VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

BEHAVIORÁLNÍ ANALÝZA SÍŤOVÝCH ÚTOKŮ TYPU DDOS

BEHAVIORAL ANALYSIS OF DDOS NETWORK ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ONDREJ KVASNICA**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Kvasnica Ondrej**
Program: Informační technologie
Název: **Behaviorální analýza síťových útoků typu DDoS**
Behavioral Analysis of DDoS Network Attacks
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s principy detekce anomálií v prostředí systémů počítačových sítí se zaměřením na útoky typu DDoS.
2. Analyzujte požadavky na systém umožňující analýzu metadat nižších vrstev modelu OSI za pomoci statistiky, případně vybraných metod umělé inteligence.
3. Navrhněte systém pro detekci a charakterizaci společných znaků útoků typu DDoS dle předchozího bodu.
4. Navržený systém implementujte dle instrukcí vedoucího práce.
5. Implementovaný systém ověřte na vhodně zvolených reálných, případně generovaných 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.

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. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 14. října 2021

Abstrakt

Tato bakalářská práce se zabývá detekcí anomálií v počítačových sítích pomocí metody umělé inteligence. Práce se soustředí zejména na detekci DDoS útoků na základě informací z nižších vrstev modelu OSI. Cílem je navrhnout a implementovat systém, který je schopný detekovat různé typy DDoS útoků a charakterizovat jejich společné znaky. Zvolené útoky jsou SYN záplavy, UDP záplavy a ICMP záplavy. Je obsažen popis a výběr důležitých rysů těchto útoků. Následně je navržen systém, který na základě síťových dat (organizovaných do toků) vyhodnotí, jestli data obsahují útok či nikoliv. Detekce útoků je implementována pomocí klasifikační metody XGBoost, která používá způsob učení s učitelem. Výsledný model je validován pomocí křížové validace a otestován na útocích vygenerované autorem práce.

Abstract

This bachelor thesis deals with anomaly detection in computer networks using artificial intelligence method. Main focus is on the detection of DDoS attacks based on the information from the lower layers of the OSI model. The target is to design and implement a system that is capable of detecting different types of DDoS attacks and characterize common features among them. Selected attacks are SYN flood, UDP flood and ICMP flood. Description and feature selection of the attacks is included. Furthermore, a system is designed that evaluates whether the network traffic (organized into flows) is a DDoS attack or not. Attacks are detected using the XGBoost method, which uses supervised learning. The final model is validated using cross-validation and tested on attacks generated by the author.

Klíčová slova

DoS, DDoS, detekce anomálií, XGBoost, SHAP hodnoty

Keywords

DoS, DDoS, anomaly detection, XGBoost, SHAP values

Citace

KVASNICA, Ondrej. *Behaviorální analýza síťových útoků typu DDoS*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Ing. Pavel Očenášek, Ph.D.

Behaviorální analýza síťových útoků typu DDoS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Mgr. Ing. Pavla Očenáška, Ph.D. Další informace mi poskytl pan Ing. Petr Chmelař. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondrej Kvasnica
10. května 2022

Poděkování

Rád bych poděkoval svému vedoucímu panu Mgr. Ing. Pavlu Očenáškovi, Ph.D. za vedení práce a pedagogickou pomoc. Dále bych rád poděkoval panu Ing. Petru Chmelařovi za jeho odbornou pomoc.

Obsah

Kapitola 1

Úvod

V současné době jsou počítačové sítě všudypřítomné. Vyskytují se tam, kde je potřeba rychlý přenos informací. Dnes se do počítačové sítě zapojují běžné domácí spotřebiče. Tento fakt způsobuje rychlý růst počtu počítačových sítí, a důraz na bezpečnost se zvyšuje. Všudypřítomnost počítačových sítí motivuje útočníky zmocnit se sítě, změnit předávané informace v ní, nebo ji vyřadit z provozu. Pokud se síť na určitou dobu nechová běžným způsobem, jedná se o síťovou anomálii. Detekování těchto anomálií je nezbytné k zajištění správného provozu sítě.

Jedna z anomálií je znemožnění služby (angl. Denial of Service) ve zkratce DoS. Jedná se o útok, který má za cíl vyřadit určitou službu z provozu. Útok je na Internetu velmi častý díky své jednoduchosti a efektivitě. Přichází-li z více zdrojů, nazýváme jej distribuované odepření služby (angl. Distributed Denial of Service), zkráceně DDoS. Typicky zaplaví určitou službu velkým počtem dotazů a služba poté nestíhá na všechny odpovědět. Kvůli své podstatě je jeho detekce výpočetně náročným procesem. Následky útoku mohou být kritické. Jen pár minutové vyřazení sítě z provozu může pro firmu znamenat velké finanční ztráty. Pro nemocnice, které bývají častou obětí, můžou být ztráty mnohem více katastrofální.

V oboru, kterým se zabývá téma bakalářské práce, je zapotřebí vyzkoušet nové způsoby detekce, aby byla detekce co nejpřesnější a za co nejkratší dobu. Bezpečnost na Internetu je považována za důležité téma, které nesmí být opomíjené. Zmíněné důvody byly motivací k tvorbě této bakalářské práce.

Cílem práce je navrhnout a implementovat systém, který bude schopný spolehlivě detekovat vybrané typy útoků DDoS na základě informací převážně z nižších vrstev modelu OSI. Dalším cílem je získání dodatečných znalostí o chování těchto útoků. Pro uskutečnění cílů je zvolena metoda umělé inteligence s názvem XGBoost. Nejenže je schopná naučit se vzory v datech a oddělit normální komunikaci od komunikace obsahující DDoS, ale je schopna také vysvětlit jednotlivé predikce.

V druhé kapitole je detailně probrán charakter útoků znemožnění služby. Jsou zde vysvětleny principy těchto útoků. Obsahuje také shrnutí jejich historie a popisuje, co útoky umožňuje. Na závěr kapitoly jsou vysvětleny odlišnosti v typech DDoS útoků – SYN záplavy, UDP záplavy, ICMP záplavy, Smurf útok.

Třetí kapitola se věnuje návrhu systému pro detekci a charakterizaci těchto DDoS útoků. Obsahuje výběr detekovaných typů útoku a výběr metody pro jejich detekci. Dále je proveden výběr rysů implementovaných v systému.

Pátá kapitola se zabývá implementací navrženého systému. Je zde popsáno, která data byla použita pro vytvoření modelu a jak byla získána. Následně je přiblíženo, jakým způso-

bem jsou zpracována, aby byla vhodná pro zvolenou metodu. Nakonec je v kapitole vytvořen model. Aby byl úspěšný, je provedena validace a ladění parametrů modelu.

Poslední kapitola se zabývá testováním vytvořeného modelu a analýzou chování DDoS a DoS útoků. Testování proběhlo ve dvou fázích – test na všech typech útoků, test smíšených dat normální komunikace a útoků. Analýza chování neboli charakterizace proběhla pomocí tzv. SHAP hodnot. Umožňují zjistit důležitost rysů a vysvětlit jednotlivé predikce modelu.

Kapitola 2

DoS a DDoS

2.1 DoS

Znemožnění služby (Denial of service) ve zkratce DoS je útok, který má za cíl znemožnit přístup právoplatným uživatelům k určité službě. Tento útok je typický tím, že přichází od jednoho hostitelského počítače, nebo jednoho síťového uzlu. Jeho důsledky mohou být např. zamezení příchodu všech zpráv k cílené destinaci, nebo narušení běžného chodu celé sítě. Je schopný ji úplně deaktivovat, nebo ji přetížit zprávami takovým způsobem, jež vede ke snížení výpočetního výkonu [36].

2.2 DDoS

Když DoS útok přichází od více hostitelských počítačů, nazýváme ho distribuované znemožnění služby (Distributed Denial of Service) ve zkratce DDoS. Tento útok je závažnější hrozba než DoS útok. Typicky velké množství hostitelských počítačů zaplaví servery, počítačové sítě, dokonce i systémy koncových uživatelů bezvýznamnou síťovou komunikací, což způsobí nepřístupnost počítačových systémů běžným uživatelům. Tento proces se nazývá zaplavení (angl. flooding) [36]. DDoS má jeden nepříjemný vedlejší účinek na internetovou síť. Často se stává, že vytvoří přetížení v síti po cestě od zdroje útoku k cíli. Toto způsobí, že internetové operace nejsou narušeny pouze oběti, ale i všem jiným uživatelům po cestě [38].

Historie DDoS

K Internetu je připojeno čím dál více lidí a většina z nich nemá dostatečné informace o bezpečnosti na Internetu. Špatně spravované stroje se dají snadněji napadnout, můžou být potenciálně využity k rozsáhlému DDoS útoku. Kvůli těmto vlastnostem a schopnosti generovat obrovské množství nežádoucího provozu je DDoS velká hrozba pro Internet. První velký útok se stal v srpnu roku 1999 na Minnesotskou universitu. Tento útok způsobil nedostupnost sítě na více jak dva dny. V roce 2000 byl DDoS útok použit na velké komerční stránky jako Yahoo a CNN. Útok vyřadil stránku Yahoo na 3 hodiny, webová stránka nezvládla v dané době nevídanou rychlost provozu $1GB/S$. V únoru roku 2001 bylo zjištěno více jak 12000 útoků mířených na více jak 5000 jedinečných počítačových sítí [35].

Nejdříve se myslelo, že tyto útoky jsou problematické pouze pro servery, jež komunikují synchronně. Všechny tyto servery byly vyřazeny z provozu, ale pokračující útoky na webové stránky ukázaly, že nezáleží na způsobu komunikace [35].

Co umožňuje DDoS útoky

Tato sekce pochází z průzkumu [39]. Při návrhu Internetu byla nejdůležitějším faktorem funkčnost, bezpečnost nebyla prioritou. Soubor protokolů TCP/IP předpokládá, že žádní komunikující nemají zlé úmysly. Není zde žádný bezpečnostní prvek, který by bránil před útokem hostitelského počítače na jiný hostitelský počítač. Pro příklad, protokol TCP předpokládá, že hostitelské počítače sníží rychlost přenosu paketů, když je detekována ztráta paketů v důsledku přetížení. Pokud ale daný počítač nezareaguje, tak se přilehlé spoje můžou lehce přetížit. Tyto chyby v návrhu protokolů můžou vést k potenciálním DDoS útokům. Další důvody proč je DDoS útok možný:

- Není důležité, jak moc je zabezpečený jeden konkretní hostitelský počítač. Pokud jsou jiné počítače špatně zabezpečené, tak i ty dobře zabezpečené se mohou stát obětí DDoS útoku. Hostitelské systémy, které jsou špatně zabezpečené, jsou potencionálním zdrojem pro útok.

- Obtížnost zjištění zdroje útoku. Naprostá většina Internetu běží na souborech protokolů TCP/IP. IP protokol nevytváří spojení. Na každé zastávce po cestě od zdroje k cíli se rozhoduje o dalším hostiteli, kterému bude paket adresován. Tato rozhodnutí, kam se paket pošle, se tvoří na základě cílové IP adresy. V IP protokolu je možné vytvořit paket, který má nesprávnou zdrojovou IP adresu. Tyto pakety lze potom použít pro DDoS útok. Tato technika se nazývá IP spoofing. Uživatelé v systému s dostatečným oprávněním můžou takto fabrikovat podvržené pakety. Např. v operačním systému Linux lze vytvářet tzv. raw sockety, které uživateli s dostatečným oprávněním umožní volit informace obsažené v hlavičce paketu. Díky tomuto je velmi obtížné detekovat opravdové zdroje útoku. Útočníci ale nemění pouze zdrojovou IP adresu, náhodně mění všechny informace v hlavičce kromě cílové IP adresy. Z toho důvodu se těžko filtrují podvržené pakety podle určitých vlastností. Existují DDoS útoky, které využívají podvrženou zdrojovou IP adresu určitého hostitelského počítače jako cíl útoku. Příkladem je tzv. Smurf útok.

- Omezené výpočetní zdroje hostitelů a počítačových sítí jsou důležitým faktorem pro existenci DDoS útoku. Snaží se využít omezeného výpočetního výkonu a paměti. Pokud by do těchto zdrojů bylo více investováno, tak by bylo obtížnější vytvořit dostatečný DDoS útok, který by dokázal tyto zdroje ohrozit. Ale útok nemusí využít všechny zdroje oběti a úplně ho vyřadit z provozu, útočníkům stačí, když využijí určitou část těchto zdrojů. Tím dosáhnou zhoršení kvality služby, která vede k finančním ztrátám.

- Existují tisíce a tisíce hostitelů a sítí, které jsou zranitelné. Tudíž je lehké získat velký počet hostitelů pro uskutečnění efektivního DDoS útoku.

- Je snadnější proniknout do systému než ho vytvořit. Každý hostitel včetně směrovačů na Internetu očekává pakety v určitém formátu a očekává určité chování komunikace. Toto může vést k nečekanému chování sítě. Např. hostitelské počítače a směrovače čekají a přidělují paměť pro fragmenty ještě neobdrženého datagramu. Pokud takový směrovač dostane špatně zformátované fragmenty datagramu, které indikují velkou velikost daného datagramu, tak směrovač bude pokračovat v alokování paměti, dokud mu nedojde paměť. Pokud dojde směrovači paměť, tak nebude moci zpracovat další datagram.

Získání sítě určenou pro útok

Aby útočník mohl provést DDoS útok, musí se zmocnit velkého počtu počítačů. Tyto počítače nazýváme zombie. Jedná se o počítač, na kterém běží program, jež útočí na jiné počítače. Zmocnění se velkého počtu počítačů se děje v těchto krocích:

1. Je zapotřebí software, který dokáže vyvolat DDoS útok. Musí být spustitelný na velkém počtu počítačů a musí být na daném počítači utajený. Je nutné, aby software mohl komunikovat s útočníkem, nebo aby měl zabudovaný časovací systém, který útok ve vybraném čase spustí.

2. Velký počet systémů musí být zranitelný. Útočník si musí být vědom chyby v těchto systémech a využít ji na instalaci zombie software. Častá chyba uživatelů je nedostatečné aktualizování software.

3. Strategie lokalizování zranitelných počítačů. Tento proces nazýváme skenování.

Útočník nejprve procesem skenování zjistí, které počítače jsou zranitelné a infikuje je zombie softwarem. Tento zombie software zase zjistí, které další počítače jsou zranitelné a ty infikuje. Takto se postupuje až do té doby, než bude mít útočník dostatečný počet zombie ve své síti na DDoS útok. Existují různé strategie skenování:

- Při náhodném skenování infikovaný počítač náhodně vybere IP adresu ze stejného adresového prostoru. Tato technika produkuje velké množství provozu na Internetu, může dokonce způsobit narušení správné komunikace i před počátkem DDoS útoku.

- Útočník sestaví tzv. hit-list, jedná se o seznam potenciálně zranitelných počítačů. Tento proces může trvat dlouhou dobu, aby útočník zamezil detekci. Dále každý infikovaný počítač dostane část daného seznamu, který bude skenovat. Tato technika je hůře detekovatelná, protože skenování na jednotlivých počítačích probíhá krátkou dobu.

- Topologická strategie využívá informace na daném infikovaném počítači. Tyto informace použije k nalezení dalších zranitelných počítačů.

- Tato metoda využívá IP adresy z lokální podsítě. Pokud je počítač infikován za bránou firewall, pak zombie software hledá potencionální cíle ve své vlastní podsíti.

Blíže popsáno v knize [36].

Typy DDoS útoků

Podle práce [35] DDoS útoky dělíme zejména na dvě skupiny: záplavové útoky a logické útoky.

Při záplavových útocích útočník posílá oběti dotazové pakety v určité frekvenci. Kvůli mechanismu zvaném Congestion control legitimní uživatelé sníží počet posílaných paketů na server. Jakmile je celkový počet požadavků na server větší než rychlost odpovídání služby, tak jsou dotazové pakety uloženy do vyrovnávací paměti serveru. Legitimních požadavků na server bude čím dál méně do té doby, než je celá šířka pásma využita pouze na odpovídání útočným paketům.

Podstata logických útoků je zaslání malého množství poškozených paketů, které využijí již zjištěné chyby v software. Tomuto typu útoku lze poměrně jednoduše zabránit včasnou

instalací aktualizací, které zabrání těmto chybám v software, nebo přidáním speciálních pravidel pro bránu firewall, která vyfiltruje poškozené pakety ještě před interakcí se systémem.

SYN záplavy

Tento záplavový útok využívá vlastností mechanismu, který uzavírá spojení v protokolu TCP, zvaného třícestný handshake (anglicky three-way handshake). Pro pochopení SYN záplavy je nutné porozumět tomuto mechanismu. Zahájení TCP spojení probíhá v těchto krocích [28]:

1. Klientova strana pošle na server speciální TCP paket. Ten neobsahuje žádná aplikační data, ale je v něm nastavený SYN bit na 1. Z tohoto důvodu tento speciální paket nazýváme SYN paket. Je zapotřebí, aby klient náhodně vybral počáteční pořadové číslo (anglicky sequence number) paketu a vložil ho do správného pole TCP hlavičky. Takto nastavený paket se zapouzdří a pošle na server.

2. Až dorazí poslaný IP datagram na server, tak server vyjme SYN paket z IP datagramu a pro nově vytvořené TCP spojení alokuje paměť. Dále server pošle paket klientovi, který potvrzuje, že proběhlo vytvoření spojení. Tento paket také neobsahuje žádná aplikační data, je v něm nastavený SYN bit na 1 a pole s názvem potvrzovací číslo (anglicky acknowledgment number) je nastaveno na hodnotu pořadového čísla přijatého SYN paketu inkrementovanou jedničkou. Server také musí vygenerovat a nastavit nové pořadové číslo tomuto paketu. Tento potvrzovací paket nazýváme SYNACK paket.

3. Jakmile klient přijme SYNACK paket, tak alokuje paměť pro toto spojení. Klient následně pošle poslední paket, který potvrzuje SYNACK paket od serveru. Taktéž neobsahuje žádné aplikační data. Potvrzovací číslo nastaví na hodnotu pořadového čísla SYNACK paketu inkrementovanou o jedničku. SYN bit je nastaven na 0, protože spojení už bylo uzavřeno. Nyní je možné posílat data serveru.

Jak můžeme vidět z postupu třícestného handshake, tak server alokuje zdroje po každém přijetí SYN paketu. Tento fakt způsobuje náchylnost k útokům typu vyčerpání zdrojů. Záplava SYN spočívá v zaslání velkého počtu žádostí o synchronizaci (SYN paket) na server, což způsobí rozvrat v paměti serveru. Tyto SYN pakety se ukládají do fronty obsluhy, kde čekají na příchozí SYNACK paket od klienta. Pro každý přijatý SYN paket server pošle SYNACK zpět klientovi a vyhradí místo v již popsané frontě. Spojení v tomto stavu nazýváme polootevřené (anglicky halfopen). Řazení těchto spojení do fronty spotřebovává paměť serveru, pokud bude žádostí o spojení velký počet, tak některým z nich nebude vyhověno. Útočník může sestrojit takové SYN pakety, které budou mít podvrženou IP adresu. To způsobí, že server nedostane potvrzení SYNACK od klienta s podvrženou IP adresou a fronta nevyřízených požadavků bude narůstat.

TCP obsahuje mechanismus, který se vypořádává s těmito nepotvrzenými pakety ve frontě, pro které nepřišla odpověď. Využívá tzv. timeouty (časové prodlevy), hodnota značí, po jaký časový úsek se má nechat nepřijatá žádost o spojení ve frontě. Typická hodnota pro tento timeout je 3 minuty. Během této doby lze poslat obrovské množství předstíraných požadavků a fronta s konečnou délkou může být naplněna podvrženými pakety a legitimním požadavkům nebude vyhověno [10].

UDP záplavy

Tento typ záplavového útoku je stejně jako záplavový útok SYN velmi častý. Útočník posílá UDP paket na náhodný port systému oběti. Až tento paket dorazí na systém oběti, tak systém bude předpokládat, že na portu obsaženém v UDP hlavičce běží aplikace. Po zjištění, že tam žádná aplikace neběží, se pošle ICMP zpráva (Destination unreachable) zdrojové adrese uvedené v paketu. Tato zdrojová adresa je podvržená útočníkem. Pokud bude těchto UDP paketů poslán velký počet na porty oběti způsobem, jakým bylo zmíněno, tak systém oběti může havarovat [16].

ICMP záplavy

ICMP záplavy je další typ záplavového útoku, který typicky využívá infikovanou síť k zaslání velkého počtu ICMP zpráv na určitý server. ICMP zprávy jsou obsaženy v klasických IP datagramech a slouží k diagnostickým účelům v dané síti. ICMP protokol obsahuje zprávy ECHO REQUEST a ECHO REPLY, které nám umožňují např. zjistit dostupnost určitého síťového uzlu. Cílem útoku je zahltit šířku pásma serveru a tím znemožnit službu legitimním uživatelům serveru. Každé zařízení v infikované síti pošle ICMP ECHO REPLY zprávu oběti. Tyto zprávy vyžadují odpověď. Oběť musí poslat odpověď na každou zprávu zvlášť zpět do Internetu. To může zapříčinit přetížení sítě, ve které se oběť nachází, což může vést k potížím s připojením [17], [9].

Smurf útok

Tento útok neposílá ICMP zprávy přímo od zdroje k cíli, ale místo toho pošle ICMP ECHO REQUEST požadavky na broadcast adresu špatně nakonfigurované sítě. Každé zařízení v dané síti, které ICMP zprávu přijme, pošle ICMP odpověď na zdrojovou adresu ICMP požadavku. Podvržením zdrojové IP adresy ICMP požadavku na IP adresu oběti se zajistí, že všechny odpovědi budou směřovány k ní. Těmto typům útoků, které na jednu zprávu pošlou mnoho odpovědí, říkáme zesilující typy útoků. V tomto útoku není zasažena pouze oběť, ale také zprostředkovávající zařízení [9], [8].

Kapitola 3

Detekce anomálií

Ještě než se práce zaměří na problém detekce DDoS útoků, tak je zapotřebí zjistit, jaká existují současná řešení v detekci obecných podezřelých aktivit v počítačové síti. Tímto problémem se zabývá obranný systém s názvem IDS (systém pro odhalení průniku). V této kapitole je probráno, jakým způsobem IDS odhaluje podezřelé aktivity a jaké různé typy IDS existují. Tyto podezřelé aktivity můžeme také nazvat síťové anomálie. Pro ně je typické, že se liší od normálního chování v síti. DDoS útoky spadají do skupiny podezřelých aktivit neboli anomálií, co IDS musí umět detekovat. Dále jsou anomálie rozděleny do skupin podle své charakteristiky. Nakonec se kapitola zabývá nejdůležitější částí této práce a to detekci DDoS.

3.1 IDS

IDS (angl. Intrusion Detection System, česky systém pro odhalení průniku) je obranný systém, který pozoruje a analyzuje síťový provoz. Jakmile narazí na potenciálně podezřelou aktivitu, tak ji ohlásí. Systém, který filtruje tyto podezřelé aktivity, se nazyvá IPS (angl. Intrusion Prevention System, česky systém prevence průniku). Práce se kvůli zaměření zabývá zejména systémem IDS. Odhalení podezřelých aktivit neboli průniků vychází z předpokladu, že chování útočníka se liší od chování legitimního uživatele takovým způsobem, jež lze kvantifikovat. IDS může být použit k detekci různých typů útoků včetně mapování sítě (nástroj nmap), skenování portů, DoS útoky, červy a viry, útoky na zranitelnost operačního systému a aplikací. Určitá organizace typicky zavede jeden nebo více IDS senzorů do své počítačové sítě. Často se zavádí pouze jeden IDS senzor poblíž směrovače [28], [36]. Hlavní cíle IDS podle práce [5] jsou následující:

1. Detekovat širokou škálu průniků. IDS by se měl zaměřit jak na externí průniky, tak i na ty interní. Dále by měl být schopný detekovat dříve známé, ale i neznámé útoky.

2. Detekovat průniky včas. Včas nemusí přímo znamenat v reálném čase. Často stačí, když jsou průniky detekovány za krátkou dobu od reálného průniku.

3. Prezentovat výsledky analýzy jednoduše, aby byly lehce srozumitelné. Naneštěstí průniky a podezřelé aktivity nejsou tak jednoznačné, IDS musí jasně prezentovat tato komplexní data dohlížejícímu technikovi.

4. Dosáhnout vysoké přesnosti v rozlišování jednotlivých průniků. V tomto případě existují dva typy chyb. Falešně pozitivní chyba, která vzniká, když IDS ohlásí, že probíhá

útok, ale žádný útok se nekoná. Falešně negativní chyba nastane, když IDS neohlásí právě probíhající útok. Tato chyba je horší z toho důvodu, že hlavním smyslem IDS je ohlašovat útoky/průniky. Cílem v této části je minimalizovat oba dva typy chyb.

IDS systémy jsou obecně klasifikovány podle principu do dvou skupin – detekce na základě signatur (angl. signature-based systems) a detekce anomálií.

3.2 Detekce na základě signatur

IDS systémy provádějící detekci na základě signatur potřebují předešlou znalost potenciálních zranitelností systému, které můžou útočníci využít pro své účely. Tento systém si musí uchovávat rozsáhlou databázi signatur již známých útoků. Každá signatura je množina pravidel týkající se dané podezřelé aktivity. Když síťová data procházejí systémem, tak se na ně tato pravidla aplikují a rozhoduje se, zda pravidla odpovídají. Pokud ano, tak systém upozorní na tento útok. Upozornění může vypadat jakkoliv, může se jednat o zaslání e-mail zprávy síťovému administrátorovi, nebo systému spravující danou síť. Signatura může být seznam charakteristik jednoho paketu (např. číslo zdrojového portu, typ protokolu nebo určitá posloupnost bitů v datech), nebo se může jednat o sérii paketů. Tyto signatury jsou většinou tvořeny zkušenými síťovými specialisty, kteří se podílí na zkoumání známých útoků. Tyto systémy neumí detekovat útoky, které nejsou obsaženy v databázi signatur, neporadí si také ani s jinými variantami známých útoků. Další nevýhodou je, že tato pravidla můžou sedět na paket nebo sérii paketů, které nejsou útokem [28], [5].

3.3 Detekce anomálií

Systémy, co používají princip založený na detekci anomálií, vytváří svůj profil síťového provozu na základě normálního chodu sítě. To znamená, že IDS posbírá kolekci síťových dat, která odpovídají chování legitimních uživatelů, a na základě toho vytvoří profil. Nově příchozí síťová data se statisticky porovnají s tímto profilem a IDS dospěje k rozhodnutí, jestli se jedná o normální chování či nikoliv. Hledají se tedy posloupnosti paketů, které jsou statisticky neobvyklé v porovnání s normálem. Může se jednat např. o vyšší procento ICMP paketů než bývá v normálním provozu. Velkou výhodou detekcí anomálií je možnost odhalení útoku, který IDS ještě nespatřil. Nespoléhají se tedy na znalosti všech možných útoků. Na druhou stranu je poměrně náročné vytvořit takový systém, který dokáže od sebe rozlišit normální a anomální chování. Z toho důvodu většina IDS systémů používá detekci na základě signatur [28], [36].

Hybridní systémy

Hybridní systémy kombinují IDS systémy, které využívají techniku detekce na základě signatur a detekce anomálií. Využívají se k tomu, aby zvýšily přesnost v detekci známých útoků a snížily míru falešně pozitivních chyb. Autor článku [6] tvrdí, že nenalezl mnoho systémů, které byly založené čistě na detekci anomálií. Většina systémů byla založená na hybridní technice.

3.4 Rozdělení anomálií

Než se přejde na rozdělení anomálií, tak je zapotřebí jasně definovat, co to je anomálie. V práci se už pojem anomálie objevil, byl spojen s podezřelou aktivitou v počítačové síti. Anomálie jsou definované jako vzory v datech, které neodpovídají dobře definované charakteristice normálních vzorů [2]. Může se jednat například o síťový útok, nebo neautorizovaný přístup. Na základě práce [2] se anomálie dělí podle své podstaty do třech skupin – bodová anomálie, kontextová anomálie a kolektivní anomálie. Níže v sekci jsou jednotlivé skupiny popsány.

Bodová anomálie

Když se konkrétní instance dat zásadně liší od normálního vzoru datového souboru, tak mluvíme o bodové anomálii. Pro příklad, když normální člověk spotřebuje 5 litrů benzínu za den, ale najednou se jeho spotřeba zvýší na 50 litrů denně, tak se jedná o bodovou anomálii [2]. V kontextu počítačových sítí řekněme, že normální nasledovaná hodnota TTL je 64, ale určitý paket má hodnotu 255. Zase se jedná o bodovou anomálii.

Kontextová anomálie

Kontextová anomálie vzniká, když se instance dat chová anomálně v určitém kontextu. Pro příklad vyšší útrata na kreditní kartě během vánočních svátků se může jevit jako anomální. Jenže když se dá tato událost do kontextu, tak se zjistí, že vyšší útrata během vánočních svátků je normální. Na druhou stranu pokud je stejně vysoká útrata na kreditní kartě zjištěna v normálním období, tak se může jednat o kontextovou anomálii. Dalším příkladem kontextové anomálie z oboru počítačových sítí je skenování sítě, kdy se útočník snaží získat informace o síti, které pak využije k zlomyslným účelům [2]. Jenže získávání informace o síti nemusí byt vždy anomální, záleží to na kontextu. Z toho důvodu je skenování sítě kontextovou anomálií.

Kolektivní anomálie

Když se soubor podobných instancí dat chová anomálně v porovnání se zbytkem datového souboru, tak je tento soubor instancí dat nazýván kolektivní anomálie. Například když se zkoumá výstup testu EKG u lidí, tak nízké hodnoty po delší dobu můžou znamenat určitý nález (tedy anomálii). Jenže ojedinělá nízká hodnota je normální. V případě DoS útoku se posílají úplně normální pakety, které nejsou nijak zvláštní, ale jakmile je těchto paketů více, tak se může jednat o DoS útok. Tedy útoky DoS a DDoS jsou kolektivní anomálie [2].

3.5 Detekce DoS/DDoS

Prevence DoS/DDoS útoku pomocí filtrování vstupu může pomoci v redukování některých typů těchto útoků např. útoky založené na spoofing IP adresy. Jenomže tyto preventivní techniky často nelze použít pro mitigaci DoS/DDoS útoků, často jsou spíše zapotřebí reaktivní techniky, kde už je potřeba konkrétní DoS/DDoS útoky detekovat. Jak je již známo, tak DoS/DDoS útok je pouze jeden typ útoku, na které IDS systémy musí reagovat. Záplavové útoky je náročnější detekovat než ty logické, protože jsou posílány normální pakety pouze ve velkém množství. Od normálních paketů se prakticky neliší. Tento fakt způsobuje,

že při detekci vzniknou falešně pozitivní chyby. Dobrá detekční technika by měla reagovat rychle a měla by mít malou míru falešně pozitivních chyb.

Stejně jako obecné IDS systémy i detekční techniky DoS/DDoS útoků jsou rozdělené do dvou skupin – detekce na základě signatur a detekce anomálií. Jsou definovány stejně jako u IDS. Detekce na základě signatur porovnává vzory známých DoS/DDoS útoků s příchozí komunikací a detekce anomálií porovnává komunikaci se svým profilem normálního chování [3].

Kapitola 4

Návrh a analýza požadavků

Tato kapitola se zabývá návrhem systému, který určuje, jestli komunikace obsahuje DDoS útok. V návrhu jsou zohledněny požadavky, které musí systém splňovat, aby byl úspěšný. Pro cíle této práce není nutné se zabývat zpracováváním jednotlivých paketů, ale spíše se zaměřit na detekci skupin paketů. Z tohoto důvodu systém předpokládá síťovou komunikaci již organizovanou do síťových toků. Výstupem systému je klasifikace těchto toků, která říká, jestli se jedná o DDoS útok, nebo nikoliv. To znamená, že k jednotlivým tokům bude přiřazená hodnota, která značí jestli systém vyhodnotil, že se jedná o útok, nebo normální provoz.

Při návrhu takového systému je zapotřebí zvážit mnoho aspektů. V sekci 4.1 je vybrána metoda, která je použita v systému pro detekci útoků. Sekce obsahuje popis zvažovaných metod a požadavky, které by měla metoda splňovat. Dále v sekci 4.2 jsou zváženy různé typy DDoS/DoS útoků, které systém musí detekovat. Následně v sekci 4.3 jsou vybrány jednotlivé rysy/vlastnosti/atributy, které jsou použité ve vybrané detekční metodě. Tyto parametry jsou důležité v odlišení normální komunikace od útoků.

4.1 Výběr metody pro detekci

Při výběru detekční metody pro útoky DDoS je zapotřebí zvážit jaké metody se hodí pro tento úkol. V přehledu [2] metod pro detekování síťových anomálií jsou jednotlivé metody porovnány. Je tam i stanoveno, jaké typy anomálií preferují dané metody. Pro detekci DDoS útoků jsou vhodné podle autorů klasifikační a shlukovací metody. Na základě tohoto přehledu je v návrhu zvážena Metoda podpůrných vektorů (anglicky Support Vector Machines – SVM) a shlukovací algoritmus k-means.

Metoda podpůrných vektorů je blíže popsána níže, ale podstatné je, že tento algoritmus ukázal poměrně vysokou přesnost a dokázal správně označit různé typy útoků, ale může být neefektivní při velkém počtu dat a také může být obtížné interpretovat konečný model [11]. Nemožnost interpretovat rozhodnutí neuronových sítí zároveň diskvalifikuje neuronové sítě.

K-means je shlukovací algoritmus, který používá učení bez učitele. Tento algoritmus se prokázal jako účinný v detekci DDoS útoků, ale má v klasifikaci poměrně vysoké procento falešně pozitivních chyb [15].

XGBoost (Extreme Gradient Boosting) je klasifikační algoritmus strojového učení, který používá pro implementaci rozhodovací stromy. XGBoost používá učení s učitelem, tedy potřebuje předem označená data. Pokud určitý problém vyžaduje strukturovaná data, což

v případě síťových toků platí, tak je velmi vhodné použití rozhodovacích stromů. Tento algoritmus často vyhrává soutěže na různé složité problémy, zejména na stránce Kaggle.com. XGBoost získává na popularitě díky svému výkonu a škálovatelnosti [14]. Je blíže popsán v aktuální sekci níže. V práci Chena a ostatních byl XGBoost použit na klasifikování toho, jestli síťový tok obsahuje DDoS útok, či nikoliv. Při srovnání bylo poznamenáno, že algoritmy SVM, Náhodný les (angl. Random Forest), GBDT (Gradient Boosted Decision trees) docílily přesnosti 97.19%, 96.33%, a 97.69%. XGBoost dosáhl nejvyšší přesnosti v klasifikaci až 98.53% a celkově dosáhl nejlepšího výsledku. Nicméně Sanjeetha a ostatní ukázali, že Gaussian Naive Bayes je horší v klasifikaci pouze o 1% (97.57% místo 98.62%). Algoritmus je však řádově (až 100 krát) rychlejší než XGBoost jak v detekci, tak i v trénování a dokáže se stejným modelem obojí - klasifikaci i detekci anomálií [32]. I tento algoritmus byl zvážen v návrhu, ale nakonec byl vybrán XGBoost kvůli své přesnosti a možnosti získání dodatečných informací o výsledném modelu, které můžou pomoci charakterizovat útoky.

XGBoost

Jak již bylo zmíněno, tak XGBoost využívá rozhodovací stromy. Rozhodovací strom bere v úvahu atributy (vlastnosti nebo rysy) datového souboru. Tyto atributy vystupují v rozhodovacím stromu jako vnitřní uzly. Každý vnitřní uzel dělí data na základě podmínky do větví, to stejné platí i pro kořenový uzel. Uzel, ze kterého už nevedou žádné další větve, se nazývá list. Dosažení tohoto listu znamená, že se rozhodovací strom dopracoval ke klasifikaci určitého objektu.

Algoritmus XGBoost je popsán na základě práce Chena a spoluautorů [7] a práce dvou autorů Gouveia a Correia [14].

Koncept trénování modelu ve strojovém učení vede k matematickému procesu, kde se využívají trénovací data x_i obsahující atributy vstupních dat na predikci cílové hodnoty y_i. Aby mohl model správně predikovat cílové hodnoty, tak je zapotřebí určit parametry modelu. Tyto parametry modelu se získávají na základě trénovacích dat. Trénování modelu znamená hledání parametrů modelu, které nejvíce odpovídají x_i a y_i. Značíme je θ. V případě XGBoost se při trénování sestavují rozhodovací stromy a hledají se jejich parametry takovým způsobem, aby model byl co nejpřesnější. Funkce, která optimalizuje model a posuzuje jeho přesnost, se nazývá objektivní funkce (angl. objective function). Je obecně definována jako:

$$F_{Obj} = L(\theta) + \Omega(\theta), \text{ kde } L(\theta) = l(\hat{y}_i, y_i) \tag{4.1}$$

Objektivní funkce 4.1 se skládá ze dvou částí: $L(\theta)$ a $\Omega(\theta)$. $L(\theta)$ je ztrátová funkce, která měří rozdíl mezi predikcí $\hat{y}_i$ a reálnou cílovou hodnotou y_i. Hlavním účelem této funkce je vyjadřovat přesnost modelu v predikci dat, které jsou v trénovacím souboru. Často používaná ztrátová funkce je např. střední kvadratická chyba definována jako $l(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$. $\Omega(\theta)$ je výraz, který penalizuje komplexní modely. To znamená, že pokud je vytvořený strom v daném kroku moc komplexní, tak má objektivní funkce větší hodnotu. XGBoost model, který je upravován objektivní funkcí, lze zapsat jako:

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

Kde K je počet stromů a f je definice funkce reprezentující strukturu stromu a váhu jeho listů, tato funkce po dosazení x_i vrací hodnotu reprezentující predikci jednoho stromu. F je množina všech možných klasifikačních a regresních stromů. Pro tento konkrétní model, je objektivní funkce definována jako:

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

Funkce 4.3 sčítá rozdíl mezi predikcemi každého stromu a reálnými hodnotami z trénovacích dat, dále připočte ještě komplexitu každého stromu. Při sestavování stromů se snaží algoritmus docílit co nejmenší hodnoty F_{Obj}. I když rozdíl mezi predikcí a reálnou hodnotou může být velmi malý, tak hodnota ohodnocení modelu objektivní funkcí nemusí být nízká. To by znamenalo, že model sice správně predikuje na základě trénovacích dat, ale stromy jsou moc složité. To může vést ke špatnému výsledku při klasifikaci jiných než trénovacích dat. V takto definovaném modelu by se stromy učily nebo sestavovaly paralelně, což způsobuje praktické problémy. Z tohoto důvodu XGBoost využívá aditivní strategii. Tedy v každém kroku t se iterativně přičítají jednotlivé stromy. Tento přetvořený aditivní model 4.4 následně vypadá jako:

$$\hat{y_i}^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y_i}^{(t-1)} + f_t(x_i) \tag{4.4}$$

Objektivní funkce 4.5 se změní na:

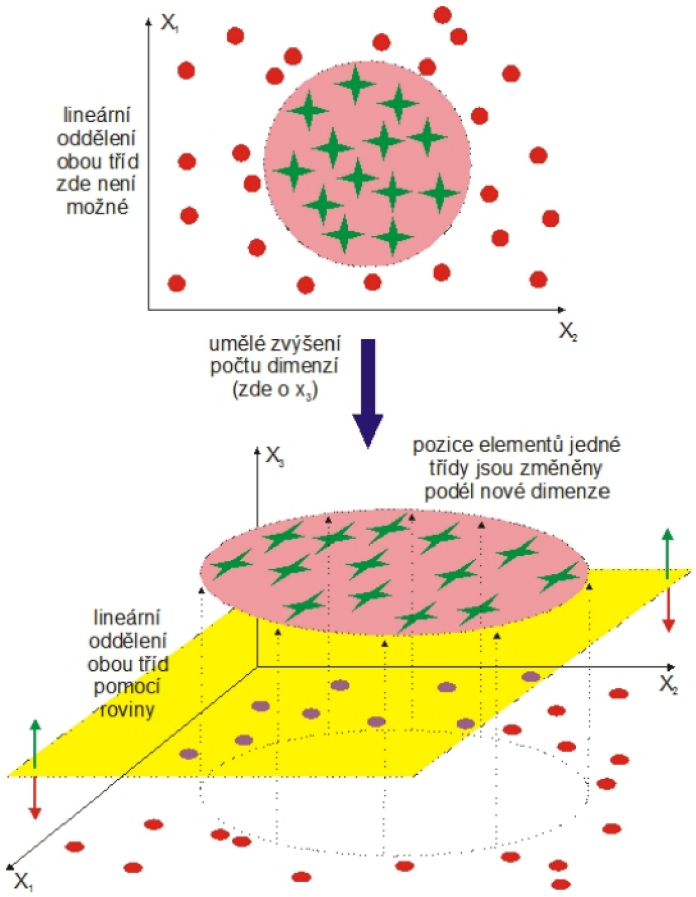
$$F_{Obj}^{(t)}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega f_t + constant \tag{4.5}$$

V každém kroku se připočte nový strom, který optimalizuje objektivní funkci.

Metoda podpůrných vektorů

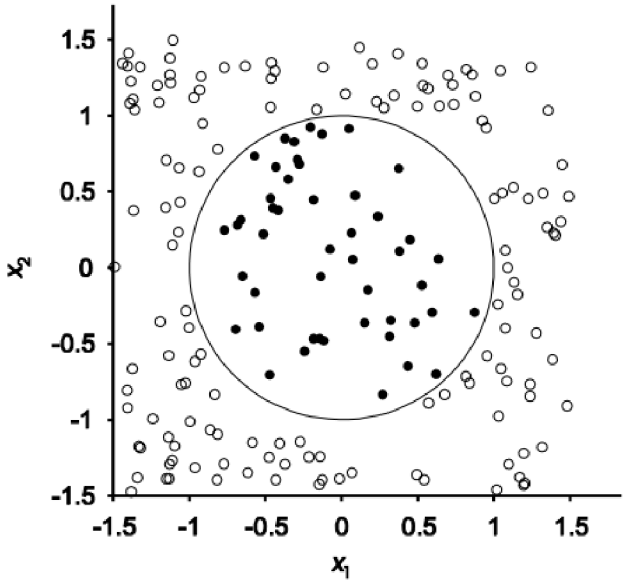
V návrhu byla zvážena klasifikační metoda podpůrných vektorů (anglicky Support Vector Machines – SVM), která používá učení s učitelem. Metoda vyžaduje mít předem označená trénovací data, která říkají, jestli se jedná o útok, či nikoliv.

Tato metoda je jednou z metod, které tvoří kategorii zvanou jádrové algoritmy (anglicky kernel machines). Popis této metody je převzat z dokumentu [40], kde je blíže popsán. Tyto metody využívají výhody poskytované efektivními algoritmy pro nalezení lineární hranice a zároveň jsou schopny reprezentovat vysoce složité nelineární funkce. Jedním ze základních principů je převod daného původního vstupního prostoru do jiného, vícedimensionálního, kde již lze od sebe oddělit třídy lineárně. Tento způsob je znázorněn na obrázku 4.1, kde je ukázáno, jak pro prostor se dvěma dimenzemi nelze třídy rozdělit lineárně (jedná se o kružnici), ale po přidání 3. dimenze to už možné je.



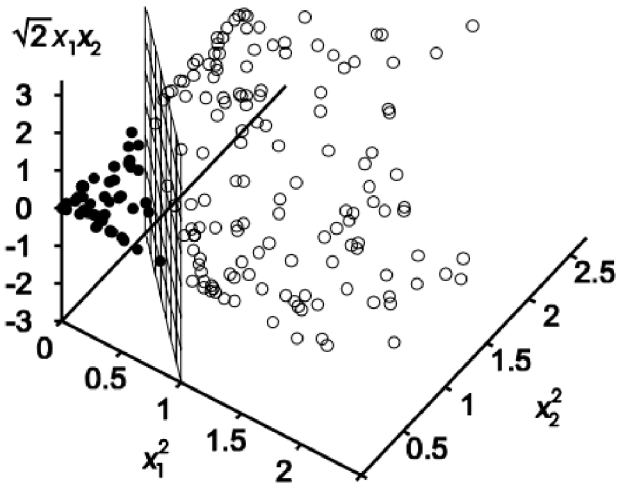
Obrázek 4.1: Způsob oddělení tříd přidáním 3. dimenze. Převzato z dokumentu [40].

V případě příkladu výše máme dvourozměrný vstupní prostor definovaný atributy $\mathbf{x} = (x_1, x_2)$. Testovací hodnoty $y = 1$ jsou uvnitř kruhu a $y = -1$ jsou mimo kruh. Pro kruh nejde vytvořit lineární hranici, která by oddělovala dané trénovací příklady, hranice je reprezentovaná funkcí $x_1^2 + x_2^2 \leq 1$. Tento prostor je znázorněný detailněji na obrázku 4.2.



Obrázek 4.2: Původní dvojrozměrný prostor s kruhovou hranicí. Převzato z dokumentu [40].

Tento původní vektor $\mathbf{x}$ můžeme modifikovat přidáním nového atributu a změněním původních, tento nový 3-D vektor značíme $F(\mathbf{x})$. Jednotlivé atributy definujeme funkcemi: $f_1 = x_1^2, f_2 = x_2^2, f_3 = \sqrt{2}\ x_1 x_2$.



Obrázek 4.3: Mapovaná vstupní data do trojrozměrného prostoru. Převzato z dokumentu [40].

Nový prostor lze vidět na obrázku 4.3. Po transformaci je z obrázku jasné, že jdou původní třídy lineárně separovat. Tento jev platí obecně. Pro N datových bodů je možno vždy (kromě některých speciálních případů) lineárně oddělit v prostoru s $N-1$ nebo více dimenzemi. Hrozí ale nebezpečí tzv. přetrénování, stane se pokud $d \approx N$, kde d je počet dimenzí prostoru, v tomto případě dojde ke ztrátě obecnosti klasifikátoru.

Důležitou částí metody podpůrných vektorů je volba optimálního lineárního oddělovače. Lineární oddělovač nazveme optimálním, pokud poskytuje co nejširší pásmo mezi ním a pozitivními příklady na jedné straně a negativními příklady na straně druhé. Musí také podporovat robustnost klasifikace. Tyto optimální lineární oddělovače se hledají pomocí metody kvadratického programování. Problém hledání optimálního oddělovače spočívá v hledání hodnot parametrů α_i, které maximalizují výraz 4.6:

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{4.6}$$

s omezením $\alpha_i \geq 0$ a $\sum_i \alpha_i y_i$. Po vypočítání optimálních parametrů α_i je oddělovač dán rovnicí 4.7:

$$h(\mathbf{x}) = sign\left(\sum_i \alpha_i y_i (\mathbf{x} \cdot \mathbf{x}_i)\right) \tag{4.7}$$

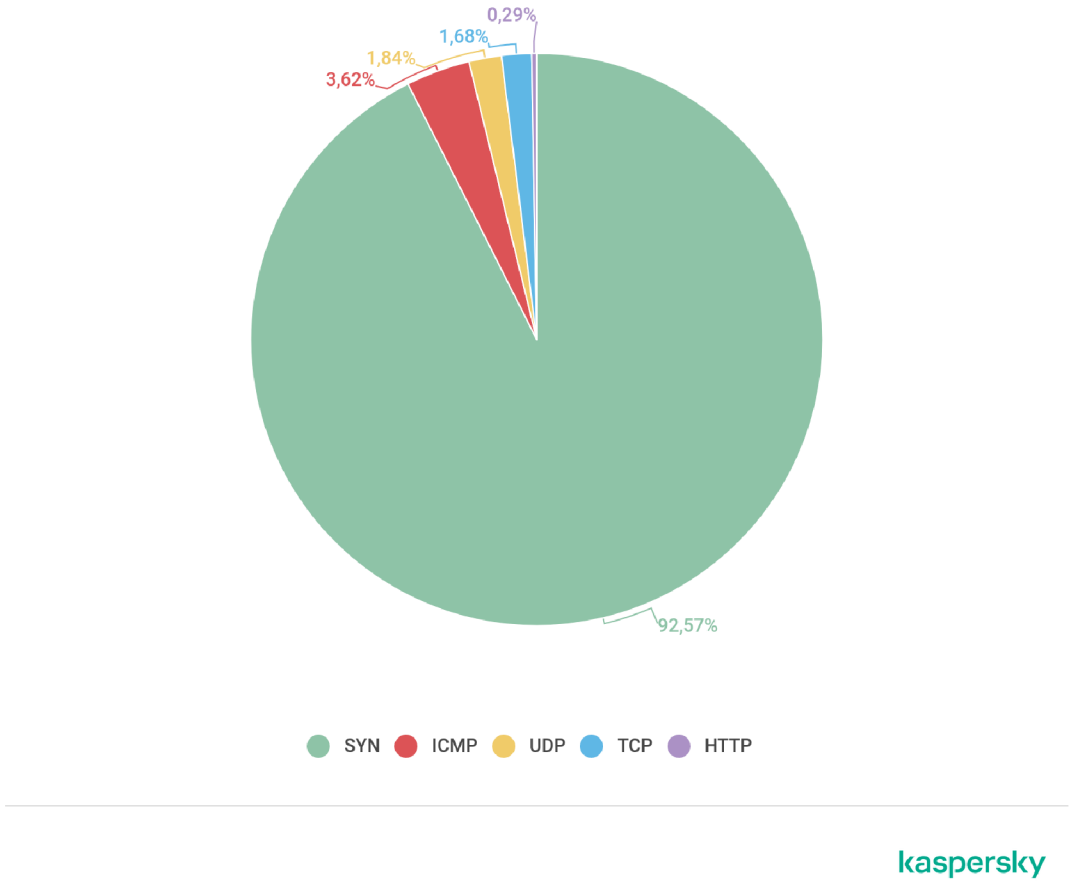
Při hledání optimálních parametrů má lineární oddělovač nulové váhy α_i pro každý datový bod kromě těch bodů, které jsou nejblíže vlastnímu oddělovači. Tyto nejbližší body mají název *podpůrné vektory* (anglicky support vectors), podle kterých je metoda pojmenována. To znamená, že lineární oddělovač se volí pouze na základě těchto podpůrných vektorů, jiné nejsou zapotřebí. Tato metoda tedy nevyužívá všechny trénovací příklady, ale pouze ty, které jsou relevantní pro volbu lineárního oddělovače. Tento fakt dělá z metody podpůrných vektorů efektivní algoritmus.

Nemůžeme ale očekávat, že najdeme lineární oddělovač v originálním vstupním prostoru $\mathbf{x}$. Z toho důvodu je zapotřebí upravit vstupní prostor určitou funkcí, která na něm není lineárně závislá. Tuto funkci nazýváme *jádrová funkce* a značíme ji $K(\mathbf{x}_i, \mathbf{x}_j)$. V souvislosti s touto metodou je jádrová funkce použita na dvojice vstupních dat k vyhodnocení skalárního součinu v nějakém odpovídajícím prostoru. Z toho vychází, že můžeme najít lineární oddělovač v novém prostoru $F(\mathbf{x})$ nahrazením $\mathbf{x}_i \cdot \mathbf{x}_j$ za jádrovou funkci $K(\mathbf{x}_i, \mathbf{x}_j)$. Těchto jádrových funkcí je více, např. polynomická, Gaussova jádrová funkce.

Tyto lineární oddělovače lze mapovat zpět od původního prostoru, kde už nebudou lineární, ale určitým způsobem zvlněné.

4.2 Výběr detekovaných typů útoku

Z obrázku 4.4 lze vidět, že nejčastějšími útoky v 1. čtvrtletí roku 2020 byly SYN záplavy, ICMP záplavy a UDP záplavy. Z tohoto důvodu bude v implementaci dbáno na to, aby tyto útoky byly spolehlivě detekovány. Útok typu SYN je mnohonásobně častější než jakýkoliv jiný útok, proto bude jeho detekce nejvíce důležitá. Jelikož útoky na aplikační vrstvu jsou vzácnější, systém se bude soustředit na detekci útoků nižších vrstev. Tyto informace vyplývají z článku [27]. Přestože hlavní zaměření práce je detekce útoků formou DDoS, jsou v návrhu zvoleny i SYN, ICMP, UDP záplavy formou DoS. Tedy ve výběru detekovaných útoků jsou zváženy již zmíněné záplavové útoky formou DDoS i DoS. Při detekci není brán takový důraz na útoky DoS jako na útoky typu DDoS.



Obrázek 4.4: Distribuce útoků DDoS podle typu v 1. čtvrtletí 2020. Převzato ze zprávy [27].

4.3 Výběr rysů/atributů

Aby byla klasifikace přesná, je zapotřebí zvolit rysy (atributy nebo vlastnosti) pro proces klasifikace. Volbou rysů chceme zajistit správnost klasifikace a minimalizaci chyb v ní, dále můžeme dobrou volbou zajistit efektivitu systému. Pro tuto práci byla zvolena většina rysů k dispozici po zpracování paketů do toků. Některé rysy jsou důležité pro určité typy útoků (např. SYN bit v TCP hlavičce pro SYN záplavy), jiné jsou zase společné pro všechny detekované typy. Kvůli vybrané metodě umělé inteligence (XGBoost) není nutné provést podrobný výběr rysů. XGBoost použije ve svých rozhodovacích stromech rysy s nejvyšší informační hodnotou. V tabulce C.1 jsou všechny rysy vyjmenovány a popsány. Všechny obsažené informace se vztahuji k jednom toku.

Důležité rysy SYN záplavy

SYN záplavy útočí velmi specificky, využívají způsob navázání komunikace TCP protokolu třícestný handshake, blíže popsáno v kapitole 2.2. Z tohoto důvodu je potřeba sledovat jednotlivé příznaky v TCP hlavičce, tvrdí autor studie [38].

Pro útok je typické mít nastavený SYN příznak v TCP hlavičce a zároveň ACK příznak v odpovědi. Z tohoto důvodu je důležité, aby byly počty paketů s těmito nastavenými příznaky zahrnuty v návrhu. Důležitost těchto příznaků je znázorněna ve více pracích [29], [33]. V rysech se sledují počty všech TCP příznaků v daném toku. To by mělo pomoci v detekci SYN záplav.

Podle genetického algoritmu použitého v již zmíněné práci [29] je důležitý parametr maximum skoků (anglicky Time To Live – TTL) v hlavičce IP paketu. Obecně tato hodnota bývá dost nepředvídatelná, ale v generovaných DDoS útocích není tak rozmanitá jako v normální komunikaci. Tato hodnota je sledována informacemi src_TTL_min, dst_TTL_min a src_TTL_max, dst_TTL_max a je obsažena v rysech trénovacích dat.

Důležité rysy UDP záplavy

V článku [4] je uvedeno, že při průběhu UDP záplavy je velikost UDP paketu značně menší, než při normálním provozu. Na základě zmíněné skutečnosti jsou v rysech obsaženy různé informace o velikosti paketu např. src_bytes nebo dst_bytes. Při UDP záplavě je očekávaná hodnota menší.

Důležité rysy ICMP záplavy

Pro ICMP záplavy je podle práce [37] důležité sledovat IP příznaky z hlavičky IP a velikost rámce. Autor zjistil, že v normální situaci je velikost rámce vyšší - konkrétně 98 bajtů a při útoku nižší (42 bajtů). Autor také zjistil, že typicky nejsou nastaveny IP příznaky a v normálním provozu jsou nastaveny často. Ve vlastní práci jsou tedy zahrnuty tyto informace pod názvem src_bytes, což odpovídá velikosti rámce od zdroje a src_TCP_IP_cumulative, kde jsou obsaženy IP příznaky také od zdroje.

4.4 Výběr přístupu detekce

Jak je známo ze sekce 3, je zapotřebí určit, kterou techniku detekce bude navrhovaný systém využívat. V sekci bylo řečeno, že velká výhoda při detekci na základě anomálií spočívá v možnosti odhalení nových útoků, které ještě systém nespatřil. Na druhou stranu detekce na základě signatur má vysokou přesnost a poměrně nízkou míru falešně pozitivních chyb. Z práce [6] vyplývá, že nejlepší modely mají co nejvyšší přesnost v odhalení útoků, ale zároveň minimalizují falešně pozitivní a falešně negativní chyby. Jak již je známo, práce se soustředí na detekci 3 útoků SYN, UDP, ICMP záplavy. Jelikož je záměr detekovat hlavně tyto tři útoky, je vhodné použití určité formy detekce na základě signatur. Rozhodovací stromy obsažené v algoritmu XGBoost se učí co je typické pro tento typ útoků a dokáží je rozeznat v síťové komunikaci. V tomto popsaném případě se ale nejedná o typický přístup detekce na základě signatur, tak jak je popsaný v sekci 3.2, ale spíš hybridní systém. Hybridní z toho důvodu, že algoritmus si tvoří určitý profil normálu komunikace a je zde potenciál, že algoritmus správně označí útoky, co předtím neviděl.

4.5 Způsob provedení charakterizace společných znaků

Díky specifické struktuře rozhodovacích stromů použitých v metodě *XGBoost* je poměrně přímočaré zjistit, které vlastnosti datového souboru jsou důležité v rozhodování vytvořeného modelu. Jak je již známo, všechny uzly v rozhodovacích stromech jsou podmínkami nad konkrétní vlastností datového souboru. Tyto podmínky jsou navržené tak, aby rozdělily datový soubor. Výběr ideální podmínky je založen na získání co největšího informačního zisku. To platí pro klasifikační problémy. Tedy podle tohoto měřítka lze seřadit vlastnosti datového souboru podle důležitosti. Seřadí se takovým způsobem, že se vypočítá průměr informačního zisku pro individuální vlastnost ve všech rozhodovacích stromech. Vlastnost s největším informačním ziskem je nejdůležitější pro vytváření predikcí.

Jenže to, že lze zjistit důležitost vlastností ve všech stromech, nám neřekne, jakým způsobem byla vytvořena jedna individuální predikce. Přesně k tomu slouží hodnoty SHAP. Hodnoty umožňují interpretaci modelu takovým způsobem, že lze ukázat, proč jedna datová instance byla zařazena do jedné z kategorií. Tento způsob se ukázal jako jediný konzistentní a lokálně přesný v zjišťování jednotlivých rozhodnutí [30].

V návrhu práce jsou použity hodnoty SHAP jak pro zjištění důležitost vlastností ve všech stromech, tak v interpretaci rozhodování jednotlivých predikcí. Tato metoda se ukázala jako jedna z mála spolehlivých v interpretaci.

4.6 Shrnutí návrhu

Systém implementovaný v jazyce Python 3 označuje jednotlivé síťové toky hodnotou 0 nebo 1. Jedná se o výstup. Hodnota 0 znamená, že daný síťový tok systém označil za normální. Hodnota 1 značí, že konkretní síťový tok je považován za útok. Aby mohl existovat systém, který je schopný takto komunikaci správně označit, je zapotřebí získat reálnou síťovou komunikaci. Ta bude použita při trénování, testování a validaci modelu. Typický formát pro zachycení komunikace je PCAP. V tomto formátu budou získány jednotlivé soubory síťové komunikace. Je zapotřebí získat jak soubory s normální komunikací, tak i soubory obsahující DoS/DDoS útok. Data použitá při trénování jsou z jiných zdrojů a data použitá k testování jsou generována a následně zachycena programem Wireshark[1]. Normální komunikace je výhradně převzata z různých zdrojů (detailněji popsáno níže). V návrhu je zvoleno, že se model vytvoří na základě informací sdružených do síťových toků. Další možnost se jevila nijak nezpracovat PCAP komunikaci a pracovat s informacemi jednotlivých paketů. Tato možnost byla vyloučena. Formát PCAP obsahuje mnoho zbytečných informací a jeho zpracování není podstatou této práce. Z toho důvodu je PCAP zpracován do síťových toků. Síťové toky musí obsahovat informace popsané v sekci 4.3. Takto rozdělená komunikace se více hodí pro zpracování metodou umělé inteligence, protože obsahuje souhrnné informace, které zachycují podstatu více paketů. Pro jednoduchost organizace a zpracování se zavede databázový systém PostgreSQL[2], do kterého jsou všechny datové soubory vloženy. Při vytváření modelu se datové soubory podle potřeby načtou z databáze a použijí.

V návrhu je určeno, že samotný model bude zpracován aplikací Jupyter Notebook v jazyku Python 3, pomocné funkce budou v odděleném souboru jazyku Python 3 a budou načteny do hlavního Jupyter dokumentu. Jupyter Notebook je typický nástroj pro práci s umělou inteligencí. Obsahuje vhodné nástroje pro vytvoření modelu a umožňuje vhodně

[1] https://www.wireshark.org/
[2] https://www.postgresql.org/

prezentovat výsledky, což je důležité z hlediska zaměření práce spíše na výzkumnou oblast. Modul, který obsahuje model, si pomocí dotazů načte data z databáze. Tato data potřebují další zpracování. Ze sekce 4.1 je známo, že model je tvořen metodou umělé inteligence XGBoost. Píše se tam, že XGBoost potřebuje označená data. V přípravě dat tato skutečnost musí být zohledněna. Je tedy zapotřebí načíst odděleně data obsahující útok a normální data a jednotlivě je označit. Dále je potřeba připravit data do formátu vhodného pro XGBoost. Výběr rysů ze sekce 4.3 je třeba zohlednit a upravit data takovým způsobem, aby odpovídala specifikaci již popsané. Tato část práce lze provést knihovnou Pythonu 3 pandas[3].

Následně proběhne tvoření modelu neboli trénování. V této části práce je zapotřebí určit trénovací a validační data. Dále je zapotřebí určit parametry modelu. Tyto části podstatně ovlivňují, jak úspěšný se model stane. Z toho důvodu je nezbytné je vhodně zvolit a jednotlivé možnosti otestovat. V průběhu tvoření modelu se musí konstantně zohledňovat nové informace, které nám vytvořený model poskytl, a použít je pro vytvoření nového lepšího modelu.

Po dokončení trénování se provádí testování na reálných datech. Nejprve se provede testování modelu nad jednotlivými útoky, které byly vybrány v sekci 4.2. Zjišťuje se, jestli model dokáže spolehlivě detekovat tyto útoky. Aby bylo testování kompletní, vytvoří se testovací datový soubor, který obsahuje namíchaná data útoků a normálních dat. Tento soubor zjistí, jak úspěšný je model. Úspěch se zhodnotí za pomocí různých metrik např. AUC, F1 skóre, míra falešně pozitivních a falešně negativních chyb apod.

Způsob provedení charakterizace společných znaků je popsán v sekci 4.5. V sekci je popsáno jakým způsobem zjistit, které rysy byly použity v rozhodování konkrétního útoku. Díky tzv. SHAP hodnotám je možnost zjistit dodatečné informace o tom, jak se rozhodoval model při predikci jedné instance dat. To nám dává možnost se dozvědět, které hodnoty parametrů jsou typické nebo důležité pro DoS/DDoS útok.

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

Kapitola 5

Implementace systému

V kapitole je do detailu vysvětlen postup implementace systému. V první sekci 5.1 je popsán způsob, jakým byla data vybrána a získána. Dále v sekci 5.2 je po krocích popsáno, jak jsou data připravena pro proces klasifikace. Model byl natrénován v sekci 5.3. Obsahuje popis hledání parametrů modelu a jeho celkové vytvoření. Jak je již stanoveno v návrhu, tak k implementaci je použitý nástroj Jupyter Notebook. Tento nástroj umožňuje vhodně prezentovat výsledky a usnadňuje práci s velkými daty. Aby byly výsledky jasně srozumitelné, Jupyter soubor neobsahuje vlastně definované funkce. Ty jsou definovány zvlášť v samostatném Python souboru.

5.1 Získání a výběr dat

Jeden z nejdůležitějších aspektů tvorby modelu je získání vhodných dat pro trénování, testování a validaci modelu. Zejména tedy pro trénování. V této práci bylo rozhodnuto, že je vhodné použít vlastně generovaná data spolu s daty od jiných autorů, aby byl výsledný model co nejvíce obecný. Získáním dat z různých zdrojů je vyvinuta snaha o snížení zkreslení dat.

Normální data

Aby bylo možné sestavit profil normálního chování, je nutnost vybrat vhodná data normálního provozu. Data slouží modelu k porovnání s daty útoků. Bylo rozhodnuto, že normální data jsou získána ze třech různých zdrojů.

Hlavní zdroj normální komunikace je datový soubor od Kanadského institutu pro kyberbezpečnost s názvem CIC-IDS2017 [34]. Datový soubor obsahuje normální komunikaci ve formátu PCAP. Autoři datového souboru se soustředili na vygenerování reálného provozu. První den generování věnovali pouze normální komunikaci. To znamená, že z datového souboru byla použita data pouze ze dne pondělí. Aby toho docílili, použili tzv B-Profile systém. Vytvořili abstraktní chování 25 uživatelů na základě poštovních protokolů a HTTP, HTTPS, FTP, SSH protokolů.

Další datový soubor, který byl použit pro profil normální komunikaci, se jmenuje bigFlows[1]. Provoz byl zachycen na zaneprázdněném přístupovém bodu k Internetu. Obsahuje mnohé síťové toky a mnohé aplikace.

[1] https://tcpreplay.appneta.com/wiki/captures.html

Finální datový soubor normální komunikace se jmenuje CTU-Normal-20 [13] . Byl vytvořen jako součást projektu na ČVUT univerzitě v Praze. Obsahuje mnoho připojení k častým HTTPS webovým stránkám.

V tabulce 5.1 jsou vypsány použité PCAP soubory normální komunikace, odkud pochází a jak se jmenují v databázi.

Tabulka 5.1: Shrnutí normálních datových souborů.

Původ dat. souboru	Originální název PCAP souboru	Jméno v databázi
CIC-IDS2017	Monday-WorkingHours.pcap	normal_CIC_IDS2017_1
CIC-IDS2017	Monday-WorkingHours.pcap	normal_CIC_IDS2017_2
CIC-IDS2017	Monday-WorkingHours.pcap	normal_CIC_IDS2017_3
CIC-IDS2017	Monday-WorkingHours.pcap	normal_CIC_IDS2017_4
bigFlows	bigFlows.pcap	normal_bigFlows
CTU-Normal-20	2017-04-30_win-normal.pcap	normal_CTU_Normal_20

Data DoS/DDoS útoků

Pří získávání datových souborů DoS/DDoS (SYN, ICMP, UDP) útoků byl zvolen přístup, který zajišťuje, že ke každému útoku jsou k dispozici alespoň 2 různé datové soubory. Bylo tak rozhodnuto, aby se dala spolehlivě ověřit detekce útoku na jiném datovém souboru, než byl použit v trénovacích datech. Zdroj dat je smíšený, některé datové soubory jsou vygenerované autorem práce. Další jsou zase od jiných autorů.

První datový soubor s názvem modbus [12] byl vyvinut při výzkumu použití strojového učení na problémy kyberbezpečnosti. Obsahuje 2 útoky. ICMP záplavy formou DDoS a SYN záplavy též formou DDoS. Konkrétně se jedná o datové soubory ze složky `captures2`. Blíže popsané v tabulce 5.2.

Následující datový soubor se jmenuje Bot-IoT, jeho specifikace je popsána v pracích [26], [25], [24], [22], [23], [21]. Jedná se o datový soubor, který byl podle autorů realizován ve skutečném prostředí. Jsou zde obsaženy originální PCAP soubory útoků UDP záplavy a SYN záplavy. Obsahují jak formu DDoS, tak i formu DoS.

Další datový soubor se jmenuje SDN-DDOS (ICMP,TCP,UDP)[20]. Jak již název napovídá, obsahuje všechny vybrané typy útoků. Tedy ICMP, UDP, SYN záplavy formou DDoS a DoS. Všechny útoky jsou obsažené v jednom souboru formátu PCAP. PCAP pochází ze složky `TRAIN-DATA`.

Poslední datový soubor z jiného zdroje pochází z nástroje pflua-bench. Byl použit pouze PCAP soubor s názvem `ping-flood.pcap`[2].

Tabulka 5.2: Shrnutí všech DoS/DDoS útoků, které nebyly generovány autorem práce.

Původ dat. souboru	**Originální název PCAP souboru**	**Jméno v databázi**
modbus	eth2dump-pingFloodDDoS30m-1h_1.pcap	icmp_ddos_modbus
modbus	eth2dump-tcpSYNFloodDDoS15m-6h_1.pcap	syn_ddos_modbus

[2] https://github.com/Igalia/pflua-bench/blob/master/savefiles/ping-flood.pcap

Bot-IoT	IoT_Dataset_TCP_DoS___00008 _20180604130802.pcap	tcp_dos_Bot_IoT
Bot-IoT	IoT_Dataset_UDP_DoS___00002 _20180604140231.pcap	udp_dos_Bot_IoT
Bot-IoT	IoT_Dataset_TCP_DDoS___00003 _20180604170243.pcap	tcp_ddos_Bot_IoT
Bot-IoT	IoT_Dataset_UDP_DDoS___00019 _20180604180729.pcap	udp_ddos_Bot_IoT
SDN-DDOS	TRAIN-DATA.pcap	all_SDN_DDOS
pflua-bench	ping-flood.pcap	icmp_dos_pflua

Generování vlastních DoS/DDoS útoků

Ke generování DoS/DDoS útoků je použit nástroj Hping3[3]. Tento nástroj je schopný vytvářet ICMP/UDP/TCP pakety, které odpovídají nastavení nástroje. Pro útok formou DDoS je schopný vytvářet podvržené zdrojové IP adresy. V případě SYN a UDP záplavy jsou útoky poslány na port 80, což je typické pro tento typ útoku. Pro rozsáhlejší testování je útok poslán i na jiné porty. Útok byl následně zachycen programem Wireshark.

SYN záplava

Pro generování SYN záplavy formou DoS byl použit příkaz 5.1:

```
hping3 -S --flood -p 80 *IP adresa*
```

Výpis 5.1: Příkaz pro generování SYN záplavy (forma DoS).

První parametr `-S` značí, že se má nastavit v TCP hlavičce SYN bit na 1. Parametr `--flood` říká, že se mají pakety posílat co nejrychleji to jde. Parametr `-p 80` říká, že se pakety mají poslat na port 80.

Pro generování SYN záplavy formou DDoS byl použit příkaz 5.2:

```
hping3 -S --flood --rand-source -p 80 *IP adresa*
```

Výpis 5.2: Příkaz pro generování SYN záplavy (forma DDoS).

Kde jsou použity stejné parametry jako v prvním útoku, ale navíc je zde parametr `--randsource`, který do paketů vloží náhodnou zdrojovou IP adresu.

Jsou generovány další 4 útoky s tím rozdílem, že komunikace je poslaná na jiný cílový port. SYN záplava formou DoS na port 443 byla vygenerován příkazem 5.3:

```
hping3 -S --flood -p 443 *IP adresa*
```

Výpis 5.3: Příkaz pro generování SYN záplavy na port 443 (forma DoS).

Ten samý útok jen formou DDoS byl vygenerován příkazem 5.4:

```
hping3 -S --flood --rand-source -p 443 *IP adresa*
```

Výpis 5.4: Příkaz pro generování SYN záplavy na port 443 (forma DDoS).

Dále byl vygenerován stejný útok na port 22. Nejprve formou DoS 5.5:

[3] https://www.kali.org/tools/hping3/

```
hping3 -S --flood -p 22 *IP adresa*
```

Výpis 5.5: Příkaz pro generování SYN záplavy na port 22 (forma DoS).

Dále formou DDoS příkazem 5.6:

```
hping3 -S --flood --rand-source -p 22 *IP adresa*
```

Výpis 5.6: Příkaz pro generování SYN záplavy na port 22 (forma DDoS).

ICMP záplava

Pro generování ICMP záplavy formou DoS byl použit příkaz 5.7:

```
hping3 --icmp --flood *IP adresa*
```

Výpis 5.7: Příkaz pro generování ICMP záplavy (forma DoS).

První parametr `--icmp` značí, že se mají použít ICMP pakety. Parametr `--flood` je již popsán.

Pro generování ICMP záplavy formou DDoS byl použit příkaz 5.8:

```
hping3 --icmp --rand-source --flood *IP adresa*
```

Výpis 5.8: Příkaz pro generování ICMP záplavy (forma DDoS).

Kde jsou použity stejné parametry jako v předchozím útoku, ale navíc je zde parametr `--rand-source`, který je již popsán.

UDP záplava

Pro generování UDP záplavy formou DoS byl použit příkaz 5.9:

```
hping3 --udp --flood -p 80 *IP adresa*
```

Výpis 5.9: Příkaz pro generování UDP záplavy (forma DoS).

První parametr `--udp` značí, že se mají použít UDP pakety. Ostatní parametry jsou již popsány.

Pro generování UDP záplavy formou DDoS byl použit příkaz 5.10:

```
hping3 --udp --flood --rand-source -p 80 *IP adresa*
```

Výpis 5.10: Příkaz pro generování UDP záplavy (forma DDoS).

Kde jsou použity stejné parametry jako v předchozím útoku, ale navíc je zde parametr `--rand-source`, který je již popsán.

Jsou generovány další 4 útoky s tím rozdílem, že komunikace je poslaná na jiný cílový port. UDP záplava formou DoS na port 53 byl vygenerován příkazem 5.11:

```
hping3 --udp --flood -p 53 *IP adresa*
```

Výpis 5.11: Příkaz pro generování UDP záplavy na port 53 (forma DoS).

Ten samý útok jen formou DDoS byl vygenerován příkazem 5.12:

```
hping3 --udp --flood --rand-source -p 53 *IP adresa*
```

Výpis 5.12: Příkaz pro generování UDP záplavy na port 53 (forma DDoS).

Dále byl vygenerován stejný útok na port 67. Nejprve formou DoS 5.13:

```
hping3 --udp --flood -p 67 *IP adresa*
```

Výpis 5.13: Příkaz pro generování UDP záplavy na port 67 (forma DoS).

Dále formou DDoS příkaze 5.14:

```
hping3 --udp --flood --rand-source -p 67 *IP adresa*
```

Výpis 5.14: Příkaz pro generování UDP záplavy na port 67 (forma DDoS).

Výběr trénovacích, validačních a testovacích dat

Jak je popsáno v sekci 4.1, při učení s učitelem se používají trénovací data. To je kolekce předem označených dat, ze kterých se metoda umělé inteligence učí vzory v datech. Jenže není možné použít pouze trénovací data při vytváření modelu. Musí se také vyhodnotit, jak je model úspěšný. Úspěšnost modelu nelze vyhodnotit z dat, ze kterých byl vytvořen. Mohlo by se zdát, že perfektně klasifikuje, ale v realitě při použití jiných dat by mohl být špatný. Aby byl model úspěšný, je zapotřebí mít odlišná data od těch použitých při trénování. Tato odlišná data se používají k vyhodnocení úspěšnosti modelu a jsou označována jako testovací data. Jenže vzniká problém. Jakým způsobem zvolit parametry modelu, aby byl co nejlepší? Při vytváření modelu by model neměl být vyhodnocován na základě testovacích dat, ta mají sloužit jako vyhodnocení úspěšnosti finálního modelu. Na volbu nejlepších parametrů modelu se používají ještě jiná třetí data - data validační. Úspěšnost metrik těchto dat se sleduje po každém trénovaní a na základě nich se mění parametry modelu. Validační data jsou další data navíc, která nejsou použita při trénování modelu. Z toho důvodu jsou vyvinuty metody, které dokáží sloučit validační a trénovací data dohromady a stále zachovají úspěšnost modelu. Jedna z těchto metod je křížová validace [31].

V práci je rozhodnuto, že trénovací data obsahují všechny typy útoků formou DoS i DDoS - od každého alespoň jeden. Všechny útoky v trénovacích datech jsou z jiných zdrojů odpovídají tabulce 5.2. První datový soubor použitý v trénovacích datech je modbus, ten obsahuje ICMP záplavy a SYN záplavy formou DDoS. Jako druhý datový soubor je použit BoT-IoT, který obsahuje UDP a SYN záplavy formou DoS i DDoS. Další datový soubor použitý v trénovacích datech se jmenuje SDN-DDOS. Obsahuje všechny typy útoků. Byly použity záznamy ze složky TRAIN-DATA daného datového souboru. Poslední použitý datový soubor útoku se jmenuje `ping-flood.pcap`. Je obsažen jako příklad v již zmíněném nástroji pflua-bench.

Do trénovacích dat patří taky označená normální data. V tabulce 5.1 jsou všechna vyjmenována. Pro trénování jsou použita všechna data kromě v databázi označených jako – `normal_CIC_IDS2017_1` a `normal_CIC_IDS2017_3`. Ta jsou odložena pro účel testování.

Všechny útoky jsou vyváženě zastoupeny v trénovacích datech. Až na útok typu ICMP záplava formou DoS. Zpracovaný datový soubor `ping-flood.pcap` totiž obsahuje pouze 1 tok. Separátní validační data se nevolí, protože pro validaci se použije křížová validace.

Pro testování úspěšnosti finálního modelu jsou použita data útoků, která jsou generovaná autorem práce v sekci 5.1. Model se otestuje na úspěšnosti detekce jednotlivých útoků. Hlavní testovací data jsou namíchaná vlastně generovanými daty útoků a normálními daty ze souboru CIC-IDS2017. Samozřejmě data použitá k testování ze souboru CIC-IDS2017 nejsou použita při trénování ani validaci.

5.2 Příprava dat

V této sekci je popsáno, jakým způsobem se přetvoří data ve formátu PCAP do formátu vhodného pro metodu XGBoost. Jak již bylo zmíněno v návrhu, nejprve se všechny datové soubory přetvoří do síťových toků libovolným nástrojem např. Suricata. Tyto záznamy síťových toků musí obsahovat všechny příslušné informace specifikované v sekci 4.3. V práci se takto formátované záznamy vložily do PostgreSQL databáze pro snadnou manipulaci. Dále se v Jupyter Notebook dokumentu naváže spojení na tuto lokální databázi, ve které jsou uloženy všechny datové soubory. Pomocí dotazů se z databáze načtou potřebné tabulky a provede se další zpracování v jazyce Python 3.

Příprava dat v jazyku Python

Jakmile jsou všechny datové soubory připraveny v databázi, načtou se pomocí dotazů a přetvoří se do formátu vhodného pro trénování. Pomocí knihovny SQLAlchemy[4] je navázáno spojení. Konkrétně je to provedeno funkcí `create_engine`. Ta vytvoří spojení s PostgreSQL databází. Musí se funkci specifikovat řetězec, který obsahuje přihlašovací jméno do databáze, heslo, IP adresu, port, jméno databáze.

Jakmile je navázáno spojení, je možné načíst jednotlivé tabulky. To je provedeno pomocí knihovny pandas již zmíněné v návrhu. Funkcí `read_sql_query` se načtou tabulky z databáze. Funkce má dva parametry. První je řetězec, který se řídí PostgreSQL syntaxí a lze v něj specifikovat dotazy. Druhý je identifikátor navázaného spojení provedeného v prvním kroku. Tabulka se uloží do datové struktury datový rámec (DataFrame), což je typická datová struktura knihovny pandas pro práci s umělou inteligencí. Útoky a normální data jsou načteny zvlášť, protože je nezbytné označit data příslušnou hodnotou. Hodnota 1 znamená, že daný síťový tok obsahuje útok. Hodnota 0 znamená, že daný síťový tok je normální a neobsahuje útok. Označení dat se provede ve vlastně definované funkci `add_label`. Tato funkce se nachází v odděleném Python souboru, ve kterém se nachází definice všech vlastně definovaných funkcí. Je zapotřebí jí předat informace, o který datový rámec se jedná a jestli se jedná o útok.

Následně je zapotřebí vybrat vhodné a nevhodné rysy neboli atributy (angl. features) datového souboru. V práci byla ponechaná většina rysů, protože rozhodovací stromy si samy vybírají atributy s největší informační hodnotou. V kódu jsou rysy/atributy často označované jako `cols`. Což je zkratka pro sloupce. Protože jednotlivé rysy vystupují v datovém rámci jako sloupce. Z toho důvodu se často zaměňuje mezi termínem sloupec a rys. Sloupce, které nejsou v návrhu, jsou smazány z datového rámce. Slouží k tomu vlastně definovaná funkce `drop_cols`, která obsahuje funkci datového rámce `drop`. Ta odstraní zvolené sloupce.

V aktuálním stavu jsou načtené a označené jednotlivé datové soubory a mají korektní sloupce. Jenže sloupce nejsou ve vhodném formátu pro metodu XGBoost. Např. hodnoty obsahují pole, řetězce, záporné hodnoty atd. Tyto hodnoty musejí být vhodně změněny. Ve funkci `handle_col_vals` jsou víceprvková pole rozdělena do jednotlivých sloupců. To znamená, že pokud hodnota sloupce obsahuje pole s více prvky, jeho jednotlivé prvky jsou nakopírované do samostatných sloupců. Tedy jeden sloupec, který má maximální délku pole 9, se rozdělí na 9 nových sloupců, které obsahují hodnoty původního pole. Ve stejné funkci se všechny hodnoty sloupců obsahující jednoprvková pole vyjmou z pole a vloží do stejného sloupce. Dále se odstraní všechny negativní hodnoty, které jsou v datovém rámci. To je provedeno ve funkci `drop_neg_vals`. Pro sloupce, které obsahují kategorické proměnné, je

[4] https://www.sqlalchemy.org/

změněn datový typ. Datový rámec má pro tyto sloupce datový typ `categorical`. Dále se datový rámec rozdělí na dva datové rámce. Jeden obsahuje všechny sloupce kromě toho, co obsahuje informaci, jestli se jedná o útok, nebo nikoliv. Druhý obsahuje sloupec právě s touto informací pro záznamy v prvním rámci. Nakonec se plně připravené datové rámce uloží do speciálního objektu specifického pro metodu a knihovnu XGBoost. Objekt nese název `DMatrix`, jedná se o interní datovou strukturu optimalizovanou pro danou metodu. Funkce, která tvoří tuto datovou strukturu z datového rámce, se nazývá také `DMatrix`. Funkci jsou specifikovány tyto parametry – datový rámec, ze kterého má být nový objekt vytvořen, další datový rámec obsahující značky záznamů prvního datového rámce, nastavení hodnoty `enable_categorical` na hodnotu `True`. Poslední jmenovaný parametry zajistí, že kategorické proměnné budou brány v úvahu při trénování.

Tímto způsobem jsou upravena všechna data – trénovací a testovací.

5.3 Trénování a validace modelu

Jak je popsáno v sekci 5.1, při trénování se pracuje buď se dvěma datovými soubory, nebo s jedním. Pokud je pouze jeden, jedná se o trénovací data a validace probíhá pomocí křížové validace. Pokud jsou dva, pracuje se s trénovacími daty a validačními daty. Jakmile jsou data zvolena, jediné co zbývá, je zvolit vhodně parametry metody XGBoost. Parametry modelu jsou uloženy v datové struktuře slovník, která je pojmenována `params`. Tato datová struktura se upravuje na základě výsledků křížové validace. Hlavní parametry blíže popsané v dokumentaci knihovny XGBoost [1] jsou následující:

- **eta (learning_rate)** – jedná se o míru aktualizace vytvářených stromů. Parametr zabraňuje přeučení modelu. Snižuje váhu rysům, aby se metoda učila pomaleji. Čím menší hodnota, tím déle trvá učení.
- **gamma (min_split_loss)** – velikosti prahu pro další rozdělení listů. Čím vyšší hodnota, tím jsou vytvořené stromy méně složitější. Vyšší hodnota parametru může zabránit přeučení, ale zpomalí se učení.
- **max_depth** – maximální hloubka vytvářených stromů. Vyšší hodnoty vytvoří komplexnější stromy, což může zvýšit přesnost, ale je vysoká šance přeučení.
- **min_child_weight** – minimální hodnota součtu vah v uzlu potřebný pro další rozdělení uzlu. Čím větší hodnota, tím méně jsou stromy komplexní a tím menší šance přeučení.
- **subsample** – poměr dílčího vzorku trénovacích instancí dat použitých při tvoření stromů. Při nastavení parametru na 0.5 se náhodně zvolí polovina trénovacích dat, která se použije na tvorbu rozhodovacích stromů. Čím menší hodnota tím, větší prevence přeučení. Vybírání trénovacích dat se děje při každé iteraci metody.
- **colsample_bytree** – podobný parametr jako subsample, ale s tím rozdílem, že se při iteraci limituje počet sloupců (rysů).
- **lambda** – regularizační parametr, který penalizuje složité modely.
- **alpha** – jedná se o další regularizační parametr, který penalizuje složité modely. Liší se od lambda úrovní regulariazace.

- **objective** – specifikuje cíl učení neboli podstatu řešícího problému. Nejvíce vhodná volba pro cíl práce se nazývá `binary:logistic`. Jedná se o logistickou regresi pro binární klasifikaci. Umožňuje určit pravděpodobnost, jestli predikce je 0 nebo 1.
- **tree_method** – specifikuje způsob sestavování rozhodovacích stromů. Kategorické proměnné podporují pouze možnosti `approx`, `hist` a `gpu_hist`. Z důvodu možnosti použití hardwarové akcelerace je vybrána možnost `gpu_hist`.
- **eval_metric** – určuje evaluační metriku určenou pro validační data. Knihovna má mnoho možností, některé možnosti jsou vysvětleny níže. Pro tento problém je vybrána metrika `auc`, což odpovídá metrice AUC-ROC.
- **seed** – číslo použité pro generátor pseudonáhodných čísel. Umožňuje pro stejná data získat pokaždé stejný model. Při ladění je zvolena hodnota 1.

Tyto parametry uložené v `params` jsou použity ve funkci knihovny XGBoost `train`. Funkce `train` spustí trénování objektu zvaného `Booster` ve výchozím nastavení – použití rozhodovacích stromů. Tento objekt reprezentuje model. Potřebným parametrem pro tuto funkci jsou trénovací data v datové struktuře `DMatrix`. Další parametr je počet iterací algoritmu XGBoost. Dále lze specifikovat pole, které určuje validační data, což díky křížové validaci nemusí být použito. Pokud jsou specifikovaná validační data, lze použít parametr, který se jmenuje `early_stopping_rounds`. Pokud je určen, slouží k monitorování výkonu modelu a dokáže předejít přeučení. Dosáhne toho tím, že pokud se pro specifikovaný počet iterací algoritmu nezmění sledovaná hodnota metriky, učení se zastaví. V každé iteraci algoritmu probíhá sestavování stromů. Jak je již v aktuálním odstavci zmíněno, tato hodnota je specifikována funkci `train`. Počet iterací, v knihovně XGBoost označované jako `num_boost_round`, je stejně jako parametry důležitou součástí modelu. Z toho důvodu i tato hodnota musí být laděna. V práci je považován jako jeden z dalších parametrů modelu, protože velice ovlivňuje výsledek modelu.

Postup při tvorbě modelu je zvolen následující. Po přípravě dat je k dispozici objekt `DMatrix`, ve kterém jsou uložena trénovací data. Zvolí se základní parametry modelu – datová struktura `params` - které již dál nebudou upravovány. Jedná se o tyto parametry –`objective`, `tree_method`, `eval_metric` a `seed`. Jejich nastavení již bylo popsáno. Ostatní parametry jsou ve výchozím nastavení. Parametry a trénovací data se použijí k ladění a validaci modelu. To je detailně popsáno v sekci 5.3. Výsledkem této fáze je výběr nejlepších parametrů pro daná trénovací data, včetně zjištění ideálního počtu iterací algoritmu. Jakmile jsou k dispozici tyto dvě informace, je možno přejít na vytvoření finálního modelu. Což je popsáno v předešlém odstavci. Model je připraven na testování.

Ladění hyperparametrů modelu a validace

Nativní rozhraní knihovny XGBoost, které je použito v práci, má svou vlastní implementaci křížové validace. Pomocí této funkce lze vyzkoušet různá nastavení hyperparametrů modelu a získat jejich nejlepší nastavení. Pro tuto situaci jsou vlastně definované dvě funkce `tune_param_pair` a `tune_single_param`, které využívají funkci knihovny XGBoost `cv`. Vlastně definované funkce předají po jednom funkci `cv` rozsah hodnot hyperparametrů. V případě `tune_param_pair` se hledají hodnoty dvou hyperparametrů a musí se specifikovat dva rozsahy. Při ladění pouze jednoho hyperparametru se použije funkce `tune_single_param`, ta potřebuje pouze rozsah hodnot jednoho ladícího hyperparametru. Všechny specifikované možnost parametrů se vyzkouší a vypíše se podle určité metriky

(v tomto případě AUC) nejlepší nastavení těchto dvou parametrů. Do dvojic jsou vybrány parametry, které spolu určitým způsobem souvisí – `max_depth` s `min_child_weight`, `eta` a `gamma`, `subsample` s `colsample_bytree`, `lambda` s `alpha`. Pořadí ladění je určeno na základě impaktu na model. `cv` potřebuje stejné parametry jako funkce `train` s tím rozdílem, že navíc potřebuje specifikovat, kolikanásobná křížová validace se má spustit. Typická doporučená hodnota je k=10. Iteračně se hledají nejlepší konfigurace hyperparametrů. Nejlepší volba hyperparametrů se vypíše a musí se upravit datová struktura `params` novými parametry. Parametry konečného modelu jsou zobrazeny v tabulce 5.3 níže. Je nutno dodat, že v tabulce jsou hyperparametry, které byly laděny. Další 4 parametry jsou pevně dány po celou dobu ladění – `objective = binary:logistic`, `tree_method = gpu_hist`, `seed = 1` a `eval_metric = auc`.

Konkrétní postup při ladění je zvolen následující. Na začátku Jupyter dokumentu se zvolí, že se přechází na fázi ladění parametrů, což vyjadřuje proměnná `bool_param_tuning`. Ta se nastaví na hodnotu `True`. Nejprve se zvolí parametr `eta` na nižší hodnotu aby se předešlo přeučení. V případě finálního modelu `eta=0.1`. Na této hodnotě moc nezáleží, protože se nejedná o finální nastavení parametru. `eta` bude ještě laděna. Dále jsou zvoleny hodnoty, které se mají odzkoušet. Je spuštěna funkce `tune_param_pair` pro parametry `max_depth` a `min_child_weight`. Funkce vypíše nejlepší hodnotu metriky AUC pro dvojici odzkoušených parametrů a těmito hodnotami se přepíše datová struktura `params`. To stejné se provede pro dvojici `subsample` a `colsample_bytree`. Dále se ladí parametry `eta` a `gamma`. Opět se zvolí hodnoty k odzkoušení a zase je spuštěna funkce `tune_param_pair`. Volba parametrů s nejvyšší hodnotou AUC aktualizuje strukturu `params`. Nakonec se ladí regularizační parametry – `lambda` a `alpha`. Zase se spustí `tune_param_pair` a aktualizuje se nejlepšími hodnotami `params`. Ladění je dokončeno, v `params` jsou optimální parametry podle křížové validace. Jak již bylo zmíněno funkce `train` potřebuje znát počet iterací. Počet iterací má velký vliv na výsledný model. Lze nalézt jeho ideální hodnotu stejně jako to bylo u hyperparametrů. Zase se použije funkce křížové validace `cv` s teď již finální strukturou `params`. Nejlepší počet iterací je uložen v datovém rámci `cv_results`. Konkrétně první sloupec a poslední hodnota. Ta se ještě musí inkrementovat o 1, aby hodnota odpovídala celkovému počtu iterací, ne pouze číslu poslední iterace. Interní validační soubor křížové validace dosáhl nejvyšší hodnoty v 57 iteracích. Křížová validace byla po těchto iteracích zastavena, protože hodnota metriky AUC se nezměnila za posledních 10 iterací.

Tabulka 5.3: Vybrané hyperparametry na základě ladění.

Hyperparametr	Volba hyperparametru
eta	0.25
max_depth	3
min_child_weight	1
subsample	0.9
colsample_bytree	0.6
gamma	0.0
lambda	1.0
alpha	0.0

Když je znám ideální počet iterací a nejlepší volba parametrů, stačí natrénovat model funkcí `train`. Vhodný počet iterací pro parametry v tabulce 5.3 je 57. Natrénovaný model je reprezentován objektem `Booster`, v implementaci je pojmenovaný jako `model`. Nad tímto objektem lze provádět další testování a vytvářet predikce. Pomocí křížové validace byly

zvoleny ideální parametry modelu a bylo ověřeno, že model je vhodně natrénovaný na trénovací data. Model je nakonec uložen do souboru s příponou `.model` pomocí jeho funkce `save_model`. Aby se při jeho použití nemusel znovu vytvářet a mohl se pouze načíst ze souboru.

Kapitola 6

Testování a analýza chování DoS/DDoS

V této kapitole je provedena analýza chování DoS/DDoS a testování modelu. Testování proběhlo ve dvou fázích. Nejprve se otestoval model na všech útocích. V této fázi byl prověřen, jak je schopný detekovat jednotlivé útoky. V další fázi byl model otestován na testovacích datech složených z normální komunikace a z komunikace útoků.

Potom je provedena charakterizace útoků DDoS. Jinak řečeno se v kapitole interpretuje model. Je vyvinuta snaha o získání co nejvíce nových znalostí z interpretovaného modelu. Cílem kapitoly, která se zabývá interpretací, je ukázat, co je typické pro tyto typy útoků a zjistit, jakým způsobem vytvořený model predikuje nové útoky. Jsou zde ukázány rysy, které jsou v rozhodování nejvíce důležité. Dále je ukázáno, které faktory model zvažoval při rozhodování jedné konkrétní instance. Interpretace modelu je provedena na základě SHAP hodnot, již zmíněno v sekci 4.5.

6.1 Testování modelu

V sekci jsou definovány jednotlivé evaluační metriky, které umožňují zjistit, jak kvalitní a úspěšný vytvořený model je. Model je nejprve otestován a zhodnocen na datových souborech obsahující pouze DoS/DDoS útoky. Dále je model otestován ve více reálné situaci, kde model musí klasifikovat toky v datovém souboru, který obsahuje normální data i útoky. Natrénovaný model je reprezentován objektem `Booster`, tento objekt má metodu `predict`, která po specifikování testovacích dat ve datovém souboru `DMatrix` vrátí pro každý tok pravděpodobnost, že tok obsahuje útok. Pokud pravděpodobnost je více jako 0.5, je tok označen za útok. V jiném případě je označen jako normální.

Evaluační metriky

Aby bylo možné posoudit, jak moc nebo málo úspěšný model je, je nutno vědět, jakým způsobem se vyhodnocuje. Existuje mnoho metrik, které posuzují výsledky predikcí modelu. Ve finálním modelu je použita vyhodnocovací metrika AUC ROC. Shrnutí metrik je vypracováno na základě prací [18],[19].

Chybovost (angl. error rate)

Vyjadřuje míru chybné klasifikace. Měří poměr mezi nesprávnými predikcemi a celkovým počtem predikcí. V knihovně XGBoost je označována tato evaluační metrika jako `error`.

Přesnost (angl. accuracy)

Obecně metrika přesnost měří poměr mezi správnými predikcemi vůči celkovému počtu predikcí. Jedná se o komplementární metriku k chybovosti. Dá se jednoduše převést mezi chybovostí. Rovnice k převodu je: $presnost = 1 - chybovost$. Je to velice oblíbená metrika při vytváření modelu.

Preciznost (angl. precision)

Též označovaná jako pozitivní prediktivní hodnota (PPV). Jedná se o poměr počtu správně pozitivně klasifikovaných ku počtu všech pozitivně klasifikovaných. Preciznost lze použít i na negativní třídu, což potom lze označovat jako negativní prediktivní hodnota (NPV). Což je poměr správně negativně klasifikovaných ku počtu všech negativně klasifikovaných.

Senzitivita (angl. sensitivity nebo recall)

Tato metrika se používá pro měření poměru počtu správně pozitivně klasifikovaných ku počtu všech patřících do pozitivní třídy. Zjednodušeně to vyjadřuje míru správně pozitivně klasifikovaných.

Specificita (angl. specificity)

Jedná se o tu samou metriku jako senzitivita, ale pro negativní třídu. Je to počet správně negativně klasifikovaných ku počtu všech patřících do negativní třídy. Neboli míra správně negativně klasifikovaných.

F1 skóre

Často používaná metrika pro vyhodnocování úspěšnosti v klasifikaci pozitivní třídy je F1 skóre. Závisí na preciznosti a senzitivitě. $F1skore = 2\frac{preciznost \cdot senzitivita}{preciznost + senzitivita}$. Lze měřit F1 skóre i pro negativní třídu, což vypadá následovně: $F1skore = 2\frac{NPV \cdot specificita}{NPV + specificita}$

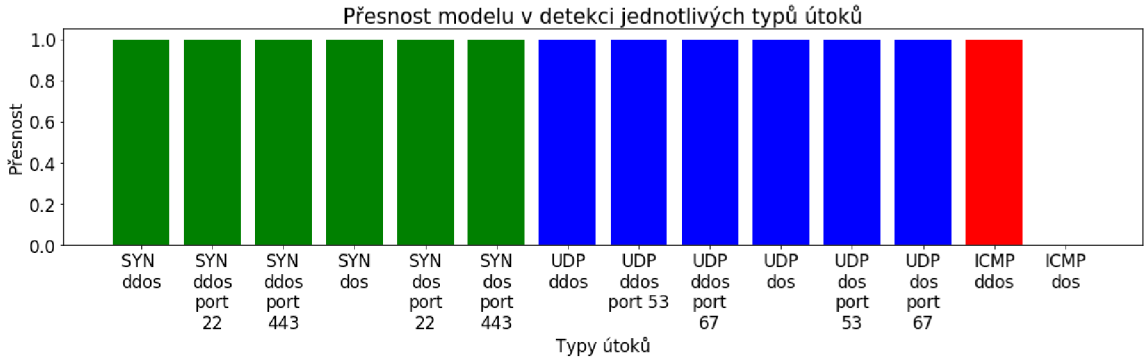
AUC ROC

Tato metrika byla prokázána jako teoreticky i empiricky lepší, než metrika přesnosti. Jako evaluační metrika se používá skalární charakteristika AUC. Je na intervalu $< 0; 1 >$. Vyjadřuje jediným číslem výkon klasifikátoru. Aby bylo možné zjistit, co vyjadřuje hodnota AUC, je zapotřebí porozumět grafu ROC (angl. Receiver Operating Characteristic). ROC slouží k přehlednému zobrazení predikčního potenciálu modelu. Na ose x je míra falešně pozitivních, což odpovídá vzorci $1 - specificita$. Na ose y je senzitivita. Jeden klasifikátor odpovídá v grafu jednomu bodu. V testování je ROC graf zobrazen. Typicky platí, že čím blíže hornímu levému rohu graf ROC je, tím je model lepší. Teď už se lze přesunout na to, co je to AUC. AUC (angl. Area under the ROC Curve) je plocha pod křivkou ROC. Vyjadřuje graf ROC jedním číslem, což lze použít na vyhodnocení kvality klasifikátoru při jeho vytváření. V XGBoost knihovně je pod názvem `auc` a je přímo použitá při vytváření modelu.

Testování detekce jednotlivých útoků

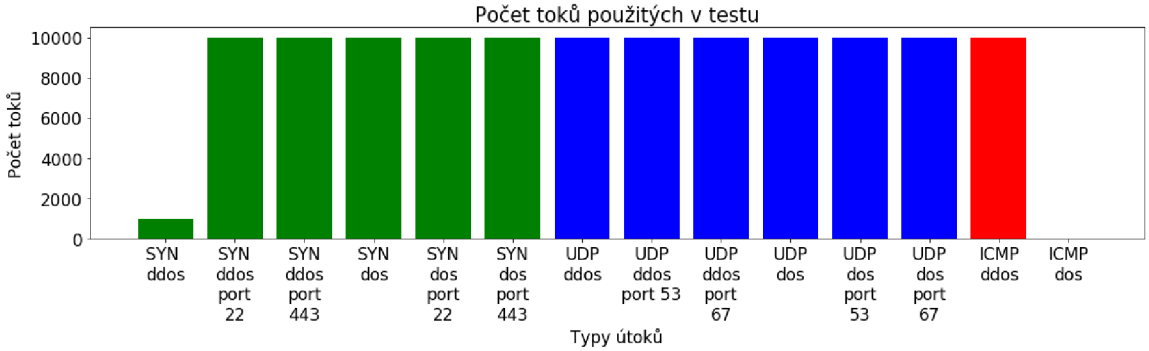
Aby bylo možné zjistit, jak je vytvořený model kvalitní, je dobré ho nejprve otestovat na jednotlivých útocích. Jelikož detekce těchto útoků je podstata navrženého systému, je tato metrika esenciální. Na modelu jsou vyzkoušeny jednotlivé útoky vygenerované v sekci 5.1. V tabulce 6.1 jsou vypsané jednotlivé útoky a jejich odpovídající datové soubory. Jelikož tyto testovací soubory obsahují pouze komunikaci útoků, není nutné ani možné použít všechny metriky ke zhodnocení výkonu modelu na útocích. Je to z důvodu nepřítomnosti druhé třídy v testovacím souboru. Máme k dispozici pouze klasifikace správně pozitivní a falešně pozitivní. Jediné metriky ze sekce 6.1, které lze plně vyjádřit a poskytnou hodnotnou informaci, jsou tyto – senzitivita, přesnost nebo chybovost. V případě testování pouze jedné třídy jsou metriky senzitivita a přesnost identické. Jelikož chybovost je komplementem přesnosti, tak i z tohoto důvodu má smysl vyjadřovat pouze přesnost. Maximální počet záznamů jednotlivých datových souborů použitých při testu je 10000. Vlastně definovaná funkce `test_one_set` provede testování specifikovaného datového souboru z databáze. Další parametry funkce: maximální počet načtených záznamů z databáze, sloupce/rysy trénovacích dat, objekt reprezentující připojení k databázi, vytvořený model, informace, jestli je testovací soubor útok nebo ne, a trénovací data.

Grafy v aktuální sekci jsou zobrazeny pomocí knihovny `Matplotlib`[1]. Útoky stejného typu mají stejnou barvu, liší se pouze formou. V prvním obrázku 6.1 lze vidět přesnost jednotlivých útoků. Při útocích SYN a UDP záplavy přesnost dosahuje 100 procent. To znamená, že všechny toky obsažené ve vyjmenovaných datových souborech byly označeny jako útok (hodnota 1). Při ICMP záplavě je to složitější. ICMP záplava formou DDoS je spolehlivě detekována, přesnost zase dosahuje až 100 procent, ale ICMP záplava formou DoS není detekována vůbec. Přesnost útoku formou DoS je nulová. Příčina této skutečnosti může být v počtu toků. Na obrázku 6.2 lze vidět graf, který zobrazuje jednotlivé útoky a jejich počet toků. Jak již bylo zmíněno, maximální počet toků byl zvolen na 10000. Jenže při ICMP záplavě formou DoS vzniká problém. Všechny pakety mají stejnou zdrojovou a cílovou IP adresu a porty ICMP protokol nevyužívá. Tedy veškerý útok je zaznamenán do jednoho toku. Vytvořený model pravděpodobně nezískal dostatečný počet informací, aby mohl tento tok správně klasifikovat.



Obrázek 6.1: Přesnost modelu na různých typech DDoS a DoS útocích. Zobrazeno pomocí knihovny Matplotlib.

[1] https://matplotlib.org/stable/api/index



Obrázek 6.2: Počet toků jednotlivých testovacích souborů. Zobrazeno pomocí knihovny Matplotlib.

V tabulce 6.1 lze vidět shrnutí aktuálního testu. Tabulka obsahuje mimo již zmíněných datových souborů a názvů útoků taky počet záznamů útoku a dosaženou přesnost. Je nutno podotknout, že vygenerované testovací útoky obsahují velmi podobné toky. Tedy dává smysl, že přesnost pro daný test je buď nula, nebo blíže sto procent. Testování jednotlivých útoků proběhlo úspěšně. Model se prokázal jako schopný v detekci všech typů kromě ICMP záplavy formou DoS. Útoky SYN a UDP záplavy jsou otestovány ve více variantách.

Tabulka 6.1: Shrnutí testu všech generovaných útoků.

Název útoku	**Název datového souboru**	**Počet záznamů testu**	**Přesnost**
SYN záplava formou DDoS	syn_ddos_own	1000	100.0
SYN záplava formou DDoS (port 22)	syn_22_ddos_own	10000	100.0
SYN záplava formou DDoS (port 443)	syn_443_ddos_own	10000	100.0
SYN záplava formou DoS	syn_dos_own	10000	99.99
SYN záplava formou DoS (port 22)	syn_22_dos_own	10000	99.99
SYN záplava formou DoS (port 443)	syn_443_dos_own	10000	100.0
UDP záplava formou DDoS	udp_ddos_own	10000	99.97
UDP záplava formou DDoS (port 53)	udp_53_ddos_own	10000	99.97
UDP záplava formou DDoS (port 67)	udp_67_ddos_own	10000	99.97
UDP záplava formou DoS	udp_dos_own	10000	99.99
UDP záplava formou DoS (port 53)	udp_53_dos_own	10000	100
UDP záplava formou DoS (port 67)	udp_67_dos_own	10000	99.99
ICMP záplava formou DDoS	icmp_ddos_own_1	10000	100.0
ICMP záplava formou DoS	icmp_dos_own	1	0.0

Testování detekce smíšených dat

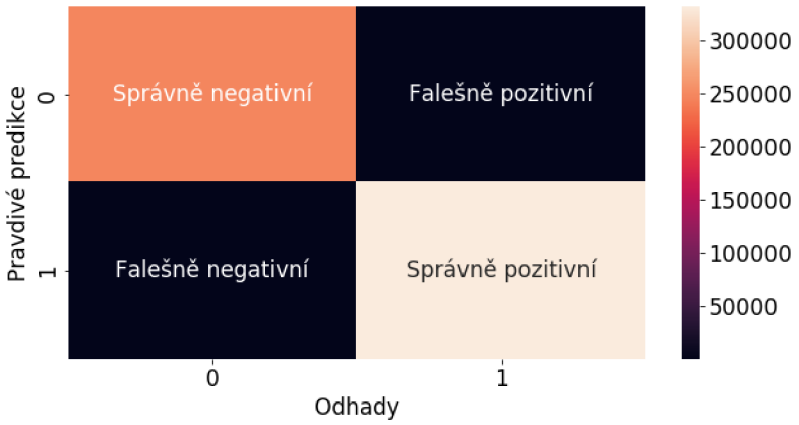
Přesnost v detekci jednotlivých útoků je již otestována. Nyní přichází na řadu více reálná situace. V aktuální sekci je model otestován na smíšených datech. To znamená, že se vy-

tvoří pouze jeden testovací soubor, který obsahuje data útoků a normální data. Vlastně definovaná funkce, která vytvoří jeden společný testovací soubor, se jmenuje `create_mixed_test_set`. Ta načte jednotlivé datové soubory a sloučí je. Dále je upraví do vhodného tvaru pro funkci `predict`. Nakonec vrátí objekt `DMatrix` obsahující jeden společný testovací soubor. Nad tímto testovacím souborem lze provést rozsáhlejší vyhodnocení modelu. Všechny metriky popsané v sekci 6.1 jsou použity k vyhodnocení. Jediná nepoužitá metrika je chybovost, protože je komplementem přesnosti.

Testovací soubor je složen z vlastně vygenerovaných útoků ze sekce 5.1. Z každého vygenerovaného souboru obsahující útok je načteno maximálně 30000 záznamů. Normální data pochází z datového souboru CIC-IDS2017, konkrétně `normal_CIC_IDS2017_1` a `normal_CIC_IDS2017_3`. Samozřejmě se jedná o odložená data, která nebyla použita při trénování ani validaci. Načteny jsou z nich všechny záznamy.

Celý testovací soubor obsahuje 248205 záznamů normální komunikace a 331740 záznamů z komunikace útoků. K zobrazení veškerých údajů v této sekci jsou použity dvě knihovny - `Matplotlib` a `seaborn`[2]. Knihovna `seaborn` umožňuje jednodušší vykreslení datových rámců.

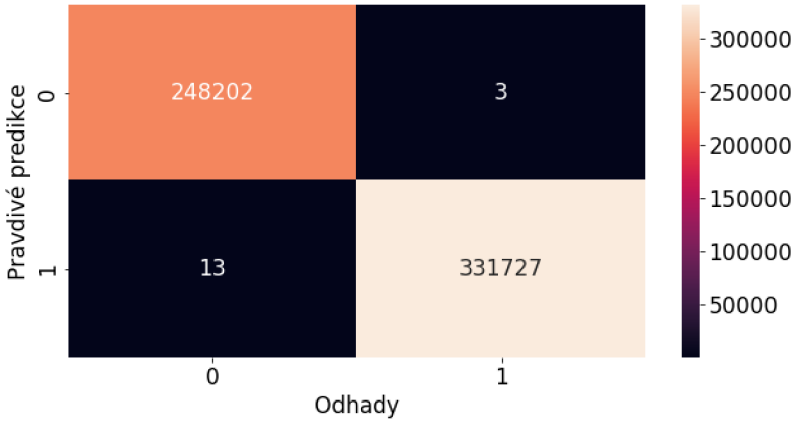
První obrázek 6.3 je vykreslený pomocí teplotní mapy (angl. heat map). Zobrazuje, co znamenají jednotlivá políčka v teplotní mapě. Nejdůležitější informace zobrazeny v obrázku 6.3 jsou statistické chyby – falešně pozitivní a falešně negativní.



Obrázek 6.3: Zobrazení pozice statistických veličin. Vypočítáno pomocí funkce `confusion_matrix` z knihovny sklearn. Zobrazeno pomocí funkce `heatmap` z knihovny seaborn.

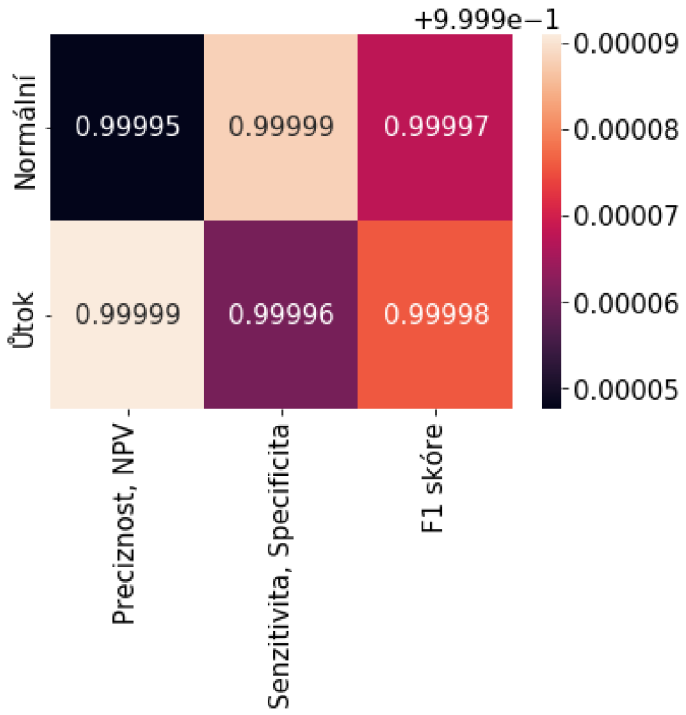
V dalším obrázku 6.4 jsou reálné naměřené hodnoty z predikce testovacího souboru. Odpovídají obrázku 6.3. Tyto hodnoty jsou esenciální ve vyhodnocování úspěšnosti modelu. Vypočítávají se z nich další metriky např. AUC, F1 skóre atd. Lze vidět, že vytvořený model správně predikuje třídy jednotlivých toků. Zejména vynikající je počet jeho falešně pozitivních chyb. Informace z obou obrázků byly získány z funkce `confusion_matrix` knihovny sklearn, jedná se konkrétně o modul[3]. Parametry funkce jsou: pravdivé označení síťového toku (útok či ne) a zaokrouhlené predikce modelu z funkce `predict`.

[2] https://seaborn.pydata.org/index.html
[3] https://scikit-learn.org/stable/modules/model_evaluation.html



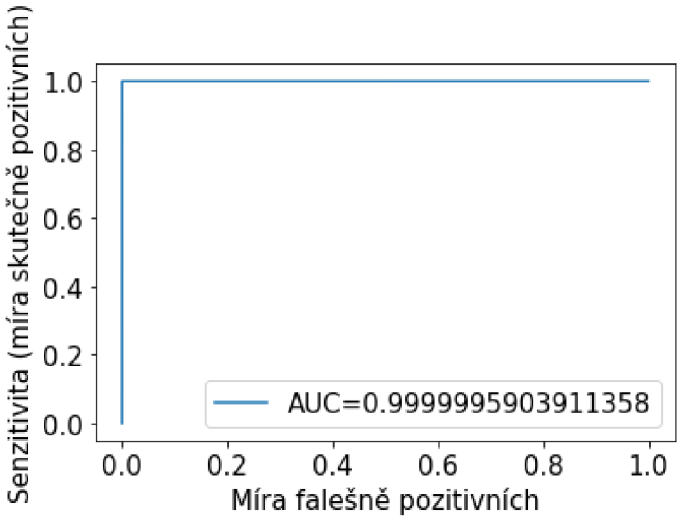
Obrázek 6.4: Reálné statistické veličiny smíšeného testovacího souboru. Vypočítáno pomocí funkce `confusion_matrix` z knihovny sklearn. Zobrazeno pomocí funkce `heatmap` z knihovny seaborn.

Následující obrázek 6.5 ukazuje některé z metrik popsaných v sekci 6.1. Výsledky funkce `classification_report` z knihovny sklearn jsou zobrazeny pomocí teplotní mapy. Již výsledky z obrázku 6.3 napovídaly, že hodnoty metrik budou vysoké. Opravdu tomu tak je. Model je ohodnocen velmi vysoce všemi uvedenými metrikami.



Obrázek 6.5: Vypočítané evaluační metriky. Vypočítáno pomocí funkce `classification_report` z knihovny sklearn. Zobrazeno pomocí funkce `heatmap` z knihovny seaborn.

Nyní je zapotřebí zhodnotit model na v tomto případě nejdůležitější metrice – AUC. Je v tomto případě nejdůležitější, protože jako jediná metrika byla použita při vytváření modelu. Z tohoto důvodu je v obrázku 6.6 zobrazen ROC graf. Čím více se křivka blíží hornímu levému rohu, tím je model lepší. Hodnota AUC, která odpovídá obsahu křivky, je zobrazena uvnitř obrázku.



Obrázek 6.6: ROC graf a AUC hodnota pro smíšený testovací soubor. Vypočítáno pomocí funkce `roc_curve` z knihovny sklearn. Zobrazeno pomocí funkce `plot` z knihovny Matplotlib.

Nakonec je vypočítána přesnost pro smíšený testovací soubor. Přesnost výsledného modelu je 99.997%. Model byl zhodnocen na různých metrikách a jeho úspěšnost byla prokázána na testovacím souboru, který obsahuje normální komunikaci i komunikaci útoků. Jediný nedostatek modelu je ten, že ICMP záplavu formou DoS nedokázal detekovat. Odůvodněno je to tím, že v trénovacích a testovacích datech je v každém pouze 1 záznam tohoto útoku.

6.2 Důležité rysy

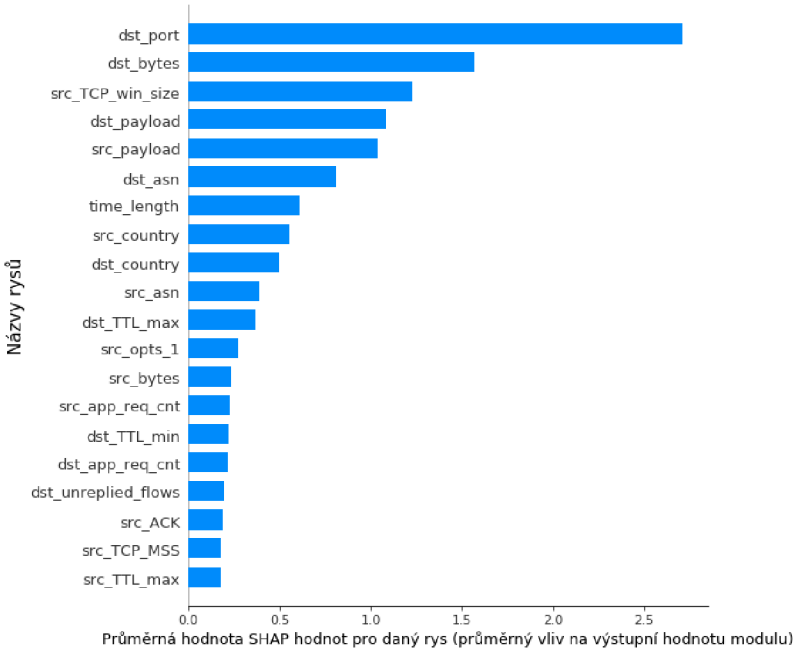
SHAP (angl. SHapley Additive exPlanations) hodnoty umožňují spolehlivě a konzistentně (jako jedna z mála metod) zjistit důležitost rysů/atributů. Jiné metody jsou často nekonzistentní [30]. Z těchto důvodů je použita právě tato metoda na zjištění důležitých rysů při rozlišování útoků od normální komunikace. Pro tento účel je použita knihovna `shap`[4]. Tato knihovna dokáže vypočítat SHAP hodnoty a obsahuje funkce, které umožňují jejich přímočarou interpretaci. Dokonce je zabudována do samotné knihovny `XGBoost`. Použito je původní rozhraní knihovny. Je vytvořen objekt `explainer`, doslovně přeloženo jako vysvětlovač, který obsahuje funkci `shap_values`. Funkce vypočítá SHAP hodnoty pro specifikovaný model a testovací data, tyto hodnoty se analyzují.

SHAP se vypočítají pro každý sloupec každého záznamu testovacího souboru. To znamená, že vznikne datová struktura, která má stejný rozměr jako testovací soubor. Ovšem v této struktuře jsou vypočítané SHAP hodnoty. Čím větší SHAP hodnota, tím víc měla

[4] https://shap.readthedocs.io/en/latest/

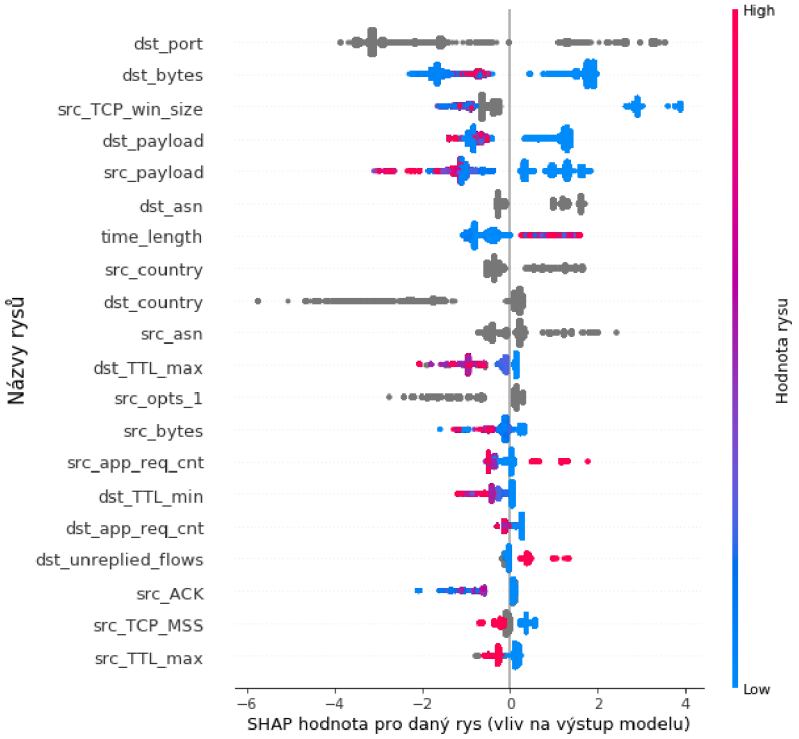
podíl na predikci dané instance. Pokud je hodnota záporná, daná hodnota rysu ovlivnila predikci ve prospěch normální třídy. Pro kladné je to zase ve prospěch predikce útoků.

V prvním obrázku 6.7 jsou seřazeny jednotlivé rysy podle důležitosti – průměru SHAP hodnot sloupce (rysu). Osa x značí průměr všech SHAP hodnot daného rysu. Na ose y jsou jednotlivé rysy. Zajímavá věc na obrázku je, že bylo vycházeno z předpokladu, že je důležité mít informaci o nastavených příznacích v TCP hlavičce. Jak lze vidět, tento model toto tvrzení nepotvrzuje. V sekci 4.3 bylo také zmíněno, že je podle jiných autorů důležité mít informaci o velikosti paketu (src_bytes a dst_bytes), popřípadě velikosti dat (src_payload a dst_payload). To se tady potvrdilo. Dále je logické, že důležitým rysem je cílový port (dst_port), protože naznačuje, která služba běží na cílovém zařízení. Útoky v trénovacích datech útočily specificky na jen některé porty.



Obrázek 6.7: Ukázka důležitých rysů. Jsou seřazeny podle důležitosti. Zobrazeno pomocí funkce `summary_plot` z knihovny shap.

Na dalším obrázku 6.8 jsou zobrazeny a seřazeny jednotlivé rysy podle důležitosti. Pointa tohoto obrázku je zjistit, jaké mají hodnoty jednotlivých rysů vliv na predikci. Červená barva značí vysoké hodnoty daného rysu a modrá barva značí nízké hodnoty. Šedé hodnoty jsou buď chybějící hodnoty, nebo kategorické proměnné. Na ose x jsou konkrétní SHAP hodnoty. Z obrázku lze např. vidět, že data (src_payload a dst_payload) s vyšší velikosti jsou typická spíše pro normální komunikaci a data (src_payload a dst_payload) s nižší velikostí jsou zase typická spíše pro útoky. Dále lze vypozorovat, že počet neodpovězených toků cíle (dst_unreplied_flows) je jednoznačně vyšší pro útoky. Zajímavé také je, že časová délka (time_length) toku je vyšší pro útoky, než pro normální komunikaci.



Obrázek 6.8: Ukázka důležitých rysů ve vztahu s jejich hodnotami. Jsou seřazeny podle důležitosti. Zobrazeno pomocí funkce `summary_plot` z knihovny shap.

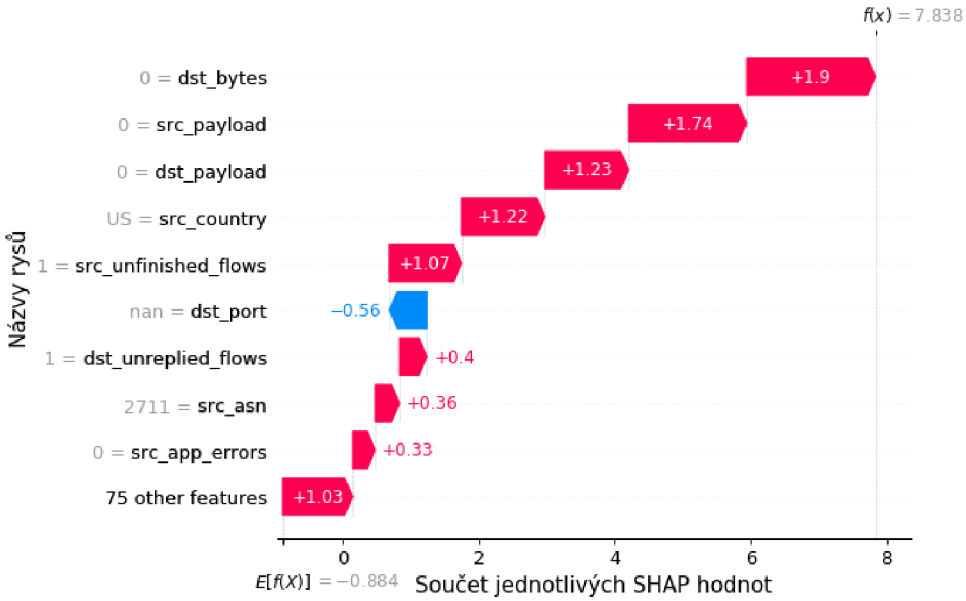
Výsledky charakterizace přinesly znalosti o rozdílu mezi útokem DDoS a normální komunikací. Dalo by se z pozorování zobecnit, že při DoS/DDoS útoku dochází k menší výměně aplikačních dat. Dále bylo zjištěno, že celkový počet bajtů při DoS/DDoS útocích směrované na nižší vrstvy je od cíle menší, než při normální komunikaci. Při útoku tok obvykle trvá déle a v normální komunikaci se častěji objevují vyšší hodnoty TTL.

6.3 Interpretace jednotlivých predikcí

Když se počítaly důležité rysy pomocí SHAP hodnot, počítal se průměr z těchto hodnot pro každý rys. Jenže to, že je známo, které rysy jsou považovány za důležité, nijak neobjasní individuální lokální predikce. Jinak řečeno, proč se model rozhodl klasifikovat konkrétní tok jako 0 nebo 1. Pochopení jednotlivých predikcí může zlepšit porozumění rozdílu mezi útokem a normálním chováním. V předešlé sekci 6.2 je již popsáno, jak se SHAP hodnoty vypočítaly. Tam se braly v úvahu všechny hodnoty. Pro vysvětlení individuálních predikcí stačí vzít SHAP hodnoty vypočítané pro každý záznam testovacího souboru. Tedy počet SHAP hodnot pro jeden záznam je dán počtem rysů.

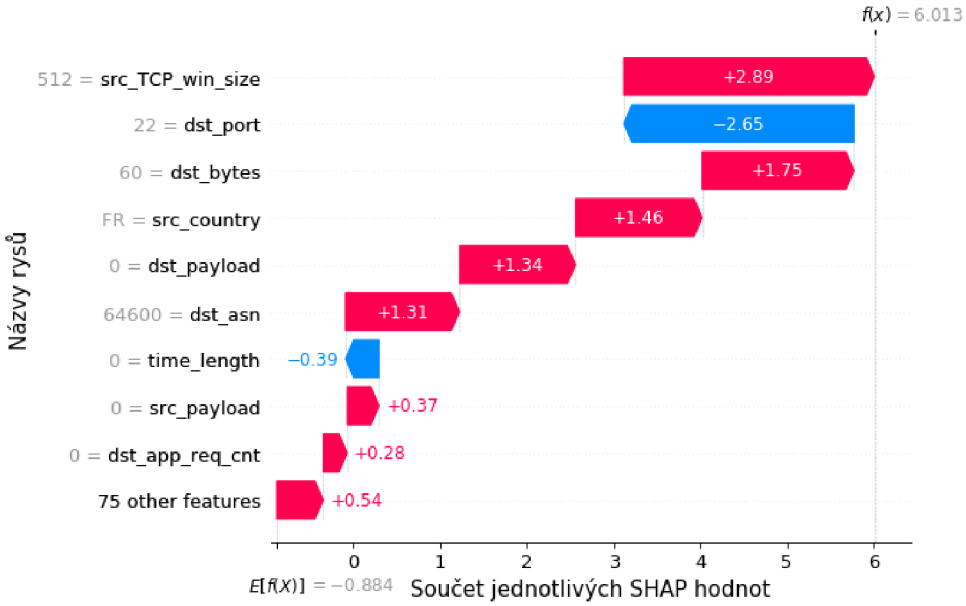
V prvním obrázku 6.9 je vysvětlena predikce toku, který obsahuje útok. Jedná se konkrétně o ICMP záplavu formou DDoS. Na levé straně (osa y) obrázku lze vidět nejdůležitější rysy použité při predikci tohoto záznamu z testovacího souboru. Jsou tam také ukázány reálné hodnoty tohoto záznamu. Na ose x jsou hodnoty SHAP. Značí, jak moc daný rys ovlivnil predikci tohoto záznamu. Smysl tohoto obrázku/grafu je ukázat, které rysy ovlivnily predikci pro tuto instanci, jak moc rys ovlivnil predikci a na jakou stranu. Modrá barva značí, že daný rys snížil pravděpodobnost útoku a červená barva značí, že

rys zvýšil pravděpodobnost útoku. V tomto konkrétním případě lze vidět, že naprostá většina hodnot rysů indikovala, že se jedná o útok. Nejvyšší váhu měla na predikci velikost bajtů od cíle (dst_bytes). Lze jasně vidět, že nulová velikost aplikačních dat (src_payload a dst_payload) také velmi přispěla k rozhodnutí. Znovu se potvrzuje, že velikost paketu a dat je důležitá informace. Stejně jako tomu bylo v sekci 6.2.



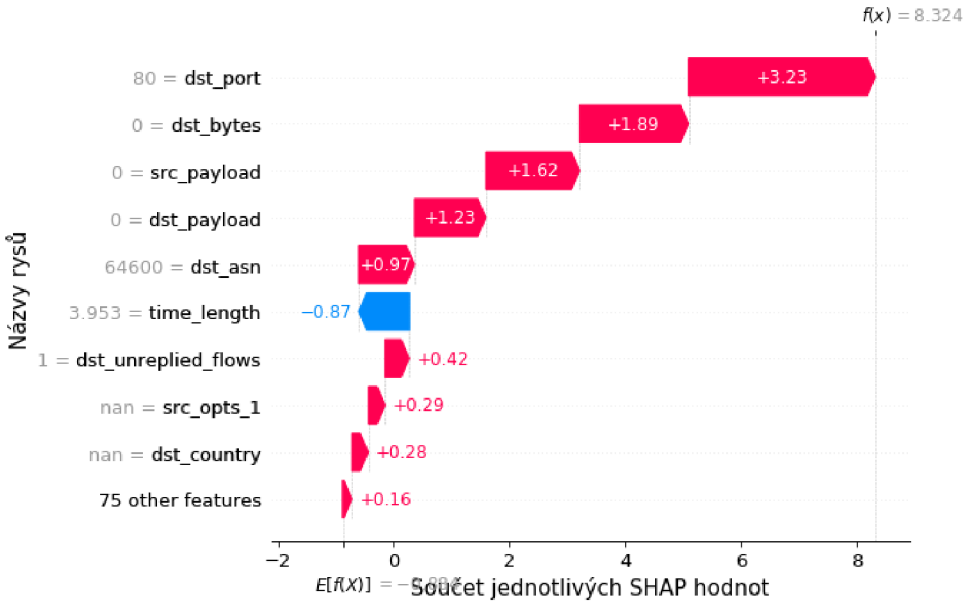
Obrázek 6.9: Vysvětlení predikce pro ICMP záplavu formou DDoS. Zobrazeno pomocí funkce `waterfall_legacy` z knihovny shap.

Další obrázek 6.10 popisuje vysvětlení predikce pro záznam SYN záplavy na port 22 formou DDoS ze smíšeného trénovacího datového souboru. Oproti předešlému obrázku je to méně jednoznačná predikce. Trénovací data neobsahovala žádné útoky, které cílily na port 22. Tím pádem model nepředpokládá útok na tento port (dst_port). I přes to na základě informací z jiných rysů dokázal správně určit původ toku. Největší vliv na to měla velikost TCP okna (src_TCP_win_size).



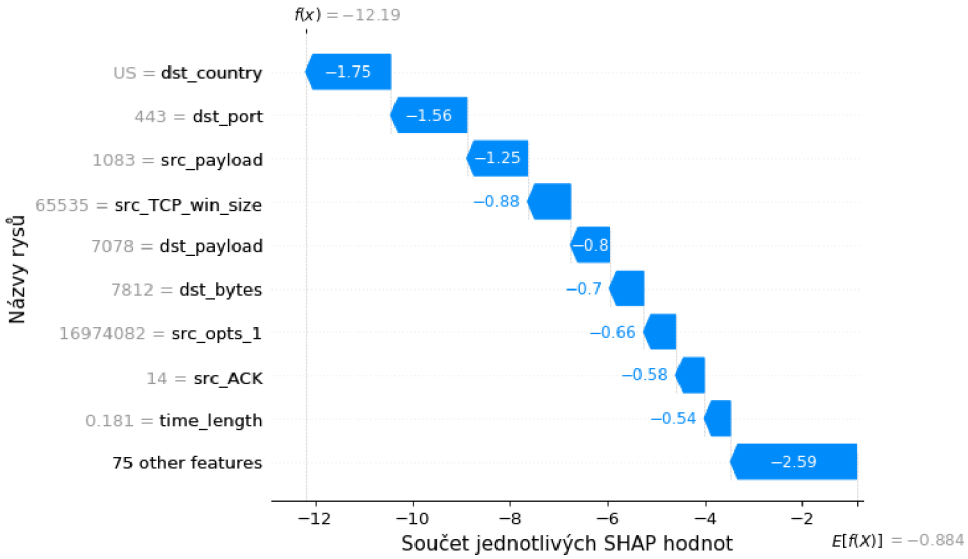
Obrázek 6.10: Vysvětlení predikce pro SYN záplavu formou DDoS. Zobrazeno pomocí funkce `waterfall_legacy` z knihovny shap.

Následující obrázek 6.11 popisuje predikci UDP záplavy formou DoS. Útok je poslán na typický port pro DoS/DDoS útok 80. Zatím byly ukázány pouze útoky formy DDoS. Je vhodné se podívat na to, v čem se liší predikce DoS a DDoS. Predikce byla velmi jednoznačně určena. Rysy indikující útok jsou velmi podobné jako při útocích DDoS. Rozdíl je pouze v tom, že cílový port (dst_port) ještě více zvyšuje šanci na útok.



Obrázek 6.11: Vysvětlení predikce pro UDP záplavu formou DoS. Zobrazeno pomocí funkce `waterfall_legacy` z knihovny shap.

Poslední obrázek 6.12 vysvětluje, proč model označil tento záznam jako normální. Z obrázku lze vidět, že všechny hodnoty jednoznačně indikovaly, že se jedná o normální tok. Velký rozdíl oproti minulým predikcím je v tom, že rysy přispěly k predikci poměrně rovnoměrně. Žádný z rysů nepřispěl mnohonásobně více než další. Samozřejmě jsou tu takové, které přispěly více než ostatní – země původu cílové IP adresy (dst_port) a cílový port (dst_port). Ale podstatné je to, že všechny, nebo skoro všechny rysy indikovaly, že se jedná o normální tok. Lze to vidět hlavně na tom, že zbylé nezobrazené rysy jednoznačně ovlivnily predikci k normálu. Toto je velmi dobré znamení, že model funguje správně.



Obrázek 6.12: Vysvětlení predikce pro normální tok. Zobrazeno pomocí funkce `waterfall_legacy` z knihovny shap.

Kapitola 7

Závěr

Cílem práce bylo navrhnout a implementovat systém, který je schopný spolehlivě detekovat vybrané typy DDoS útoků na základě informací převážně z nižších vrstev modelu OSI. Dalším cílem bylo získání dodatečných znalostí o chování těchto útoků. Záměr práce byl zcela splněn.

Aby bylo možné vyjmenované cíle splnit, bylo zapotřebí se nejprve seznámit s problematikou. To znamená zjistit, jak fungují konkrétní DDoS útoky, jaké existují typy tohoto útoku a co je umožňuje. Dále bylo nutné se seznámit s principy detekce anomálií v počítačových sítích. To zahrnuje zjištění, jak fungují IDS systémy a jaké způsoby detekce používají. Pro řešení je nezbytné znát druhy anomálií a které z nich vystihují podstatu DDoS útoku. Nakonec byly zjištěny způsoby detekce DDoS útoků.

Po nabytí nutného teoretického základu byl navrhnut systém. Jeho požadavky byly stanoveny na základě cílů práce. Nejdůležitější součást návrhu byla volba metody pro detekci a charakterizaci útoků. Na základě obsáhlé analýzy možností hodících se pro tento problém byla vybrána metoda umělé inteligence XGBoost. Bylo zjištěno, že v jiných pracích dosahovala vysoké přesnosti a umožňovala přímočarou interpretaci modelu – získání nových znalostí. V návrhu byly také vybrány rysy a typy detekovaných útoků. Vstup systému byl organizován do síťových toků, ty byly využity k trénování a testování modelu.

Návrh byl implementován podle specifikace. Při implementaci bylo nutno získat datové soubory, které byly použity pro vytváření modelu. Trénovací data byla navolena z více zdrojů pro zachování objektivity modelu. Následně došlo k validaci a ladění parametrů pomocí křížové validace. Potom byl model ve dvou fázích otestován. První fáze obsahovala pouze vlastně generované útoky – SYN záplava, UDP záplava a ICMP záplava. Celkem bylo použito 14 různých variant těchto útoků. Všechny byly detekovány s přesností 99.97-100%. Až na jeden – ICMP záplava formou DoS. V trénovacích datech byl spatřen jen jeden záznam, z toho důvodu model neměl dostatek informací na jeho správné zařazení. Pro testování v druhé fázi byl sestrojen testovací soubor, který obsahoval smíšená data generovaných útoků a odložených normálních dat. Celkem 579945 záznamů. Test na smíšených datech dopadl výborně, jeho přesnost dosáhla 99.997%. Byl vyhodnocen na více různých metrikách. Pouze 3 toky byly označeny jako falešně pozitivní chyba.

Po otestování modelu byla provedena analýza chování DoS a DDoS útoků použitých v testovacích souborech. Analýzu umožnily tzv. SHAP hodnoty. Dokázaly spolehlivě určit důležité rysy a vysvětlit jednotlivé predikce modelu. Důležité rysy pro rozpoznání útoku byly – číslo cílového portu, velikosti paketů a velikost aplikačních dat.

V práci bych chtěl pokračovat rozšířením schopnosti modelu detekovat rozsáhlejší spektrum DDoS útoků a vyzkoušet jiné metody umělé inteligence. Dále by bylo vhodné navázat na tuto práci výzkumem zabývajícím se mitigací DDoS.

Literatura

[1] *XGBoost Parameters* [online]. [cit. 2022-05-10]. Dostupné z: https://xgboost.readthedocs.io/en/stable/parameter.html.

[2] Ahmed, M., Naser Mahmood, A. a Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* [online]. 2016, sv. 60, C, s. 19–31, [cit. 2022-05-10]. ISSN 1084-8045. Dostupné z: https://www-sciencedirect-com.ezproxy.lib.vutbr.cz/science/article/pii/S1084804515002891.

[3] Alenezi, M. a Reed, M. J. Methodologies for detecting DoS/DDoS attacks against network servers. In: IARIA. *The Seventh International Conference on Systems and Networks Communications ICSNC* [online]. 2012, s. 92–98 [cit. 2022-05-10]. ISBN 978-1-61208-231-8. Dostupné z: https://www.researchgate.net/profile/Mohammed-Alenezi-5/publication/352312016_Methodologies_for_detecting_DoSDDoS_attacks_against_network_servers/links/60c31c9ba6fdcc2e6131a793/Methodologies-for-detecting-DoS-DDoS-attacks-against-network-servers.pdf.

[4] Bijalwan, A. a Singh, H. Investigation of UDP Bot Flooding Attack. *Indian Journal of Science and Technology* [online]. Červen 2016, sv. 9, č. 21, s. 1–6, [cit. 2022-05-10]. DOI: 10.17485/ijst/2016/v9i21/85014. Dostupné z: https://www.researchgate.net/profile/Anchit_Bijalwan/publication/304492068_Investigation_of_UDP_Bot_Flooding_Attack/links/5cb729e992851c8d22f109b5/Investigation-of-UDP-Bot-Flooding-Attack.pdf.

[5] Bishop, M. *Computer Security: Art and Science*. 1. vyd. Boston, MA: Addison-Wesley, 2003. ISBN 0201440997.

[6] Buczak, A. L. a Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications surveys and tutorials* [online]. PISCATAWAY: IEEE. 2016, sv. 18, č. 2, s. 1153–1176, [cit. 2022-05-10]. ISSN 1553-877X. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7307098.

[7] Chen, Z., Jiang, F., Cheng, Y., Gu, X., Liu, W. et al. XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud. In: *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)* [online]. IEEE, 2018, s. 251–256 [cit. 2022-05-10]. ISBN 1538636492. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8367124.

[8] Dasmohapatra, M., Datta, K. a Sengupta, I. A preventive measure to protect from denial of service attack. In: Wyld, D. C., Wozniak, M., Chaki, N.,

Meghanathan, N. a Nagamalai, D., ed. *Advances in Network Security and Applications* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, sv. 196, s. 157–166 [cit. 2022-05-10]. ISBN 9783642225390. Dostupné z: https://doi-org.ezproxy.lib.vutbr.cz/10.1007/978-3-642-22540-6_16.

[9] Douligeris, C. a Mitrokotsa, A. DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks* [online]. 2004, sv. 44, č. 5, s. 643–666, [cit. 2022-05-10]. ISSN 1389-1286. Dostupné z: https://www.alexandria.unisg.ch/262960/1/1-s2.0-S1389128603004250-main.pdf.

[10] Endorf, C. *Detekce a prevence počítačového útoku*. 1. vyd. Praha: Grada, 2005. ISBN 80-247-1035-8.

[11] Fernandes, J., Joel, J. P. C. R., Carvalho, L. F., Al Muhtadi, J. a Proença, J. A comprehensive survey on network anomaly detection. *Telecommunication Systems* [online]. Březen 2019, sv. 70, č. 3, s. 447–489, [cit. 2022-05-10]. Copyright - Telecommunication Systems is a copyright of Springer, (2018). All Rights Reserved; Last updated - 2019-09-05. Dostupné z: https://www.proquest.com/scholarly-journals/comprehensive-survey-on-network-anomaly-detection/docview/2063108190/se-2?accountid=17115.

[12] Frazão, I., Henriques Abreu, P., Cruz, T., Araujo, H. a Simoes, P. Denial of Service Attacks: Detecting the Frailties of Machine Learning Algorithms in the Classification Process. In: Springer, ed. *13th International Conference on Critical Information Infrastructures Security (CRITIS 2018)*. Kaunas, Lithuania: [b.n.], Prosinec 2018, s. 230–235. DOI: 10.1007/978-3-030-05849-4_19. ISBN 978-3-030-05849-4. Springer series on Security and Cryptology.

[13] Garcia, S. *Malware Capture Facility Project*. Dostupné z: https://stratosphereips.org.

[14] Gouveia, A. a Correia, M. Network Intrusion Detection with XGBoost. In: *Recent Advances in Security, Privacy, and Trust for Internet of Things (IoT) and Cyber-Physical Systems (CPS)* [online]. Chapman and Hall/CRC, 2020, s. 137–166 [cit. 2022-05-10]. DOI: 10.1201/9780429270567-6. ISBN 9780429270567. Dostupné z: https://www.gsd.inesc-id.pt/~mpc/pubs/XGBoost_chapter.pdf.

[15] Gu, Y., Li, K., Guo, Z. a Wang, Y. Semi-Supervised K-Means DDoS Detection Method Using Hybrid Feature Selection Algorithm. *IEEE Access* [online]. 2019, sv. 7, s. 64351–64365, [cit. 2022-05-10]. DOI: 10.1109/ACCESS.2019.2917532. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8717648.

[16] Gupta, B. B., Joshi, R. C. a Misra, M. Defending against Distributed Denial of Service Attacks: Issues and Challenges. *Information security journal.* [online]. Taylor and Francis Group. 2009, sv. 18, č. 5, s. 224–247, [cit. 2022-05-10]. ISSN 1939-3555. Dostupné z: http://www.tandfonline.com/doi/abs/10.1080/19393550903317070.

[17] Gupta, N., Jain, A., Saini, P. a Gupta, V. DDoS attack algorithm using ICMP flood. In: IEEE. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* [online]. Bharati Vidyapeeth, New Delhi as the Organizer of INDIACom - 2016, 2016, s. 4082–4084 [cit. 2022-05-10]. ISBN

9789380544212. Dostupné z:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7725026.

[18] Honzík, P. Strojové učení. *Elektronická skripta VUT Brno* [online]. 2006, [cit. 2022-05-10]. Dostupné z:
http://vision.uamt.feec.vutbr.cz/STU/others/Honzik%20-%20Strojove_uceni_S.pdf.

[19] Hossin, M. a M.N, S. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process* [online]. Březen 2015, sv. 5, č. 2, s. 01–11, [cit. 2022-05-10]. DOI: 10.5121/ijdkp.2015.5201. Dostupné z:
https://www.researchgate.net/profile/Mohammad-Hossin/publication/275224157_A_Review_on_Evaluation_Metrics_for_Data_Classification_Evaluations/links/57b2c95008ae95f9d8f6154f/A-Review-on-Evaluation-Metrics-for-Data-Classification-Evaluations.pdf?origin=publication_detail.

[20] Housman, O. G., Hafida, I. a Fauzi, S. *SDN-DDOS (ICMP,TCP,UDP)*. Mendeley, 2020. DOI: 10.17632/hkjbp67rsc.1.

[21] Koroniotis, N. *Designing an effective network forensic framework for the investigation of botnets in the Internet of Things*. University of New South Wales. Engineering & Information Technology, 2020.

[22] Koroniotis, N. a Moustafa, N. Enhancing network forensics with particle swarm and deep learning: The particle deep framework. *CoRR*. 2020, abs/2005.00722. Dostupné z: https://arxiv.org/abs/2005.00722.

[23] Koroniotis, N., Moustafa, N., Schiliro, F., Gauravaram, P. a Janicke, H. A Holistic Review of Cybersecurity and Reliability Perspectives in Smart Airports. *IEEE Access*. 2020, sv. 8, s. 209802–209834. DOI: 10.1109/ACCESS.2020.3036728.

[24] Koroniotis, N., Moustafa, N. a Sitnikova, E. A new network forensic framework based on deep learning for Internet of Things networks: A particle deep framework. *Future Generation Computer Systems*. 2020, sv. 110, s. 91–106. DOI: https://doi.org/10.1016/j.future.2020.03.042. ISSN 0167-739X. Dostupné z:
https://www.sciencedirect.com/science/article/pii/S0167739X19325105.

[25] Koroniotis, N., Moustafa, N., Sitnikova, E. a Slay, J. Towards Developing Network Forensic Mechanism for Botnet Activities in the IoT Based on Machine Learning Techniques. In: Hu, J., Khalil, I., Tari, Z. a Wen, S., ed. *Mobile Networks and Management*. Cham: Springer International Publishing, 2018, s. 30–44. ISBN 978-3-319-90775-8.

[26] Koroniotis, N., Moustafa, N., Sitnikova, E. a Turnbull, B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*. 2019, sv. 100, s. 779–796. DOI: https://doi.org/10.1016/j.future.2019.05.041. ISSN 0167-739X. Dostupné z:
https://www.sciencedirect.com/science/article/pii/S0167739X18327687.

[27] Kupreev, O., Badovskaya, E. a Gutnikov, A. *DDoS attacks in Q1 2020* [online]. 2020 [cit. 2022-05-10]. Dostupné z:
https://securelist.com/ddos-attacks-in-q1-2020/96837/.

[28] Kurose, J. F. *Computer networking : a top-down approach.* 7. vyd. Essex: Pearson, 2017. ISBN 978-1-292-15359-9.

[29] Mašetić, Z., Kečo, D., Doğru, N. a Hajdarević, K. SYN Flood Attack Detection in Cloud Computing using Support Vector Machine. *TEM Journal* [online]. Novi Pazar: UIKTEN - Association for Information Communication Technology Education and Science. 2017, sv. 6, č. 4, s. 752–759, [cit. 2022-05-10]. ISSN 22178309. Dostupné z: https://www.temjournal.com/content/64/TemJournalNovember2017_752_759.pdf.

[30] Meng, Y., Yang, N., Qian, Z. a Zhang, G. What makes an online review more helpful: An interpretation framework using xgboost and shap values. *Journal of theoretical and applied electronic commerce research* [online]. MDPI AG. 2021, sv. 16, č. 3, s. 466–490, [cit. 2022-05-10]. ISSN 0718-1876. Dostupné z: https://www.mdpi.com/0718-1876/16/3/29.

[31] Ripley, B. D. *Pattern Recognition and neural Networks.* 1. vyd. Cambridge (U.K.): Cambridge University Press, 1996. ISBN 0-521-46086-7.

[32] Sanjeetha, R., Raj, A., Saivenu, K., Ahmed, M. I., Sathvik, B. et al. Detection and mitigation of botnet based ddos attacks using catboost machine learning algorithm in SDN environment. *International Journal of Advanced Technology and Engineering Exploration* [online]. Bhopal: Accent Social and Welfare Society. 2021, sv. 8, č. 76, s. 445–461, [cit. 2022-05-10]. ISSN 2394-5443. Dostupné z: https://www.accentsjournals.org/PaperDirectory/Journal/IJATEE/2021/3/2.pdf.

[33] Seo, J., Lee, C., Shon, T., Cho, K.-H. a Moon, J. A new DDoS detection model using multiple SVMs and TRA. In: Enokido, T., Yan, L., Xiao, B., Kim, D., Dai, Y. et al., ed. *Embedded and Ubiquitous Computing – EUC 2005 Workshops* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, sv. 3823, s. 976–985 [cit. 2022-05-10]. ISBN 978-3-540-32296-2. Dostupné z: https://link.springer.com/content/pdf/10.1007/11596042_100.pdf.

[34] Sharafaldin., I., Habibi Lashkari., A. a Ghorbani., A. A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: INSTICC. *Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP,* [online]. SciTePress, 2018, s. 108–116 [cit. 2022-05-10]. DOI: 10.5220/0006639801080116. ISBN 978-989-758-282-0. Dostupné z: https://www.scitepress.org/papers/2018/66398/66398.pdf.

[35] Srivastava, A., Tyagi, A., Sharma, A. a Mishra, A. A Recent Survey on DDoS Attacks and Defense Mechanisms. In: Nagamalai, D., Renault, E. a Dhanuskodi, M., ed. *Advances in Parallel Distributed Computing: First International Conference on Parallel, Distributed Computing Technologies and Applications, PDCTA 2011, Tirunelveli, India, September 23-25, 2011. Proceedings* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, sv. 203, s. 570–580 [cit. 2022-05-10]. Communications in Computer and Information Science. ISBN 9783642240362. Dostupné z: https://doi.org/10.1007/978-3-642-24037-9_57.

[36] Stallings, W. *Network security essentials : applications and standards.* 6. vyd. Pearson education: Hoboken, 2017. ISBN 978-0-13-452733-8.

[37] Stiawan, D., Suryani, M. E., Susanto, Idris, M. Y., Aldalaien, M. N. et al. Ping Flood Attack Pattern Recognition Using a K-Means Algorithm in an Internet of Things (IoT) Network. *IEEE Access* [online]. 2021, sv. 9, s. 116475–116484, [cit. 2022-05-10]. DOI: 10.1109/ACCESS.2021.3105517. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9514914.

[38] Suresh, M. a Anitha, R. Evaluating machine learning algorithms for detecting DDoS attacks. In: Wyld, D. C., Wozniak, M., Chaki, N., Meghanathan, N. a Nagamalai, D., ed. *Advances in Network Security and Applications* [online]. 2011, sv. 196, s. 441–452 [cit. 2022-05-10]. ISBN 9783642225390. Dostupné z: https://doi.org/10.1007/978-3-642-22540-6_42.

[39] Zaroo, P. A Survey of DDoS attacks and some DDoS defense mechanisms Advanced Information Assurance (CS 626). *Advanced Information Assurance (CS 626)* [online]. Leden 2002, [cit. 2022-05-10]. Dostupné z: https://www.semanticscholar.org/paper/A-Survey-of-DDoS-attacks-and-some-DDoS-defense-Zaroo/c96ce29ac4f693a65019c0b3fbbc3726bb2980b8?p2df.

[40] Žižka, J. *Support vector machines (SVM)* [online]. 2005 [cit. 2022-05-10]. Masarykova Univerzita : Studijní materiály předmětu FI:PA034. Dostupné z: https://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf?fakulta=1433;obdobi=3523;kod=PA034.

Příloha A

Obsah přiloženého paměťového média

Adresářová struktura

Přiložené DVD obsahuje tyto položky:

- `/ddos_detection` – adresář obsahující zdrojové kódy pro vytvoření modelu a samotný vytvořený model.
- `/text` – adresář obsahující zdrojové kódy pro vytvoření textu bakalářské práce.
- `flows.dump` – jedná se o uloženou PostgreSQL databázi.
- `LICENSE` – licence.
- `README.txt` – textový soubor obsahující návod k instalaci a použití.
- `requirements.txt` – textový soubor obsahující potřebné knihovny a jejich verze. Tento soubor slouží pro jejich instalaci nástrojem `pip`.
- `xkvasn12.pdf` – text bakalářské práce.

Příloha B

Manuál

V aktuální příloze lze najít popis zprovoznění implementovaného systému. Jedná se o postup instalace použitý autorem. Práce byla vytvořena na systému Ubuntu 20.04.4 LTS s architekturou amd64. Model byl vytvořen pomocí hardwarové akcelerace. Při nepoužití grafické karty se výsledky budou lišit. Práce byla vyvinuta na systému s grafickou kartou: NVIDIA GeForce RTX 3060. Nainstalovaný ovladač se jmenuje nvidia-driver-510. Jedná se o proprietární ovladač.

Je zapotřebí mít aktualizované balíčky, lze to docílit příkazem:

```
sudo apt update
```

Následně je nutno nainstalovat správce balíčku pro jazyk Python `pip`. Konkrétní použitá verze je 20.0.2:

```
sudo apt install pip
```

Dále se nainstalují potřebné balíčky s použitou verzí. V kořenovém adresáři spusťte příkaz:

```
pip install -r requirements.txt
```

Instalace aplikace Jupyter Notebook. Použitá verze je 6.4.7:

```
sudo apt install jupyter-notebook
```

Dále je na vyvíjeném systému databázový systém PostgreSQL 12.9. Instalován následovně:

```
sudo apt install postgresql
```

Aby bylo možné importovat databázi, tak musí být splněny tyto kroky:

- Musí existovat uživatel `postgres` (je vytvořen při instalaci).
- Modul pro vytvoření modelu předpokládá nastavené heslo `postgres` pro tohoto uživatele. Lze docílit:

  Přístup do databázového systému PostgreSQL např. `sudo -u postgres psql`
  Nastavení hesla: `\password postgres`

- Musí existovat databáze s názvem `flows`. Lze vytvořit:

  S přístupem v PostgreSQL např. `CREATE DATABASE flows;`

Potom v adresáři obsahující soubor `flows.dump` spusťte příkaz:

```
pg_restore -j 8 -h localhost -U postgres -d flows flows.dump
```

Nakonec je nutno spustit webovou aplikaci Jupyter Notebook:

```
jupyter-notebook
```

Po spuštění se otevře prohlížeč. Navigujte do složky `ddos_detection`. Ve složce `ddos_detection` otevřete soubor s příponou `.ipynb`. Pokud jsou splněny všechny kroky, tak by se měl po spuštění všech buňěk natrénovat a otestovat implementovaný model.

Příloha C

Rysy použité v modelu

Tabulka C.1: Tabulka všech rysů.

názvy rysů toků	**popis**
ip_family	0 = žádná IP, 1 = IPv4, 2 = IPv6
src_asn / dst_asn	číslo autonomního systému zdroje nebo cíle
src_country / dst_country	původ země zdrojové nebo cílové IP adresy
ip_proto	pole Protokol v IP hlavičce
src_port / dst_port	zdrojový nebo cílový port
application	jméno aplikace
sum_flow	počet sjednocených toků
src_pktcnt / dst_pktcnt	počet paketů od zdroje / cíle
src_bytes / dst_bytes	počet bajtů od zdroje / cíle
src_payload / dst_payload	velikost dat od zdroje / cíle, velikost aplikačních dat
src_app_req_cnt / dst_app_req_cnt	počet aplikační požadavků od zdroje / cíle
time_length	délka toku v sekundách
user_exp_time_avg	průměrná časová délka od poslání prvního SYN paketu po získání dat (v s)
user_exp_time_std	směrodatná odchylka časové délky od poslání prvního SYN paketu po získání dat (v s)
user_exp_time_min	minimální časová délka od poslání prvního SYN paketu po získání dat (v s)
user_exp_time_max	maximální časová délka od poslání prvního SYN paketu po získání dat (v s)
round_trip_time_avg	průměrná délka obousměrného zpoždění (v s)
round_trip_time_std	směrodatná odchylka délky obousměrného zpoždění (v s)
round_trip_time_min	minimální délka obousměrného zpoždění (v s)
round_trip_time_max	maximální délka obousměrného zpoždění (v s)
app_server_response_time_avg	průměrná doba odezvy aplikačního serveru (v s)

app_server_response_time_std	směrodatná odchylka doby odezvy aplikačního serveru (v s)
app_server_response_time_min	minimální doba odezvy aplikačního serveru (v s)
app_server_response_time_max	maximální doba odezvy aplikačního serveru (v s)
src_TTL_min / dst_TTL_min	minimální hodnota TTL zdroje / cíle
src_TTL_max / dst_TTL_max	maximální hodnota TTL zdroje / cíle
src_DSCP_min / dst_DSCP_min	minimální hodnota pole DSCP v IP hlavičce zdroje / cíle
src_DSCP_max / dst_DSCP_max	maximální hodnota pole DSCP v IP hlavičce zdroje / cíle
src_TCP_win_size / dst_TCP_win_size	velikost okénka v TCP hlavičce zdroje / cíle
src_TCP_SYN_pckt_length / dst_TCP_SYN_pckt_length	velikost SYN paketu (v TCP hlavičce) zdroje / cíle
src_TCP_MSS / dst_TCP_MSS	maximální velikost segmentu zdroje / cíle
src_TCP_win_scale / dst_TCP_win_scale	hodnota window scale v TCP volbách (angl. options) zdroje / cíle
src_TCP_IP_cumulative / dst_TCP_IP_cumulative	kumulativní hodnota různých specifických bitů pro TCP/IP zdroje / cíle
src_SYN_pckt_payload_length / dst_SYN_pckt_payload_length	velikost dat SYN paketu (v TCP hlavičce) zdroje / cíle
src_TCP_opts_length / dst_TCP_opts_length	délka TCP voleb (angl options) zdroje / cíle
src_TCP_opts_count / dst_TCP_opts_count	počet TCP voleb (angl. options) zdroje / cíle
src_TCP_opts_padd / dst_TCP_opts_padd	výplň (angl. padding) TCP voleb (angl. options) zdroje / cíle
src_opts_1 / dst_opts_1	příznaky TCP voleb uložené v celém čísle zdroje / cíle
src_opts_2 / dst_opts_2	pokračování příznaků TCP voleb uložené v celém čísle zdroje / cíle
src_CWR / dst_CWR	počet příznaků CWR zdroje / cíle
src_ECE / dst_ECE	počet příznaků ECE zdroje / cíle
src_URG / dst_URG	počet příznaků URG zdroje / cíle
src_ACK / dst_ACK	počet příznaků ACK zdroje / cíle
src_PSH / dst_PSH	počet příznaků PSH zdroje / cíle
src_RST / dst_RST	počet příznaků RST zdroje / cíle
src_SYN / dst_SYN	počet příznaků SYN zdroje / cíle
src_FIN / dst_FIN	počet příznaků FIN zdroje / cíle
src_app_errors / dst_app_errors	chybové hlášení aplikační vrstvy zdroje / cíle
src_prohib_flows / dst_prohib_flows	zakázané, odmítnuté nebo nedosažitelné toky zdroje / cíle
src_reverse_flow / dst_reverse_flow	TCP toky, které jdou od cíle a obsahují pouze SYN a ACK příznaky.

src_unfinished_flows	počet nedokončených toků zdroje
dst_unreplied_flows	počet neodpovězených toků cíle
src_wrong_order_pkts / dst_wrong_order_pkts	počet paketů, které jsou ve špatném pořadí zdroje / cíle
src_retrans_pkts / dst_retrans_pkts	počet znovu poslaných paketů z důvodu ztráty nebo poškození zdroje / cíle
src_bad_checksum_pkts / dst_bad_checksum_pkts	součet paketů se špatným kontrolním součtem zdroje / cíle
src_ECN_pkts / dst_ECN_pkts	součet paketů s nastaveným ECN zdroje / cíle
src_zero_win_pkts / dst_zero_win_pkts	počet intervalů s nastavenou velikostí okna na nula zdroje / cíle