparametrov sa konštatne a významne menilo poradie dôležitosti jednotlivých rysov pre rovnaké klasifikačné úlohy. Aj vzhľadom na relatívne plytký výskum tejto skutočnosti by som fluiditu dôležitosti rysov stále nazval podstatnou. Pre ilustráciu som graf dôležitosti vložil ako prílohu A k tejto práci. Exponenciálny tvar rozloženia dôležitosti je, z môjho pozorovania, konštantný skrz všetky iterácie. Čo sa však mení je poradie jednotlivých rysov až na pár výnimiek.

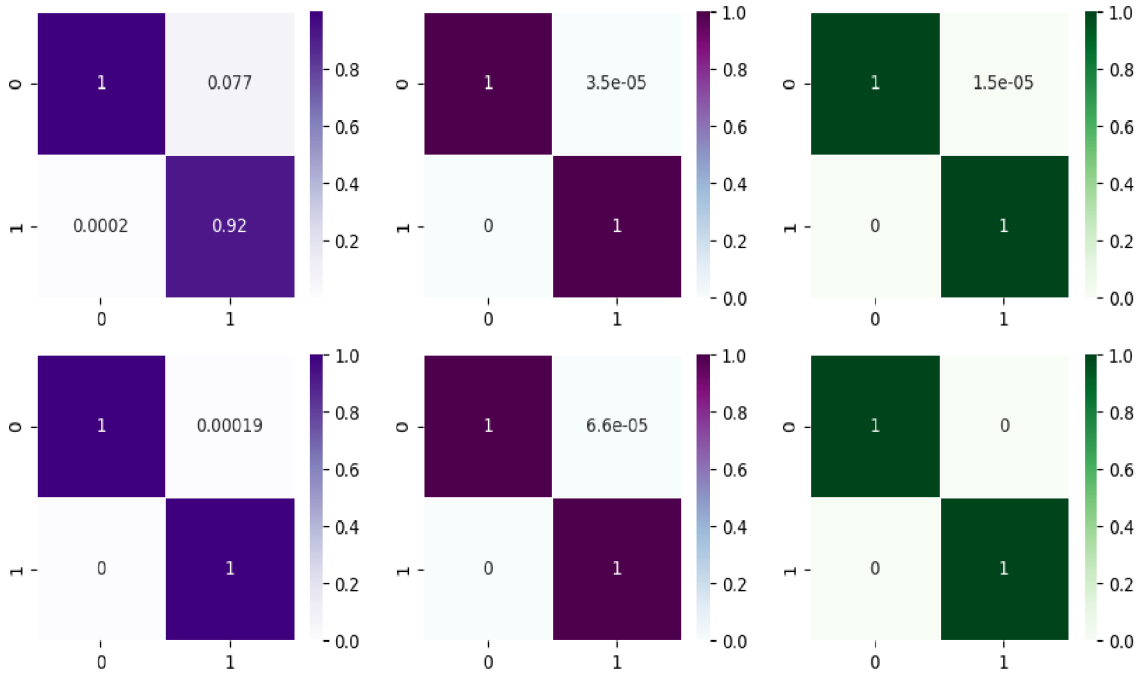
Z tohoto vyplývajú dve hlavné možnosti. Prvou z nich je fakt, že v spojení s relatívne vysoko kvalitnými výsledkami finálneho klasifikátoru to značí, že táto kvantita neškodí presnosti a naopak umožňuje vyššiu úroveň flexibility a možnosti pre model zachytiť jemnejšie nuansy sieťových tokov.

Druhá možnosť je, že rysy generované CICFlowMetrom sú zbytočne komplexné a bolo by možné ich redukciou urýchliť trénovanie modelov bez štatisticky významnej straty presnosti modelu.

Keďže cieľom tejto práce nie je vyhodnocovať túto problematiku bližšie a dosiahnuté výsledky sú kvalitné, tak som dospel k záveru, že nemá cenu sa tejto téme nadmerne venovať, no určite je tu priestor na plodný výskum.

5.4.2 Vyhodnotenie funkčnosti

Aj napriek inklúzii útokov, ktorých dáta, ako bolo ukázané v sekcii 5.3, boli nedostatočné, dosiahla úspešnosť klasifikácie mimo týchto typov F1 skóre >99.7%. Model úspešne klasifikuje typy útokov, na ktorých bol vytrénovaný a generuje minimum falošných klasifikácií.



Obr. 5.6: Ukážka dosiahnutej presnosti s modelom bez skresľujúcich dát, rôzne súbory tokov dát s jedným typom útoku. Primárny zdroj z ToN IoT datasetu. Matice sú normalizované podľa predikcií.

Oproti prvým iteráciam modelu sa jedná o zlepšenie, a výsledky sú porovnateľné s výsledkami[16] štúdií z posledných rokov. Skoro nulová prítomnosť falošne negatívnych klasifikácií je kritická pre bezpečnostné účely a vytrénovaný model toto spĺňuje. Taktiež sa podarilo vo vybraných kategóriach priblížiť k nulovému množstvu falošne pozitívnych výsledkov, čím sa redukuje možnosť, že model by negatívne ovplyvňoval analytikov. Teda minimalizuje efekt `alert-fatigue` čo bol jeden z vedľajších cieľov práce.

[16] https://paperswithcode.com/sota/network-intrusion-detection-on-cicids2017

Kapitola 6

Záver

Cieľom práce bolo vytvoriť systém schopný detegovať anomálie na dátovych tokoch s vysokou spoľahlivosťou aby sa minimalizovalo množstvo falošne pozitívnych a falošne negatívnych výsledkov.

Počas výskumnej časti tejto práce som aplikoval a prehĺbil svoje znalosti z praxe ohľadom systémov na detekciu a prevenciu útokov a objasnil som si najdôležitejšie koncepty o strojovom učení a detekcii anomálií. Následne som si zjednotil novonadobudnuté poznatky s existujúcimi a vypracoval teoretickú časť tejto práce. Aj napriek tomu, že sa nejedná o triviálnu tému sa mi na základe získaných informácií podarilo pripraviť si metodiku na porovnávanie klasifikačných metód a načrtnúť nasledujúce kroky vývoja implementačnej časti.

Následne som iteroval a testoval efektivitu jednotlivých klasifikačných metód a ich schopnosti vytvoriť multiklasifikátor s cieľom čo najpresnejšieho a efektívneho modelu na rozpoznávanie anomálií na sieťových tokoch. Počas fáze experimentácie som finalizoval výber algoritmu a spracoval teoreticky nutné množstvo rysov na detekciu viacerých typov útokov. Na základe obšírnej analýzy problematiky spracovania sieťových tokov a formy, ktorú tieto toky zvyčajne nadobúdajú som sa rozhodol pre algoritmus strojového učenia XGBoost. Mimo iné zaváţila najviac skutočnosť, že od jeho vzniku sa XGBoost stal štandardom na prácu s tabuľkovými dátami. Tento fakt sa jasne prejavil vo všetkých aspektoch práce s týmto algoritmom, vrátane rýchlosti, vysokej presnosti a samotnej jednoduchosti riešenia čo v zápätí viedlo k rapídnemu zisku vedomostí.

Fáza návrhu riešenia, aj napriek úspechom dosiahnutým vďaka skvelým výsledkom experimentácie, narazila takmer okamžite na problémy, nakoľko sa aj napriek vynaloženému úsiliu nepodarilo dosiahnuť požadovaného stavu pri práci so samotným systémom IDS Suricata. Výstupy tejto aplikácie sú síce kvalitné a svojím spôsobom aj flexibilné na integrácie s inými systémami, ale je ťažké ich pretaviť do podoby, ktorá je optimálna pre XGBoost, ktorý má problém s nekonzistenciou vstupných dát. Preto som v tejto fáze prešiel viacerými iteráciami dokým som dosiahol uspokojujúceho riešenia. Tým je práve dosť vysoko úrovňový systém, ktorý využíva externý program na extrakciu rysov a spracovanie dátových tokov.

Nakoniec bol však návrh implementovaný podľa špecifikácie. Pri implementácii bolo nutné zaobstarať objemný súbor dát, na ktorom sa následne trénoval model. Po iteráciách ladenia parametrov sa podarilo docieliť vysoko kvalitných klasifikácií, ktoré sa následne testovali a validovali. To prebiehalo vo viacerých fázach, najskôr na podobných dátach z rok staršieho datasetu od rovnakých autorov. Následne sa využilo iných datasetov v menšom počte na overenie výsledkov. Prvotný multiklasifikátor dosiahol presnosti len 98% primárne

kvôli inklúzii klasifikátoru na infiltráciu a laterálneho pohybu v sieti, ktorý dosahoval veľmi nízke presnosti. Po ich vylúčení sa síce znížil počet klasifikovateľných útokov, no model dosiahol výborných 99.7% Toto bolo vyhodnotené pomocou viacerých metrík. Celkom sa v prvej fáze spracovalo 16,2 miliónov záznamov. V nasledujúcej validačnej fáze sa využilo 2,8 milióna tokov.

V práci by som chcel pokračovať, najmä v oblasti rozšírenia spektra detekovateľných útokov a spresnenia detekcie zložitejších techník a taktík útočníkov. Taktiež považujem za vhodné pracovať na bližšej integrácií s robustným IDS/IPS systémom, ako napríklad Suricata, a umožniť obojsmernú komunikáciu a tým aj implementovať schopnosť automatizovanej odozvy systému na jednotlivé útoky. Ak by sa venoval dostatočný čas obohateniu trénovacieho súboru dát o podrobnejšie označenie fáz útokov typu infiltrácie tak je vysoká šanca na prudký rast schopnosti modelu tento typ útoku detegovať. Podobnou cestou je možné pokračovať v trénovaní klasifikátoru na väčšie množstvo typov útokov.

Literatúra

[1] *Direct-path attacks surge in 2022 making up half of all ddos attacks according to latest netscout ddos threat intelligence report.* [cit. 2023-05-14]. Dostupné z: https://www.netscout.com/press-releases/direct-path-attacks-surge-2022-making-half-all-ddos-attacks.

[2] *Yara - the pattern matching swiss knife for malware researchers.* [cit. 2023-1-30]. Dostupné z: https://virustotal.github.io/yara/.

[3] *Percentage of Web Pages Loaded by Firefox Using HTTPS.* 2019 [cit. 2023-2-1]. Dostupné z: https://letsencrypt.org/stats/#percent-pageloads.

[4] Ahmed, M., Naser Mahmood, A. a Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications.* 2016, zv. 60, s. 19–31, [cit. 2023-03-10]. DOI: https://doi.org/10.1016/j.jnca.2015.11.016. ISSN 1084-8045. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1084804515002891.

[5] Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A. a Anwar, A. TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access.* 2020, zv. 8, s. 165130–165150, [cit. 2023-05-14]. DOI: 10.1109/ACCESS.2020.3022862.

[6] Aouedi, O., Piamrat, K., Hamma, S. a Perera, J. K. M. Network Traffic Analysis using Machine Learning: an unsupervised approach to understand and slice your network. *Annals of Telecommunications - annales des télécommunications.* Nov 2021, [cit. 2023-01-29]. Dostupné z: https://hal.science/hal-03344361.

[7] Ashraf, J., Keshk, M., Moustafa, N., Abdel Basset, M., Khurshid, H. et al. IoTBoT-IDS: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities. *Sustainable Cities and Society.* 2021, zv. 72, s. 103041, [cit. 2023-05-14]. DOI: https://doi.org/10.1016/j.scs.2021.103041. ISSN 2210-6707. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2210670721003255.

[8] Bace, R. G., Mell, P. et al. Intrusion detection systems. US Department of Commerce, Technology Administration, National Institute of .... 2001, [cit. 2023-1-30]. Dostupné z: http://cs.uccs.edu/~cchow/pub/ids/NISTsp800-31.pdf.

[9] Benram, G. *XGBoost or TensorFlow?* Oct 2018 [cit. 2023-02-01]. Dostupné z: https://www.doit.com/xgboost-or-tensorflow/.

[10] Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N. a Hartog, F. T. H. d. ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets. *IEEE Internet of Things Journal*. 2022, zv. 9, č. 1, s. 485–496, [cit. 2023-05-14]. DOI: 10.1109/JIOT.2021.3085194.

[11] Chen, T. *Introduction to Boosted Trees*. 2014 [cit. 2023-03-12]. Dostupné z: https://web.njit.edu/~usman/courses/cs675_fall16/BoostedTree.pdf.

[12] Chen, T. a Guestrin, C. XGBoost. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug 2016 [cit. 2023-03-11]. DOI: 10.1145/2939672.2939785. Dostupné z: https://doi.org/10.1145%2F2939672.2939785.

[13] Dew, T. *Be Proactive against Threats in Your Network with Fidelis Decryption*. Jul 2020 [cit. 2023-1-30]. Dostupné z: https://fidelissecurity.com/threatgeek/network-security/be-proactive-against-threats-in-your-network-with-fidelis-decryption/.

[14] Franck, D. h.-d. *What does AUC stand for and what is it?* [Cross Validated]. URL:https://stats.stackexchange.com/q/132832 (version: 2018-12-19). Dostupné z: https://stats.stackexchange.com/q/132832.

[15] García Teodoro, P., Díaz Verdejo, J., Maciá Fernández, G. a Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*. Feb 2009, zv. 28, 1-2, s. 18–28, [cit. 2023-1-16]. DOI: 10.1016/j.cose.2008.08.003. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167404808000692.

[16] Hossin, M. a M.N, S. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*. Marec 2015, zv. 5, s. 01–11, [cit. 2023-03-13]. DOI: 10.5121/ijdkp.2015.5201.

[17] Kumar, A. *Accuracy, Precision, Recall & F1-Score - Python Examples*. Oct 2021 [cit. 2023-05-05]. Dostupné z: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/.

[18] Lin, J., Keogh, E., Fu, A. a Van Herle, H. Approximations to magic: finding unusual medical time series. In: *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*. 2005, s. 329–334 [cit. 2023-03-10]. DOI: 10.1109/CBMS.2005.34. Dostupné z: https://ieeexplore.ieee.org/document/1467711.

[19] Moustafa, N. *New Generations of Internet of Things Datasets for Cybersecurity Applications based Machine Learning: TON_IoT Datasets EXECUTIVE SUMMARY*. 2019 [cit. 2023-05-14]. Dostupné z: https://conference.eresearch.edu.au/wp-content/uploads/2019/08/2019_eResearch_59_New-Generations-of-Internet-of-Things-Datasets-for-Cybersecurity.pdf.

[20] Moustafa, N. *A Systemic IoT-Fog-Cloud Architecture for Big-Data Analytics and Cyber Security Systems: A Review of Fog Computing*. 2019 [cit. 2023-05-14].

[21] Moustafa, N. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society*. 2021, zv. 72, s. 102994, [cit. 2023-05-14]. DOI: https://doi.org/10.1016/j.scs.2021.102994. ISSN 2210-6707. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2210670721002808.

[22] Moustafa, N., Ahmed, M. a Ahmed, S. Data Analytics-Enabled Intrusion Detection: Evaluations of ToN_IoT Linux Datasets. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020, s. 727–735 [cit. 2023-05-14]. DOI: 10.1109/TrustCom50675.2020.00100.

[23] Moustafa, N., Keshky, M., Debiez, E. a Janicke, H. Federated TON_IoT Windows Datasets for Evaluating AI-Based Security Applications. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020, s. 848–855 [cit. 2023-05-14]. DOI: 10.1109/TrustCom50675.2020.00114.

[24] Narkhede, S. *Understanding AUC - ROC Curve*. Jun 2018 [cit. 2023-02-01]. Dostupné z: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[25] Nicolò, V. *Gradient boosting in tensorflow vs xgboost_2018*. Jun 2018 [cit. 2023-02-01]. Dostupné z: https://nicolovaligi.com/articles/gradient-boosting-tensorflow-xgboost/.

[26] Ryan, M. *Cage Match: XGBoost vs. Keras Deep Learning*. May 2020 [cit. 2023-05-06]. Dostupné z: https://towardsdatascience.com/cage-match-xgboost-vs-keras-deep-learning-a8bb2f69a9ab.

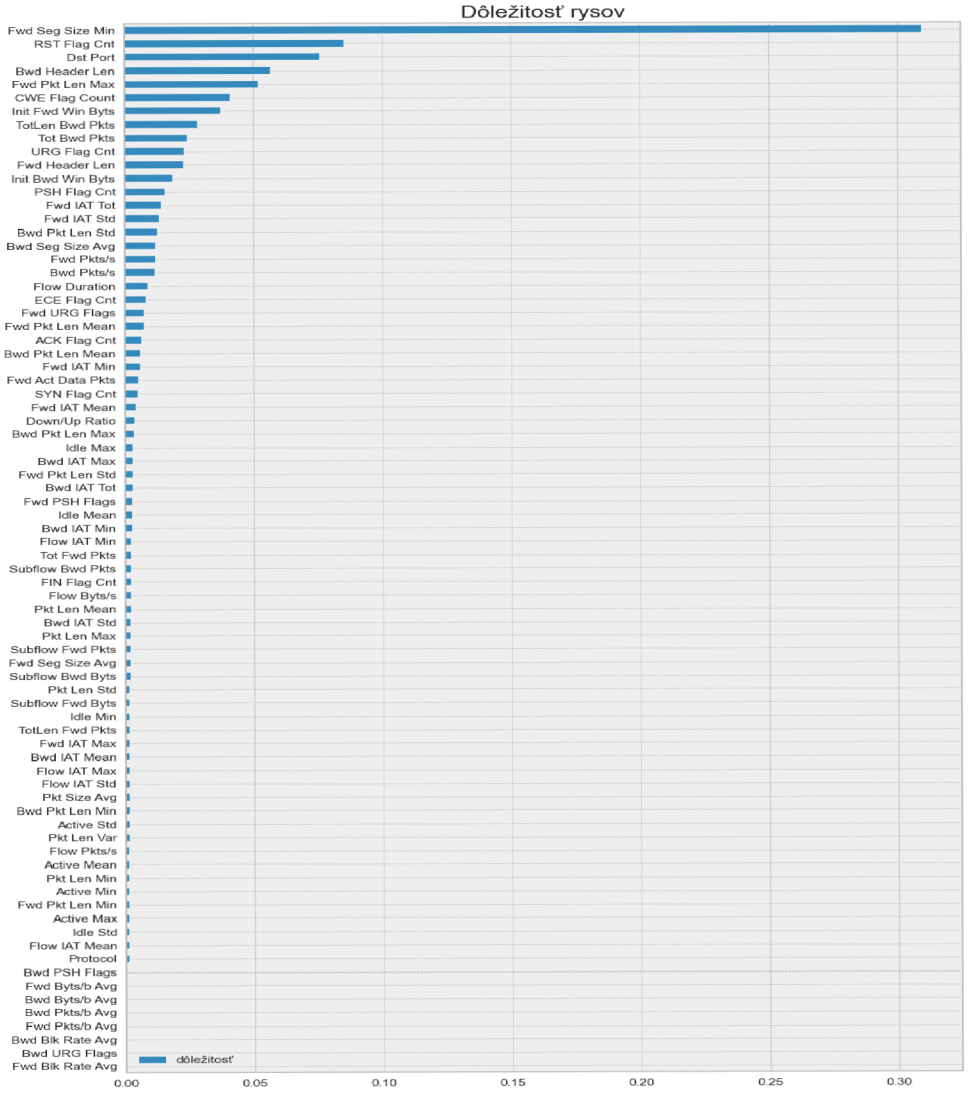
[27] Shafiq, M., Yu, X., Laghari, A., Yao, L., Karn, N. et al. Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms. In:. Október 2016, s. 2451–2455. 10.1109/CompComm.2016.7925139. Dostupné z: https://www.researchgate.net/publication/316900016_Network_Traffic_Classification_techniques_and_comparative_analysis_using_Machine_

[28] Sharafaldin, I., Habibi Lashkari, A. a Ghorbani, A. A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. 2018, [cit. 2023-02-18]. DOI: https://doi.org/10.5220/0006639801080116. Licencia vyžaduje: https://registry.opendata.aws/cse-cic-ids2018/. Dostupné z: http://www.scitepress.org/Papers/2018/66398/66398.pdf.

[29] Zhan, M., Li, Y., Yu, G., Li, B. a Wang, W. Detecting DNS over HTTPS based data exfiltration. *Computer Networks*. 2022, zv. 209, s. 108919, [cit. 2023-03-10]. DOI: https://doi.org/10.1016/j.comnet.2022.108919. ISSN 1389-1286. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1389128622001104.

Príloha A

Ukážka výstupu dôležitosti rysov



Obr. A.1: Ukážka výstupu dôležitosti rysov vytrénovaného modelu

Príloha B

Obsah pamäťového média

```
/
├── docs/ ..... Priečinok s prácou a jej zdrojovými súbormi
│   ├── latex/ ..... Priečinok obsahujúci zdrojové súbory LaTeX
│   └── xgawro00-Detekcia-anomalii.pdf ..... Súbor s touto prácou
├── src/ ..... Priečinok obsahujúci zdrojové súbory práce
│   ├── data/ ..... Malá vzorka dát na účely PoC
│   ├── model/ ..... Vytrénovaný model
│   └── scripts/ ..... Priečinok obsahujúca skripty a notebooky
├── requirements.txt ..... Zoznam závislostí jazyka Python
└── README.md ..... Návod na inštaláciu
```