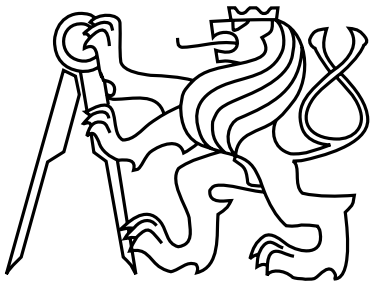
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky



Diplomová práce

Detekce anomálií v sítích

Bc. Jiří Anděra

Vedoucí práce: Ing. Radek Mařík, CSc.

Studijní program: Elektronika a komunikace, Magisterský

Obor: Komunikační sítě a internet

14. 8. 2020

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 14. 8. 2020 .....

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Anděra** Jméno: **Jiří** Osobní číslo: **439545**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

Specializace: **Komunikační sítě a internet**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Detekce anomálií v sítích

Název diplomové práce anglicky:

Anomaly Detection in Networks

Pokyny pro vypracování:

Postup:
1) Vytvořte přehled metod pro detekci anomálií v sítích.
2) Vyberte vhodné metody, které jsou buď implementované či je naimplementujte.
3) Na syntetických datech a vhodně vybraných reálných datech demonstrujte vlastnosti implementovaných metod.
4) Proveďte diskusi získaných výsledků.

Seznam doporučené literatury:

Literaturu dodá vedoucí diplomové práce.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Radek Mařík, CSc., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **28.01.2020** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **30.09.2021**

Ing. Radek Mařík, CSc. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ _____

Datum převzetí zadání Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Poděkování

Rád bych poděkoval vedoucímu mé diplomové práce Ing. Radkovi Maříkovi, CSc. za inspirativní a technické rady a za jeho čas, který mi věnoval. Velký vděk také patřím blízkým za pomoc a trpělivost při mém studiu.

Abstract

This diploma thesis deals with the topic of anomaly detection in telecommunication. In the theoretical background a review of methods is created dealing with anomaly detections in networks and a brief summary of telecommunication attacks. The most suitable method for the problem of anomaly detection is selected. Two algorithms (Isolation Forest and Local Outlier Factor) are selected for data processing. These algorithms are implemented using Python with help of a library scikit-learn. In the practical part, properties of the implemented methods are demonstrated using both synthetic and real datasets. An analysis of detected anomalies has been performed. In the end a discussion of the obtained results is provided.

Keywords: anomaly detection, machine learning, network security, Isolation Forest, Local Outlier Factor, scikit-learn, Python

Abstrakt

Tato diplomová práce se zaobírá problematikou detekce anomálií v sítích. V teoretické části jsou sepsány metody, které využívající statistický přístup, až po metody strojového učení. Následně je vybrána jedna z popsaných metod. V rámci dané metody došlo ke zvolení dvou algoritmů, *Isolation Forest* a *Local Outlier Factor* . Bylo vytvořeno předzpracování dat v jazyce *Python*, dále se přistoupilo k implementaci obou algoritmů za pomocí knihovny *scikit-learn* nejdříve na syntetických a následně na reálných datech. V poslední fázi dochází k rozboru detekovaných síťových anomálií a diskuze nad dosaženými výsledky.

Klíčová slova: detekce anomálií, síťová bezpečnost, strojové učení, Isolation Forest, Local Outlier Factor, scikit-learn, Python

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V dnešní době, kdy exponenciálním způsobem vzrůstá počet informací v každém oboru, jako například bankovnictví, finanční sektor, informace z vesmíru nebo také v síťovém provozu je problémem tyto informace zpracovávat, případně v nich i hledat nějaké vzory. Jedním z řešení tohoto problému je detekce anomálií, které obsahují několik základních algoritmů navržených právě pro nalezení vzorů v datech, které by člověk nebyl schopen nikdy odhalit. Většinou dochází ke zpracování dat, která obsahují i několik stovek dimenzí. Běžný člověk je schopen si představit v grafické reprezentaci dat maximálně tři dimenze, a to kolikrát ještě s problémem. Metody detekce anomálií lze použít na téměř všechna data, která jsou rozumným způsobem předzpracována. Příkladem aplikace algoritmů detekce anomálií může být tzv. AML *Anti-money laundering*, což značí jeden ze způsobů, jakým se bojuje proti praní špinavých peněz v bankovním sektoru.

V telekomunikacích, případně síťovém provozu nastává ten samý problém, existuje stále více a více informací, které je nutné nějakým způsobem alespoň třídit - klasifikovat. Základní klasifikace by měla býti alespoň na úrovni normální provoz a nenormální provoz. S přibývajícím počtem síťových zařízení je tedy stále složitější detekovat nenormální provoz. Je totiž složité se vyznat v ohromném balíku dat, která jsou každou minutou generována. Řešení přináší metody strojového učení, u kterých jsme schopni vytvořit model, který má informaci o tom, jak vypadá normální provoz a vše co neodpovídá normálnímu provozu označí za anomálii. Tento přístup je dobrý především pro možnost rozpoznat i nové útoky na telekomunikační síť. Metody, které se používaly zpočátku tuto možnost neměly, byly sice schopny detekovat známé útoky, ale nebyly schopny říci, že se jedná o dosud nedetekovaný druh útoku. Občasným problémem při aplikaci algoritmů pro detekci anomálií můžou být neoznačkovaná data. Pokud pracujeme s označkovanými daty, které nám jasně říkají, který datový vzorek patří do jaké skupiny, můžeme použít metodu strojového učení s učitelem. Model lze trénovat tak dlouho, dokud se správně nenaučí, jak vypadají jednotlivé druhy datových vzorků. Naopak při práci s neoznačkovanými daty, u kterých nevíme, kam jednotlivá data patří, je možné použít metodu strojového učení, učení bez učitele.

Algoritmy, které využívají strojové učení bez učitele,

mají schopnost rozdělit data do několika skupin, na základě určitých metrik, které jsou definovány pro každý algoritmus. Tento přístup nám umožňuje najít skryté shluky ve velkém množství dat. Tyto shluky se posléze lépe analyzují, což značně usnadňuje práci s daty.

Diplomová práce se věnuje problematice detekce anomálií v sítích. Přístup byl pojat ze širší perspektivy. To znamená, že dochází nejdříve k všeobecnému členění metod detekce anomálií a posléze ke konkrétním případům. Rozebrány jsou i jednotlivé hrozby, které mohou nastat v dnešním síťovém světě, ať už od neoprávněných přístupů do sítě, po hromadné DoS útoky.

Cílem práce je vytvoření přehledu metod pro detekci anomálií v sítích, vybrání vhodných metod a jejich implementace na syntetických a reálných datech.

Kapitola 2

Útoky na telekomunikační síť

Útok na telekomunikační síť lze definovat jako pokus o neautorizovaný přístup se záměrem poškodit, odepřít, znehodnotit nebo zničit informace, popřípadě služby počítačové sítě firmy, organizace, veřejného orgánu nebo domácí sítě. Útočníci docílí těchto záměrů z velké většiny převážně skrze datový tok, který je veden do sítě. Cílem je poškodit integritu a důvěrnost informací, či celkovou dostupnost počítačové sítě. Cílem nemusí být jen velké korporáty, bankovní společnosti nebo celosvětové společnosti, útok je možné vést i na emailový server, webový server nebo databázový server malé firmy a poté využít přístupu k dojednání určité dohody o znovu zprovoznění. Síťový útok může být klasifikován od obtěžujícího emailu směrovaného na normálního uživatele po útok směrovaný na kritická místa páteřní sítě. Prevencí před všemi těmito nežádoucími situacemi je síťová bezpečnost, která začíná autorizací, většinou uživatelským jménem a heslem. Síťová bezpečnost je zajištěna správnými pravidly nastavenými síťovým administrátorem tak, aby zabraňovala a monitorovala neautorizovaný přístup, odepření služeb nebo úpravu systému vhodně nastavenými právy uživatelům [21].

Příkladem útoků na telekomunikační/počítačovou síť může být:

- virus, který je v příloze, případně v odkazu těla emailu,
- internetový červ, jehož funkce je podobná jako červům v reálném světě, a to rozšířit se pokud možno na co nejvíce hostitelů a zneužívat jej ve svůj prospěch,
- neautorizované použití systému,
- odepření konkrétních služeb (například dostupnost webového serveru),
- zjištění chyby v programu a její využití ve prospěch útočníka.

Nezanedbatelným problémem v síťové bezpečnosti je termín, kterému se říká *Social Engineering*. *Social Engineering* je způsob získání přístupu k citlivým datům a jiným údajům skrze člověka. Je využito psychologické manipulace s člověkem (většinou) skrze email za účelem docílení odhalení citlivých informací, například přístupových údajů. Rozrůstající trend využívání telefonního spojení, kde je útočník kontaktován dobrovolně ze strany uživatele za účelem vyřešení technického problému s vlastním počítačem podporuje tento problém. Útočník pomocí naučených manévrů přesvědčí uživatele, že je jeho bankovní účet v ohrožení. Následně je uživatel nevědomky donucen (pasivně, ale i aktivně) sdělit přihlašovací údaje

k jedné z mnoha platebních bran, které jsou využívány všude po světě. Na tento problém navazuje další z možných přístupů útočníka *Masquerading* (maskovaný útok). *Masquerading* je typ útoku ve kterém útočník předstírá, že je jeden z autorizovaných uživatelů systému. Pokud se útočníkovi podaří dostat do systému s těmito ukradenými údaji, je pro něj již snadné ukrást data, nebo šířit svůj vliv dále k zajímavějším a citlivějším informacím [21].

2.1 Všeobecná klasifikace útoků

Klasifikace útoků je důležitá zejména kvůli sjednocenému pohledu na hrozby, které mohou nastat. Díky sjednocenému pohledu na klasifikaci jsme schopni vyměňovat informace o popisu útoku, který nastal například na druhé straně světa. V další části je popsán přístup klasifikace dle Hansman [52].

V práci dle Hansman [52] autoři poukazují na trend neustálého vzrůstajícího počtu útoku, ale přitom výrazný pokles technických znalostí útočníka. Z tohoto faktu plyne, že nástroje na útok se stávají přívětivé i pro běžného uživatele, který nerozumí problematice a může napáchat ještě více škod, než původně zamýšlel. Byl navržen klasifikační systém útoků sestávající se ze 4 dimenzí.

- První dimenze rozlišuje dle dvou následujících možností:
  1. Pokud útok používá jednu cestu, nebo prostředek, pomocí něhož byl získán přístup do počítačové nebo telekomunikační sítě.
  2. Pokud útok nelze zařadit dle bodu výše, je zařazen do následujících devíti kategorií, které jsou součástí první dimenze.
- Druhá dimenze klasifikuje(specifikuje) cíl, na který je veden útok.
- Třetí dimenze klasifikuje zranitelnosti cíle a odhaluje konkrétní zneužití ze strany útočníka.
- Čtvrtá dimenze slouží k vypořádání se s útoky, které mají svoji vlastní datovou část (*payload*).

První dimenze

V první dimenzi byly útoky rozděleny do devíti následujících kategorií:

1. **Virus**: samoreplikační program, které je součástí jiného existujícího programu, tento existující program většinou stáhne uživatel sám, infikuje systém bez vědomí uživatele.
2. **Červ**: samoreplikační program, který je šířen sítí na počítače uživatelů bez jejich vědomí.
3. **Trojan**: název dle trojského koně, jde o program, který uživatel stáhne za určitým účelem, nicméně program vytváří podezřelou aktivitu bez vědomí uživatele.
4. **Přetečení vyrovnávací paměti**: jde o proces, který získá kontrolu nad jiným procesem díky přetečení vyrovnávací paměti, to je možné provést přepsáním hranic pro vyrovnávací paměť.

5. **Odmítnutí služby**: útok, který znemožní oprávněným uživatelům přístup, nebo použití počítačové sítě.
6. **Síťový útok**: útok, který znemožní používání počítačové sítě díky manipulaci se síťovými protokoly.
7. **Fyzický útok**: útok směřovaný na zničení fyzické infrastruktury sítě, nebo počítače.
8. **Odhalení hesla hrubou silou**: je útok, při němž je cílem získat heslo uživatele. Tento útok bývá indikován sérií špatných pokusů o přihlášení.
9. **Shromažďování informací**: je útok, který má za cíl získat co nejvíce informací o zranitelnosti počítačové sítě díky jejímu skenování.

Druhá dimenze

Ve druhé dimenzi jde o zjištění cílů, na které je útok veden. Útok může obsahovat více než jeden cíl. Důležité je cíl útoku co nejvíce specifikovat. Například pokud je útok veden na server A, podstatná pro nás není informace o samotném útoku na server A, ale fakt, že je útok veden na operační systém serveru A, nebo na určitou službu serveru A.
Jedná se o specifické určení cíle, například IIS, operační systém MAC, operační systém UNIX, operační systém LINUX atd. Cíle útoků lze směřovat i na hardware počítače jako například síťová rozhraní, pevný disk, procesor, nebo grafická karta. Útokům se neubrání ani síťová zařízení, jako směrovač, přepínač nebo hub. K oblasti síťových útoků patří také útok na síť samotnou, konkrétně na její protokoly, jako například TCP. Nelze opomenout útoky na aplikační vrstvu, konkrétně třeba webový server nebo uživatelské aplikace, jako například emailový klient.

Třetí dimenze

Ve třetí dimenzi jsou pokryty zranitelnosti systému využity ze strany útočníka. Stejně jako ve druhé dimenzi, může být i zde více zranitelností, na které je daný útok soustředěn. Proto může být aplikováno více záznamů kvůli jednomu útoku. Klasifikování zranitelností systémů je umožněno díky projektu CVE (*Common Vulnerabilities and Exposures*) [37]. CVE je projekt navržen na společné definice zjištěných zranitelností. Jelikož škála zranitelností je velice široká a různorodá. Jakmile je zranitelnost odhalena, je zapsána do databáze CVE a může být v následující verzi opravena. Zápisem do databáze se také nabízí možnost náhledu na zranitelnosti všem uživatelům, což jim umožňuje se efektivně bránit.

Čtvrtá dimenze

Ve čtvrté dimenzi je řešena datová část daných útoků. Rozdílné datové části můžou mít rozdílný efekt na výsledek útoku. Jedná o problém, kde například útok červ má ve své datové části schovaného trojského koně. Tato dimenze byla vytvořena právě pro tyto případy. V první dimenzi je možné klasifikovat určitý typ útoku, ale díky čtvrté dimenzi jsme schopni rozpoznat, co má daný útok v sobě schovaného za další hrozbu. Datové části jsou rozděleny do následujících pěti částí:

- Útok první dimenze: je definován v první dimenzi, neobsahuje žádné další dodatečné datové části
- Poškození informací: skrytá část útoku za účelem poškodit nebo zničit informace v uživatelském systému.

- Zveřejnění informací: jde o využití autentizace oběti a následné využití informací ve svůj prospěch
- Krádež služby
- Převzetí kontroly nad službou nebo celým systémem

2.2 Skenování systému

Skenování, je metoda zjišťování informací o počítačové nebo telekomunikační síti za účelem zjištění zranitelností, které mohou být využity k útokům na daný objekt. Konkrétně se může jednat o odhalování zranitelností různých serverových služeb, jako například Apache, Sendmail, FTP server, MySQL server a jiné. Případně i zranitelnost samotného operačního systému. Dále v práci jsou popsány některé z vybraných metod skenování systému [21].

2.2.1 IPSweep a PortSweep

IPSweep je metoda pro zjišťování IP adres sítě. Nastává, když útočník pošle několik ICMP paketů na různé hosty vně předdefinovaného intervalu a očekává, že alespoň jeden odpoví. Pokud host odpoví, získá útočník jeho IP adresu.

PortSweep je metoda pro zjištění informací o stavu portů na daném hostu. Nastane, když jedna zdrojová IP adresa pošle několik paketů obsahující TCP SYN segmenty na jednu cílovou IP adresu, ale na několik portů zároveň. *IPSweep* a *PortSweep* společně určí aktuálně běžícího hosta a jeho typ služby, tyto informace lze následně využít pro hledání zranitelnosti případně útokům na systém [31].

2.2.2 NMap

NMap (Network Mapper) je softwarový nástroj, který je schopen mnoha věcí, a je vyvíjen otevřeně (*OpenSource*). Konkrétně se jedná o bezpečnostní skener IP adres, portů, firewallů nebo operačních systémů. NMap je nástroj využívaný primárně pro vytvoření přehledné mapy sítě. Nepoužívá předdefinované časy v určité periodě, ale počítá i s různými stavy sítě, například kolísání zpozdění, nebo zahlcená kapacita sítě. NMap je nástroj na velmi vysoké úrovni a ve špatných rukách je obtížné jej detekovat, především díky distribuci vícero zdrojových IP adres a pomalému skenování sítě, které probíhá delší dobu a ne jednorázově [23][21].

2.2.3 SAINT

SAINT (*Security Administrator's Integrated Network Tool*) je softwarový nástroj používaný pro skenování a zjišťování zranitelnosti počítačové sítě. SAINT získává velké množství informací ze síťových služeb, například informace o otisku, FTP, telnet atd. Nejedná se útočný nástroj ve svém základu, nicméně informace, které je schopen získat ze sítě jsou v takovém množství, že můžou pomoci útočníkovi odhalit zranitelnost sítě. SAINT je schopen zjišťovat zranitelnosti z DNS (*Domain Name Server*), RPC (*Remote Procedure Call*) nebo

také nezabezpečených NFS (*Name File Server*), TCP porty jako například HTTP, FTP, Telnet, SMTP, nebo UDP porty [21].

2.3 Privilegované útoky

Privilegované útoky jsou takové útoky, ve kterých útočník zneužívá zjištěných chyb v softwarové výbavě počítače za účelem získat přístup k informacím, které jsou za normálních okolností skryty. To může vést k následné eskalaci a zvýšení zájmu útočníka o specifickou oblast, která byla zpočátku skryta. Tento problém lze rozdělit do dvou kategorií:

- uživatel s nízkými právy může zjistit chybu v systému a následně využije funkce, nebo obsah, který byl původně privilegovaný pouze pro super uživatele,
- neuživatel zneužije systémové chyby a využije funkce, nebo obsah určený pro normálního uživatele.

2.3.1 Přetečení vyrovnávací paměti

Vyrovnávací paměť je část paměti, která slouží pro dočasné uchování dat, přenášených mezi dvěma lokacemi. Používá se také pro vyrovnávání rozdílů přenosových rychlostí. Její přetečení nastane, když proces, nebo program se snaží uložit do vyrovnávací paměti více dat, než se do ní vejde. Vyrovnávací paměti nemají nekonečnou kapacitu. Výsledkem tohoto problému je, že data, který se již nevejdou do paměti musejí jít někam jinam, konkrétně do sousedních vyrovnávacích pamětí. Málo místa v paměti způsobí změnu cesty, který má program nastavenou jako vykonávací. To může vést k získání soukromých informací. Útok vedený na přetečení vyrovnávací paměti podstrčí svůj vlastní kód, který se snaží útočník spustit v přetečené oblasti vyrovnávací paměti (*Overflow*) viz obrázek 2.1. Útočník může přepsat návratovou adresu funkcí tak, aby směrovaly zpět do vyrovnávací paměti a využít tohoto faktu k získání práv super uživatele, poškození souborů uživatele, změně dat, nebo zneužití citlivých informací. Na obrázku 2.1 lze vidět příklad přetečení vyrovnávací paměti (buffer). Jedná se například o přihlašovací údaje, které jsou očekávány jako vstupní data o velikosti 8B, pokud se pošlou data o velikosti 10B, dojde k přetečení zásobníku a zbývající data mohou přepsat následující místa v paměti, kde mohou být uloženy instrukce jiného kódu, což může vyvolat problém [26][21].

Obrázek 2.1: Přetečení vyrovnávací paměti - příklad [26]

Tento typ útoku byl použit útočníky na produkt společnosti Microsoft Outlook a Microsoft Outlook Express v roce 2000. Útočníci využili chybu, kterou v programu nalezli a byli

tak schopni spustit jakýkoliv kód na počítači oběti pouhým posláním emailové zprávy. Uživatelé se nemohli bránit pouhým neotevřením příloh emailu, jelikož Outlook obsahoval výše zmíněnou chybu, která umožňovala útočníkům přepsat část programu tak, aby byl proces aktivován ihned po přijetí emailu do emailové schránky, nebylo tedy nutno email nutně otevírat, případně stahovat přílohu [21].

2.3.2 Problém špatné konfigurace

Všeobecně konfigurace, ať už operačního systému, programu, nebo celého serveru, je jeden ze zásadních problémů, který ohrožuje bezpečnost v IT. Každý bezpečnostní systém je potřeba správně nakonfigurovat systémovým administrátorem tak, aby dodržoval bezpečnostní pravidla, co nejlepším způsobem a zároveň musí poskytovat určitou funkcionalitu uživatelům. S přibývajícím počtem služeb vyžadovaných služeb uživateli, stoupá bezpečnostní riziko útoku na systém. Jedním z problému je samotná špatná konfigurace těchto vychytávek, která směřuje k odhalení zranitelností a využití ze strany útočníka. Dalším problémem jsou nevyužívané služby, které jsou umístěny na serveru a nikdo je delší dobu nespravuje. Takové služby představují bezpečnostní riziko. Je dokázáno, že špatná konfigurace, nebo nedostatečná péče o běžící služby je jedním z největších rizik, které ohrožují kybernetickou bezpečnost.

Jedním ze známých problémů je slovníkový útok, který lze nazvat útok hrubou silou. Pro útok hrubou silou je typické hádání určitých výrazů, například výchozí nastavení přihlašovacích údajů lze snadno odhalit pomocí slovníkového útoku. Slovník obsahuje kombinace nejčastějších výchozích kombinací pro přihlašovací jméno a heslo. Pokud systémový administrátor nezměnil výchozí hodnoty, je snadné prolomit přihlašovací údaje. Slovníkovému útoku se dá v určité míře zamezit maximálním počtem povolených přihlašovacích pokusů, případně zavést časovou mezeru mezi pokusy [21].

2.3.3 Útok typu Man in the Middle

Útok typu MITM (*Man in the Middle*) je všeobecně vzato poslouchání konverzace/komunikace mezi ostatními uživateli, bez jejich vědomí. Konkrétně v oblasti IT bezpečnosti se jedná o akt, kde útočník kontroluje celou konverzaci mezi obětmi tím, že vytvoří nezávislá spojení s každou obětí. Útočník "tlumočí" zprávy mezi obětmi a nutí je si myslet, že vedou konverzaci mezi sebou.
Celý útok probíhá takto: Mějme tři osoby, Jan, Petr a Marek. Jan s Markem si chtějí vyměnit informace, Petr je uprostřed a vše odposlouchává.

1. Jan pošle zprávu: "Ahoj Marku, jaký je tvůj klíč?"
2. Petr zprávu přijme a přepošle ji Markovi.
3. Marek odpoví zpět se zdáním, že odpovídá Janovi. Marek posílá svůj klíč.
4. Petr přijme zprávu od Marka, Markovo klíč si nechá a místo toho pošle Janovi svůj klíč.
5. Jan zašifruje zprávu s Petrovým klíčem se zdáním, že ji zašifroval s klíčem Marka.

6. Petr přijme zprávu od Jana, dešifruje ji svým privátním klíčem, v této fázi ji může upravit a následně ji zašifrovat Markovo klíčem. Zprávu posílá Markovi.

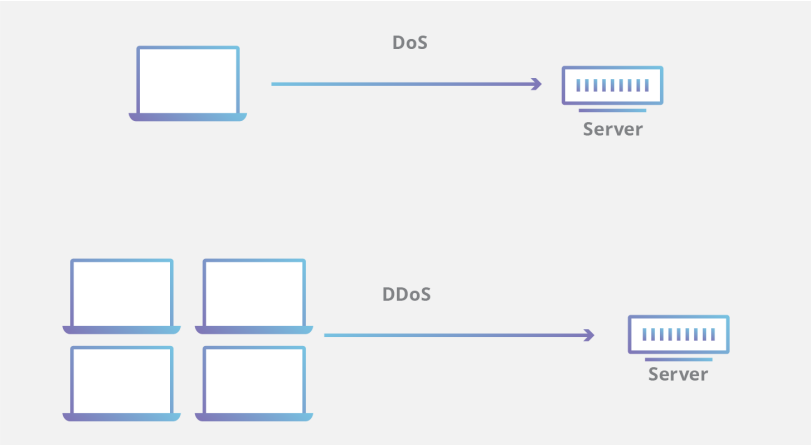
7. Marek si myslí, že mu dorazila zpráva od Jana [21].

2.3.4 Útoky s využitím lidského faktoru

Útoky s využitím lidského faktoru, o kterých již byla řeč v úvodu, se v anglicky psané literatuře nazývají *Social Engineering Attacks*. Lidský faktor hraje obrovskou roli v zabezpečení systému, jelikož člověk není počítač, má na něj vliv nespočet faktorů, které mohou vést k bezpečnostním rizikům. Útoky s využitím lidského faktoru mají za úkol oklamat jinou osobu, ať už agresivním přístupem, psychologickou manipulací nebo jinými osobnostními schopnostmi, za účelem získání přístupu k chráněným informacím. Útoky s využitím lidského faktoru rozdělujeme do dvou kategorií, fyzické a psychologické.
Fyzická kategorie představuje pracoviště, kde útočník může jednoduše okoukat heslo tím, že se postaví do vhodné pozice a bude pozorovat uživatele při jeho zadávání. Možné je také využití lži k přesvědčení, že heslo od uživatele, potřebuje například k opravě systémové chyby. Do fyzické kategorie dále řadíme odpad, ve formě papíru, který je velmi cenným zdrojem informací, zejména pokud firma nakládá s citlivými informacemi nezodpovědně.
Psychologická kategorie obsahuje psychologickou manipulaci a přesvědčování. Jedná se zejména o využití emailového a telefonního spojení. Útočník může zavolat do hlasového centra (call centrum) či na firemní podporu a předstírat, že je zaměstnanec firmy. Vhodnými manipulacemi je schopný útočník vytáhnout z operátora cenné informace o firmě, zabezpečovacím systému nebo uživatelích. Většina zaměstnanců v těchto centrech není obeznámena s bezpečnostními riziky a představují jednu ze zranitelností bezpečnostního systému. Další z možností je využít tzv. *email phishing*, což je pokus útočníka získat citlivé informace přímo od uživatele. V dnešní době se využívají dvě metody pro získání citlivých informací. První metoda je postavená na důvěřivosti uživatele. Druhá metoda je založena vydírání uživatele, kde útočník vytvoří falešné informace o dané osobě, následně na ni vyvine tlak ve formě emailu.
Detekce situací útoků, kde je využíván lidský faktor, je složitá, jelikož se nejedná o žádnou zranitelnost ani chybu v systému. Nejslabším článkem sebelepšího systému zůstane vždy lidský faktor [21].

2.4 Útoky typu DoS

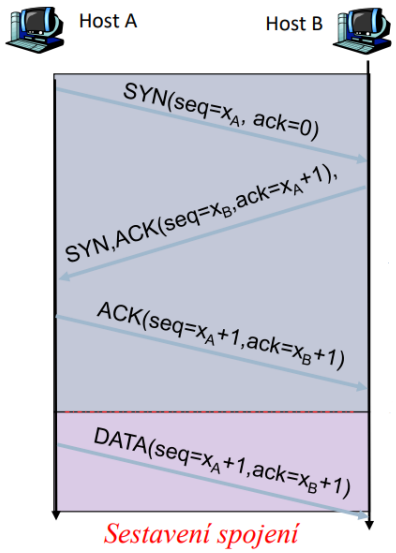
Útoky typu DoS (*Denial of Service*) mají za úkol narušit chod síťových služeb tím, že zcela zahltí a vyčerpají kapacitu zdrojů, ze kterých má být služba poskytována. Konkrétně se může jednat o celou síť, nebo jen jeden článek sítě. Zdroje, které lze vyčerpat jsou: kapacita sítě, kapacita směrovače, výpočetní síla serveru, nebo operačního systému. Během útoku typu DoS dochází ke generování velkého množství toku, který vlastně nedává smysl. Například ohromné množství požadavků na webovou stránku, upravené IP pakety, nedokončené požadavky o TCP spojení atd. Některé škodlivé softwary, které se dostanou do počítače uživatelům v sobě mají zakořeněn také DoS útok. Tento škodlivý software je pak schopný využít zdroje napadeného počítače k produkování DoS útoku. Pokud je takto napadených zařízení více a všechny vytváření DoS útok na jeden cíl, pak je typ útoku nazýván jako Distribuovaný DoS (DDoS - *Distributed Denial of Service*). Obrázek 2.2 zobrazuje rozdíl v útokách DoS a DDoS [17].
Rozsah útoku DDoS může dosahovat až několika gigabitů za sekundu. Jelikož v reálných sítích je většina útoků směrována na výkonné servery, které jsou napojené na optická vlákna, přičemž útočník má omezené výpočetní i kapacitní vlastnosti, jeví se varianta DDoS v tomto ohledu jako velice účinná. Proto zkušený útočník použije při útoku na robustní servery spíše chytře distribuovaný útočný skript na více zařízeních. Těmto zařízením se říká "*zombies*". Většinou jde o zařízení, které je využíváno díky své chybě v konfiguraci, nebo obsahuje bezpečnostní bug [1][21].



Obrázek 2.2: Útok DoS vs útok DDoS [9]

2.4.1 Příklady útoku DoS a DDoS

- **TCP-SYN Flood** je útok, který vytváří doslova záplavu paketů SYN, které jsou využívány pro navázání TCP spojení. Pro navázání TCP spojení je využíván tzv. **třícestný handshake**, který je viděn na obrázku 2.3.



Obrázek 2.3: Třícestný handshake - navázání TCP spojení [15]

Nejprve pošle host A hostu B požadavek na navázání spojení pomocí paketu s příznakem SYN (synchronizace), host B přijme požadavek o navázání spojení a odpoví odpovědí s příznakem SYN, ACK. V tomto bodě zůstává host B (server) ve stavu z půlky otevřen, jelikož čeká na potvrzení ze strany klienta. Jakmile host A pošle hostu B paket s příznakem ACK, je navázáno spojení a mohou být přenášena data. SYN-Flood útok generuje velké množství paketů první části navázání spojení, čili zcela zahlcuje server požadavky, které nejsou nikdy naplněny ze strany klienta. Server poté není schopen navázat žádná další TCP spojení s klienty, kteří to potřebují. SYN-Flood je velice známý typ útoku a moderní servery si s ním dokáží poradit, zejména pokud nealokují prostředky pro spojení hned při každém novém přijatém požadavku o SYN. Ze strany útočníka je možné buďto záměrně neposílat ACK pakety, a nebo mohou využít tzv. IP spoofingu, kde se pošle požadavek na SYN se špatně uvedenou adresou. Server poté odpoví na danou adresu, ale z té se odpovědi nedočká, jelikož žádný požadavek vlastně nevznesla, nebo o něm alespoň neví. Hůře se bojuje proti distribuované verzi tohoto útoku, v kombinaci s IP spoofingem [6].

- **ICMP/UDP Flood Attack:** UDP a ICMP protokoly jsou bezestavové, jejich pakety lze poslat mezi dvěma zařízeními velmi vysokými rychlostmi. Pokud se vygeneruje velké množství paketů UDP nebo ICMP, může dojít k zahlcení serveru, nebo konkrétních portů. Problémem UDP protokolu v tomto ohledu je fakt, že UDP datagramy mohou být poslány ve velké míře téměř na každého hosta v síti bez omezení rychlosti. V kombinaci s IP spoofingem se jedná o velice efektivní útok [21].

- **Ping of Death:** Útočník pošle ohromné množství ICMP požadavků na echo s velkým množstvím informací tak, aby cíl byl zcela zahlcen daty. Vyrovnávací paměť je na hostu přeplněna díky pokusům odpovědět na všechny požadavky, což může vést na celkové selhání systému. Útok typu "ping smrti" spadá do kategorie útoků založených na fragmentaci. Což je proces, kdy se velké IP pakety fragmentují na menší, aby se vešly do MTU (*Maximum transmission unit* - Maximální přenosová jednotka). Pokud dojde k fragmentaci, tak každý IP fragment musí obsahovat informaci o tom, na jaké místo v paketu patří. Tato informace se nachází v IP hlavičce v části fragment offset. Vhodnou úpravou offsetu je útočník schopen vygenerovat IP pakety o velikosti větší, než je maximální povolená 65535B [68].

- **Smurf attack** je distribuovaný útok, ve kterém útočník využívá ICMP protokol. Za normálního provozu pošle host A echo požadavek hostu B. Host B odpoví zpět stejnou zprávou hostu A. Pointa tohoto útoku spočívá v tom, že host A podvrhne adresu hosta C a pošle požadavek na echo broadcastem na určitou síť, nebo část sítě. To způsobí hromadné odpovědi od zasažených stanic, které začnou odpovídat na podvrhlou adresu hosta C. Jako výsledek může dojít k zahlcení linek a zařízeních (směrovačů, přepínačů) vedoucí k hostu C. Dojde k zaplavení provozem odpovědí, což může pracovní stanici výrazně zpomalit, nebo dostat do stavu, kdy s ní není již možné pracovat. Smurf útok lze snadno odhalit pozorováním síťového provozu s velkým počtem echo požadavků na určité místo systému [21].

- **Útok na e-mailový server** je jeden z typických příkladů DoS útoku na aplikační vrstvě, konkrétně vedený na SMTP/POP server. Jeho úkolem je zahltit emailovou frontu serveru tím, že začne posílat velké množství emailů na emailový server. Bránit se tomuto útoku dá nejlépe díky správné konfigurace serveru, například omezením maximálního počtu příchozích emailů pro určitou skupinu uživatelů, nebo na jednoho konkrétního uživatele [21].

- **Útok na webový server** je dnes velice běžná záležitost. Webové stránky jsou velice navštěvovanou věcí na internetu a jejich nefunkčnost může v některých případech znamenat nemalé peněžité ztráty. Útok spočívá v poslání požadavku na webový server (např. Apache), který obsahuje spousty HTTP hlaviček. Server je nucen použít na zpracování těchto hlaviček velké množství procesorového času, což může vést k odepření přístupu na webový server jiným uživatelům. Detekce je možná sečtením počtu HTTP hlaviček, které jsou v požadavku. Pokud je počet na limitem, server zprávu zahodí [21].

2.5 Replikační útoky - Worms attacks

Počítačoví červi jsou druh škodlivého softwaru, který je schopen se sám replikovat a šířit do dalších částí sítě. Využije daný stroj, na kterém právě sídlí, aby zjistil, co je kolem něj a jakým směrem se může dále rozšířit. Na rozdíl od virů, které se potřebují "přichytit"k určitému souboru, aby se mohli šířit, červi existují jako samostatná entita a nepotřebují žádný jiný soubor, který by je šířil dále. Rozdíl mezi viry a červi byl dříve v odlišnosti chování v systému. Virus se téměř vždy snaží modifikovat souboru na cíleném počítači. Červi se o toto

nesnažili, ale naopak spotřebovávali například část z kapacity síťového připojení. Zpočátku byli červi navrženi, aby se pouze šířili, ale s příchodem nových verzí se z nich stali škůdci, kteří ubírají na celkovém výkonu systému.
Jedním z prvních červů vypuštěných do sítě Internetu byl červ od *Robert Tappan Morris*, přezdívaný *Morris worm*. Tento červ nebyl navržen, aby způsobil škody v systému, nýbrž aby odhalil jeho zranitelnosti [21].

Červi se mohou šířit skrze systémové zranitelnosti, emailovou poštou, nebo také přes platformy, jako je například Facebook. Jakmile se otevře odkaz, který je v poště, dojde k automatickému stažení a nainstalování programu na hostitelský počítač, bez vědomí uživatele. Jakmile je program nainstalován, začne pracovat v pozadí bez známek prezence. Dnes se červi zcela změnili a jsou využíváni ke špatným věcem, jako například: mazání a úprava souborů, nebo stáhnutí dalšího škodlivého softwaru [43].

2.5.1 Příklady samoreplikačních útoků

- **Stuxnet** je první samoreplikační program (worm), který byl použit jako kybernetická zbraň. V roce 2010 docházelo k incidentům v počítačové síti ve státě Írán. Jednalo se o velice komplexní samoreplikační program, který byl navržen k sabotování Íránského nukleárního programu, konkrétně PLC (*Programmable Logic Controllers*) zařízení, která byla používána k automatizaci výrobních procesů [43].

- **Blaster** byl červ navržen přímo na míru společnosti Microsoft. Šířil se na počítače v síti v roce 2003, které měly operační systém Windows XP a Windows 2000. Využíval zranitelnosti přetečení vyrovnávací paměti ve službě DCOM RPC. Jakmile se dostal do počítače, měl za úkol spustit SYN Flood útok na portu 80 na webovou stránku společnosti Microsoft. Stránka musela být na nějaký čas shozena úplně, aby se zabránilo dalším škodám, záplata přišla v další verzi [21].

- **Slammer** červ byl vypuštěn do světa také v roce 2003, nakazil přes 75000 strojů během deseti minut. Je možné ho považovat za jednoho z nejrychlejších šíření replikačního programu. Využíval opět chyby v přetečení vyrovnávací paměti softwaru Microsoft SQL server. Replikoval sám sebe tím, že posílal pakety na náhodně vygenerované IP adresy, čímž neustále vyhledával další a další oběti. V tehdejší době to byl celkem veliký problém, jelikož to dokonce zpomalovalo celý Internet. Směrovače nestíhaly odbavovat síťový provoz a docházelo k jejich přetěžování, což mohlo vést k jejich úplnému vypnutí nebo vypovězení provozu. Tento problém vedl k neustálému aktualizování směrovacích tabulek, což mělo za následek ubývající kapacitu telekomunikační sítě [21].

Kapitola 3

Detekce útoků

3.1 Motivace

Telekomunikační, či počítačové sítě jsou ve většině moderních zemích považovány za standard, což značí o jejich robustnosti napříč světem. Svět je dnes propojený jako nikde dříve, člověk může zavolat videohovorem svému známému a během pár sekund s ním být on-line v živém přenosu, nicméně tato možnost, co nám nabízí telekomunikační sítě má i své stinné stránky. Stejně jako se dva kamarádi mohou dovolat z jedné půlky planety na druhou, tak může i zločinec nabourat síť, počítač, či server na druhé straně planety. Se vzrůstající velikostí sítě stoupá i počet útočníků, kriminálníků, co se snaží ukrást cenná data, nebo rovnou peníze z bankovního účtu.
Právě pro tyto problémy, které s sebou přináší rozlehlá síť, se muselo přijít i se způsobem, jak se takovým útokům bránit. Firewall byla jedna z prvních možností, jak zamezit alespoň základním útokům, nicméně tato ochrana není zdaleka dostačující. Kvůli nedostatečné ochraně a přívalem nových hrozeb se zavádí nové systémy pro detekci útoků (*instrusion detection systems* – IDS), které je možné rozdělit do dvou základních kategorií, detekce založená na detekci signatur a detekce založená na detekci anomálií [4].

3.2 Systém pro detekci útoků

Systém pro detekci útoků (IDS) je softwarový nástroj, který by měl doplňovat firewall v jeho funkci a používá se k detekci neautorizovaného přístupu do počítačového systému, či sítě. Systém monitoruje provoz v síti a vyhledává podezřelé aktivity. Pokud podezřelou aktivitu nalezne, hlásí ji síťovému administrátorovi. Systém pro detekci útoků by měl být schopen objevit všechny typy škodlivého provozu, jako například útok proti zranitelným službám, útoky na webové aplikace, nebo neautorizované přihlášení. IDS shromažďuje pakety a analyzuje je dle daného rozlišovacího algoritmu.

3.2.1 Rozdělení systémů pro detekci útoků

1. **Síťový systém pro detekci útoků (*Network Intrusion Detection System* - NIDS)**

   NIDS je umístěn vně sítě na určité strategické místo (například router nebo switch) tak, aby se z tohoto místa dala kontrolovat celá podsíť. Kontroluje tedy veškerý datový provoz, který jde a vychází ze všech zařízení dané sítě. Dnešní sítě disponují rychlostmi v řádu Gbit/s a proto kontrola veškerého provozu snižuje celkovou přenosovou rychlost. Jakmile je zjištěn útok, nebo podezřelé chování síťového provozu, je tato skutečnost oznámena síťovému administrátorovi.
   Jedním z dalších umístění systému pro detekci útoků muže být přímo na zařízení, kde je umístěn firewall, aby se zjistily i útoky na něj. Tento druh systému lze rozdělit na on-line *NIDS* a off-line *NIDS*, kde on-line pracuje v reálném čase a off-line pracuje s uloženým balíkem dat [60][4].

2. **Hostitelský systém pro detekci útoků (*Host intrusion detection systems* - HIDS)**
   HIDS je umístěn na jednom konkrétním zařízení v síti. Kontroluje příchozí a odchozí provoz na síťové kartě. *HIDS* je rozdílný oproti systému *NIDS* především v tom, že vytváří časové snímky obsahu souborů na daném zařízení a zpětně vyhodnocuje jejich rozdílnost. Tímto způsobem lze identifikovat možná narušení jednotlivých souborů na disku [60][4].

3.3 Metody detekce

Metody detekce lze rozdělit do dvou základních kategorií, detekce signatur a detekci anomálií.

Systém pro detekci signatur je založen na známé informaci o průběhu útoků. Z čehož plyne, že lze detekovat pouze známé útoky, jejich rysy, vlastnosti a charakteristiky musejí být někde uchovány, aby mohly být porovnány s přicházejícím síťovým provozem. Jednou z hlavních výhod systému založeném na detekci signatur je relativně malé množství tzv. *false positive* alarmů v porovnání s metodou detekce anomálií. Nevýhoda spočívá v tom, jak již bylo řečeno, že je zapotřebí znát informace o průběhu daného útoku, v opačném případě není možné tento útok detekovat [65].

Systém detekce anomálií řeší problém z jiného úhlu. Staví model normálního chování síťového provozu a detekuje odchylky od daného modelu. Modelu se poskytnou "normální" data, na kterých se systém naučí, jak má správný provoz vypadat. Tomuto procesu se říká také trénovací fáze. Na model se poté aplikuje na datový provoz, ve kterém již může být obsažen nějaký druh útoku. Jelikož model vidí daný útok poprvé, vyhodnotí jej jako anomálii. Výhoda detekce anomálií spočívá v možnosti detekovat i doposud neznámé útoky. Nevýhoda je ve velkém počtu *false positive* alarmů [65].

Kapitola 4

Metody detekce anomálií

4.1 Statistická detekce anomálií

Systém generuje dva profily chování daného subjektu. Profil by neměl obsahovat celé balíky dat, ale pouze statistické údaje, jako například frekvence výskytů jednotlivých protokolů, frekvence výskytů portů, odchylky, střední hodnoty atd. Tyto dva profily jsou generovány pro každý subjekt, jde o profil, který je aktuální a profil, který je uložený k pozdějšímu porovnávání.
Aktuální profil je neustále aktualizován se vzrůstajícím počtem událostí, které nastaly a v průběhu se periodicky počítá skóre anomality. Skóre anomality se zjišťuje z porovnání uloženého profilu a aktuálního profilu. Pokud je skóre anomality vyšší než daná určitá hranice, systém vygeneruje poplašnou zprávu.
Tento systém nepotřebuje znát charakter průběhu útoků, což mu dává možnost reagovat i na doposud neznámé druhy útoků, například skenování portů se dá s tímto druhem systému velice jednoduše odhalit, jelikož tento útok je vyhodnocen s velikým skórem anomality. Nevýhody tohoto systému spočívají ve vysokém počtu falešných alarmů, což je způsobeno malou statistickou přesností, ne všechno chování dokáže být modelováno pouze za použití statistických metod. Další značnou nevýhodou je fakt, že útočník, který má alespoň základní informace o druhu systému dokáže vytrénovat daný systém, a poupravit jeho profil tak, aby abnormální chování provozu sítě bylo považováno za normální. Obtížné je také určení správné hranice, při které dochází k detekování anomálie. Pokud se hranice určí špatně, může docházet k velkému množství detekcí anomálií, které ovšem nejsou útoky, v opačném případě při nastavení hodně "nízké" hranice nemusí systém detekovat probíhající útok. Otázka určení správné hranice je velice složitá.
Příklad systému založeném na statistické detekci anomálie je v práci od Haystack et al. [57], jejichž přístup je dost podobný jako v práci od Denning et al. [16]. Tento systém modeluje systémové parametry jako nezávislé s Gaussovým náhodným rozložením. Pro každý parametr definuje rozsah jeho proměnné, ve kterém je tato proměnná považována za normální. Každý parametr má své skóre anomality, které není závislé na ostatních parametrech. Toto skóre se zvýší pokaždé, jakmile je překročen nastavený rozsah hodnot pro daný parametr. Pokud se toto skóre zvýší několikrát po sobě a přesáhne hranici pro detekci anomálie, dojde ke generování alarmu. Systém také udržuje svou vlastní databázi profilů, ve které má celé skupiny uživatelů, ale také jednotlivce.

Haystack [57] byl navržen k detekci šesti druhů útoků:

- Pokus o vstup do systému (neautorizovaných uživatelů) se detekuje monitorováním přihlašovacích pokusů. Úspěšný pokus o prolomení do systému se považuje, když se útočníkovi podaří přesvědčit systém, že se jedná o autorizovaného uživatele. K detekci dochází díky atypickému chování profilu uživatele.
- Pokus o zamaskování útoku, útočník se snaží přesvědčit systém, že je něco, co není. Maskuje svůj útok za něčím jiným. K detekci dochází opět atypickým chováním profilu nebo porušením bezpečnostních podmínek.
- Pokus o vniknutí do bezpečnostního systému, kde je možné změnit například přihlašovací údaje.
- Únik je přenos citlivých informací ven ze systému, například jejich zobrazení v terminálu, odkud je možné je zachytit.
- Odepření služby (*Denial of Service* - DOS) je detekováno netypickým použitím systémových prostředků.
- Detekování škodlivého softwaru se opět detekuje pomocí abnormálního chování profilu.

Haystack [57] byl ovšem navržen pro práci v off-line režimu. Naopak jedním z prvních systémů, který byl schopný pracovat v reálném čase byl systém *NIDES* (*Next-Generation Intrusion Detection Expert System*) [10] [11]. Systém *NIDES* vychází ze systému *IDES* (*Intrusion Detection Expert System*). Z těchto dvou systémů, je *IDES* čistě statistický, *NIDES* je hybridní, čili využívá statistický přístup a k němu má nějaký dodatek. Oba systémy pracují s množinou proměnných, které jsou získány pomocí statistické analýzy. Tyto získané proměnné slouží následně k porovnání očekávaných hodnot a aktuálních. Pokud aktivita dané proměnné výrazně vybočuje z hodnot, které jsou očekávané, dojde k označení anomálie. Každá proměnná, která je uložená v daném profilu říká, jaké hodnoty jsou v souladu s normálním chováním, jelikož je profil vybudován za chování systému, který je bez anomálií. Hodnoty, které určují rozsah normálního chování se počítají spojením daného měření a dané náhodné veličiny. Vzniká frekvenční distribuce, která je vytvářena v průběhu určité periody a v rámci této periody také pravidelně aktualizována. Frekvenční distribuce je uchovávána ve formě histogramu s pravděpodobnostmi možných rozsahů, které může proměnná nabýt. Systém počítá "vzdálenost", jak moc jsou právě změřená data vzdálená od normálního stavu. Toho je docíleno kombinací proměnných z jednotlivých měření, ale také vzájemné korelace v měření.

SPADE [54] (*Statistical Packet Anomaly Detection Engine*) je jedním z dalších příkladů detekce anomálie za pomoci statistického přístupu, který je přímo cílený na telekomunikační sítě. Tento systém byl jedním z prvních, který používal pro detekci útoku skenování portů skóre anomality, namísto přístupu, kde se počítalo x pokusů za y čas.
SPADE používá přístup, čím méně byl daný paket viděn na určitém portu, tím větší má skóre anomality, čili je velká šance, že se jedná o útok. SPADE má ovšem veliký stupeň falešných alarmů, je to způsobeno tím, že veškeré pakety, které nikdy neviděl vyhodnocuje jako útok. Skóre anomality:

$$A(x) = -\log_{10} P(x)$$

je vypočítáno za předpokladu, že je známa aktuální distribuce normálního provozu na síti na dané porty a cíle. P(x) je pravděpodobnost, že v normálním provozu by daný paket směroval na nějaký specifický port. Potom má daný paket vypočítané skóre anomality dle vzorce výše [54].

4.2 Detekce anomálií založena na strojovém učení

Strojové učení lze definovat jako schopnost systému se učit a vylepšovat své schopnosti v určité oblasti, do které byl navržen. Cílem strojového učení je zcela to samé jako ve statistickém přístupu, nicméně se zde neklade důraz na proces porozumění generování dat. Strojové učení klade důraz na celý proces učení se v závislosti na předchozích výsledcích. Systém založený na strojovém učení má schopnost se přizpůsobovat změnám podmínek, ve kterých pracuje, aby byl neustále schopný dosahovat požadovaných výsledků. V posledních letech byly učící se techniky široce používány v detekci anomálií, jelikož jsou schopny určit normální chování například sítě a na základě vybudovaného modelu rozeznávat příchozí hrozby. Učící se modely dělíme na učení s *učitelem* a *bez učitele* [4][21].

Strojové učení s učitelem sestavuje profil normálního chování během fáze učení, během níž jsou mu k dispozici označená data. V trénovací fázi se systém na základě označkovaných dat snaží sestavit proces, který je schopen přiřadit značky na základě vstupních dat v co největším souladu se značkami učitele.

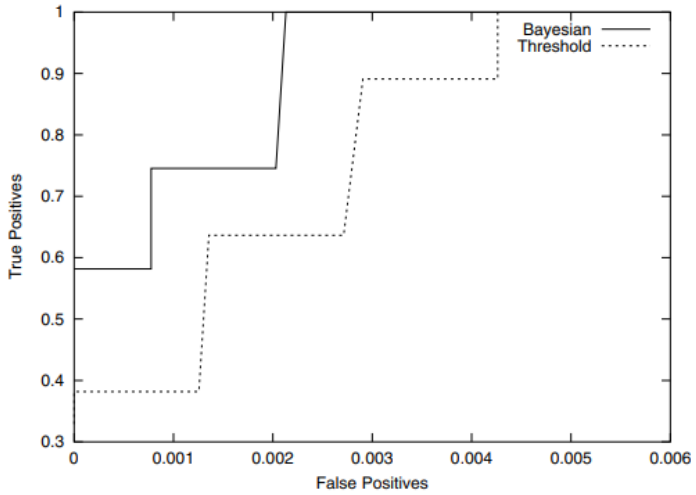
Naopak strojové učení bez učitele se snaží odhalit útoky bez jakékoliv informace o tom, co je v konceptu normality nebo znalosti o průběhu útoku. Označkovaná data jsou jedním z nevýhod strojového učení s učitelem, jelikož ne vždy máme taková data k dispozici. Časová náročnost celého procesu stoupá především s počtem dimenzí dat, se kterými se pracuje [21].

4.2.1 Bayesovská síť

Bayesovská síť je grafický model, který reprezentuje pravděpodobnostní vztahy mezi proměnnými, což lze využít především, pokud chybí nějaká data. Tato síť dokáže také zachytit kauzální vztahy, což znamená, že dokáže predikovat následky události, která nastane. Proto se Bayesovské sítě používají tam, kde je potřeba kombinovat data a předchozí znalosti.
Příkladem práce s Bayesovskou síti může být publikace od Valdes et al. [2], kde byl vytvořen detekční systém využívající naivní Bayesovskou síť, což je síť, která má pouze dvě vrstvy a předpokládá nezávislost mezi uzly. Jedná se o jeden hlavní uzel (*root uzel*), který má několik podřízených uzlů (*child nodes*). Všechny *child nodes* mají pouze jeden *parent node*, jinak mezi nimi není žádná jiná spojitost. Práce od Valdes et al. má schopnost detekovat distribuované útoky, tyto útoky je složité odhalit, protože jsou rozprostřeny do více spojení, což je dělá špatně detekovatelné. Jedna z nevýhod tohoto systému je, že schopnost rozeznávat útoky je téměř stejná jako u systémů založených na určité hranici. Problémem je také, že *child nodes* nemají mezi sebou žádné vazby a jejich výstup ovlivní pouze pravděpodobnost uzlu jejich rodiče (*root node*) [13][4].

Bayesovská síť se nepoužívá jen přímo v daném modelu systému, ale je také využívána jako určitý potlačovač falešných alarmů, kterých vzniká opravdu velké množství. Toho je

využito v práci od Kruegel et al.[5], kde byla místo porovnání součtu výsledků s určitou hranicí právě Bayesovská síť.



Obrázek 4.1: Porovnání úspěšnosti určení správného a nesprávného výsledku [5]

Na obrázku 4.1 můžeme vidět, jak se zlepšila úspěšnost určení správného výsledku. Jde o porovnání v práci [5], kde bylo docíleno dvojnásobného zlepšení v případě použití Bayesovské sítě namísto tradičního přístupu s určenou hranicí. Ideální klasifikátor by měl mít *true positive* na hodnotě 1 a *false positive* na hodnotě 0. V obou případech můžeme vidět, že při absolutně správném určení všech útoků, které nastaly, se v systému i nadále vyskytují případy, kdy byl ohlášen alarm, ale o útok nešlo. Nejvýznamnější rozdíl v obou přístupech je vidět při stoprocentním odhalení hrozeb.

4.2.2 Analýza hlavních komponent

Analýza hlavních komponent je transformace, která se používá ke snížení dimenze dat, při pokud možno co nejmenší ztrátě informace. Data, která jsou používána pro detekci anomálií jsou velice paměťově náročná a obsahují mnoho dimenzí. Čím složitější síť, tím více informací, které je potřeba zpracovat [4].
K vyřešení problému více dimenzí se používá analýza hlavních komponent. Jedná se o techniku, která z x náhodných korelovaných proměnných vytvoří $y < x$ nekorelovaných proměnných. Nekorelované proměnné, které byly vytvořeny pomocí transformace jsou lineárními kombinacemi originálních proměnných a mohou být tedy použity jako určité formy originálních dat.
Obvykle první složka s největším rozptylem, která vznikne transformací, je lineární kombinací původních hodnot. Jde o projekci ve směru takovém, ve kterém je největší rozptyl. Ve většině datových balících stačí ponechat prvních pár komponent a zbytek zahodit s minimální ztrátou informace [4].
Příkladem práce je Shyu et al. [34], kde studovali robustnost PCA v detekci extrémních (nepravidelných, nezapadajících hodnot) a následné aplikaci v detekci anomálií.

V práci počítali Mahalanobisonovu vzdálenost jako součet kvadrátů normovaných hlavních komponent pro každé pozorování. Svoji metodu použili na data *KDD CUP99* a výsledkem jejich práce byl lepší detekční poměr, než většina známých metod, jako například LOF (*Local Outlier Factor*), KNN (*K - Nearest Neighbors*) [4] .

4.2.3 Markovovy modely

Markovovy modely jsou velmi často využívány v problematice detekce anomálií. V práci Ye etl al. [42] používali právě tyto modely pro reprezentaci profilu událostí v počítačovém a síťovém systému za běžného chodu. Markovův model normálního chování systému je generován z historických dat běžných aktivit. Pozorované aktivity jsou analyzovány a jsou z nich vyvozeny pravděpodobnosti přechodů z a do daných stavů Markovova modelu. Čím menší je pravděpodobnost daného přechodu, který byl vytvořen z normálního chování systému, tím větší je šance, že se jedná o anomálii [42] [4].
Skrytý Markovův model se také široce používá v odvětví detekce anomálií. Jde o techniku, která modeluje systém jako Markovův proces s neznámými(skrytými) parametry. Snahou je dosáhnout odhalení těchto skrytých parametrů z parametrů, které je možné pozorovat. Na rozdíl od běžného Markovova modelu, kde jsou známe pravděpodobnosti přechodu jednotlivých stavů a stav systému je možné pozorovat ihned. Ve skrytém Markovově modelu jsou jediným pozorovaným parametrem výstupní proměnné systému, které jsou ovlivněny stavem systému, který je skryt. Skryté stavy Markovova modelu reprezentují nepozorovatelné podmínky sytému [4].
Skrytý Markovův model lze použít pro modelování chování systému, je nutné odhadnout jeho parametry. Toho je docíleno sekvencemi událostí, které nastaly za normálního chodu systému. Tyto sekvence událostí jsou použity v trénovací fázi. Pro odhad parametrů se využívají sekvence událostí v kombinaci s použitím algoritmu EM (*expectation-maximization*). Jakmile je skrytý Markovův model vytrénován a podroben testovacím datům, je možné určit pravděpodobnostní hranici pro detekci anomálií [4].
Pokud chceme použít skrytý Markovův model pro detekci anomálií, je možné řešit tři základní problémy:

1. *Evaluation problem*: je problém určení pravděpodobnosti, že nastane daná sekvence pozorování.

2. *Learning problem*: je budování modelu z trénovacích dat, který správně popisuje chování systému.

3. *Decoding problem*: je určení s největší pravděpodobností množinu skrytých stavů, které vedly k danému pozorování.

Příkladem práce, kde je využit skrytý Markovovo model může být například Yeung et al. [12]

4.2.4 Příklady metod strojového učení

Metody založené na strojovém učení jako například *PHAD* (Packet Header Anomaly Detector) [39], *LERAD* (Learning Rules for Anomaly Detection) [40] nebo *ALAD* (Application

Layer Anomaly Detector) [41] řeší problém detekce anomálií pomocí zkoumání hlaviček paketů. Jde o systematickou aplikaci učících se technik k získání profilu normálního chování pro protokoly na **různých** vrstvách. V těchto metodách je použit časový model, ve kterém je pravděpodobnost události určena pomocí času, kdy nastala naposledy. Pro každou vlastnost se určí rozsah povolených hodnot, hodnoty, které jsou mimo interval se označí jako anomálie. *PHAD*, *ALAD* a *LERAD* monitoruje různé vlastnosti síťového provozu. *PHAD* pracuje s atributy z hlaviček na linkové, síťové a transportní vrstvě. *ALAD* modeluje příchozí TCP požadavky, zdrojové a cílové IP adresy, TCP příznaky a první slovo z datové části paketu. V závislosti na každé vlastnosti vytváří oddělené modely pro každé navázané spojení, nebo pro kombinaci zdrojová IP / port. Přestože autoři používají pouze data z hlaviček paketů, jedná se stále o multidimenzionální data. Proto využívají rozkladu dat na data s jednou dimenzí a počítají součet váhovaných výsledků z každé dimenze. Tento přístup sice dělá celý postup mnohem méně výpočetně náročný, ale ztrácejí se vazby mezi dimenzemi a je tedy náročné detekovat specifické typy útoků [4].

4.3 Detekce anomálií založena na dolování dat

Metoda dolování dat je technika, která dokáže z balíku dat odhalit vzory, nepozorovatelná spojení mezi proměnnými, statistické struktury nebo události v datech, která nejsou na první pohled zřejmá [50].
Dolování dat pomáhá v procesu detekce anomálií přidáním nové roviny, na kterou se dá zaměřit pozorování.

4.3.1 Detekce anomálií založena na klasifikaci

Systém založeny na klasifikaci dat rozděluje data na normální a abnormální na základě pravidel a vzorů.
Klasifikační proces obsahuje tyto kroky:

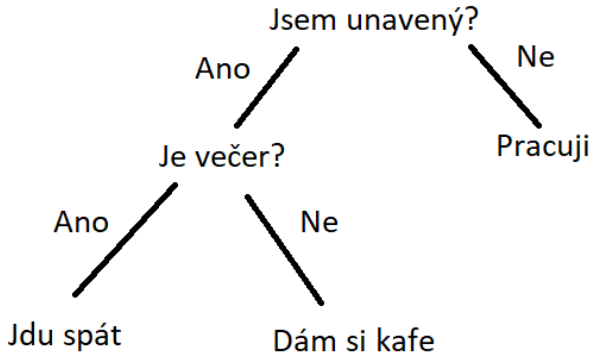
1. Identifikace rozdělovacích skupin z trénovacích dat (jak rozdělit data).
2. Identifikace vlastností pro klasifikaci.
3. Naučení modelu z trénovacích dat.
4. Použití naučeného modelu ke klasifikaci neznámých dat.

Algoritmy pro určení pravidel rozdělování dat
Pomocí těchto algoritmů jsme schopni vyhledávat vztahy mezi daty. Jde o aplikaci asociačních pravidel ve tvaru, pokud se objevila událost X, pak se pravděpodobně objeví i událost Y. X a Y mohou tedy být popsány jako množina párů hodnot, kde se hledá množina X a Y taková, že X implikuje Y. V doméně klasifikační se většinou zafixuje Y a hledá se množina X, která predikuje správně Y.
Učení s učitelem obvykle odvodí pravidla spojená pouze s jednou vlastností, naopak odvození obecných pravidel je doménou učení bez učitele a lze je použít na kteroukoliv vlastnost dat. Výhoda použití pravidel je, že jsou jednoduchá, intuitivní a nestrukturovaná. Nevýhoda

pravidel spočívá ve složité údržbě.

Algoritmů pro odvozování pravidel je celá řada. Jedním z přístupů je použití **rozhodovacích stromů**.

Rozhodovací stromy fungují na principu pokládání otázek na základě vlastností dat. Dle odpovědí se data rozdělují do dalších a dalších větví stromu. Otázky musejí být co nejjednoznačnější. Rozhodovací strom se skládá ze tří částí: uzel, listy, větve. Uzly jsou jednotlivé prvky (otázky), první otázce ve stromu se říká kořenová. List je poslední otázka, po které již nenásleduje žádná další otázka. Větev spojuje uzly a listy s uzly.

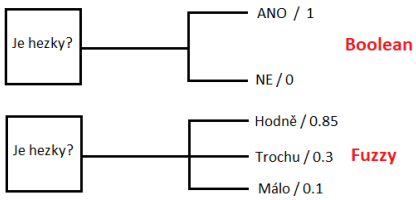


Obrázek 4.2: Rozhodovací strom [38]

Na obrázku 4.2 můžeme vidět příklad rozhodovacího stromu, v němž se rozhoduje další činnost na základě otázek. Datový bod je možné na základě otázek, počínaje prvotní otázkou a postupným procházením celého stromu klasifikovat do jedné ze skupin. Příkladem nemusí být pouze odvozování pravidel pro detekci anomálií, ale i třeba otázka "Co je to na obrázku?" Existují i algoritmy jako RIPPER[69] nebo C4.5[30], které určují pravidla přímo z dat přístupem rozděl a panuj. Jakmile se algoritmus přestane učit dojde k promazání nějakých pravidel, aby došlo ke zvýšení klasifikační přesnosti. RIPPER byl již úspěšně použit v případech detekce anomálií založeném na dolování dat. Generovaná pravidla algoritmem RIPPER jsou jednoduchá a snadno ověřitelná.

Fuzzy logika

Termín fuzzy značí věci, které nejsou čistě jasné, jako například pravda/lež, studená/-teplá. V reálném světě je spousta situací, kde nemůžeme jednoznačně vytvořit tvrzení o absolutní pravdě. Ve fuzzy logice ohodnocujeme výroky mírou pravdivosti. Fuzzy logika je schopná matematicky vyjádřit pojmy jako „trochu“, „dost“ nebo „hodně“. Jde o vyjádření příslušnosti k množině. Příklad porovnání Fuzzy logiky s Boolean logikou lze vidět na obrázku 4.3 [62].
Fuzzy logika se široce používá v oblasti počítačů a síťové bezpečnosti již od roku 1990. Její využití se opírá o fakt, že na kvantitativní parametry, které jsou použity v detekci anomálií lze pohlížet jako na fuzzy parametry. Samotný pojem bezpečnosti lze považovat za fuzzy.



Obrázek 4.3: Boolean a fuzzy logika - *vlastní zpracování*

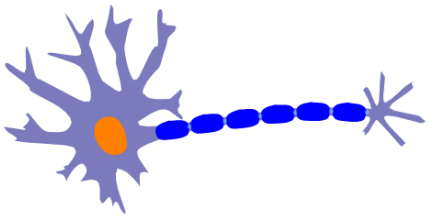
Pomocí fuzzy logiky jsme tedy schopni vytvořit plynulý přechod mezi daty, která jsou normální a abnormální [25][59].
Systém FIRE [27] (*Fuzzy Intrusion Recognition Engine*) používá jednoduchou metodu dolování dat ke zpracování síťového provozu a generování fuzzy množin pro každou pozorovanou vlastnost. Fuzzy množiny jsou poté použity k definování fuzzy pravidel pro detekování útoků. FIRE nesestavuje žádný model reprezentující aktuální stav systému, ale místo toho se spoléhá na specifická fuzzy pravidla. Bylo zjištěno, že tato metoda je velice efektivní proti skenování portů a různým metodám zjišťujícím informace o síti.

Genetické algoritmy

Je technika používaná k nalezení přibližného řešení a optimalizaci vyhledávacích problémů. Používá se široce v odvětví rozlišení abnormálního a normálního chování sítě. Jedná se o flexibilní a rozsáhlou vyhledávací techniku, která konverguje k řešení z několika různých směrů a je založena na pravděpodobnostních pravidlech. Genetické algoritmy lze použít k odvození klasifikačních pravidel, nebo k určení vhodných vlastností, které lze sledovat v síťovém provozu [67].

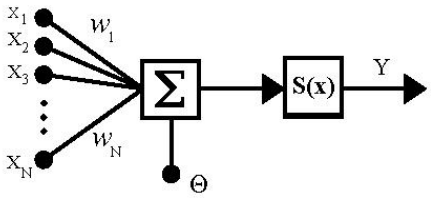
Neuronové sítě

Neuronová síť je výpočetní model umělé inteligence. Vychází z biologických struktur. Umělé neurony, kterou jsou používány se inspirovaly neurony biologickými. Neuron je buňka, která dokáže generovat signál na svém výstupu, v závislosti na svém vstupu [51].



Obrázek 4.4: Neuron

Na obrázku 4.4 lze vidět neuron, vlevo jsou dendrity (vstupy), jádro a terminály. Tomuto obrázku odpovídá z technického pohledu model neuronu, který je viděn na obrázku 4.5. Jádro rozhoduje, zda bude neuron aktivován, či ne. Pomocí terminálu je připojen neuron k dalším neuronům. Neurony jsou vzájemně propojené a předávají si signály.
Neuron lze modelovat, modelů existuje celá řada. Nejjednodušším modelem je **perceptron**, který se sestává pouze z jednoho neuronu a používá skokovou funkci pro aktivaci (signál se buď přenese nebo nepřenese) [51].



Obrázek 4.5: Model neuronu [28]

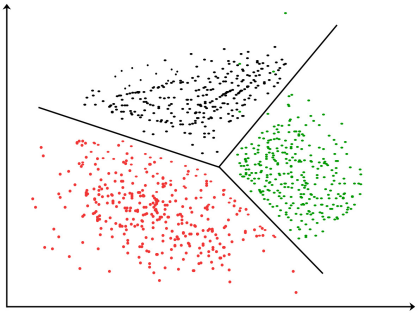
Perceptron se skládá ze:

- Vstupů neuronu,
- biasu (práh), což je prahová hodnota aktivace neuronu,
- váhy spojení (synaptické váhy), která vyjadřuje uložení zkušeností do neuronu, čím vyšší hodnota, tím je daný vstup důležitější,
- sumy,
- přenosové funkce (aktivační funkce),
- a výstup neuronu.

V doméně detekce anomálií se používají systémy založené na neuronových sítích přímo na hostitelském systému, soustředí se na odhalení odchylek v chování programu. Neuronové sítě se dají také použít k predikování chování uživatelů nebo daemonů systému. Tolerance těchto systému k neurčitým informacím a nepřesným datům je jedna z výhod, mají schopnost určovat řešení z dat bez znalostí o datech samotných. Nicméně mají i své nevýhody, v některých případech se nenalezne dostačující řešení, buďto kvůli málo dostačujícím datům, nebo špatně nastavené učící se funkci. Problémem je i výpočetně náročná a pomalá trénovací fáze [51].

4.3.2 Shlukování a detekce abnormalit

Shlukování (*clustering*) je technika strojového učení bez učitele, která slouží pro rozdělení množiny objektů do skupin (*clusterů*), kde objekty ve stejné skupině mají společnou nejméně jednu z vlastností [3]. Jde o nalezení vzorů v neoznačkovaných datech s mnoha dimenzemi. Kde dimenze znamená počet vlastností (atribut) dat. Výhody těchto techniky je schopnost učit se a detekovat anomálie v datech, bez potřeby popisu daného útoku, jinými slovy nepotřebujeme znát povahu, průběh, důvod útoku ani žádné další informace. Výhoda spočívá také v menším množství dat, které jsme nuceni poskytnout v učící se fázi oproti jiným metodám detekce anomálií. Pokud jde o detekci anomálie, tak z pohledu shlukovacího algoritmu je anomálie datový bod, který není ve společné skupině s ostatními daty. Příklad rozdělení datových bodů do skupin viz. obrázek 4.6 [4].



Obrázek 4.6: Příklad rozdělení datových bodů do skupin - shluková analýza [61]

Jednu z hlavních rolí ve shlukové analýze hraje vzdálenost. Nejvíce populární je Euklidovská vzdálenost. Tato vzdálenost není bohužel vyhovující, jelikož každá vlastnost, kterou data mají, přispívá stejným dílem. Problémem je také rozdílná variabilita (rozptyl hodnot) u vlastností daných dat. Řešením tohoto problému je Mahalanobisova vzdálenost, která bere v úvahu korelaci mezi parametry, dává dimenzím různou váhu dle jejich variability a je nezávislá na rozsahu hodnot parametrů. Lze s ní vypočítat vzdálenost mezi dvěma body v systému souřadnic, jehož osy na sebe nemusí být kolmé [4][29].

Shlukování lze provést dvěma způsoby, v prvním přístupu je model pro detekci anomálií vytrénován na neoznačkovaných datech, která obsahují jak normální provoz, ale také útoky. Myšlenka je taková, že data, která mohou být považována za útočná zabírají pouze malé procento celkového datového toku. Pokud tato myšlenka přetrvá, potom může být shluková analýza použita v detekci anomálií. Ve druhém přístupu se jedná o rozdělení dat na "špatná"a "dobrá", data jsou trénována na datech, ve kterých není nic podezřelého [4][53].

Ve studii od Ramaswamy et al. [53] byl představen přístup detekce anomálie na základě vzdálenosti bodu k jeho k-nejbližším sousedům. Ohodnocuje se každý bod P na bázi Euklidovské vzdálenosti k jeho k-nejbližším sousedům, pokud je v sousedství vyšetřovaného bodu P méně než p předdefinovaných bodů, je tomuto bodu přiřazeno vysoké skóre. Prvních x bodů v žebříčku se prohlásí za anomálie. Algoritmus pro hledání k-nejbližších sousedů rozděluje datové body do skupin dle frekventovanosti jejich sousedů. Čili pro daný datový bod P se určí parametr $d_k(P)$, který zaznamenává vzdálenost z bodu P k jeho k-nejbližším sousedům, parametr $d_k(P)$ lze nazvat jako stupeň anomality. Problémem je vhodné určení parametru k.

Jedním ze zajímavých přístupu je LOF (*Local Outlier Factor*), který byl využit v práci MINDS (*The Minnesota Intrusion Detection System*) [32]. V této práci přiřazují LOF skóre každému datovému bodu. Algoritmus LOF bude popsán více v další kapitole.

Kapitola 5

Vyhodnocení modelů

Pokud jsme dospěli do fáze, kdy máme připravený a vytrénovaný model, je důležité ho správně ohodnotit. To je možné zejména, pokud máme označkovaná data, jinak řečeno, známe informaci o tom, do jaké třídy patří vyšetřovaný datový bod.

V další části jsou popsány základní pojmy a metriky právě pro vyhodnocení vytvořených modelů.

5.1 Základní pojmy

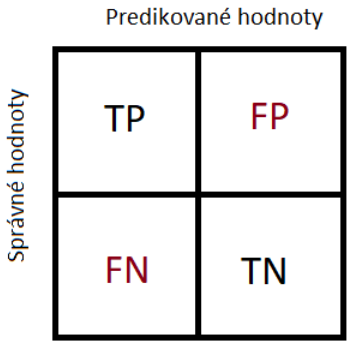
True Positive (TP) je označení pro případ, kdy došlo ke správnému odhalení (predikování) pozitivního datového bodu. Jinými slovy, datový bod patřil do kategorie anomální a byl tak modelem odhalen.

True Negative (TN) je označení pro případ, kdy došlo ke správnému odhalení (predikování) normálního datového bodu. Datový bod patřil do kategorie "normální"a byl tak modelem klasifikován.

False Positive (FP) je označení nesprávně odhaleného datového bodu. Datový bod je ve svém původním označení "normální", ale modelem byl označen za anomálii.

False Negative (FN) je označení nesprávně odhalené anomálie, respektive její neodhalení. Datový bod byl označen za "normální", ale ve skutečnosti se jedná o abnormální bod [22].

Tyto čtyři základní pojmy formují tzv. *Confusion Matrix* (česky matice záměn), kterou lze vidět na obrázku 5.1. Matice záměn se čte po řádcích, kde každý řádek dává dohromady v součtu obou sloupců hodnotu 100 %. Ve sloupcích jsou predikované hodnoty, které model označil. Zbytek odpovídá již výše zmíněným pojmům. Záměrem každého vytvořeného modelu je dosáhnutí co nejnižších hodnot parametrů *False Positive* a *False Negative*. Model s nulovými hodnotami **FP** a **FN** odhalil všechny datové body naprosto správně [22].



Obrázek 5.1: *Confusion Matrix* - matice záměn, vlastní zpracování

5.2 Metriky

Přesnost (*Accuracy*) modelu nám říká, jak moc se vytvořený model mýlil. Přesnost se vypočítá jako podíl správných predikcí a celkového počtu predikcí, viz. obrázek 5.2. Nicméně i relativně pěkný výsledek přesnosti (například 95 % bylo predikováno správně) nemusí znamenat úspěch. Je nutné si uvědomit, že přesnost se počítá přes všechny predikce, a pokud je v datech jen malé množství anomálií, může výsledek velice klamat. Dejme tomu, že datový vzorek obsahuje například 1000 záznamů, v nichž je 10 anomálií a 990 normálních dat. Přesnost může vyjít velice blízká hodnotě 100 %, ale výsledek nám neříká nic o počtu správně detekovaných anomálií, to může mít fatální dopad například pokud je model využíván ve zdravotnictví. Přesnost je spíše všeobecná metrika a je nutné se na ni dívat z širší perspektivy [22].

$$\text{Přesnost} = \frac{TN + TP}{TN + TP + FN + FP}$$

Obrázek 5.2: Způsob výpočtu přesnosti modelu (*accuracy*)

Preciznost (*Precision*) se snaží zodpovědět na otázku, jaký podíl pozitivně označených dat modelem bylo opravdu správně, viz. obrázek 5.3. Preciznost je opět jen jiný pohled na výsledky, pokud bychom se dívali z pohledu preciznosti, unikly by nám zásadní informace o skutečném stavu anomálií [22].

$$\text{Preciznost} = \frac{TP}{TP + FP}$$

Obrázek 5.3: Způsob výpočtu přesnosti modelu (*accuracy*)

Recall je zásadní parametr pro detekci anomálií, říká nám, jaká část pozitivně označkovaných dat byla identifikována správně. Výpočet parametru *Recall* viz. obrázek 5.6. Model,

který identifikoval veškeré anomálie správně, nabývá hodnoty *Recall* roven jedné (neprodukuje žádné FN) [22].

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Obrázek 5.4: Způsob výpočtu přesnosti modelu (*accuracy*)

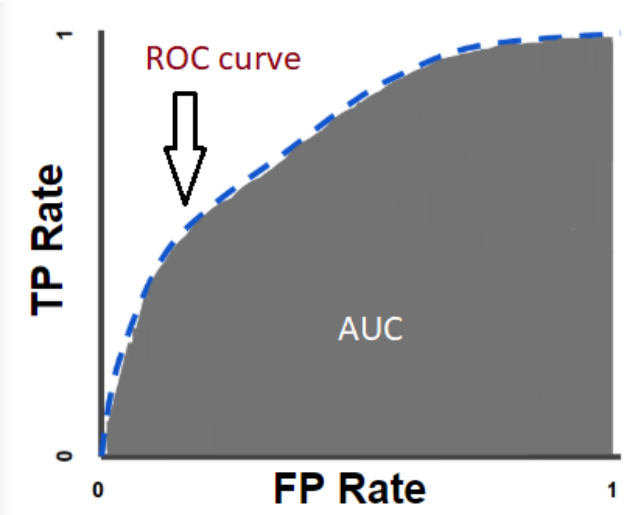
5.3 ROC křivka

ROC (*Receiver Operating characteristic Curve*) [55] křivka je vynesena v grafu a ukazuje nám, jak si daný model vede přes všechny klasifikační hranice, které jsme modelu určili. Graf obsahuje na ose y parametr *Recall* (v případě ROC křivky nazvaný ***TPR - True Positive Rate***) a na ose x parametr zvaný **FPR** (*False Positive Rate*), který se vypočítá jako:

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

Obrázek 5.5: *False Positive Rate*

Parametr **AUC** značí *Area under the ROC Curve*, jedná se o změření oblasti pod ROC křivkou. AUC nám říká, jak moc je model schopný rozlišit mezi jednotlivými třídami. Čím vyšší je hodnota AUC, tím větší je pravděpodobnost, že model správně rozliší jednotlivé třídy tak, jak mají být [55].



Obrázek 5.6: Zobrazení křivky ROC a parametru AUC [22]

Kapitola 6

Vybrané metody

6.1 Isolation Forest

6.1.1 Úvod

Isolation Forest je pojmenování algoritmu, který byl představen v práci od Liu et al. [18]. Jedná se o algoritmus, který spadá do sekce strojové učení bez učitele. Algoritmus *Isolation Forest* je odlišný od přístupů a metod, o které se dříve opíralo celé téma detekce anomálií. To je již popsaný přístup vytvoření určitého modelu, ve kterém se definuje profil normálních instancí a data, která neodpovídají vygenerovanému profilu normálního chování, jsou označena jako anomálie. *Isolation Forest* funguje na principu izolace anomálií, čehož je dosaženo především kvůli dvěma vlastnostem anomálií:

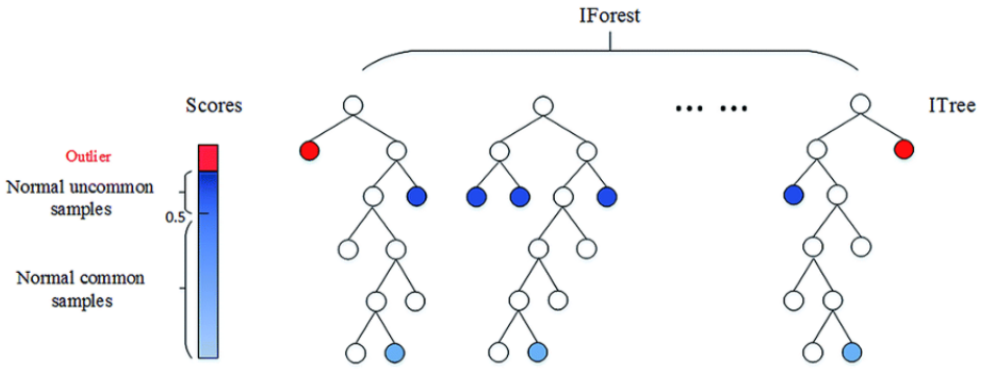
1. není jich moc,
2. jsou odlišné v určité vlastnosti nebo vlastnostech od normálních dat.

Anomálie je snažší oddělit od normálních dat, pokud jsou splněny výše zmíněné dvě podmínky, především, pokud je aplikován algoritmus *Isolation Forest*. Izolace anomálií je dosaženo díky vytváření tzv. izolačních stromů.
Isolation Forest algoritmus má nízkou lineární časovou závislost a nízké paměťové nároky. Zároveň je schopen si poradit i s vícedimenzionálními daty, což z něj dělá ideální nástroj, který nepřehltí výkon výpočetního stroje. To vše bylo ukázáno v práci od Liu et al. [18].

6.1.2 Popis metody

Metoda funguje na principu izolace, což znamená oddělování jednotlivých instancí od ostatních instancí. Dochází k náhodné separaci instancí tak dlouho, dokud nejsou všechny izolované.
Jinými slovy, algoritmus generuje rozdělení vzorku dat tak, že náhodně vybere příslušnou vlastnost a poté náhodně vybere hodnotu, kterou příslušná vlastnost nabývá, z intervalu hodnot, které může nabýt. Bylo zjištěno, že právě takového náhodné rozdělování je klíčem k detekci anomálií, jelikož anomálie vykazují kratší délku cest od kořene stromu, v porovnání s ostatními normálními daty. Délka cesty datového bodu je definována jako součet hran

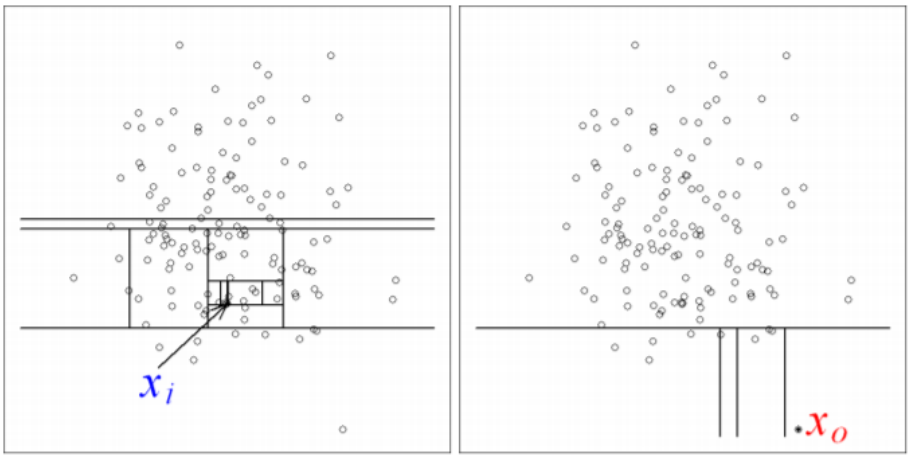
izolačního stromu, které vedou od kořene stromu na konec stromu příslušící datovému bodu, příklad lze vidět na obrázku 6.1, kde anomálie má kratší délku cesty (počet hran), než normální datový bod [18].



Obrázek 6.1: Příklad vytvoření izolačního stromu (*IForest*) [58]

Na obrázku 6.2 lze vypozorovat porovnání procesu rozdělování normálního datového bodu a anomálie. Vlevo lze vidět, že k úplné separaci normálního datového bodu (x_i) je potřeba mnohem více kroků, než je tomu v případě anomálie (x_0).

Pokud postup zopakujeme například 1000x, dojdeme k názvu algoritmu (*IsolationForest*). Při opakování tohoto procesu získáváme stále více náhodně generovaných izolačních stromů a s tím odpovídající počet délek cest jednotlivých vzorků dat. V dalším kroku dochází k operaci aritmetického průměru nade všemi délkami cest z jednotlivých izolačních stromů. Datové body, které lze považovat za anomálie, mají nižší průměrnou délku cest, než body, které reprezentují normální data [18].



Obrázek 6.2: Proces izolace a rozdělování dat algoritmu IsolationForest[18]

Isolation Forest při své činnosti může, či nemusí stavět strom do jeho úplné formy. Pokud bychom chtěli přistupovat k problému více s ohledem na paměťové a procesorové nároky,

nemusíme provádět budování stromu u normálních dat až do úplného konce, kde je již nelze rozdělit. Tento přístup značně sníží celkové nároky na zpracování. Tato skutečnost vychází z myšlenky, že datový bod má výrazně kratší délku cesty ve stromu, než normální data.
Pokud je algoritmus použit na data, která máme označkovaná, nemusí ani využít veškerý normální datový provoz, ale pouze jeho část. Vystaví si model na menším vzorku normálních dat, než je tomu zapotřebí a získá tak potřebné cesty v grafu, které reprezentují chování normální provoz [18].

6.1.3 Sub-sampling (Dílčí vzorky)

Sub-sampling je metoda používaná ke snížení objemu dat tak, že dojde k oddělení určité předem dané části dat a s tou se poté v daném kroku pracuje. Používá se zejména kvůli dvěma důvodům:

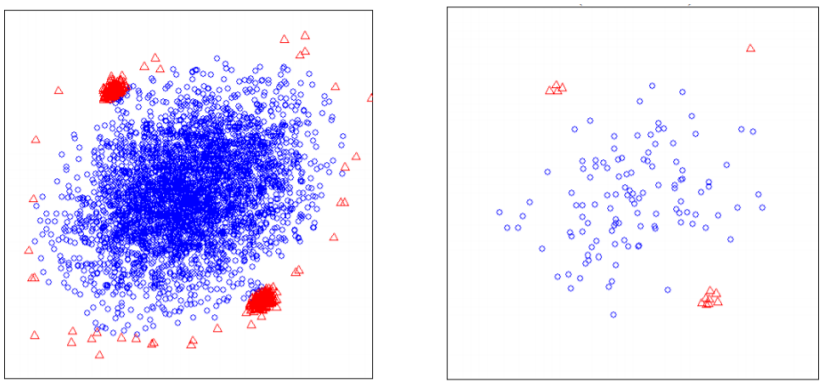
1. **Swamping**

2. **Masking**

Swamping je označení problému špatné identifikace normálních dat. Pokud jsou normální data příliš blízko k anomáliím, dochází k jejich nesprávnému rozdělení vlivem zvýšeného počtu kroků potřebnému k separaci anomálií.

Masking nastane, pokud množství anomálií v daném datasetu je příliš veliký na oddělení od ostatních dat. Data tvoří svoji vlastní skupinu a z pohledu algoritmu se jeví jako normální provoz. Pokud navíc je shluk, který tvoří větší množství anomálií rozhlehlý a hustý, dochází také k většímu počtu kroků, které jsou potřeba vykonat k jejich separaci.
Všeobecná myšlenka detekce amomálií je směrována na fakt, že čím více dat máme k dispozici, tím lepší by měl být náš model, jelikož se může učit z více dat. Bohužel všeho moc škodí, a právě *masking* a *swamping* jsou problémy, které jsou následkem příliš velkých objemů dat.

Používání metody *sub-sampling* v algoritmu Isolation Forest vede ke snížení vlivů *swamping* a *masking*. Což je to způsobeno především snížením velikosti dat, která jsou použita v jednom kroku. Navíc je dosaženo také lepšího rozdělení anomálií, kterých je příliš mnoho u sebe. Jelikož izolační strom, který se tvoří, je budován v každém cyklu s různě vybranými datovými body [18].



(a) Originální dataset (b) Dataset po aplikaci sub-sampling

Obrázek 6.3: Příklad datového setu s a bez sub-sampling[18]

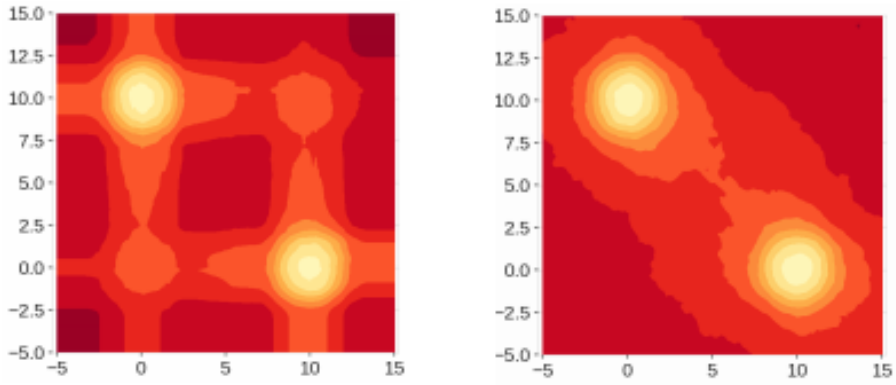
Na obrázku 6.3 lze vidět data před a po aplikaci sub-sampling. Dochází k významnému zmenšení objemu dat, což má pozitivní vliv na výpočetní vlastnosti algoritmu a zároveň nepřijdeme o užitečné informace. Jelikož struktura dat zůstane zachována díky jejich náhodnému výběru, která budou zpracována v aktuálním kroku [18].

6.1.4 Rozšíření metody a aplikace

Isolation Forest byl použit již v několika aplikacích, například v práci [63], kde využívají IF pro detekci anomálií v datových tocích. Nutnou potřebou pro fungování navrženého řešení je datový balík, který obsahuje pouze normální data. Poté je navržené řešení schopno detekovat anomálie, kterých by nemělo býti mnoho. Vytvořený algoritmus byl nazván *Streaming HS-Trees*. Výhodou tohoto algoritmu je velice dobrá časová a paměťová náročnost. V práci [66] se snaží o řešení podobného problému detekce anomálií, ale v multidimenzionálních datových tocích.

V práci [14] se snaží navrhnout nenáročný a distribuovaný algoritmus na principu izolace anomálií . Vkládají subdetektor do každého uzlu v bezdrátové síti, informace z jednotlivých detektorů v uzlech jsou poté poslány všesměrovým vysíláním (*broadcastem*) do sousedních uzlů. Role navrženého detektoru anomálií je aktivní dohled nad celou bezdrátovou sítí a zajištění kvality dat.

Základní verze algoritmu *Isolation Forest* [18] má v sobě obsažen problém přidělování skóre anomality. Skóre anomality je totiž ovlivněno směrem os. Řešení tohoto problému přináší nová verze algoritmu s názvem *Extended Isolation Forest* [24]. Základní myšlenku lze vidět na obrázku 6.4. Je patrné, že v původní verzi algoritmu byl problém s ovlivněním přidělování skóre ve směru os. Na obrázku jsou znázorněny dva shluky, vlevo je základní verze IF, vpravo je rozšířená verze IF. U rozšířené verze algoritmu IF je patrné, že vliv přidělování skóre anomality ve směru os byl odstraněn.



Obrázek 6.4: Porovnání přidělování skóre IF a Extended IF [24]

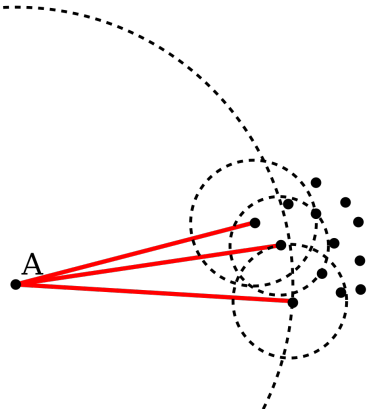
6.2 Local Outlier Factor

6.2.1 Úvod

Local Outlier Factor je algoritmus představený v práci od Breunig et al. [33], používaný v oblasti umělé inteligence, jako metoda učení bez učitele (*unsupervised learning*). Funguje na principu počítání lokální hustoty datového bodu s ohledem na jeho sousedy. Dříve se datový bod buďto prohlásil za anomálii nebo za normální, nebylo nic mezi tím. Přístup, který je aplikovaný v LOF spočívá v přiřazování skóre každému datovému bodu. Toto skóre nám říká informaci o tom, jak moc datový bod lze považovat za anomálii. Tato skutečnost značně rozšiřuje spektrum vnímání datových bodů.

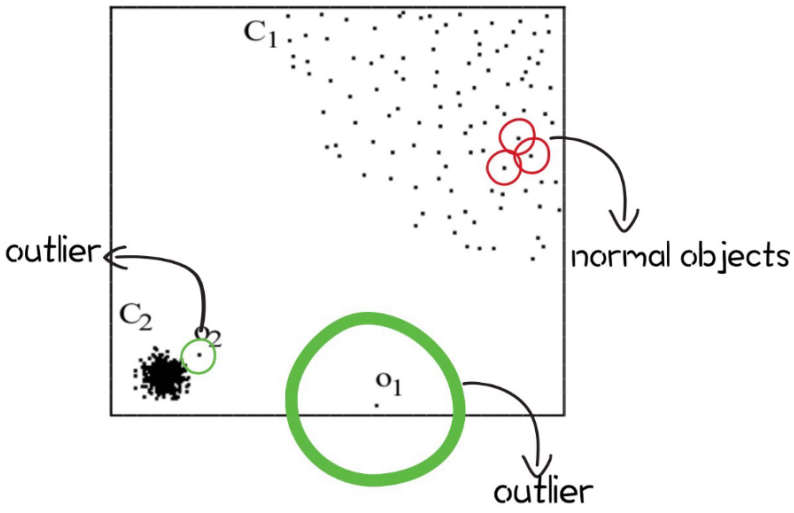
6.2.2 Popis metody

LOF skóre se počítá jako poměr průměrné lokální hustoty k-nejbližších sousedů a lokální hustoty samotného datového bodu, pro který se zrovna počítá LOF. Nejdříve se hledá nejmenší hyperkoule se středem v datovém bodě obsahující k-nejbližších sousedů. Poté je lokální hustota vypočítána jako počet nejbližších sousedů vyděleno objemem hyperkoule. Pro body, které mohou být považovány za normální provoz, bude jejich lokální hustota podobná jako s body, se kterými sousedí. Pro body, které jsou určitým způsobem anomální, bude jejich lokální hustota menší než lokální hustota jeho k-nejbližších sousedů. Na obrázku 6.5 lze vidět základní myšlenku LOF, datový bod A má mnohem menší lokální hustotu než jeho sousedé, proto bude jeho LOF faktor vysoký. Velká výhoda tohoto přístupu spočívá ve schopnosti detekovat téměř všechny formy *outliers*, a to i takové, které nemohou být detekovány pomocí algoritmů založených pouze na výpočtu vzdálenosti [7].



Obrázek 6.5: Ukázka lokální hustoty - LOF [8]

LOF řeší problém detekce lokálních *outliers* (anomálií). Jde o problém na obrázku 6.6 Existují dva velké shluky dat C1/C2 a dva datové body, které vykazují značné odlišnosti. Datový bod o_1 je vyhodnocen jako anomálie téměř vždy, nicméně datový bod je z pohledu přístupu založeném na vzdálenosti možné vyhodnotit jako normální, i když se jedná o anomálii. Právě tento problém řeší LOF. Lokální hustota datového bodu o_2 je jiná, než lokální hustota bodů ve shluku C2 a proto je možné odhalit i tyto lokální *outliers* [33].



Obrázek 6.6: Ukázka problému detekce lokálních a globálních outliers [8]

Kapitola 7

Praktická realizace

7.1 Úvod

Praktická část spočívá v aplikaci popsaných algoritmů na syntetická a reálná data. Algoritmy byly nejdříve vyzkoušeny na syntetických označkovaných datech, u kterých je možné dané algoritmy a vytvořené modely vyhodnocovat, jelikož je k dispozici informace o pravé povaze právě testovaného provozu. Je tedy možné určit, jak moc se model liší, v čem udělal nejvíce chyb, jaký je počet falešných alarmů atd. Na syntetických datech je také možné vyzkoušet vliv parametrů na chybovost vytvořeného modelu. K získaným výsledkům vlivu jednotlivých parametrů bylo přihlédnuto v aplikaci algoritmů na reálných datech.

7.2 Práce s jazykem Python

Praktická část byla vytvářena v jazyce Python, k aplikaci popsaných algoritmů a vytváření jim odpovídajících modelů byla použita knihovna **scikit-learn** [45], která je volně dostupná pro všechny a zaměřena na oblast strojového učení. *Scikit-learn* je vystavena na NumPy, SciPy a Matplotlib knihovnách, což značně usnadňuje celou práci s kódem. V následující části budou krátce popsány způsoby a metody se kterými se pracovalo.

Načítání dat je základem při postupu zpracování dat. Načítání dat lze provést v případě datového balíku *KDDCUP99* pomocí knihovny *sklearn*, konkrétně pomocí modulu *datasets*. Z modulu *datasets* je možné zavolat celou řadu datových balíků. V diplomové práci je volán konkrétně datový balík *KDDCUP99* pomocí příkazu [46]:

```
from sklearn import datasets

data = datasets.fetch_kddcup99(subset = "SA", percent10 = True)
```

Tímto příkazem dochází k načtení jedné desetiny části *SA* datového balíku *KDDCUP99* do proměnné data, ze které vznikne slovník jako objekt s datovou částí, značkou odpovídající datové části a popisem datového balíku. Při načítání reálných dat dochází k využití knihovny *Pandas*, konkrétně metody čtení *csv* souboru:

```
import pandas as pd

data = pd.read_csv('./dataHuawei/ALL_packets.csv', sep=';')
```

V další fázi dochází u syntetických dat **vytvoření datového rámce** z právě načtených dat pomocí knihovny *Pandas*:

```
import pandas as pd

dataSA = pd.DataFrame(data.data, columns = names)
```

Jako data do datového rámce vstupuje datová část z již zmíněné proměnné data a názvy sloupců jsou použity dle [36].

Knihovna *Pandas* byla v práci použita také pro získání dalších informací, například počet jednotlivých útoků v datovém rámci. V další fázi dochází k rozdělení několika tříd útoků do pouze dvou tříd, abychom byli schopni lépe hodnotit vytvořené modely. Tedy všechny útoky jsou označeny hodnotou -1 a všechen normální provoz hodnotou +1. Tyto hodnoty byly zvoleny dle použitých metod pro oba algoritmy LOF a IF, jelikož ty oba vracejí výsledky jednotlivých datových vzorků právě v tomto formátu.

```
import pandas as pd

dataSA['binary_target'] = [1 if x =='normal.' else -1 for x in dataSA['
    target']]
```

Do nově vytvořeného sloupce s názvem *'binary_ target'* je vložen celý list, který je vytvořen dle hodnot ve sloupci *'target'*.

Převod tzv. *categorical features* (vlastnosti datových vzorků, které nejsou ve vhodném číselném tvaru pro model) na vhodné číselné hodnoty je proveden pomocí knihovny *sklearn* a jejího balíčku *preprocessing*.

```
from sklearn import preprocessing

le = preprocessing.LabelEncoder()
protocol_type = le.fit_transform(list(dataSA["protocol_type"]))
```

Nejdříve dochází k vytvoření třídy *LabelEncoder* pojmenovanou jako *'le'*. Poté je zavolána metoda *fit_ transform* této třídy, která je schopna zakódovat žádaný sloupec tak, že vytvoří n - 1 počet odpovídajících numerických tříd (ve smyslu klasifikace), kde n je počet tříd původního datového sloupce [47].

Normalizace dat je realizována opět pomocí balíčku *preprocessing* v rámci knihovny *sklearn*.

```
from sklearn import preprocessing
import pandas as pd

dataSA_normed = preprocessing.normalize(dataSA.drop(['target', '
    binary_target'], axis=1))
```

Jednoduše dojde k zavolání metody *normalize*, do které lze jako vstupní parametr vložit i celý datový rámec, jako k tomu dochází v ukázce kódu výše. Metoda následně vrátí normalizovaný každý sloupec v datovém rámci. V ukázce také dochází k zavolání metody *drop*, která způsobí, že vytvářený datový normovaný rámec nebude obsahovat informaci o značkách, tato informace není potřeba, jelikož je již obsažena v původním datovém rámci.

Zejména ve zkoušecí fázi algoritmů je nutné také data **rozdělit** na testovací a trénovací. Daný datový model se nejdříve vytrénuje na trénovacích datech a poté dojde k aplikaci vytrénovaného modelu na testovacích datech. Pokud navíc máme informaci o značkách datových vzorků, jsme schopni říci, jak si model vede.

```
import pandas as pd
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(dataSA.drop(['target', '
    binary_target'], axis=1), dataSA['binary_target'], test_size=0.3)
```

K rozdělení dat je využita metoda *train_ test_ split*, která rozdělí pole, nebo matici do náhodných trénovacích a testovacích podvzorků. Tato metoda vyžaduje informaci o vstupních datech, které bude rozdělovat, informaci o jejich značkách a v neposlední řadě v jakém poměru má data rozdělit [48].

Samotná **realizace vybraných algoritmů** je vytvořena za pomocí knihovny *sklearn*.

```
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor

IF = IsolationForest(max_samples= 0.25, random_state=11, contamination
    =0.0001, n_estimators=150, n_jobs=-1)
LOF = LocalOutlierFactor(n_neighbors=15, metric='euclidean', algorithm='auto
    ', contamination=0.0001, n_jobs=-1)

IF.fit(x_train,y_train)
y_pred_IF = IF.predict(x_test)
y_pred_LOF = LOF.fit_predict(data_new_drop)
```

Za asistence balíčku *ensemble* a *neighbors* jsme schopni vytvořit odpovídající třídy daných modelů za použití určitých parametrů. V další fázi je nutné model vytrénovat na trénovacích datech. To probíhá u algoritmu *Isolation Forest* pomocí metody *fit*, kde dochází k "napasování"dat do modelu a následně je možné použít metodu *predict* na testovací data, abychom zjistili, jak vytvořený model rozděluje vstupní testovací data na normální a nenormální provoz.
Algoritmus *Local Outlier Factor* je použit ve stavu detekce anomálií, to znamená, že se musí použít metoda *fit_ predict*. Tato metoda "napasuje"data do modelu a ihned vypočítá skóre datových vzorků, na základě vypočítaného skóre označení datové vzorky za anomálie nebo za normální data. Metodu lze použít i ve formě detekce nových anomálií, u níž by se použila klasická metoda *fit* a poté *predict*, jako je tomu u algoritmu *Isolation Forest*.

Vyhodnocení vytvořených modelů je realizováno pomocí balíčku *metrics*, který je součástí knihovny *scikit-learn*. Jedná se o tři metody *classification_ report*, *confusion_ matrix* a *roc_ auc_ score*.

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score

classification_report(dataSA['binary_target'], y_pred_LOF, target_names=['
    anomaly', 'normal'])
roc_auc_score(dataSA['binary_target'], y_pred_LOF)
cm = confusion_matrix(y_test, y_pred_LOF)
```

Tři zmíněné metody umožňují pohled na hodnocení modelu z několika podobných pohledů. *Classification_report* vrátí parametry jako preciznost nebo *recall*. *Roc_auc_score* naopak vypočítá parametr *AUC* pro celý predikovaný datový balík. Metoda *confusion_matrix* vrátí list s hodnotami parametrů TP, TN, FP, FN. Hodnoty všech získaných parametrů lze po jejich vypočtení příslušným způsobem vizualizovat.

Ověřování vlivu parametrů je realizováno metodou *GridSearchCV*, která vypočítá vliv určených parametrů na přesnost modelu.

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer
from sklearn.metrics import recall_score

scoring = {'AUC': 'roc_auc', 'Recall': make_scorer(recall_score, pos_label
    =-1)}
gs = GridSearchCV(LocalOutlierFactor(metric='euclidean', algorithm='auto',
    contamination= 0.2, n_jobs=-1, novelty= True), param_grid={'n_neighbors':
     range(2, 20, 1)}, scoring=scoring, refit='Recall', return_train_score=
    True)

gs.fit(x_train, y_train)
results = gs.cv_results_
```

Využit je balík *model_selection* pro *GridSearchCV* a balík *metrics* pro výpočet zvolených metrik. Metoda *make_scorer* předpřipravý metriku *recall_score* do vhodného formátu tak, aby mohla být následně použita při vytváření třídy gs (pro *GridSearchCV*). Poté už pouze postačí použít již výše zmíněnou metodu *fit*, což způsobí úplné propočítání vlivu zvolených parametrů na úspěšnost vytvořeného modelu. Výsledky lze uložit do proměnné jako volání metody třídy gs (*gs.cv_results_*).

Vizualizace parametrů jako například *AUC*, *Recall*, *GridSearchCV* nebo TP/TN/FP/FN je realizována za pomocí příkladů z knihovny *sklearn*, případně *Matplotlib*. Veškeré odkazy jsou uvedeny přímo v kódu vždy u konkrétní metody. Základní použité funkce se jmenují: *plot_hist*, *plot_confusion_matrix* a *plot_gridsearch_cv*.

7.3 Syntetická data

Jako syntetická data mohl být zvolen jeden z několika desítek dostupných datasetů. Nicméně existuje datový balík, který se zaměřuje přímo na problematiku anomálií síťového provozu - KDDCUP99.

Dataset KDDCUP99 [35] byl použit pro soutěž v dolování dat. Účastníci měli za úkol vymyslet, co nejlepší systém pro rozlišování normálního provozu a "špatného" provozu. Dataset KDDCUP99 byl vytvořen pomocí paketového analyzátoru *tcpdump* v *MIT Lincoln Lab*. Data byla generována v uzavřené síti a jednotlivé útoky (anomálie) byly přidány ručně, aby bylo dosáhnuto široké škály útoků v datasetu. Záměrem bylo vytvořit dataset, který by se dal použít pro strojové učení - učení s učitelem (*supervised learning*). Vytvářené modely je možné naučit základní typy útoků a lze je poté použít ke klasifikačním technikám. KDDCUP99 dataset ve svém základu obsahuje opravdu hodně abnormálního provozu (cca 80%), což je nevhodné k použití u metod učení bez učitele (*unsupervised learning*), jelikož se zaměřujeme na detekci abnormálních dat, kterých by mělo být v datasetu trochu v poměru s

normálními daty a měly by být v některé vlastnosti značně odlišné od normálního provozu. Tento problém řeší knihovna *scikit-learn* [45], která dataset KDDCUP99 upravila tak, aby byl vhodný k řešení problému detekce anomálií. Rozdělila dataset do dvou částí, část SA a část SF. V práci se pracuje s částí SA, která obsahuje všechna normální data z původního datasetu a k nimž je přidáno malé množství anomálií v jednotkách procent. V tabulce 7.1 jsou vypsány druhy útoků, které dataset obsahuje.

Druh útoku	popis	příklad útoku
DoS	odmítnutí služby	SYN FLOOD
R2L	neautorizovaný pokus o přístup ze vzdáleného stroje	hádání hesla
U2R	neautorizovaný pokus o přístup na superuživatele	přetečení zásobníku
sondování	pokusy o sledování a zjišťování informací	skenování portů

Tabulka 7.1: Druhy útoků obsažených v datasetu KDDCUP99 [35]

7.3.1 Základní rozbor dat

Dataset KDDCUP99 ve své základní formě obsahuje 4898431 vzorků, 80 % vzorků je abnormální povahy, zbytek jsou data normálního provozu. Zredukovaný dataset SA obsahuje 976158 vzorků, kde každý vzorek se skládá ze 41 vlastností (dimenzionalita vzorku = 41). Vlastnosti jsou ve formátu *int*, *float* a *string*. Dataset obsahuje dále značky, podle kterých je možné jednoznačně determinovat povahu vzorků. Značky jsou ve formátu *string*.

Střední hodnota a variabilita

V nadcházející části jsou použity pojmy jako střední hodnota a variabilita dat. To se může zdát zvláštní, jelikož například MAC adresy nebo IP adresy nenabývají číselného datového typu. Celý princip předzpracování takovýchto dat probíhá převodem z formátu (většinou) *string* do formátu *int64*/*float64*. Až po převedení je na tento formát dat aplikována operace pro výpočet střední hodnoty a variability. Samotný princip výpočtu střední hodnoty v tomto kontextu nemusí dávat na první pohled smysl, nicméně se jedná o další dodatečné informace, se kterými lze pracovat. Například pokud existuje 20 IP adres, a je jedno jakých, jsme schopni přiřadit jim čísla a z nich vypočítat střední hodnotu. Z této informace jsme schopni prohlásit, okolo jaké IP adresy se točí většinový tok z pohledu počtu paketů.
Naopak informace o vypočítané variabilitě se dá v kontextu těchto dat využít k posouzení, jak moc se daná vlastnost datového balíku mění. Například pokud je variabilita určité dimenze nulová, můžeme prohlásit, že se tato dimenze v kontextu všech datových vzorků vůbec nemění. Na základě této informace můžeme přistoupit k dalšímu kroku, což může být i samotné vyloučení dané dimenze z datového balíku.

Celkový seznam útoků obsažených v datasetu lze vidět v tabulce 7.2. Jedná se vždy o několik zástupců nejznámějších druhů útoků. Výsledný dataset se skládá z vyjmenovaných útoků a dat, která lze považovat za normální provoz.

Druh útoku	Útoky
DoS	back, land, neptune, pod, smurf, teardrop
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
U2R	buffer_overflow, loadmodule, perl, rootkit
sondování	ipsweep, nmap, portsweep, satan

Tabulka 7.2: Výpis všech útoků, které obsahuje dataset KDDCUP99 [35]

Dataset KDDCUP99 obsahuje 41 vlastností pro každý jednotlivý záznam, některé z těchto vlastností lze vidět v tabulce 7.3. Některé vlastnosti se odkazují na stejného hosta, nebo službu, což je myšleno jako pozorování v intervalu 2 s (okno o délce 2 s).

Střední hodnota a variabilita je vynesena v tabulce 7.4. Lze si všimnout, že největší rozptyl nabývají vlastnosti *src_bytes*, *dst_bytes* a *duration*. To může znamenat, že jejich hodnota je nestálá při pohledu na všechny TCP toky a jsou vhodné pro detekci anomálií. Naopak vlastnosti *num_outbound_cmds a is_host_login* nabývají nulových rozptylů i nulových středních hodnot, to znamená, že se jejich hodnota nemění vůbec a jsou irelevantní z pohledu detekce anomálií.

Vlastnost	Popis	Datový typ
duration	délka trvání spojení v sekundách	číslo (spojité)
protocol_type	typ protokolu (TCP, UDP, ICMP)	řetězec (diskrétní)
service	typ služby (HTTP, Telnet atd.)	řetězec (diskrétní)
src_bytes	počet přenesených byte od zdroje do cíle	číslo (spojité)
dst_bytes	počet přenesených byte od cíle do zdroje	číslo (spojité)
flag	informace o spojení, zda skončilo chybou	číslo (diskrétní)
land	navázání dalšího spojení z nebo na stejného hosta	číslo (diskrétní)
wrong_fragment	počet chybových fragmentů	číslo (spojité)
urgent	počet urgentních paketů	číslo (spojité)
hot	počet "hot"indikátorů	číslo (spojité)
num_failed_logins	počet neplatných pokusů o přihlášení	číslo (spojité)
logged_in	úspěšné/neúspěšné přihlášení	číslo (diskrétní)
num_compromised	počet kompromitujících stavů	číslo (spojité)
root_shell	úspěšné/neúspěšné přihlášení na root	číslo (diskrétní)
su_attempted	pokus o využití příkazu "su root"	číslo (diskrétní)
num_root	počet úspěšných přihlášení na root	číslo (spojité)
num_file_creations	počet vytvořených souborů	číslo (spojité)
num_shells	počet otevřených příkazových řádek	číslo (spojité)
num_access_files	počet operací s jádrovými soubory	číslo (spojité)
num_outbound_cmds	počet odcházejících příkazů v FTP	číslo (spojité)
is_hot_login	přihlášení patří/nepatří na "hot"list	číslo (diskrétní)
is_guest_login	přihlášení patří/nepatří do kategorie "host"	číslo (diskrétní)
count	počet pokusů o spojení na stejného hosta jako je právě aktuální v posledních dvou sekundách	číslo (spojité)
	Vlastnosti, které se vztahují ke spojením, směřovaných na stejného hosta	
serror_rate	procento spojení, která mají "SYN"chybu	číslo (spojité)
rerror_rate	procento spojení, která mají "REJ"chybu	číslo (spojité)
same_srv_rate	procento spojení směřovaných na stejnou službu	číslo (spojité)
diff_srv_rate	procento spojení směřovaných na jinou službu	číslo (spojité)
srv_count	počet spojení na stejnou službu jako aktuální spojení v posledních dvou sekundách	číslo (spojité)
	Vlastnosti, které se vztahují ke spojením, směřovaných na stejnou službu	
srv_serror_rate	procento spojení, která mají "SYN"chybu	číslo (spojité)
srv_rerror_rate	procento spojení, která mají "REJ"chybu	číslo (spojité)
srv_diff_host_rate	procento spojení směřovaných na jiného hosta	číslo (spojité)

Tabulka 7.3: Základní vlastnosti jednotlivých TCP spojení v KDDCUP99 [35]

Vlastnost	Střední hodnota	Variabilita
duration	210.036362	1.809410e+06
protocol_type	1.154279	2.031633e-01
service	18.837047	5.761575e+01
flag	7.574894	2.681097e+00
src_bytes	1150.688302	1.132680e+09
dst_bytes	3323.967553	1.628366e+09
land	0.000010	9.934926e-06
wrong_fragment	0.000278	7.549845e-04
urgent	0.000030	8.941434e-05
hot	0.044240	7.363244e-01
num_failed_logins	0.000179	4.172391e-04
logged_in	0.695147	2.119198e-01
num_compromised	0.027768	1.582758e+01
root_shell	0.000229	2.284534e-04
su_attempted	0.000179	2.980188e-04
num_root	0.054205	1.983093e+01
num_file_creations	0.004580	3.980854e-02
num_shells	0.000437	4.369500e-04
num_access_files	0.004848	6.394520e-03
num_outbound_cmds	0.000000	0.000000e+00
is_host_login	0.000000	0.000000e+00
is_guest_login	0.003716	3.701893e-03
count	21.583478	6.332964e+03
srv_count	22.541086	6.093027e+03
serror_rate	0.009153	8.274984e-03
srv_serror_rate	0.009318	8.181398e-03
rerror_rate	0.056057	5.255567e-02
srv_rerror_rate	0.056375	5.206039e-02
same_srv_rate	0.977128	1.621826e-02
diff_srv_rate	0.018379	1.332636e-02
srv_diff_host_rate	0.129541	7.549694e-02
dst_host_count	152.011306	1.070229e+04
dst_host_srv_count	201.422662	7.726615e+03
dst_host_same_srv_rate	0.840884	9.683437e-02
dst_host_diff_srv_rate	0.055473	3.161331e-02
dst_host_same_src_port_rate	0.153224	9.380979e-02
dst_host_srv_diff_host_rate	0.023404	2.432121e-03
dst_host_serror_rate	0.009661	8.340810e-03
dst_host_srv_serror_rate	0.008626	7.757017e-03
dst_host_rerror_rate	0.057785	5.077529e-02
dst_host_srv_rerror_rate	0.055963	4.818472e-02

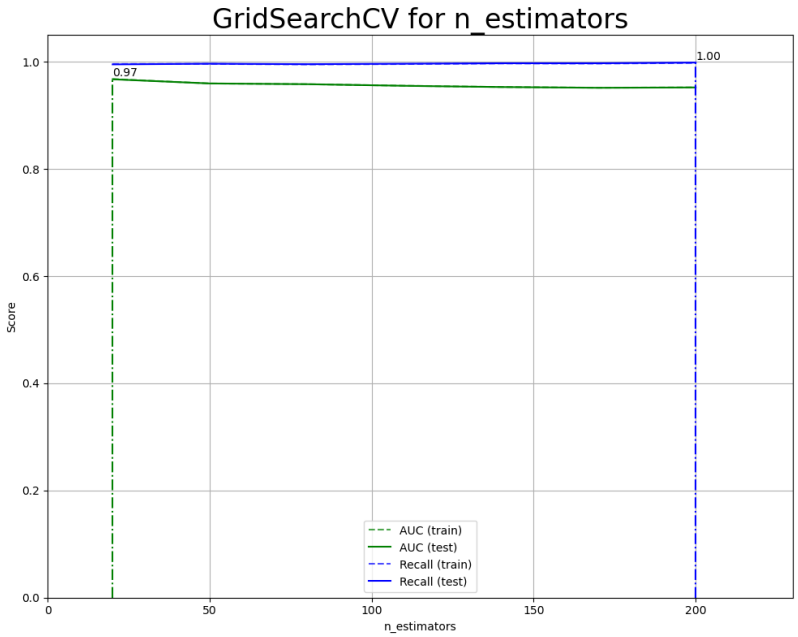
Tabulka 7.4: Střední hodnota a variabilita jednotlivých vlastností

7.3.2 Aplikace algoritmu Isolation Forest na syntetická data

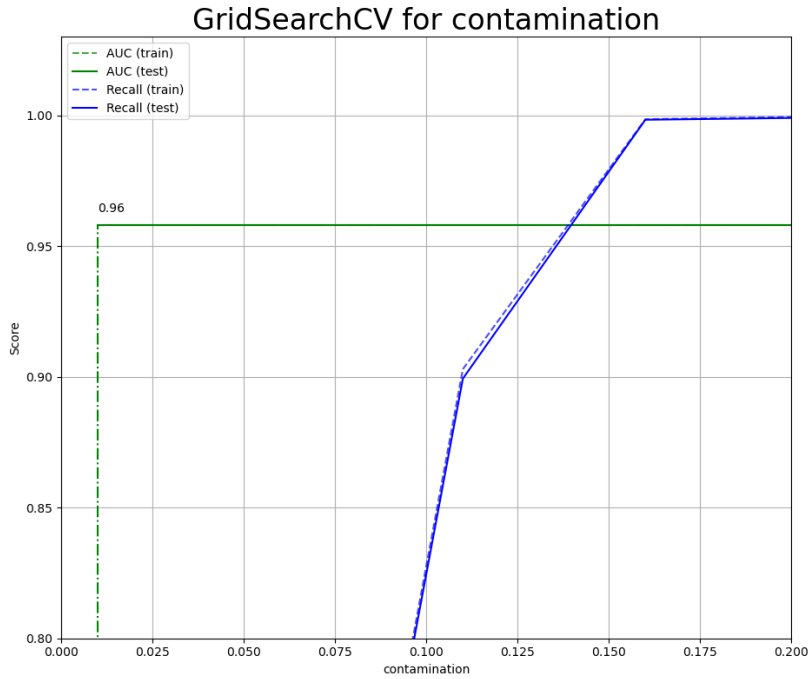
Vytvářený model detekce anomálií pomocí algoritmu *Isolation Forest* je možné realizovat knihovnou **sklearn**[45]. *Isolation Forest* pracuje se třemi hlavními parametry:

- **n_estimators** udává kolik stromů (*ITrees*) bude vystavěno v lese stromů (*Forest*),
- **contamination** se udává, pokud je Isolation Forest použit jako supervised metoda. Jedná se o procentuální část dat, která by měla být abnormální,
- **max_samples** udává, kolik dat bude odděleno (*sub-sampling*) k vytvoření jednoho izolačního stromu. Myšlenka je taková, že právě tento parametr pomáhá najít skryté vzory v datech.

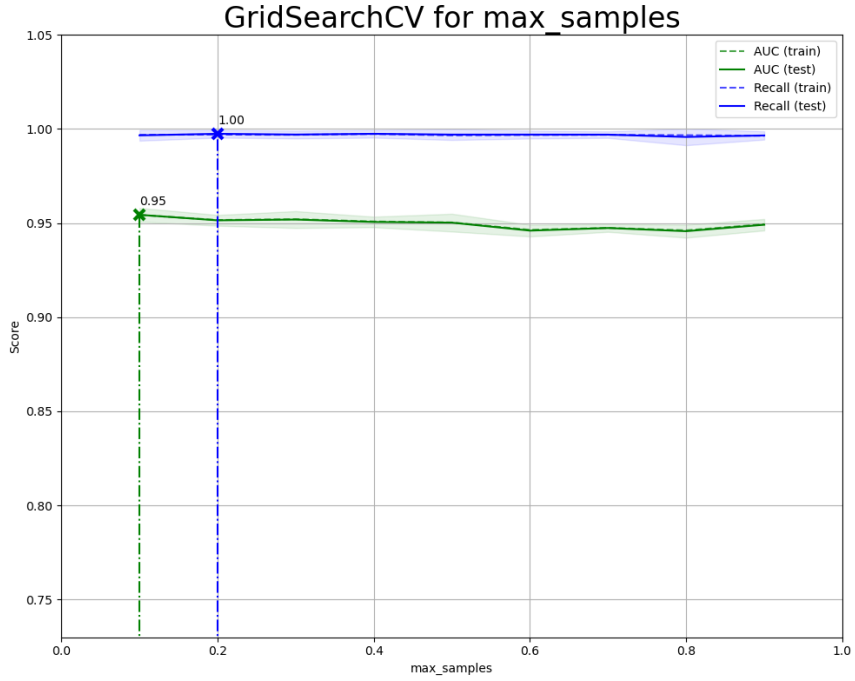
Pro normalizovaná a nenormalizovaná data byly vytvořeny testovací scénáře pro ověření vlivu tří zmíněných parametrů. Dochází k otestování vlivu parametrů na přesnost modelu, konkrétně na výsledky metrik AUC a Recall, jejichž průběh je zobrazen v grafech níže.



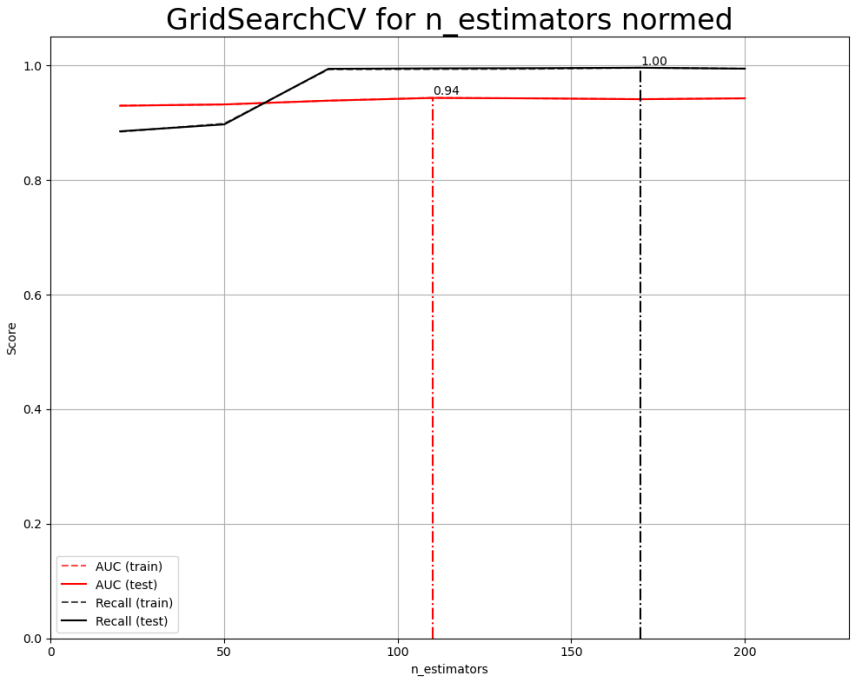
Obrázek 7.1: Vliv parametru *n_ estimators* na přesnost modelu - data nenormalizována



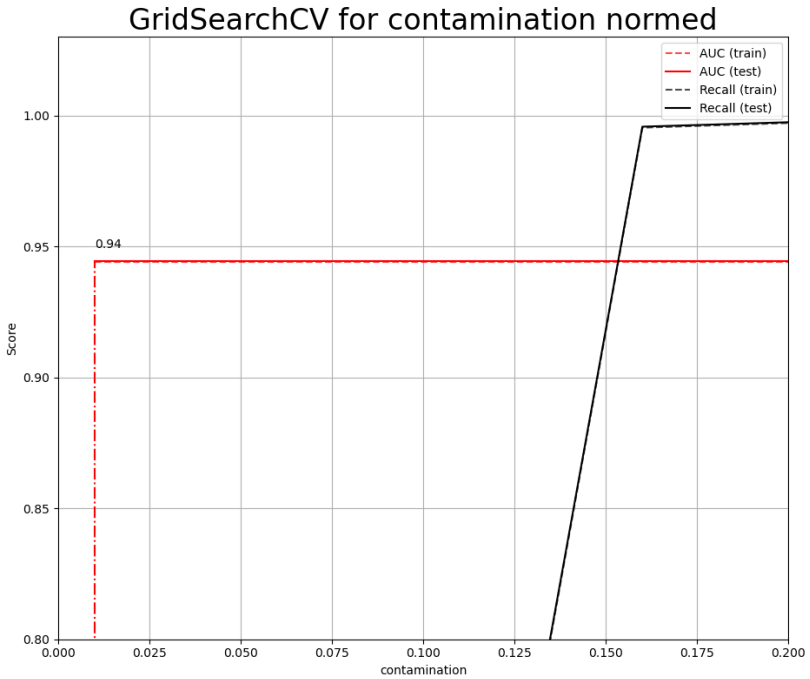
Obrázek 7.2: Vliv parametru *contamination* na přesnost modelu - data nenormalizována



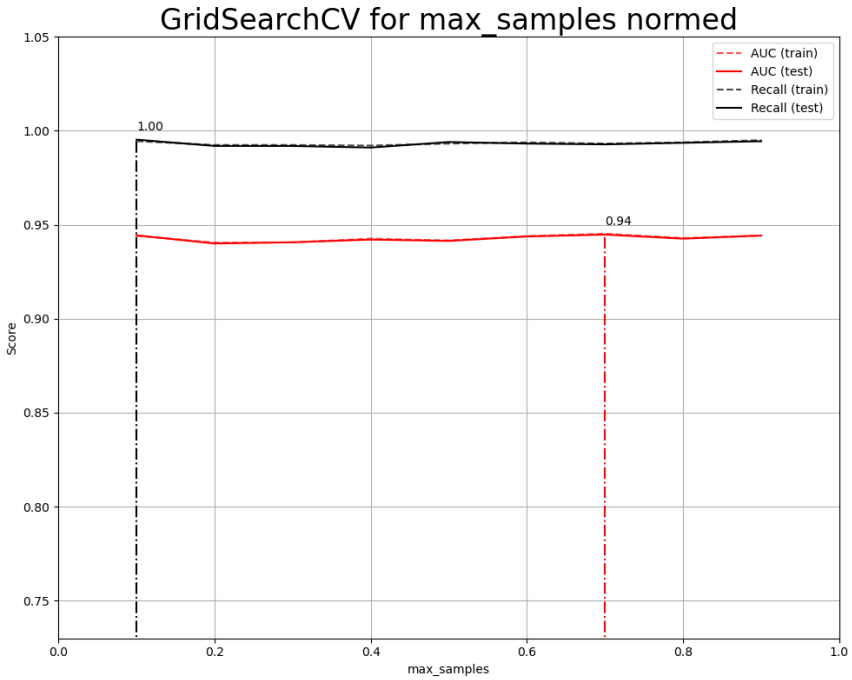
Obrázek 7.3: Vliv parametru *max_samples* na přesnost modelu - data nenormalizována



Obrázek 7.4: Vliv parametru *n_ estimators* na přesnost modelu - data normalizována



Obrázek 7.5: Vliv parametru *contamination* na přesnost modelu - data normalizována



Obrázek 7.6: Vliv parametru *max_ samples* na přesnost modelu - data normalizována

Testování se provádí pomocí metody GridSearch[49], což je metoda na počítání skóre daného modelu pro zvolený rozsah parametrů. Lze nakombinovat i více parametrů s různými hodnotami, nicméně časová náročnost takového výpočtu stoupá exponenciálně, proto bylo provedeno hledání pro každý parametr zvlášť.

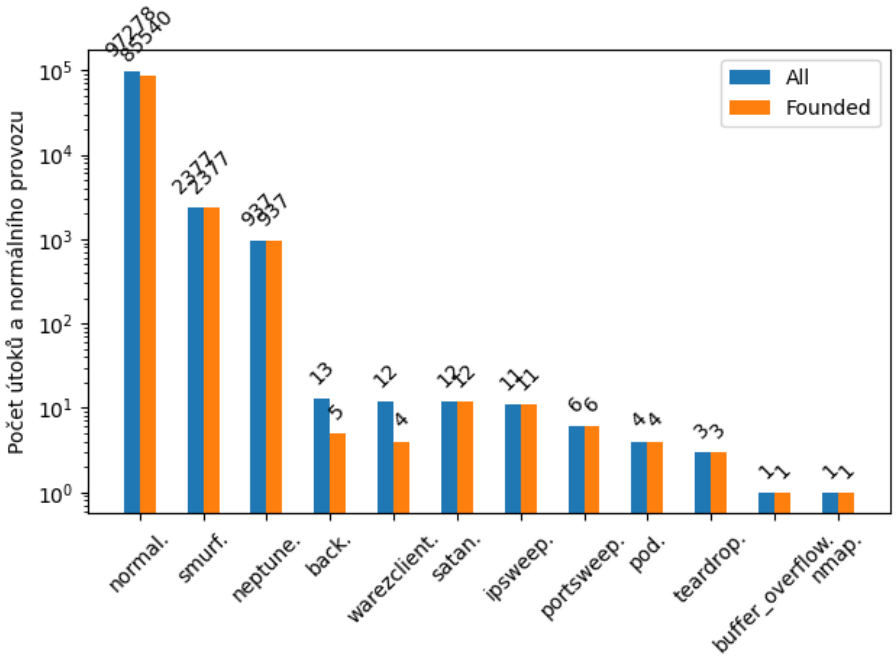
Vliv parametru ***n_ estimators*** na výsledek Recall a AUC pro nenormalizovaná data lze vidět na obrázku 7.1 a pro normalizovaná data na obrázku 7.4. Z průběhů je patrné, že pro tento konkrétní data set, nemá parametr ***n_ estimators*** téměř žádný vliv na přesnost modelu, což potvrzují výsledky prezentované v původní studii o *Isolation Forest* [18]. Algoritmus je tedy pro většinu datasetů necitlivý na parametr ***n_ estimators***. To může být způsobeno samotným principem algoritmu, jelikož anomálie vykazují mnohem kratší cestu vybudovaným lesem stromů oproti normálním datům. Z toho plyne, že není potřeba budovat mnoho stromů pro izolaci veškerých datových bodů, jelikož požadovaného výsledku je dosáhnuto již při nízkém počtu vybudovaných stromů. Rozdíl v průběhu normovaných a nenormovaných dat je minimální, k povšimnutí stojí pouze trochu jiný průběh u normovaných dat, kdy není dosáhnuto maximálního parametru recall ihned z počátku, jako je tomu u nenormovaných dat. U obou způsobů zpracování dat je dosáhnuto pěkných výsledků. Zároveň není žádný rozdíl mezi trénovacím a testovacím výběrem dat.

Naopak vliv parametru ***contamination*** na přesnost algoritmu je znatelný, viz. obrázek 7.2 a 7.5, jelikož při práci s tímto datasetem používáme algoritmus jako *supervised* metodu. Parametr ***contamination*** je použit jako pomyslná hranice mezi abnormálními a normálními daty. Tímto parametrem říkáme algoritmu, jak velký podíl dat může očekávat jako abnormální. Výsledky průběhu odpovídají předpokladům pro určování metrik Recall a AUC. Znatelný vliv na parametr ***contamination*** zaznamenává právě Recall, což odpovídá faktu,

že data obsahují malé procento anomálií a při narůstání hodnot ***contamination*** dochází k lepším výsledkům odhalování těchto abnormálních dat. Rozdíl mezi průběhy metrik pro normalizovaná a nenormalizovaná data je minimální.

Vliv parametru ***max_samples*** na výsledky metrik Recall a AUC pro nenormalizovaná data lze vidět na obrázku 7.3 a pro normalizovaná data na obrázku 7.6. Algoritmus vykazuje značnou necitlivost na vliv parametru ***max_samples***, což není zcela ve shodě s původní studií o *Isolation Forest* [18]. Průběhy se ale mohou lišit pro různé datasety a je tedy možné, že právě tento dataset nepotřebuje k odhalení hluboko ukrytých vzorů menší hodnotu parametru ***max_samples***. Všeobecně vzato by každý dataset měl mít svůj vrchol parametru ***max_samples*** někde jinde, jelikož složitost a způsob, jakým jsou data propletena je pokaždé jiný. Vliv normovaných a nenormovaných dat na celkový průběh je téměř zcela identický, proto lze usuzovat, že algoritmus *Isolation Forest* nepotřebuje k lepším metrickým výsledkům normalizovaná data.

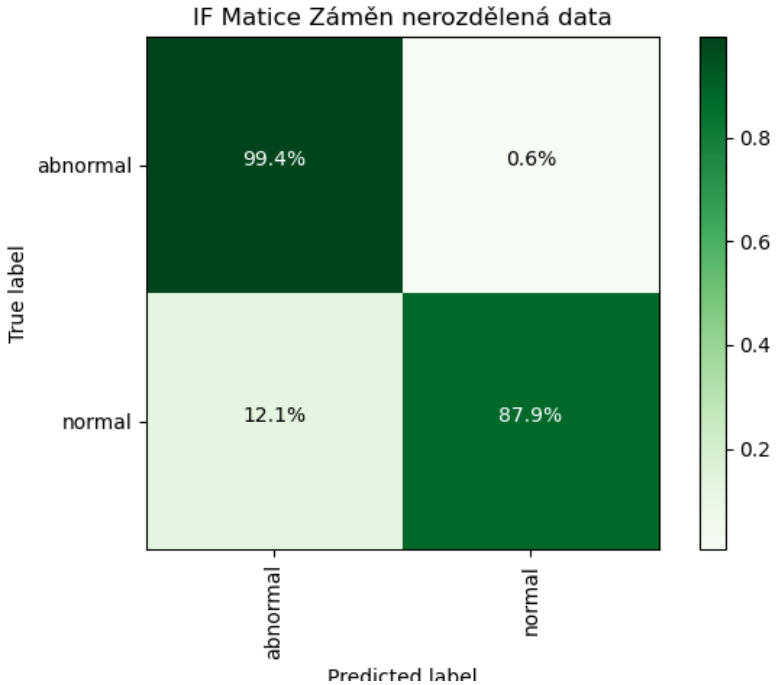
V další fázi byl algoritmus vyzkoušen na datech *KDDCUP99*[35], pro model byly zvoleny nejvhodnější parametry vyplývající ze grafů výše. Nicméně parametry ***max_samples*** a ***n_estimators*** nezaznamenaly znatelný vliv na průběh, bylo tedy nutné správně určit především parametr ***contamination***, který byl nastaven na hodnotu 15%.



Obrázek 7.7: Úspěšnost nalezení jednotlivých druhů provozu

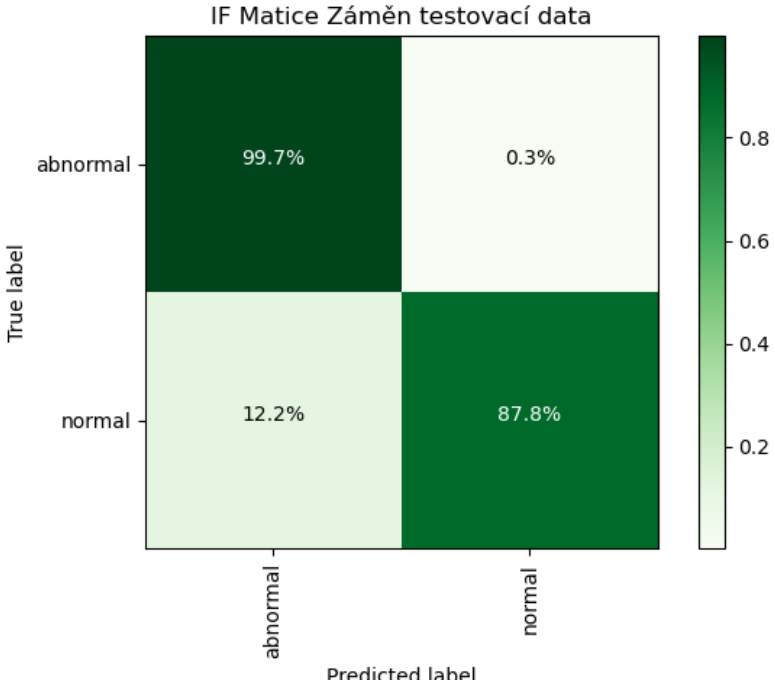
Na obrázku 7.7 lze vidět vyzkoušení míry úspěchu odhalení jednotlivých útoků pro algoritmus *Isolation Forest*. Jde o zobrazení celkové distribuce provozu (modrá barva) a správně odhalené distribuce druhů provozu (oranžová barva). Při zkoumání této distribuce nebylo použito žádné rozdělení na trénovací a testovací data. Model pracuje s celou částí SA datasetu *KDDCUP99*, kromě přiřazených značek. Značky jsou použity až při vyhodnocování

výsledků vytvořeného modelu. Z grafu 7.7 je patrné, že si vytvořený model vede velice dobře při detekci anomálií a odlišení jich od normálního provozu. Bylo dosáhnuto 100% přesnosti u téměř všech druhů útoků. Útoky *back.* a *warezclient.* vykazují menší míru úspěšného odhalení. Skript pro výpočet úspěšné distribuce útoků byl spuštěn několikrát, ale procentuální výsledky byly téměř vždy stejné, měnil se pouze počet jednotlivých útoků a normálního provozu, jelikož při opětovném spuštění dojde pokaždé k vygenerování jiného náhodného datasetu z knihovny *scikit-learn.*



Obrázek 7.8: Zobrazení úspěšnosti predikce pomocí matice záměn

Na obrázku 7.8 byla zobrazena úspěšnost predikce modelu na vložená data. Model dosahuje hodnoty 99,4% **TP** (*True Positive*) a 87,9% **TN** (*True Negative*). Výsledky detekce anomálií jsou opravdu dobré, lze říci, že byly objeveny téměř všechny útoky obsažené v datasetu. Hodnoty **FP** (*False Positive*) dosahují 12,1%, což vede ke zvětšení celkového balíku dat, které nám model vrátí jako anomálie a zhoršuje tak správnou identifikaci útoků. Model vyhodnocuje některé normální toky jako abnormální.



Obrázek 7.9: Zobrazení úspěšnosti predikce pomocí matice záměn pro testovací data

V další fázi bylo vyzkoušeno rozdělení dat na trénovací a testovací. Je to metoda, která je aplikovaná velmi často v modelech učení s učitelem. Myšlenka je taková, že model nejdříve "vytrénujeme" na trénovacích datech, řekneme mu co je špatné a správné. Poté do něj vložíme testovací data a pozorujeme míru úspěšnosti predikce. Stejný postup byl aplikován a výsledky lze vidět na obrázku 7.9. Bylo dosáhnuto mírného zlepšení oproti modelu, který byl vytvořen na detekci anomálií přímo, bez trénovací fáze.

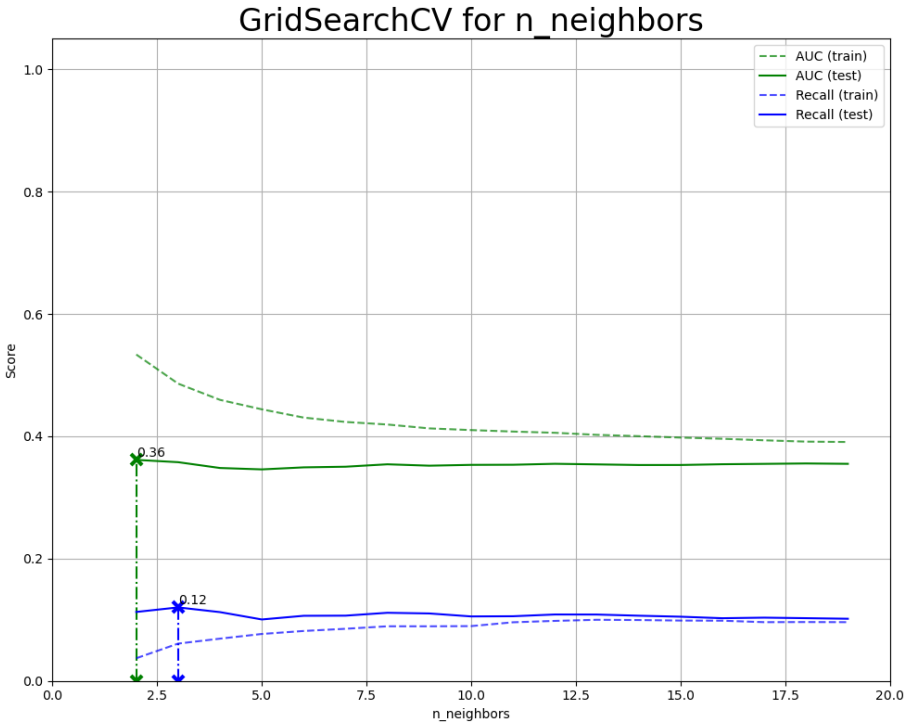
V celkovém hodnocení si *Isolation Forest* vedl neočekávaně dobře, odhalil téměř všechny útoky, které byly obsaženy v datasetu a není závislý na nutnosti normovat data. Poradil si velice dobře s velkým počtem dimenzí. Jediný parametr, na který algoritmus vykázal citlivost, byl parametr ***contamination***.

7.3.3 Aplikace algoritmu Local Outlier Factor na syntetická data

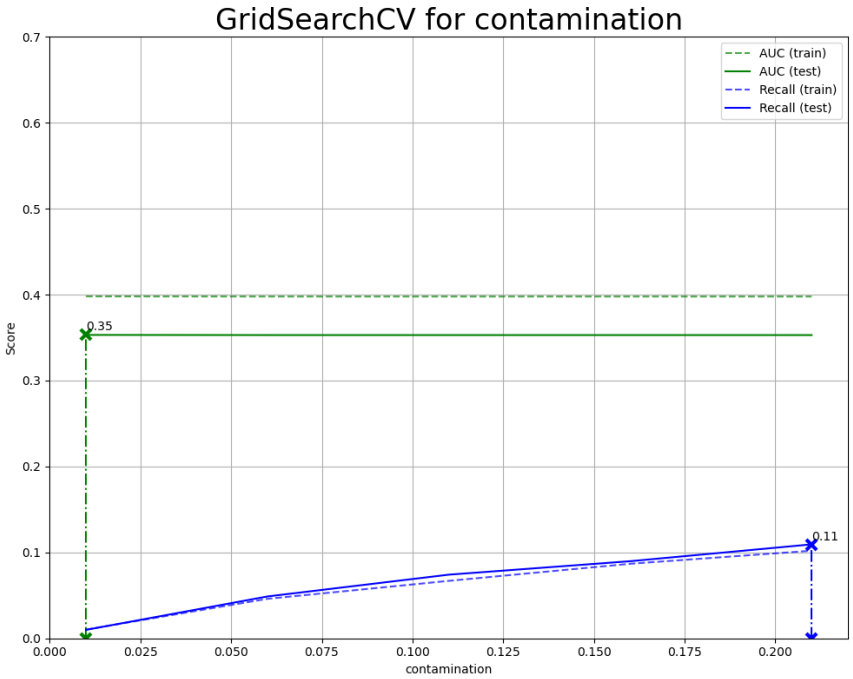
Vytvářený model pro detekci anomálií pomocí algoritmu *Local Outlier Factor* je realizován za pomocí knihovny **scikit-learn**. Byly vytvořeny testovací scénáře pro zjištění vlivu parametrů na přesnost algoritmu. Tyto parametry jsou:

- **n_neighbors** udává počet sousedů, se kterými se porovnává lokální hustota právě vyšetřovaného vzorku,
- **contamination** parametr udává, podíl anomálií v datasetu, když dochází k pasování dat do modelu, je právě tato hodnota použita jako hranice pro určení normálního a abnormálního provozu,
- **metric** parametr udává, jakým způsobem se bude počítat vzdálenost mezi datovými body (např. Euklidovská vzdálenost).

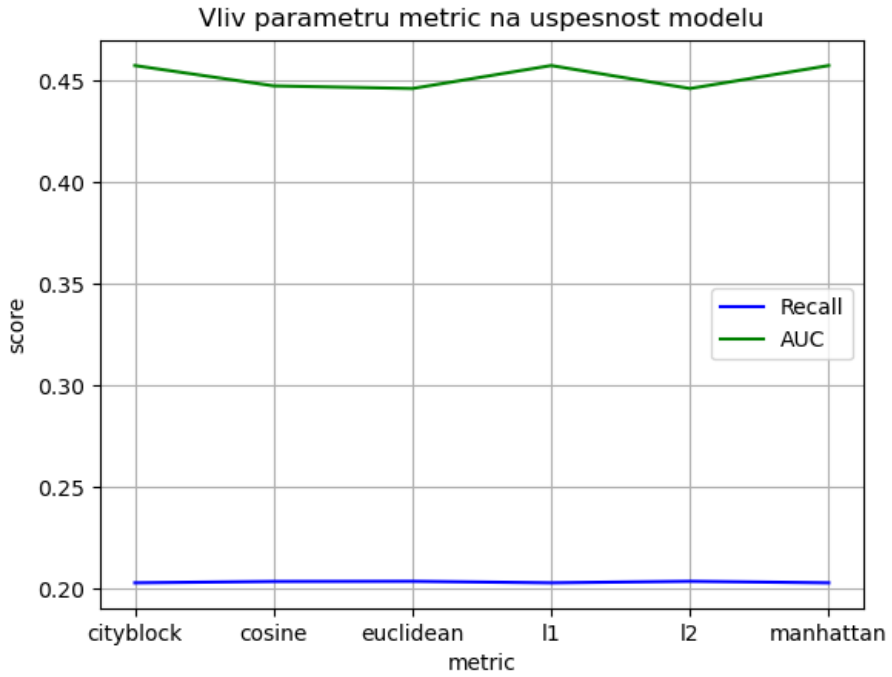
Pro normalizovaná a nenormalizovaná data byly vytvořeny testovací scénáře pro ověření těchto třech parametrů. Dochází k otestování vlivu parametrů na přesnost modelu, konkrétně na výsledky metrik AUC a Recall. Testování bylo provedeno pomocí metody GridSearch [49].



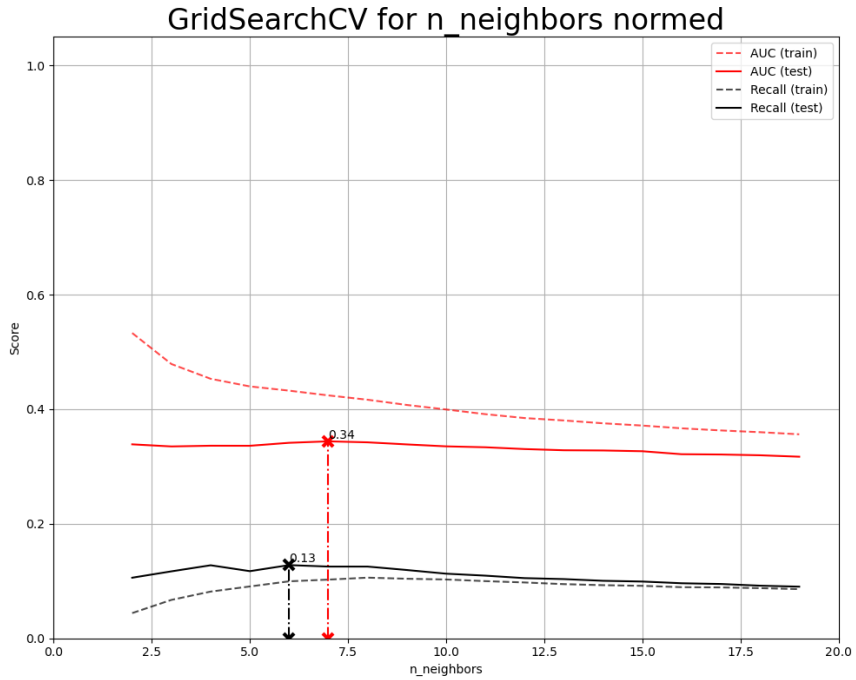
Obrázek 7.10: LOF - vliv parametru *n_ neighbors* na přesnost modelu - data nenormalizována



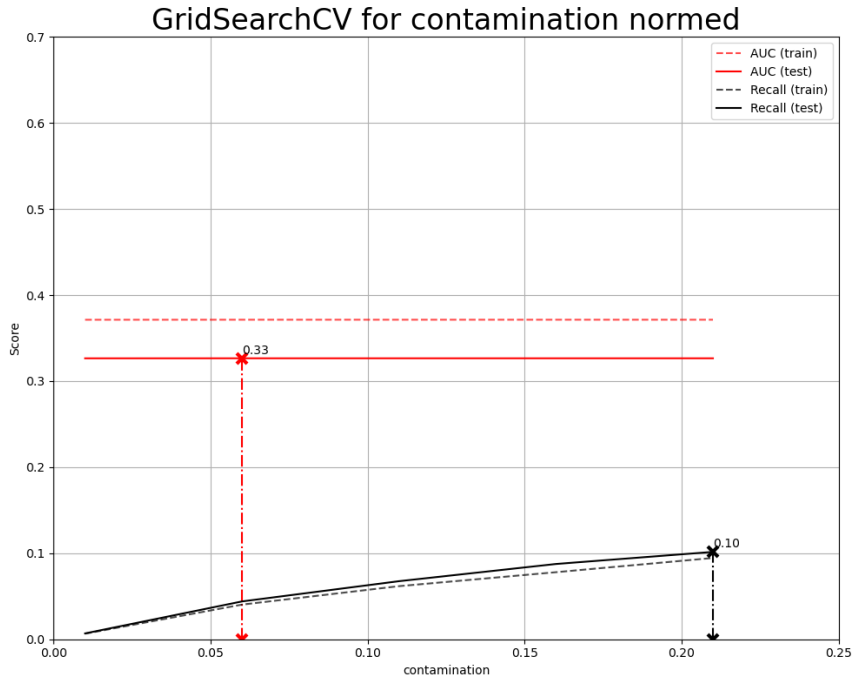
Obrázek 7.11: LOF - vliv parametru *contamination* na přesnost modelu - data nenormalizována



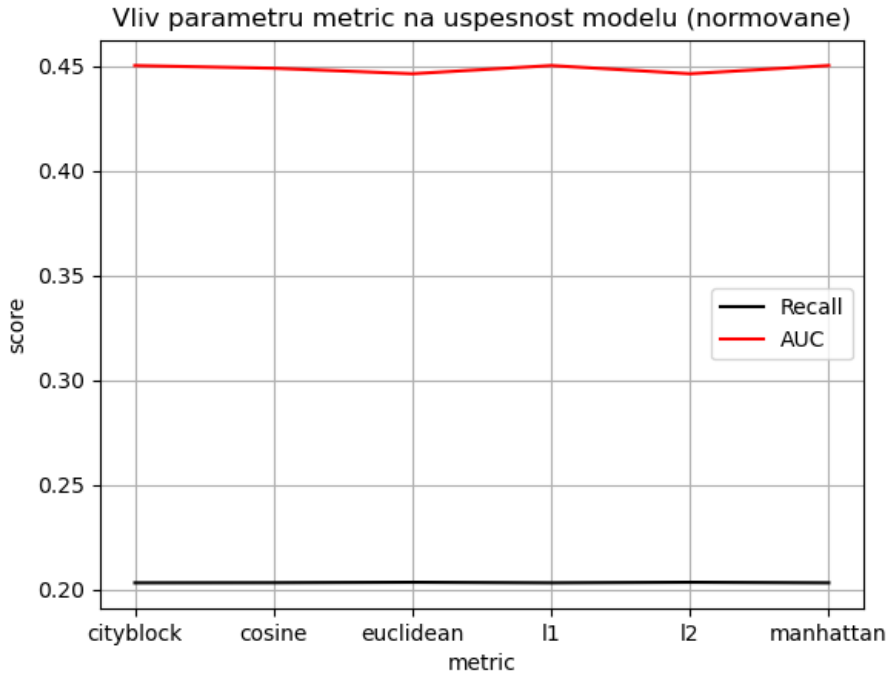
Obrázek 7.12: LOF - vliv parametru *metric* na přesnost modelu - data nenormalizována



Obrázek 7.13: LOF - vliv parametru *n_ neighbors* na přesnost modelu - data normalizována



Obrázek 7.14: LOF - vliv parametru *contamination* na přesnost modelu - data normalizována



Obrázek 7.15: LOF - vliv parametru *metric* na přesnost modelu - data normalizována

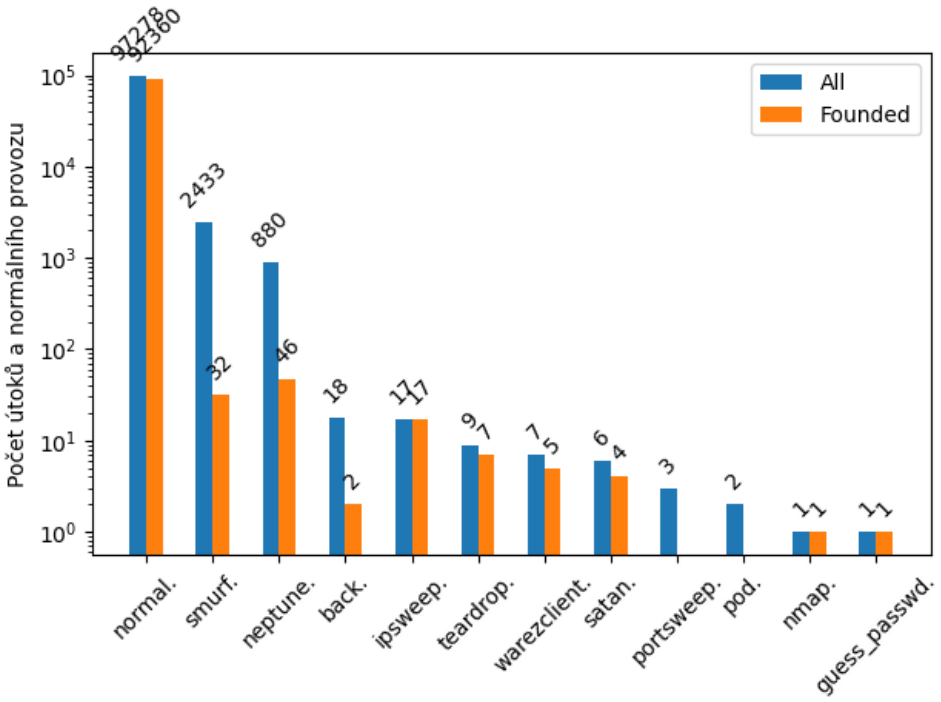
Vliv parametru ***n_neighbors*** na vyhodnocení modelu pro nenormalizovaná data lze vidět na obrázku 7.10 a pro normalizovaná data na obrázku 7.13. Byly vyzkoušeny hodnoty v rozsahu od 2 do 19 s krokem 1. Lze usoudit závěr, že algoritmus je méně citlivý na vliv parametru ***n_neighbors*** pro tento konkrétní dataset. Metriky *Recall* a *AUC* nabývají podprůměrných hodnot, navíc nebylo zaznamenáno ani žádné větší zvýšení přesnosti v průběhu testování různých hodnot ***n_neighbors***. Z grafů 7.10 a 7.13 také plyne, že bylo dosaženo mírně lepších hodnot pro trénovací data, než pro testovací data. Tato skutečnost je velice běžná. Všeobecně vzato mívá algoritmus Local Outlier Factor potíže s velkým počtem dimenzí, což může mít vliv na špatný výsledek vyplývající z grafů. Normování bývá u algoritmu LOF doporučeno, nebyl zjištěn žádný vliv na přesnost modelu. Zaznamenaný vliv byl pouze u dobu výpočtu, kdy pro normalizovaná data byl celý proces o něco rychlejší. Další řešení je, že algoritmus nevyhovuje zkoumanému datasetu, ne všechny algoritmy pro strojové učení jsou vhodné pro všechny druhy dat.

Parametr ***contamination*** má stejně jako u IF vliv na správné vyhodnocení dat modelem. Průběh pro nenormalizovaná data lze vidět na obrázku 7.11 a pro normalizovaná data na obrázku 7.14. Vliv parametru ***contamination*** lze zaznamenat především u metriky *Recall*, jelikož na tu mají vliv pouze detekované a nedetekované anomálie. Z průběhů je patrné, že správně zvolená hodnota parametru ***contamination*** má vliv na lepší výsledek metriky *Recall*.

Vliv parametru ***metric*** lze vidět na obrázku 7.12 pro nenormalizovaná data a na obrázku 7.15 pro normalizovaná data. Z grafů je patrné, že parametr ***metric*** nemá zásadní vliv na výsledky správné predikce modelu. Minimální změny jsou vidět u euklidovské vzdálenosti,

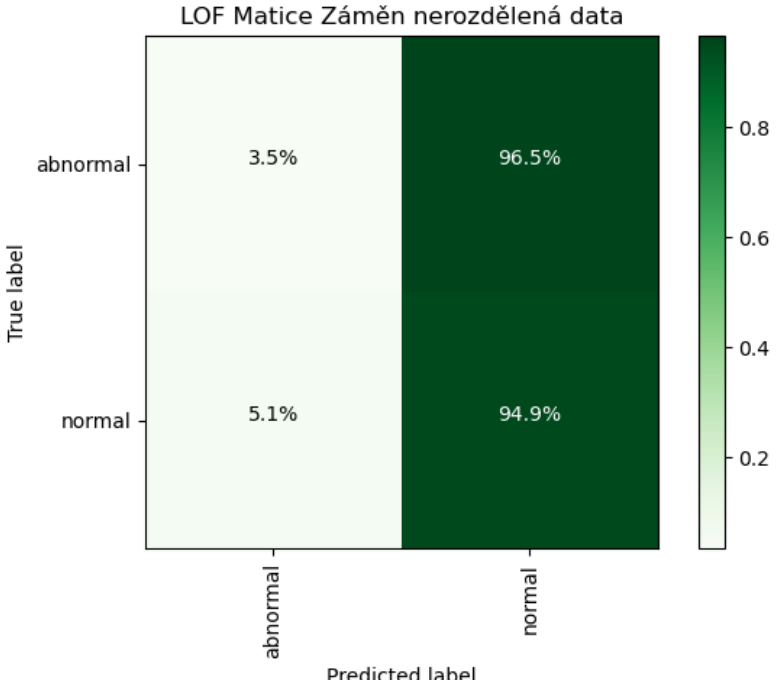
kdy dochází k mírnému poklesu u parametru *AUC*.

I přes neuspokojivé výsledky při zkoumání vlivu parametrů na přesnost modelu, byl algoritmus vyzkoušen na datech *KDDCUP99*[35].

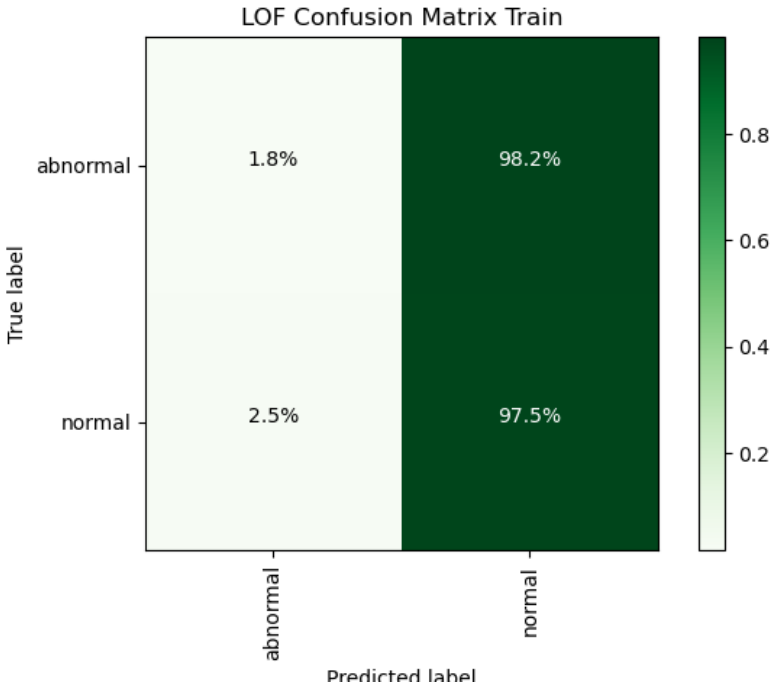


Obrázek 7.16: LOF - Úspěšnost nalezení jednotlivých druhů provozu

Na obrázku 7.16 je zobrazena úspěšná detekce jednotlivých druhů provozu algoritmu *Local Outlier Factor*. Modrou barvou je označena celková distribuce a oranžovou barvou nalezená distribuce. Algoritmus si nevedl moc dobře, zejména u anomálií, kterých je více, jako *smurf.* a *neptune.*. Řídké anomálie jsou detekovány spolehlivěji. Skript pro vytváření distribučního grafu byl spuštěn několikrát, bylo zjištěno, že LOF si nejvíce dokáže poradit s útoky *ipsweep*, *warezclient* a *nmap*. Model v tomto případě pracuje s celou částí datasetu *SA*. Značky byly přidány až po úspěšném modelování. Z grafu je patrné, že algoritmus má značný problém s anomáliemi, kterých je více než "trochu". To může být způsobeno samotným principem algoritmu, kdy takto početné anomálie utváří svůj vlastní shluk (*cluster*) a výpočet vzdálenosti, k jeho k-nejbližším sousedům a následnému porovnání lokálních hustot, vede k závěru, že datový bod není anomálie.



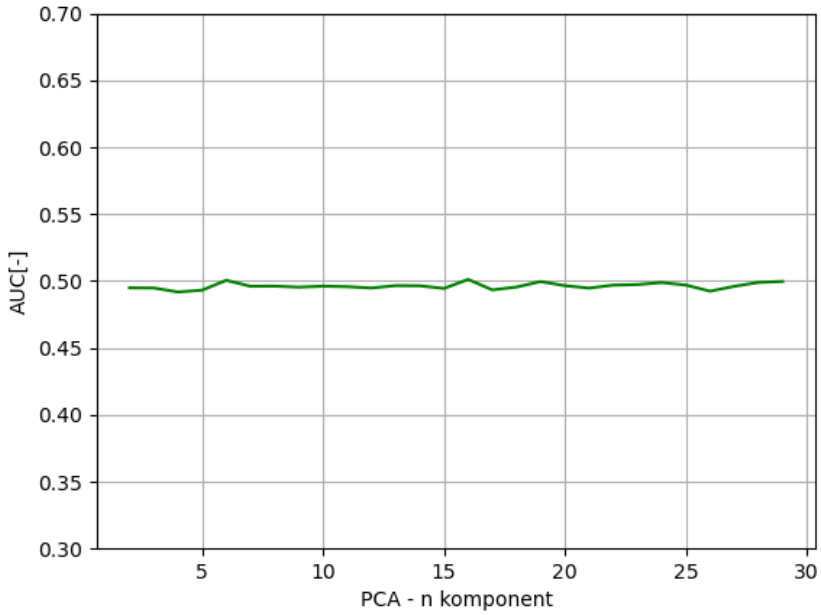
Obrázek 7.17: LOF - Zobrazení úspěšnosti predikce pomocí matice záměn



Obrázek 7.18: LOF - Zobrazení úspěšnosti predikce pomocí matice záměn pro testovací data

Na obrázku 7.17 lze vidět úspěšnost predikce pro celá nerozdělená data *SA* a na obrázku

7.18 pro rozdělená data *SA*. V obou případech si algoritmus nevedl moc dobře, zejména pro detekci anomálií, kde dochází k velkému počtu FN (*False Negative*). Naopak v detekci normálních dat si vedl algoritmus poměrně slušně, odhalil správně cca 95 % normálního provozu, tz. TN = 95 % (*True Negative*). Mírné zlepšení přináší na straně rozlišení normálního toku obrázek 7.18, kdy jsou data ještě více zmenšena, a v balíku je méně anomálií.



Obrázek 7.19: LOF - testování vlivu počtu dimenzí na výsledek metriky AUC

Závěrem lze říci, že si algoritmus *Local Outlier Factor* nevedl příliš dobře pro tento dataset. Dle mého názoru je problém v samotném fungování algoritmu, konkrétně o způsob, jakým se určují datové body za anomálie. Pokud obsahuje datový balík samotný velký shluk anomálií, nastává problém s určováním lokální hustoty datového bodu. Při malých hodnotách parametru ***n_ neighbors*** (řádově desítky) nedochází k dostatečnému ověření lokální hustoty k - nejbližších sousedů, jelikož jeho k-nejbližších sousedů jsou právě datové body v jednom menším shluku, jako je právě vyšetřovaný bod. Naopak při velkých hodnotách parametru ***n_ neighbors*** sice dochází k ověření lokální hustoty i s body, které nejsou v daném shluku, ale řešení to není, jelikož výsledek je stejně ovlivněn shlukem, ve kterém se nachází vyšetřovaný datový bod. Tuto myšlenku částečně potvrzuje distribuční graf provozu 7.16, kdy došlo k odhalení několika útoků v celkem slušném měřítku. Tyto útoky splňovaly podmínku, že je jich málo.
Jedním z původních problému, který byl zkoumán, byl velký počet dimenzí datasetu. Právě proto byla aplikována metoda analýza hlavních komponent pro snížení dimenzí, což lze vidět na obrázku 7.19. Průběh ukazuje závislost proměnné AUC na počtu vytvořených komponent. Nebyl zaznamenán větší ani menší vliv na přesnost modelu v určování skutečného původu dat.

7.4 Data z reálného provozu

V této části se aplikují algoritmy *Local Outlier Factor* a *Isolation Forest* na data z reálného provozu. Dochází k základnímu rozboru dat, je aplikována detekce anomálií na data a v neposlední řadě je realizován krátký rozbor detekovaných anomálií. Data byla poskytnuta katedrou telekomunikačních technologií na ČVUT v Praze.

7.4.1 Použité nástroje

Wireshark je protokolový a paketový analyzér. Jedná se o nástroj podobný *tcpdump*. Ve *Wiresharku* si lze zobrazit si daný síťový provoz a rozebrat jednotlivé pakety do detailních informací, včetně jejich rozdělení dle zapouzdření. V práci byl nástroj *Wireshark* použit k následné analýze některých odhalených anomálií [20].

TShark je součástí nástroje *Wireshark* a v jeho základním použití funguje jako *tcpdump*. Jedná se o protokolový analyzér. Dokáže zachytávat živý provoz, nebo číst již uložený síťový provoz. V diplomové práci je *TShark* použit zejména pro export dat ve formátu *pcap* do formátu *csv* [19].

Mergecap je program, který dokáže spojit více datových souborů ve formátu *pcap* a je součástí nástroje *Wireshark*. V diplomové práci byl použit ke spojení záchytů, které byly právě ve formátu *pcap* [56].

yEd je počítačový nástroj vytvořený ke rychlému generování grafů a diagramů. Jedná se o běžně dostupný nástroj, který je efektivní a zdarma. V diplomové práci byl využit ke generování základních charakteristik sítě [70].

Wonka Flows Exporter je skript, napsaný v jazyce *Python*, který dokáže z uloženého síťového záchytu vytvořit jednotlivé TCP toky tak, jak byly navazovány. V diplomové práci byl tento skript využit právě pro extrakci TCP toků z uložených záchytů ve formátu *pcap*[64].

7.4.2 Základní rozbor dat

Data byla poskytnuta v datovém formátu **pcap**. Jedná se o několik realizovaných záchytů během celého měsíce května (2019) z laboratoře na ČVUT FEL. Záchyty jsou rozdělené do samostatných **pcap** souborů dle jednotlivých dnů a hodin. Celková velikost záchytu je ve formátu *pcap* 700 MB a obsahuje 1 725 601 paketů.

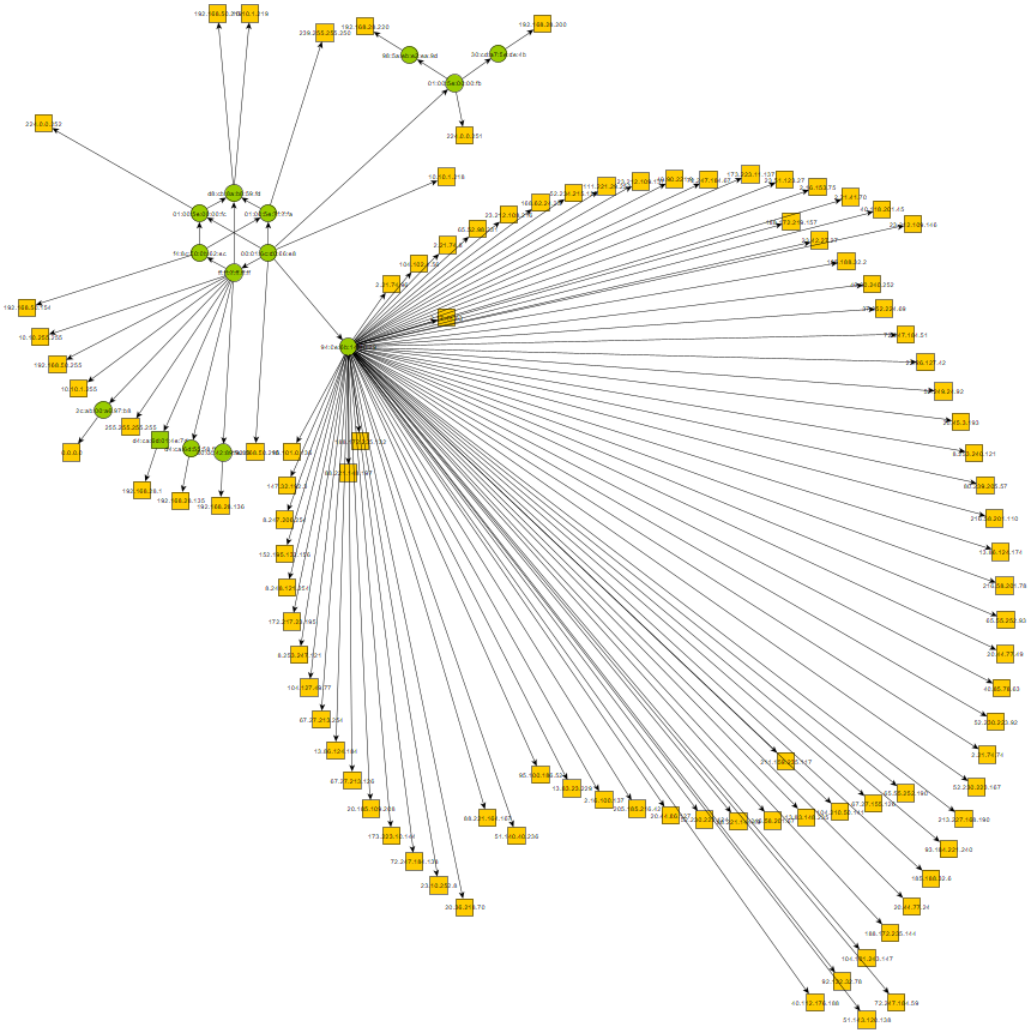
Základní údaje o datovém balíku byly zjištěny pomocí *Python* skriptu a jsou zapsány v tabulce 7.5:

Počet unikátních MAC adres	14
Počet zdrojových IP adres	92
Počet cílových IP adres	90
Počet přenesených TCP/UDP paketů	1028558
Celkový počet přenesených paketů	1725601

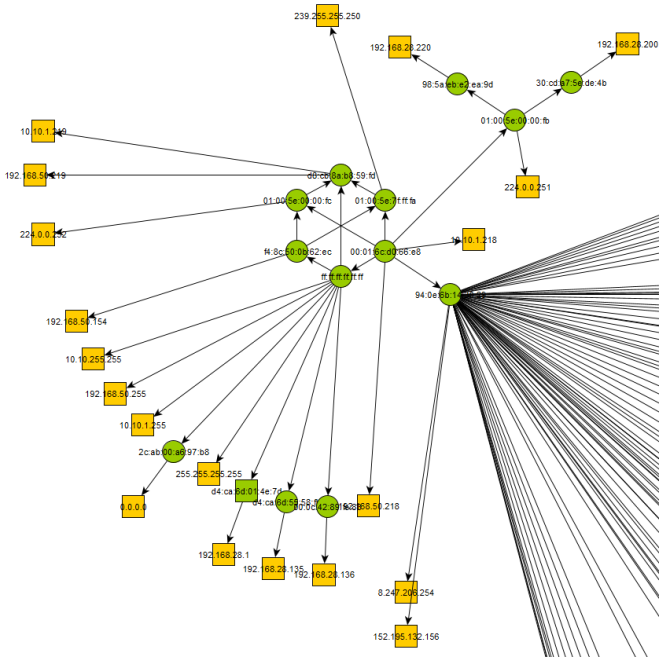
Tabulka 7.5: Základní údaje o reálném datovém balíku

Z tabulky 7.5 lze vidět, že datový okruh obsahuje 14 síťových zařízení, které využívají na třetí vrstvě 90-ti IP adres. Dále si lze všimnout, že TCP/UDP provoz obsahuje něco málo přes 50% z celkového datového balíku. Z programu *Wireshark* bylo dále zjištěno, že velkou část z celkově přenesených paketů zaobírá protokol ARP (*Address Resolution Protocol*) cca 37%.

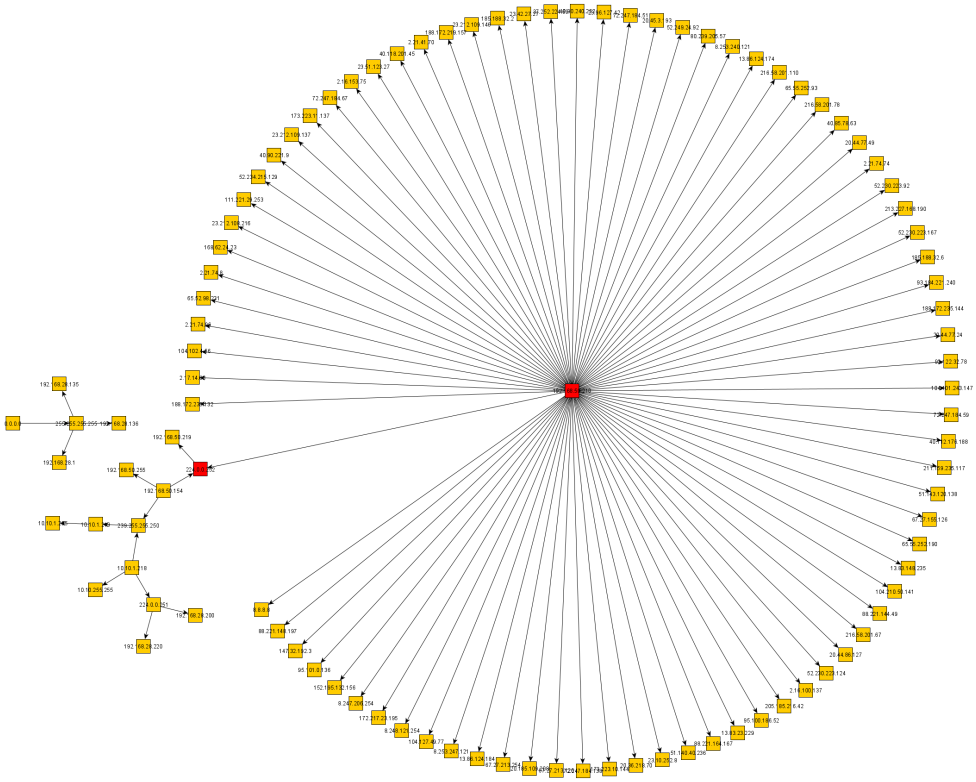
Na obrázku 7.20 lze vidět zobrazení vizualizace lokální komunikační sítě mezi MAC a IP adresami. Pokud se jednalo o TCP nebo UDP spojení, byla přidána hrana pokaždé, mezi danou zdrojovou MAC a IP adresou, mezi cílovou MAC a IP adresou a mezi zdrojovou a cílovou MAC adresou. Tím bylo dosaženo zobrazení MAC adres a jím příslušících IP adres, které využívají. Jelikož se jedná o velký počet bodů, nejsou IP a MAC adresy více čitelné, nicméně nějaké informace z obrázku lze vyčíst. Zeleně jsou označeny MAC adresy a žlutě IP adresy. K povšimnutí stojí jedna MAC adresa, které využívá spousty dalších IP adres. Dále lze vidět na obrázku 7.21 detail komunikujících MAC adres, kde jádro sítě tvoří 6 MAC adres.



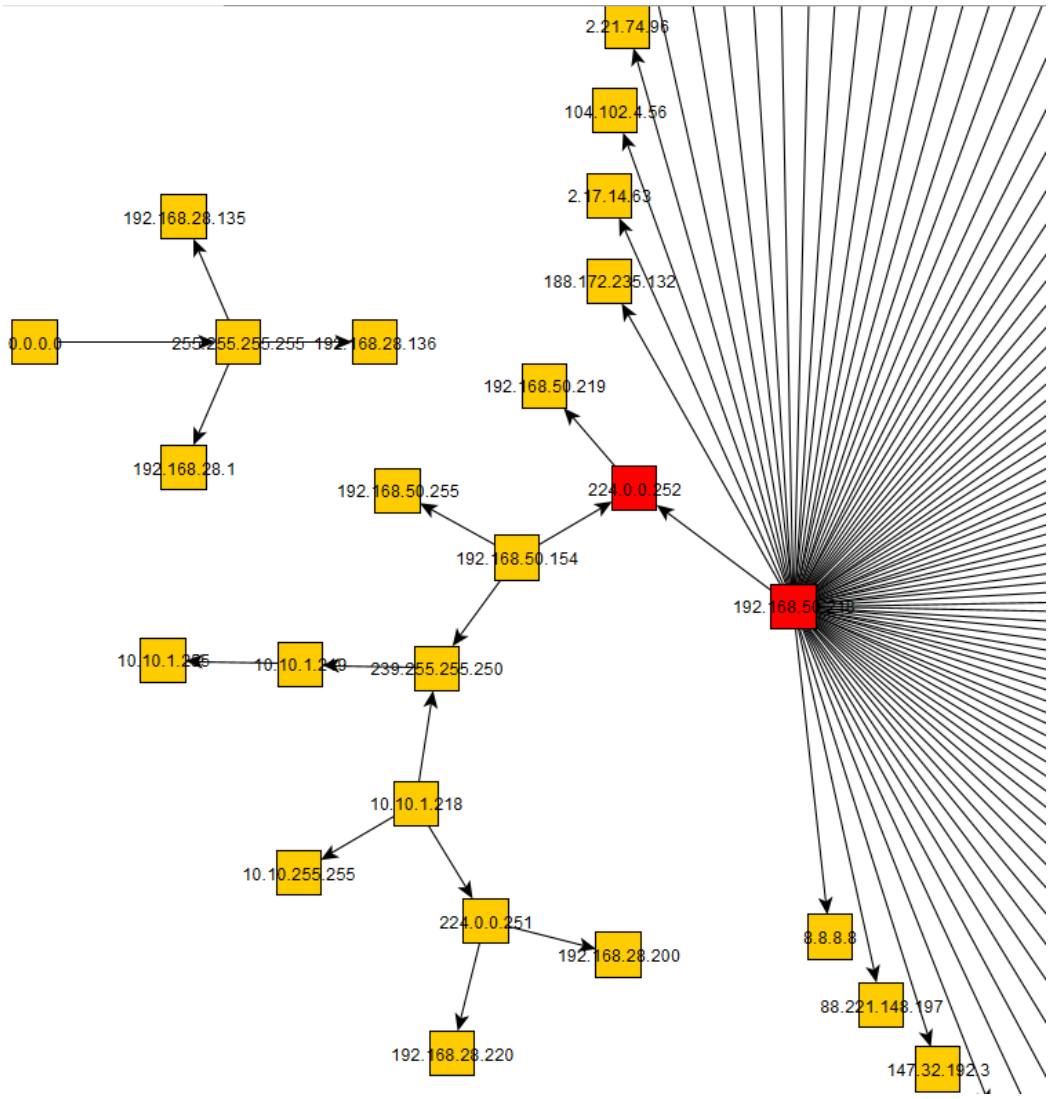
Obrázek 7.20: Zobrazení vizualizace lokální komunikační sítě mezi MAC a IP adresami



Obrázek 7.21: Zobrazení vizualizace lokální komunikační sítě mezi MAC a IP adresami detail



Obrázek 7.22: Zobrazení vizualizace komunikační sítě mezi IP adresami



Obrázek 7.23: Zobrazení vizualizace komunikační sítě mezi IP adresami - detail

Na obrázku 7.22 lze vidět zobrazení vizualizace komunikační sítě mezi IP adresami (3. vrstva), pokud byl mezi nimi přenesen alespoň jeden paket. Na IP vrstvě bylo potvrzeno, že většina komunikace probíhá kolem jednoho hlavního prvku, jako je tomu na obrázku 7.20. Na obrázku 7.23 je zobrazena stejná vizualizace, akorát ve větším detailu vedlejší komunikace, která probíhá mimo většinový provoz. Červeně je vyznačena centrální IP adresa a IP adresa, která s ní jako jediná nějakým způsobem komunikovala.

7.4.3 Zpracování dat

Zpracování dat bylo provedeno následovně:

1. Data, která byla v několika stovkách *pcap* souborech byla pomocí programu **Mergecap**[56] spojena do jednoho velkého souboru *pcap*. Byl vytvořen i druhý způsob, kde se *Mergecap* nepoužíval a docházelo k převodu jednotlivých *pcap* souborů do formátu *csv* a jejich následné sloučení probíhalo pomocí Python skriptu a knihovny *Pandas*, kde byl využit *Pandas* datový rámec. Nicméně ve finálním zpracování byl použit již výše zmíněný **Mergecap**.

2. Data byla následně exportována do formátu *csv* pomocí nástroje **TShark**[19]. Data byla při exportu filtrována tak, aby vzniklý výsledný export obsahoval všechny vlastnosti pro každý jednotlivý paket, který byl exportován a žádná nechyběla. Důvodem byl fakt, že při chybějících datech v určitých vlastnostech a u určitých vzorků dat by bylo nutné přistoupit k metodám, které řeší chybějící data v rámci algoritmů detekce anomálií. Vzniklé výstupní soubory byly dva a obsahují rozdělení dat podle filtru:

   - TCP + UDP
   - všechny pakety (bez filtrace)

   Důvodem exportu do formátu *csv* byl již zavedený zvyk práce s tímto formátem ve spojitosti s jazykem *Python*, nicméně existuje i varianta, kde lze načítat jednotlivé *pcap* souboru pomocí *Python* skriptu přímo pomocí knihovny *Scapy*.

3. V další fázi byl vytvořen ještě jeden soubor ke zpracování pomocí algoritmů detekce anomálií. Jedná se o exportování TCP toků ze souboru, který vznikl v bodě 1. K exportu byl využit nástroj **Wonka Flows Exporter**[64], který ze vstupního souboru (*pcap* soubor) exportuje *TCP* toky do formátu csv.

4. Následně byla data zpracována *Python* skriptem, ve kterém na ně byly aplikovány algoritmy *Isolation Forest* a *Local Outlier Factor*.

5. V této fázi dochází k výběru malého množství anomálií, především ty, které byly označeny jak algoritmem IF a LOF zároveň, nebo také ty, které dosáhly největších hodnot skóre anomality.

6. Poslední fáze je základní analýza dat označených za anomálie. V této fázi se nepouštím do žádných hlubokých analýz a ani se nepřikláním k jednoznačnému označení nějakého útoku, případně neoprávněnému odesílání dat ze zařízení *Huawei*. Dochází zde pouze k rozboru, proč byly data označeny za anomálie.

7.4.4 Aplikace LOF a IF na data z reálného provozu

Aplikace algoritmů se prováděla dohromady na třech vytvořených souborech z původního velkého *pcap* souboru. Jmenovitě na:

- soubor obsahující TCP toky,
- soubor, ve kterém jsou pouze TCP a UDP pakety,
- soubor, který obsahuje všechny původní pakety (žádná filtrace).

Postup zpracování dat v Python skriptu je následující:

1. Načtení vytvořeného souboru pomocí knihovny *Pandas*[44] do datového rámce.
2. Nalezení a smazání duplicitních řádků (vzorků) v datovém rámci.
3. U souboru s TCP a UDP pakety dochází k sjednocení sloupečků *tcp.srcport*/*udp.srcport* a *tcp.dstport*/*udp.dstport*, jelikož na místech, kde se nachází tcp paket, tak vlastnost daného vzorku ve sloupečku *udp.srcport*/*udp.dstport* je prázdná a to samé platí také naopak. Nebylo nutné využívat žádné metody na doplňování prázdných polí, ale naopak se přistoupilo u tohoto souboru k jejich sjednocení.
4. V další fázi dochází k vybrání těch vlastností, které jsou tzv. *categorical features*, jelikož model potřebuje pracovat hlavně s čísly, a tyto vlastnosti tuto podmínku nesplňují. Proto dochází k jejich odhalení a následnému zpracování na číselné hodnoty.
5. Jako další krok je vypočítání základních údajů každé vlastnosti, zejména střední hodnota a variabilita. Tento krok slouží spíše jako informativní a nemá dopad na výsledek modelu.
6. Následně dochází k aplikaci algoritmů *Isolation Forest* a *Local Outlier Factor* na zpracovávaná data.
7. Vytvořené modely pomocí knihovny *sklearn*[45] mají jako návratový parametr možnost zobrazit, která data byla označena za anomálie, a také vypsat jejich skóre anomality. Z označených anomálií jsou vzaty jejich indexy a je provedena operace průnik (*intersection*) těchto dvou množin.
8. V posledním kroku jsou nahrány původní data s jejich výsledky (informace o tom, zda je vzorek anomálie a skóre anomality) do *csv* souboru, který lze posléze nahrát přímo do programu *Microsoft Excel* a seřadit si data například dle skóre anomality.

Aplikace algoritmů na data obsahující TCP a UDP pakety

Soubor obsahující TCP a UDP pakety má 1 029 180 řádků a 12 dimenzí (vlastností), ale ve výsledném zpracování, které se vkládá do modelů, obsahuje soubor po aplikaci smazání duplicit a sjednocení výše zmíněných sloupců velikost 95 003 řádků a 8 dimenzí. To se může zdát na první pohled celkem radikální smazání většiny dat. Může to indikovat nekvalitní datový balík, kde se hodně dat opakuje, což je pravda, nicméně takto velká redukce je způsobena i výběrem menšího počtu dimenzí. Při zvyšování počtu dimenzí by stoupal počet záznamů, které nejsou duplicitní a tudíž by nedošlo k tak velké redukci.

Vlastnost	Střední hodnota	Variabilita
frame.len	290.097123	1.575585e+05
frame.protocols	29.940076	2.628289e+02
eth.src	2.161016	4.109008e+00
eth.dst	3.241434	1.214565e+01
ip.src	31.749829	5.555817e+02
ip.dst	28.339273	4.949138e+02
srcport	26787.579340	8.192822e+08
dstport	30772.025420	8.215025e+08

Tabulka 7.6: Střední hodnot a variabilita jednotlivých vlastností datasetu TCP + UDP

Z tabulky 7.6 lze vidět, že nejnižší variabilitu vykazuje zdrojová a cílová MAC adresa, to je způsobeno jednak jejich počtem, který byl již analyzován v kapitole *Základní rozbor dat*, ale také mírou využívání jednotlivých MAC adres, z tohoto výsledku lze usuzovat, že ne všechny MAC adresy jsou využívány rovnoměrně. Dále stojí za povšimnutí variabilita zdrojových a cílových portů, která je vysoká. Takto vysoká variabilita je způsobena především nerealizováním operace převodu hodnot, ve kterých je daná vlastnost převedena na čísla, jako tomu bylo například u zdrojových a cílových MAC adres. Variabilita je v tomto případě poněkud zavádějící. Ze střední hodnoty jsme naopak schopni říci, že se využívaly více porty s vyšším číselným označením. Důležité je neprováděn ihned jakékoliv závěry pouze z hodnot, které nám říká variabilita. Právě proto nemají tyto údaje vliv na výsledek modelů, aby se předešlo zbytečným chybám.

Vstupní soubor je pojmenován jako "*TCP_UDP.csv*" a výstupní soubor jako "*anomalies_UDP_TCP.csv*". Výstupní soubor si lze snadno načíst ve kterémkoliv programu podporujícím *csv* formát. Výhodou je, že lze vidět základní informace o provozu a k nim i dodatečnou informaci, které ze vzorků označily algoritmy jako anomálie.

V tabulce 7.7 jsou vypsány indexy vzorků a jím odpovídající skóre anomality, které vypočítaly algoritmy *Isolation Forest* a *Local Outlier Factor*. Oba algoritmy vrátily několik podezřelých vzorků s vypočítaným skóre anomality. Tabulka vznikla tak, že byly zpracovány výsledky obou těchto algoritmů a na nich byla provedena operace průnik. Dochází tedy k dvojímu ověření datových vzorků. Z tabulky 7.7 si lze všimnout, že indexy označených anomálií jsou velice blízko u sebe, což značí malé shluky dat.

Index	Číslo rámce	IF skóre	LOF skóre
131	2019	-0.067	-8.540
134	2023	-0.043	-5.745
137	2027	-0.068	-8.461
139	2031	-0.042	-5.736
141	2033	-0.044	-6.053
44182	728696	-0.042	-11.203
44184	728698	-0.048	-10.165
44199	728718	-0.044	-10.417
44200	728719	-0.063	-12.046
44201	728720	-0.068	-16.875
44202	728723	-0.057	-14.629
44203	728724	-0.063	-12.031
44204	728726	-0.052	-11.085
44205	728727	-0.064	-10.163
44209	728732	-0.052	-11.759
44210	728734	-0.067	-13.149
44211	728737	-0.147	-4.737
2624	30339	-0.106	-82.788
67280	1309785	-0.131	-44.342
43102	524379	-0.118	-74.967
4449	50977	-0.039	-5.546
63214	1261240	-0.116	-29.944

Tabulka 7.7: Výsledné skóre algoritmů IF a LOF a odpovídající indexy v souboru

Aplikace algoritmů na veškerá data, obsahuje všechny původní pakety

Soubor obsahující veškerý datová provoz má 1 725 601 řádků a 5 dimenzí (vlastností). Po aplikaci smazání duplicit má výsledný datový soubor se kterým se pracuje 552 193 řádků, tedy téměř třikrát méně, což značně usnadňuje časovou náročnost výpočtu.

Z tabulky 7.8 je patrné, že největší rozptyl má délka datového rámce. Naopak nejmenší rozptyl mají opět MAC adresy s důvodem stejným, jako je uveden při aplikaci na TCP a UDP pakety. Důvodem k aplikaci algoritmů na tato data byl fakt, že jsem nechtěl přistoupit k úplnému odstranění některých datových vzorků, jelikož ty by právě mohly být považovány za anomálie. V tomto případě je to zejména přidání protokolu ARP datového setu (oproti původnímu filtrovanému provozu, kde byly pouze TCP a UDP). Bohužel důsledkem této skutečnosti je fakt, že bylo nutné zajistit společné vlastnosti pro všechny síťové vrstvy. Čím níže se pracuje v síťovém vrstvovém modelu, tím méně lze nalézt společné vlastnosti, které by u jiných vzorků neobsahovaly prázdná pole.

Vstupní soubor je pojmenován jako "*ALL_packets.csv*" a výstupní soubor jako "*anomalies_ALL.csv*". Výstupní soubor si lze snadno načíst ve kterémkoliv programu podporujícím *csv* formát. Výhodou je, že lze vidět základní informace o provozu a k nim i dodatečnou informaci, které ze vzorků označily algoritmy jako anomálie.

V tabulce 7.8 lze vidět seřazené indexy z datového souboru podle jejich skóre anomality, které jim algoritmy IF a LOF přiřadily. Byla vyzkoušena různá úroveň parametru *contamination*, což mělo za následek pouze přibývání počtu detekovaných anomálií. Nicméně anomálie s nejvyšším skóre zůstávaly nepozměněné a přibývaly pouze anomálie s menším skóre.

Vlastnost	**Střední hodnota**	**Variabilita**
frame.len	47.944775	1.827059e+04
frame.protocols	7.791528	2.117150e+02
eth.src	4.874638	1.426380e+01
eth.dst	12.530320	6.341256e+01
frame.time_delta	210343.941098	2.027941e+10

Tabulka 7.8: Střední hodnot a variabilita jednotlivých vlastností datasetu TCP toků

Index	**Číslo rámce**	**IF skóre**	**LOF skóre**
550666	1664257	-0.054	-29.257
551642	1717429	-0.050	-29.278
31382	50974	-0.043	-13.307
226882	443217	-0.041	-11.412
154637	291899	-0.029	-11.664
543405	1647477	-0.026	-11.584
104298	192805	-0.010	-11.171
515788	1582173	-0.009	-11.707
202950	391493	-0.009	-11.603
226608	442634	-0.009	-11.571
210711	407738	-0.009	-11.078
234964	459637	-0.008	-11.718
544479	1649904	-0.002	-11.521
530201	1615215	-0.002	-11.152
129346	241971	-0.000	-11.306

Tabulka 7.9: Výsledné skóre algoritmů IF a LOF a odpovídající indexy v souboru

Aplikace algoritmů na data obsahující pouze TCP toky

Soubor obsahující pouze navázaná a udržovaná TCP spojení byl vytvořen pomocí nástroje **Wonka Flows Exporter**[64] a obsahuje 4 038 záznamů a 11 dimenzí (vlastnosti). Při zpracování tohoto souboru není aplikováno žádné mazání duplicit.

Vlastnost	Střední hodnota	Variabilita
protocol_name	0.876919	1.079586e-01
srcIP	0.000000	0.000000e+00
sport	1928.811540	1.243340e+06
dstIP	15.292719	2.683285e+02
dport	0.880634	1.125751e-01
proto	0.000000	0.000000e+00
push_flag_ratio	94.022288	5.316740e+02
average_len	255.359832	8.906661e+03
average_payload_len	1132.261020	4.412921e+05
pkt_count	14.642397	1.511944e+02
flow_average_inter_arrival_time	49.143388	4.803076e+03

Tabulka 7.10: Střední hodnot a variabilita jednotlivých vlastností datasetu TCP toků

Index	IF skóre	LOF skóre
1859	-0.179	-3.013
1990	-0.175	-2.854
0	-0.169	-2.679
1833	-0.160	-1.381
1832	-0.159	-1.394
2522	-0.134	-1.483
1084	-0.130	-1.546
241	-0.127	-1.351
1440	-0.119	-1.371
2636	-0.091	-1.421
1930	-0.079	-1.843
1877	-0.076	-1.756
1826	-0.074	-1.617
2034	-0.074	-2.225
3116	-0.067	-1.731
731	-0.064	-1.337
3079	-0.058	-1.795
2037	-0.054	-1.497
239	-0.047	-1.670
1939	-0.016	-1.776
630	-0.003	-1.363
1654	-0.001	-1.565

Tabulka 7.11: Výsledné skóre algoritmů IF a LOF a odpovídající indexy v souboru

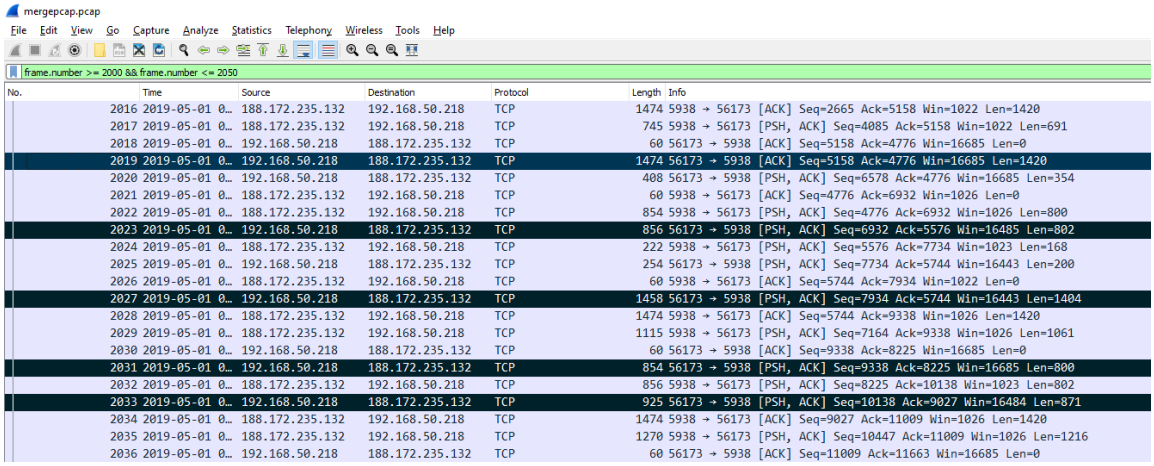
V tabulce 7.10 jsou zapsány základní údaje o datových vlastnostech. K povšimnutí stojí nulová variabilita zdrojové IP adresy, to značí, že k navázání TCP toků byla použita vždy jedna a ta samá IP adresa. Tato informace značí výchozí bránu, přes kterou se komunikuje dále do sítě internet. Zajímavá je také informace o cílovém portu, kde se variabilita moc neliší, to značí spousty stejných cílových portů a mezi nimi zamícháno pár ne zcela běžně používaných. Nulová variabilita u použitého protokolu je způsobena použitím pouze protokolu TCP k navázání spojení.

Z tabulky 7.11 je možné vyčíst informaci o indexech, které měly největší přiřazené skóre anomality. Tato skutečnost nemusí hned automaticky znamenat, že se jedná o útok, je nutné se na TCP toky podívat blíže a zhodnotit tento fakt. V následující sekci dojde k základnímu rozboru detekovaných anomálií.

7.4.5 Rozbor detekovaných anomálií

Rozbor detekce anomálií je prováděn pouze okrajově. V diplomové práci se nepouštím k odvozování závěrů, zda se jedná o útok nebo dokonce o úniky dat v rámci kauzy s firmou *Huawei*. Samotná analýza datového toku by vyšla na další diplomovou práci, proto je udělán pouze základní rozbor a u některých dat je i navrhnuta myšlenka, proč právě tato data byla označena za anomálie. K základní analýze byl použit nástroj ***Wireshark***[20].

Datový balík skládající se pouze z TCP a UDP paketů obsahuje na první pohled dle tabulky 7.7 celkem shluknuté anomálie. První shluk je tvořen kolem čísel rámců 2025 a druhý shluk okolo čísel rámců 728725. Posledních pět řádků v tabulce obsahuje relativně oddělené datové body.



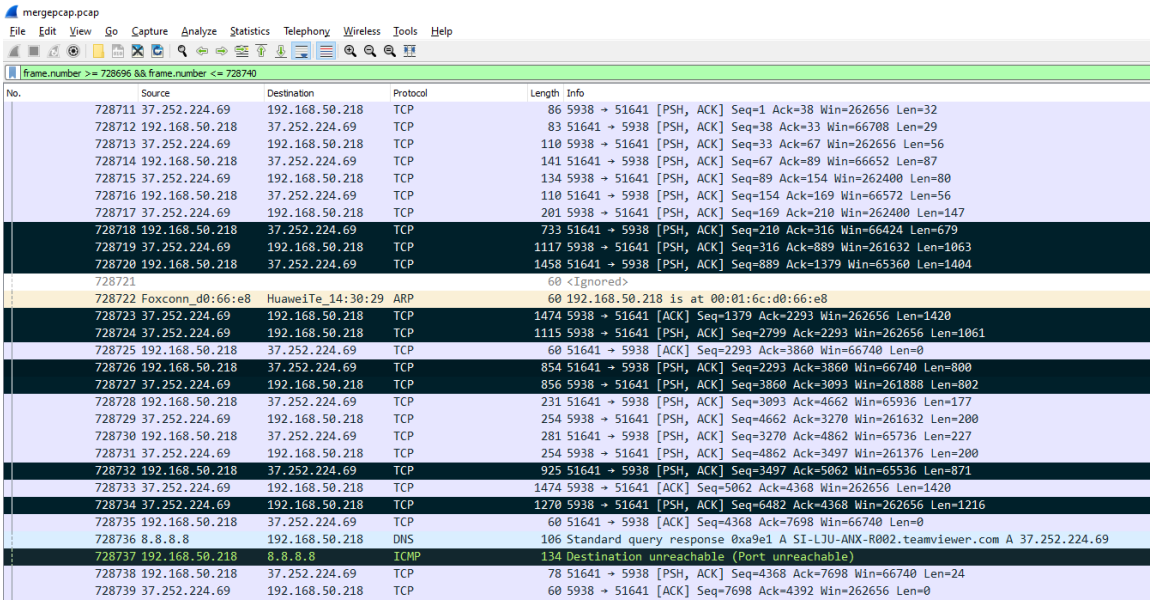
Obrázek 7.24: Zobrazení označených paketů s číslem rámce 2000 až 2050

Na obrázku 7.24 je zobrazen filtr označených paketů dle obou algoritmů. Filtr není aplikován pouze na daná čísla rámců, ale je možné vidět i jejich okolí. Z obrázku je na první pohled patrné, že byly označeny pakety se zdrojovou IP adresou 192.168.50.218 a cílovou IP adresou 188.172.235.132. Při bližším prozkoumání si lze všimnout, proč nebyly označeny i další pakety se stejnou zdrojovou a cílovou adresou. Je to způsobené především délkou rámce, jelikož označené pakety mají jednu z nejdelších délek rámce v rámci tohoto malého shluku.

Zde se projevila síla algoritmu LOF, který je schopen detekovat právě i takovéto anomálie v již vytvořených shlucích, za podmínky, správně zvolené hodnoty parametru *n_neighbors* vzhledem k velikosti vytvořeného shluku. Algoritmus IF si těchto anomálií také všiml, ale přidělil jim o dost menší skóre, než ostatním označeným datovým bodům. Určitou míru anomality zde vykazuje i cílový port, na který je veden provoz. Při pohledu z širší perspektivy (na úrovni TCP toků, ne pouze TCP paketů) bylo zjištěno, že i celý TCP tok, kterého se týkají TCP pakety v rámci tohoto datasetu, byl označen za anomálii i v druhém datasetu. Konkrétně dataset obsahující TCP toky (*tcpFlows.csv*).
V rámci shrnutí byly tyto datové rámce označeny dle mého názoru kvůli méně používanému portu 5938 a také díky velikosti datového rámce.

Datové rámce okolo shluku s číslem rámce 728725 lze vidět na obrázku 7.25. Označené anomálie se soustřeďují na IP adresy 192.168.50.218 a 37.252.224.69. V rámci komunikace s IP adresou 37.252.224.69 se jedná o minoritní provoz, to je první krok k tomu být označen jako anomálie. Při bližším pohledu není ihned patrné, proč ale byly označeny zrovna tyto pakety v rámci malého shluku, který je tvořen těmito dvěma IP adresami. Lepší pohled zobrazuje obrázek 7.26, kde jsou vidět právě označené indexy anomálií. Důvodem je opět řazení dle délky rámce.

Datový rámec s číslem 728737 do výše zmíněného shluku nespadá, byl označen nejvyšším skórem anomality dle algoritmu IF. Zde je vidět, jak algoritmus IF je schopen si poradit právě s takovýmito anomáliemi. Jedná se protokol ICMP, konkrétně o příkaz *ping*, který je poslán na známou IP adresu 8.8.8.8. Toto lze pokládat za důkaz, že algoritmus funguje dle očekávání správně, jelikož objevil v celé řadě datových vzorků jeden, který se liší extrémně od ostatních.

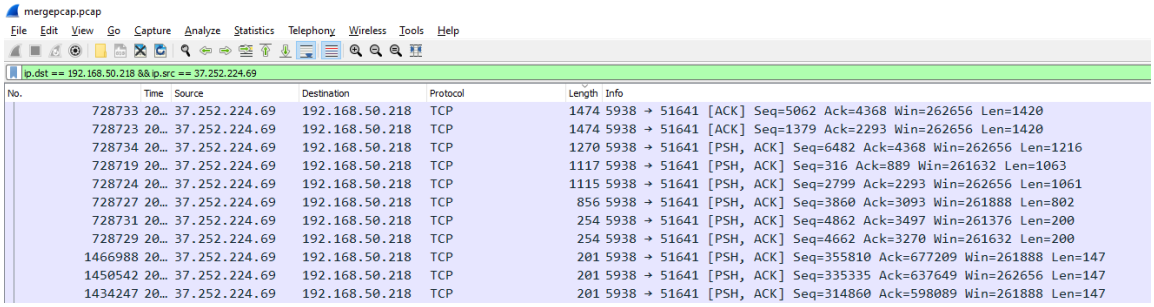

mergepcap.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

frame.number >= 728696 && frame.number <= 728740

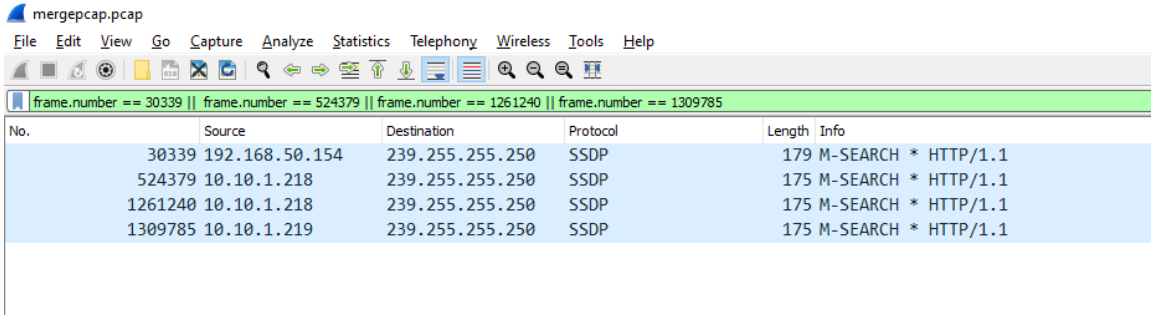
No.	Source	Destination	Protocol	Length	Info
728711	37.252.224.69	192.168.50.218	TCP	86	5938 → 51641 [PSH, ACK] Seq=1 Ack=38 Win=262656 Len=32
728712	192.168.50.218	37.252.224.69	TCP	83	51641 → 5938 [PSH, ACK] Seq=38 Ack=33 Win=66708 Len=29
728713	37.252.224.69	192.168.50.218	TCP	110	5938 → 51641 [PSH, ACK] Seq=33 Ack=67 Win=262656 Len=56
728714	192.168.50.218	37.252.224.69	TCP	141	51641 → 5938 [PSH, ACK] Seq=67 Ack=89 Win=66652 Len=87
728715	37.252.224.69	192.168.50.218	TCP	134	5938 → 51641 [PSH, ACK] Seq=89 Ack=154 Win=262400 Len=80
728716	192.168.50.218	37.252.224.69	TCP	110	51641 → 5938 [PSH, ACK] Seq=154 Ack=169 Win=66572 Len=56
728717	37.252.224.69	192.168.50.218	TCP	201	5938 → 51641 [PSH, ACK] Seq=169 Ack=210 Win=262400 Len=147
728718	192.168.50.218	37.252.224.69	TCP	733	51641 → 5938 [PSH, ACK] Seq=210 Ack=316 Win=66424 Len=679
728719	37.252.224.69	192.168.50.218	TCP	1117	5938 → 51641 [PSH, ACK] Seq=316 Ack=889 Win=261632 Len=1063
728720	192.168.50.218	37.252.224.69	TCP	1458	51641 → 5938 [PSH, ACK] Seq=889 Ack=1379 Win=65360 Len=1404
728721				60	<Ignored>
728722	Foxconn_d0:66:e8	HuaweiTe_14:30:29	ARP	60	192.168.50.218 is at 00:01:6c:d0:66:e8
728723	37.252.224.69	192.168.50.218	TCP	1474	5938 → 51641 [ACK] Seq=1379 Ack=2293 Win=262656 Len=1420
728724	37.252.224.69	192.168.50.218	TCP	1115	5938 → 51641 [PSH, ACK] Seq=2799 Ack=2293 Win=262656 Len=1061
728725	192.168.50.218	37.252.224.69	TCP	60	51641 → 5938 [ACK] Seq=2293 Ack=3860 Win=66740 Len=0
728726	192.168.50.218	37.252.224.69	TCP	854	51641 → 5938 [PSH, ACK] Seq=2293 Ack=3860 Win=66740 Len=800
728727	37.252.224.69	192.168.50.218	TCP	856	5938 → 51641 [PSH, ACK] Seq=3860 Ack=3093 Win=261888 Len=802
728728	192.168.50.218	37.252.224.69	TCP	231	51641 → 5938 [PSH, ACK] Seq=3093 Ack=4662 Win=65936 Len=177
728729	37.252.224.69	192.168.50.218	TCP	254	5938 → 51641 [PSH, ACK] Seq=4662 Ack=3270 Win=261632 Len=200
728730	192.168.50.218	37.252.224.69	TCP	281	51641 → 5938 [PSH, ACK] Seq=3270 Ack=4862 Win=65736 Len=227
728731	37.252.224.69	192.168.50.218	TCP	254	5938 → 51641 [PSH, ACK] Seq=4862 Ack=3497 Win=261376 Len=200
728732	192.168.50.218	37.252.224.69	TCP	925	51641 → 5938 [PSH, ACK] Seq=3497 Ack=5062 Win=65536 Len=871
728733	37.252.224.69	192.168.50.218	TCP	1474	5938 → 51641 [ACK] Seq=5062 Ack=4368 Win=262656 Len=1420
728734	37.252.224.69	192.168.50.218	TCP	1270	5938 → 51641 [PSH, ACK] Seq=6482 Ack=4368 Win=262656 Len=1216
728735	192.168.50.218	37.252.224.69	TCP	60	51641 → 5938 [ACK] Seq=4368 Ack=7698 Win=66740 Len=0
728736	8.8.8.8	192.168.50.218	DNS	106	Standard query response 0xa9e1 A SI-LJU-ANX-R002.teamviewer.com A 37.252.224.69
728737	192.168.50.218	8.8.8.8	ICMP	134	Destination unreachable (Port unreachable)
728738	192.168.50.218	37.252.224.69	TCP	78	51641 → 5938 [PSH, ACK] Seq=4368 Ack=7698 Win=66740 Len=24
728739	37.252.224.69	192.168.50.218	TCP	60	5938 → 51641 [ACK] Seq=7698 Ack=4392 Win=262656 Len=0

Obrázek 7.25: Zobrazení označených paketů s číslem rámce 728696 až 728740



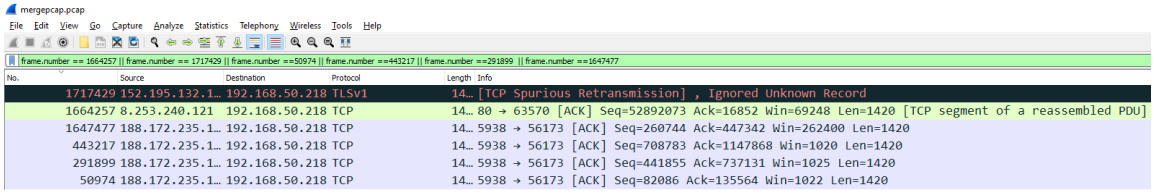
Obrázek 7.26: Zobrazení označených paketů dle IP adres a řazeno dle délky rámce

Datové rámce s čísly 30339, 524379, 1261240, 1309785 dosahují nejvyšších hodnot skóre anomality u obou algoritmů lze vidět na obrázku 7.27. Tyto datové rámce používají k rozesílání v rámci lokální sítě *multicast* adresu 239.255.255.250 a protokol *SSDP*. Nejvyšší skóre mají, protože dochází k jejich vysílání pokaždé z jiné zdrojové IP adresy, a také protože protokol *SSDP* není v této síti moc využíván. Datové rámce byly označeny takto, jelikož splňují přesně podmínky detekce anomalií, je jich málo a liší se hodně.



Obrázek 7.27: Zobrazení označených paketů s nejvyšším skórem anomality

Datový balík skládající se ze všech původních paketů byl analyzován se záměrem nepřijít o žádná data, která by mohla být považována za silné anomálie. Důsledkem tohoto požadavku bylo dosaženo pouze pěti dimenzí.

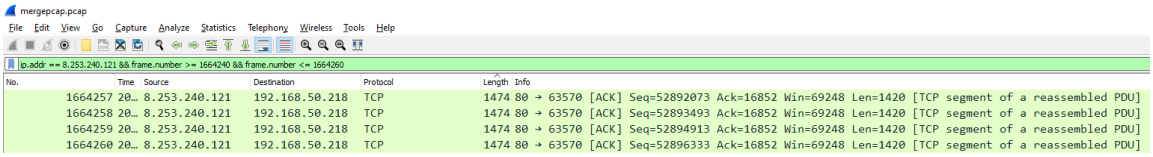


Obrázek 7.28: Zobrazení prvních šesti označených anomálií

Na obrázku 7.28 lze vidět zobrazení prvních šesti označených anomálií s nejvyšším skóre anomality. První záznam s číslem 1717429 byl označen za anomálii především kvůli použitému protokolu. Druhý záznam (z pohledu obrázku/ *Wireshark*) s číslem 1664257 dosáhl nejvyššího skóre anomality u obou algoritmů zároveň. Na první pohled není hned vidět, proč

byl tento záznam označen. Proto se přistoupilo k zobrazení okolí daného záznamu viz. obrázek 7.29, ze kterého ovšem požadovaná informace také nebyla nalezena. Při bližším prozkoumání jsem dospěl k závěru, že tento datový záznam byl označen díky vlastnosti *frame.time_ delta*, která je velice proměnlivá a závislá na předešlých datech. Její hodnota je vidět v exportu, který byl proveden pomocí *TShark*[19].

Poslední čtyři datové rámce obsahují TCP pakety, nejedná se přímo o stejné pakety jako v předchozím nálezu, kde se analyzovaly pouze TCP a UDP pakety, nicméně se jedná o pakety ze stejného TCP toku, který byl navázán.



Obrázek 7.29: Zobrazení okolí anomálie s nejvyšším skóre

Datový balík obsahující pouze TCP toky

V poslední fázi dojde k analýze výstupních hodnot, které byly získány po aplikaci algoritmů přímo na navázané TCP toky v rámci celého záchytu. Analyzovány budou první tři záznamy s nejvyššími hodnotami IF a LOF skóre.

Jmenovitě se jedná o TCP toky dle tabulky

Index	Zdrojová IP	Zdrojový port	Cílová IP	Cílový port
0	192.168.50.218	56173	188.172.235.132	5938
1859	192.168.50.218	51641	37.252.224.69	5938
1990	192.168.50.218	49167	188.172.235.144	5938

Tabulka 7.12: Základní údaje o TCP tocích, které byly označeny

TCP tok s indexem **0** a **1859** vykazovaly podivné chování (byly označeny za anomálie) již v průběhu zpracování datového setu, kde byly použity pouze TCP a UDP pakety. Došlo tedy k dvojímu potvrzení, že se tyto toky chovají jinak, než ostatní. Při bližším pohledu do souboru "anomalies_TCP_Flows" viz. obrázek 7.30 lze vidět, proč byly tyto tři toky označeny za anomálie. Soubor je řazen dle LF skóre, je tedy možné vidět zkoumané toky na prvních třech řádcích. Toky vykazují velmi podobné hodnoty v parametrech "*dstport*", "*push_flag_ ratio*", "*average_ len*" a "*average_ payload_ len*". V těchto parametrech se zároveň velice liší od ostatních navázaných TCP toků. Pokud k tomu přiřadíme také ne zcela často využívaný cílový port 5938, můžeme dojít k závěru, proč byly tyto toky označeny za anomálie.

Column1	protocol_name	src	sport	dst	dport	proto	push_flag_ratio	average_len	average_payload_len	pkt_count	flow_average_inter_arrival_time
1859	unknown	192.168.50.218	51641	37.252.224.69	5938	6	0.6	55.39541112	68.10390851	69080	74.5337
1990	unknown	192.168.50.218	49167	188.172.235.144	5938	6	0.6	58.16875121	77.55171346	25825	66.606
0	unknown	192.168.50.218	56173	188.172.235.132	5938	6	0.6	57.89733202	74.92915469	175001	64.8745
3377	unknown	192.168.50.218	56164	37.252.224.69	5938	6	0	52	0	2	0
3381	unknown	192.168.50.218	56171	37.252.224.69	5938	6	0	52	0	2	0
3380	unknown	192.168.50.218	56169	37.252.224.69	5938	6	0	52	0	2	0
3379	unknown	192.168.50.218	56167	37.252.224.69	5938	6	0	52	0	2	0
3378	unknown	192.168.50.218	56166	37.252.224.69	5938	6	0	52	0	2	0.0005
1988	unknown	192.168.50.218	49165	213.227.168.190	5938	6	0.273	44.63636364	23.27272727	11	0.0006
1855	unknown	192.168.50.218	51633	188.172.219.157	5938	6	0.273	44.63636364	23.27272727	11	0.0005
1856	unknown	192.168.50.218	51636	188.172.219.157	5938	6	0.273	44.63636364	23.27272727	11	0.0013
1854	unknown	192.168.50.218	51630	188.172.219.157	5938	6	0.273	44.63636364	23.27272727	11	0.0011
1857	unknown	192.168.50.218	51639	188.172.219.157	5938	6	0.273	44.63636364	23.27272727	11	0.0012
1989	unknown	192.168.50.218	49166	185.188.32.6	5938	6	0.273	177.5454545	453	11	0.0013
1858	unknown	192.168.50.218	51640	185.188.32.2	5938	6	0.273	180.9090909	462.7272727	11	0.0013
2023	unknown	192.168.50.218	49730	152.195.132.156	443	6	0.051	1255.89728	3497.492744	116598	29.3076
2038	unknown	192.168.50.218	50104	152.195.132.156	443	6	0.055	1239.522503	3450.586263	31885	11.1241
2027	unknown	192.168.50.218	49734	152.195.132.156	443	6	0.05	1231.868353	3429.282781	30635	12.3421
2025	unknown	192.168.50.218	49732	152.195.132.156	443	6	0.051	1228.867748	3420.003437	28801	12.1146
2040	unknown	192.168.50.218	50108	152.195.132.156	443	6	0.05	1238.022806	3446.456058	28720	10.3748
2026	unknown	192.168.50.218	49733	152.195.132.156	443	6	0.05	1230.233353	3423.917552	27302	12.4677

Obrázek 7.30: Zobrazení prvních tří toků označených za anomálie

Závěr

Hlavním cílem diplomové práce bylo prozkoumat možnosti detekce anomálií v sítích. Nejdříve byly představeny jednotlivé typy hrozeb, se kterými se lze setkat v dnešních komunikačních sítích. Všechny tyto typy hrozeb/útoků se chovají z hlediska síťového provozu jako anomálie. Dále bylo představeno základní rozdělení systémů pro detekci útoků.

V další fázi byl vytvořen základní přehled metod detekce anomálií, ze kterých byly zvoleny dva algoritmy, spadající do kategorie strojové učení, učení bez učitele. Nutno podotknout, že v době vytváření uceleného souhrnu metod detekcí anomálií nebylo rozhodnuto o způsobu, jakým se budou anomálie detekovat. K vybrání vhodné metody došlo, až po všeobecném prostudování již známých metod. Metody byly zvoleny s ohledem na již dosažené výsledky ve světě, ale také díky zalíbení v samotné myšlence fungování daných algoritmů. Po zvolení vhodných algoritmů k implementaci, došlo k jejich prostudování a následnému sepsání. Bylo nutné načerpat podrobnější informace k jejich následné realizaci.

Algoritmy *Isolation Forest* a *Local Outlier Factor* byly nejdříve otestovány na syntetických datech *KDDCUP99*. Tato data obsahují značku a je tedy možné na nich porovnávat výkony jednotlivých algoritmů. Algoritmus *Isolation Forest* si vedl při detekci anomálií velice dobře, dosáhl hodnoty AUC téměř 100 %. Samotný dataset mu velice dobře sedl, čemuž odpovídaly i výstupní metriky. Dokázal odhalit všechny druhy útoku, které byly obsaženy v datasetu. Naopak algoritmus *Local Outlier Factor* si na tomto konkrétním datovém balíku nevedl příliš dobře. Dosahoval hodnoty AUC okolo 50 %, což je jeden z nejhorších případů, odpovídá to čisté náhodě, něco jako hod mincí. Při bližším prozkoumání jsem došel k závěru, že je problém ve velkém množství určitých útoků. Algoritmus si totiž vedl celkem obstojně v identifikaci normálních dat, a menších počtů útoků. Právě proto jsem ho zcela nevyřadil, ale aplikoval jsem ho i nadále na reálná data. Ne všechny datasety jsou vhodné pro každý algoritmus.

Následně byly algoritmy implementovány na reálná data. Data byla poskytnuta ze ČVUT FEL a jednalo o záchyt, který byl realizován během celého jednoho měsíce. Nejprve bylo nutné se s daty pořádně seznámit a vymyslet strategii, jakým způsobem bude probíhat detekce anomálií. Při studii problému detekce anomálií v sítích byly získány potřebné informace, dle kterých byl filtrován datový balík pomocí nástroje *TShark* do několika různých filtrací. Bylo přihlédnuto také k faktu, že ze samotného datového balíku ve formátu *pcap* se lze dozvědět informace o průběhu TCP toků. Bylo potřeba najít nebo implementovat vhodný nástroj k jejich extrakci z několika miliónů paketů. Ve výsledku se algoritmy *Isolation Forest* a *Local Outlier Factor* testovaly na třech datových balících. Při aplikaci algoritmů jsem zohlednil špatné výkony algoritmu na syntetických datech *Local Outlier Factor* a rozhodl se pro dvojí ověřovací detekci. To je způsob, kdy se datový balík zpracuje oběma algoritmy, které vrátí

hodnoty indexů datových vzorků, které byly označeny. Následně byla realizována operace průnik a docházelo ke zpracování pouze těch datových vzorků, které byly označeny oběma algoritmy. Tím bylo dosaženo jednak zmenšení počtu detekovaných anomálií, což může být samozřejmě špatně, nicméně cílem této práce nebyl úplný rozbor všech detekovaných anomálií. Tento přístup splnil svůj účel.

Výsledky aplikace algoritmů na reálná data jsou chvalitebné. Došlo totiž k potvrzení domněnky, že algoritmus *Local Outlier Factor* nepodává špatný výkon ve všech datasetech. Oba algoritmy se jednoznačně shodly na označení jednotlivých datových vzorků za anomálie. Docházelo k přiřazení nejvyšších hodnot skóre anomality od obou algoritmů. Po úplném prozkoumání aplikace na oba dva filtrované datové balíky a třetí, který byl vytvořen, bylo dosaženo několikanásobné schody vždy napříč minimálně dvěma datovými balíky. To znamená, že pakety, které byly označeny v jednom datovém balíku za anomálie odpovídaly paketům, ze kterých se skládají celé označené TCP toky. Anomálie byly tedy úspěšně detekovány. Důležité je zdůraznit, že se jedná o anomálie z pohledu dat. To znamená, že se nemusí jednat o žádný druh útoku ani pokus o únik dat. V této fázi, kdy máme detekovány datové vzorky za anomálie se většinou posílají k dalšímu zpracování bezpečnostním expertům.

Literatura

[1] A. Hussain, J. Heidemann, and C. Papadopoulos. *A framework for classifying denial of service attacks, Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.* ACM New York, NY, USA, 2003, pp. 99–110.

[2] A. Valdes, K. Skinne. *Adaptive model-based monitoring for cyber attack detection.* Recent Advances in Intrusion Detection Toulouse, France, 2000.

[3] AGGARWAL, C. C. *Data Mining: The Textbook.* Springer Publishing Company, Incorporated, 2015. ISBN 3319141414.

[4] Animesh Patcha, Jung-Min Park. *An overview of anomaly detection techniques: Existing solutions and latest technological trends.* 2007. ISSN 1389-1286.

[5] C. Kruegel, D. Mutz, W. Robertson, F. Valeur. *Bayesian event classification for intrusion detection.* Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas, 2003.

[6] Carnegie Mellon University. *CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://resources.sei.cmu.edu/asset_files/WhitePaper/1996_019_001_496172.pdf#page=123>.

[7] Chandola, V.; Banerjee, A.; Kumar, V. *Anomaly Detection: A Survey. ACM.* Computing Surveys, 2009.

[8] Chire. *Local outlier factor* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://en.wikipedia.org/wiki/Local_outlier_factor#/media/File:LOF-idea.svg>.

[9] Cloudflare, Inc. *What is a Denial-of-Service (DoS) Attack?* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/>.

[10] D. Anderson, T. Frivold, A. Tamaru, A. Valdes. *Next generation intrusion detection expert system (NIDES), Software Users Manual, Beta-Update release.* Science Laboratory, SRI International, Menlo Park, CA, USA, 1994. SRI-CSL-95-0.

[11] D. Anderson, T. Frivold, A. Tamaru, A. Valdes. *Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES).* Science Laboratory, SRI International, Menlo Park, CA, USA, 1995. SRI-CSL-95-06.

[12] D.-Y. Yeung, Y. Ding. *Host-based intrusion detection using dynamic and static behavioral models, Pattern Recognition 36.* (2003) 229–243.

[13] David Heckerman. *A Tutorial on Learning With Bayesian Networks.* Microsoft Research Advanced Technology Division Microsoft Corporation One Microsoft Way Redmond, WA 98052, 1995. MSR-TR-95-06.

[14] DING, Z.-G. – DU, D. – FEI, M. An isolation principle based distributed anomaly detection method in wireless sensor networks. *International Journal of Automation and Computing.* 11 2014, 12, s. 402–412. doi: 10.1007/s11633-014-0847-9.

[15] Doc. Ing. Leoš Boháč, Ph.D. *Pokročilé síťové technologie - Transmission Control Protocol – TCP, Sockets* [online]. ČVUT, Fakulta elektrotechnická, katedra telekomunikačních technologií. [cit. 15. 04. 2020]. Dostupné z: <https://moodle.fel.cvut.cz/pluginfile.php/109838/course/section/29344/2-prednaska-TCP-1.pdf>.

[16] DOROTHY E. DENNING . *An Intrusion-Detection Model* [online]. in IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222-232, Feb. 1987. [cit. 8. 12. 2019].

[17] F-Secure. *Denial of Service (DoS)* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://www.f-secure.com/v-descs/articles/denial-of-service.shtml>.

[18] F. T. Liu, K. M. Ting and Z. Zhou. *Isolation Forest.* Eighth IEEE International Conference on Data Mining, Pisa, 2008, pp. 413-422, doi: 10.1109/ICDM.2008.17.

[19] Gerald Combs. *TShark [software]* [online]. [cit. 23. 5. 2020]. Dostupné z: <https://www.wireshark.org/docs/man-pages/tshark.html>.

[20] Gerald Combs. *Wireshark [software]* [online]. [cit. 5. 23. 2020]. Dostupné z: <https://www.wireshark.org/about.html>.

[21] GHORBANI, Ali, Wei LU a Mahbod TAVALLAEE. *Network intrusion detection and prevention: concepts and techniques.* New York: Springer, c2010. Advances in information security. ISBN: 978-0-387-88771-5.

[22] Google Developers. *Machine Learning Crash Course* [online]. [cit. 23. 5. 2020]. Dostupné z: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>.

[23] Gordon Lyon. *NMap* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://nmap.org/book/man.html#man-description>.

[24] HARIRI, S. – KIND, M. – BRUNNER, R. Extended Isolation Forest. 11 2018.

[25] H.H. Hosmer. Security is fuzzy!: applying the fuzzy logic paradigm to the multipolicy paradigm, in: Proceedings of the 1992–1993 Workshop on New Security Paradigms Little Compton, RI, United States, 1993.

[26] Imperva. *Buffer Overflow Attack* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://www.imperva.com/learn/application-security/buffer-overflow/>.

[27] J.E. Dickerson, J.A. Dickerson. *Fuzzy network profiling for intrusion detection, in: Proceedings of the 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS).* Atlanta, GA, 2000, pp. 301–306.

[28] Jeanlagi. *Draw by own force, CC BY 3.0* [online]. [cit. 12. 04. 2020]. Dostupné z: <https://commons.wikimedia.org/w/index.php?curid=3910972>.

[29] Jiří Jarkovský, Simona Littnerová. *Vícerozměrné statistické metody* [online]. [cit. 12. 04. 2020]. Dostupné z: <http://www.iba.muni.cz/esf/res/file/bimat-prednasky/vicerozmerne-statisticke-metody/VSM-08.pdf>.

[30] J.R. Quinlan. C4.5: Programs for Machine Learning, Morgan Kaufman, Los Altos, CA, 1993.

[31] Juniper. *ip-sweep [online]. Juniper Networks, 2020.* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://www.juniper.net/documentation/en_US/junos/topics/topic-map/security-ip-sweep-and-port-option.html>.

[32] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, P. Dokas. *The MINDS - Minnesota intrusion detection system, in: Next Generation Data Mining.* MIT Press, Boston, 2004.

[33] M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander. *LOF: identifying density-based local outliers, in: Proceedings of the ACM SIGMOD International Conference on Management of Data.* Dallas, TX, 2000, pp. 93–104.

[34] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang. *A novel anomaly detection scheme based on principal component classifier, in: Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop.* Melbourne, FL, USA, 2003, pp. 172–179.

[35] MIT Lincoln Lab. *KDD Cup 1999 Data* [online]. University of California, Irvine. [cit. 23. 05. 2020]. Dostupné z: <https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.

[36] MIT Lincoln Lab. *KDD Cup 1999 Data Column Names* [online]. University of California, Irvine. [cit. 23. 05. 2020]. Dostupné z: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup.names>.

[37] Mitre.org. *CVE. Common vulnerabilities and exposures* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://cve.mitre.org/>.

[38] MQ . *Lekce 8 - Rozhodovací stromy v Pythonu* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://www.itnetwork.cz/python/neuronove-site/rozhodovaci-stromy-v-pythonu>.

[39] M.V. Mahoney, P.K. Chan. *PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic.* . Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL, USA, Technical Report CS2001-4, April 2001.

[40] M.V. Mahoney, P.K. Chan. *Learning Models of Network Traffic for Detecting Novel Attacks Computer Science Department.* . Florida Institute of Technology CS-2002-8, August 2002.

[41] M.V. Mahoney, P.K. Chan. *Learning nonstationary models of normal network traffic for detecting novel attacks.* . Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, 2002, pp. 376–385.

[42] N. Ye, Y.Z.C.M. Borror. *Robustness of the Markov-chain model for cyber-attack detection.* IEEE Transactions on Reliability 53 (2004) 116–123.

[43] NortonLifeLock. *What is a computer worm, and how does it work?* [online]. Norton. [cit. 15. 04. 2020]. Dostupné z: <https://us.norton.com/internetsecurity-malware-what-is-a-computer-worm.html>.

[44] TEAM, T. pandas-dev/pandas: Pandas, February 2020. Dostupné z: <https://doi.org/10.5281/zenodo.3509134>.

[45] PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research.* 2011, 12, s. 2825–2830.

[46] Pedregosa, F. et al. *Scikit-learn: Machine Learning in Python - Fetch KDDCUP99* [online]. [cit. 25. 05. 2020]. Dostupné z: <https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_kddcup99.html#sklearn.datasets.fetch_kddcup99>.

[47] Pedregosa, F. et al. *Scikit-learn: Machine Learning in Python - LabelEncoder* [online]. [cit. 25. 05. 2020]. Dostupné z: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>.

[48] Pedregosa, F. et al. *Scikit-learn: Machine Learning in Python - LabelEncoder* [online]. [cit. 25. 05. 2020]. Dostupné z: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>.

[49] PEDREGOSA, F. et al. *Scikit-learn: Machine Learning in Python* [online]. [cit. 5. 23. 2020]. Dostupné z: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>.

[50] R. Grossman. *Data Mining: Challenges and Opportunities for Data Mining During the Next Decade.* 1997.

[51] ROSENBLATT, F. F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review.* 1958, 65 6, s. 386–408.

[52] S. Hansman and R. Hunt. *A taxonomy of network and computer attacks, Computers Security 24.* (2005), no. 1, 31–43.

[53] S. Ramaswamy, R. Rastogi, K. Shim. *Efficient algorithms for mining outliers from large data sets, in: Proceedings of the ACM SIGMOD International Conference on Management of Data.* Dallas, TX, USA, 2000, pp. 427–438.

[54] S. Staniford, J.A. Hoagland, J.M. McAlerney. *Practical automated detection of stealthy portscans.*

[55] Sarang Narkhede. *Understanding AUC - ROC Curve* [online]. [cit. 23. 5. 2020]. Dostupné z: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>.

[56] Scott Renfro. *Mergecap [software]* [online]. [cit. 23. 5. 2020]. Dostupné z: <https://www.wireshark.org/docs/man-pages/mergecap.html>.

[57] S.E. Smaha, Haystack. *An intrusion detection system, in: Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference,* [online]. [cit. 8. 12. 2019].

[58] Sergio Santoyo. *A Brief Overview of Outlier Detection Techniques* [online]. [cit. 5. 16. 2020]. Dostupné z: <https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561>.

[59] S.M. Bridges, R.B. Vaughn. *Fuzzy data mining and genetic algorithms applied to intrusion detection, in: Proceedings of the National Information Systems Security Conference.* Baltimore, MD, 2000.

[60] Stefan Axelsson. *Intrusion Detection Systems: A Survey and Taxonomy* [online]. Department of Computer Engineering Chalmers University of Technology Goteborg, Sweden. [cit. 8. 12. 2019]. Dostupné z: <http://neuro.bstu.by/ai/To-dom/My_research/Paper-0-again/For-research/D-mining/Anomaly-D/Intrusion-detection/taxonomy.pdf>.

[61] Surya Priy. *Clustering in Machine Learning* [online]. [cit. 12. 04. 2020]. Dostupné z: <https://www.geeksforgeeks.org/clustering-in-machine-learning/>.

[62] Surya Priy, Abhishek Rajput. *Fuzzy Logic | Introduction* [online]. [cit. 3. 4. 2020]. Dostupné z: <https://www.geeksforgeeks.org/fuzzy-logic-introduction/>.

[63] TAN, S. – TING, K. – LIU, F. T. Fast Anomaly Detection for Streaming Data. s. 1511–1516, 01 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-254.

[64] Victor Pereira. *Wonka Flows Exporter* [online]. [cit. 23. 5. 2020]. Dostupné z: <https://github.com/fichtner/flows_to_weka>.

[65] Weiyu Zhang, Qingbo Yang, Yushui Geng . *A Survey of Anomaly Detection Methods in Networks* [online]. International Symposium on Computer Network and Multimedia Technology (CNMT). [cit. 8. 12. 2019].

[66] Weng, Y. – Liu, L. A Collective Anomaly Detection Approach for Multidimensional Streams in Mobile Service Security. *IEEE Access.* 2019, 7, s. 49157–49168.

[67] WHITLEY, D. A Genetic Algorithm Tutorial. *Statistics and Computing.* 10 1998, 4. doi: 10.1007/BF00175354.

[68] Wikipedia. *Ping of death* [online]. [cit. 15. 04. 2020]. Dostupné z: <https://en.wikipedia.org/wiki/Ping_of_death>.

[69] W.W. Cohen. Fast effective rule induction, in: Proceedings of the 12th International Conference on Machine Learning, Tahoe City, CA, 1995, pp. 115–123.

[70] yWorks. *yEd [software]* [online]. [cit. 23. 5. 2020]. Dostupné z: <https://www.yworks.com/products/yed#>.

Příloha A

Seznam použitých zkratek

ALAD ApplicationLayer Anomaly Detector

AML Anti Money Laundering

API Application Programming Interface

AUC Area under the ROC Curve

CVE Common Vulnerabilities and Exposures

DDoS Distributed Denial Of Service

DNS Domain Name Server

DoS Denial Of Service

EM expectation-maximization

FIRE Fuzzy Intrusion Recognition Engine

FN False Negative

FP False Positive

FTP File Transfer Protocol

HIDS Host Intrusion Detection System

HTTP Hypertext Transfer Protocol

HW Hardware

ICMP Internet Control Message Protocol

IDES Intrusion Detection Expert System

IDS Instrusion Detection System

IF Isolation Forest

IP Internet Protocol

KNN k - Nearest Neighbors

LERAD Learning Rules for Anomaly Detection

LOF Local Outlier Factor

MAC Media Access Control

MINDS The Minnesota Intrusion Detection System

MITM Man in the Middle

MTU Maximum Transmission Unit

NFS Name File Server

NIDES Next-GenerationIntrusion Detection Expert System

NIDS Network Intrusion Detection System

NMap Network Mapper

PCA Principal Component Analysis

PHAD Packet Header Anomaly Detector

PLC Programmable Logic Controllers

ROC Receiver Operating characteristic Curve

RPC Remote Procedure Call

SAINT Security Administrator's Integrated Network Tool

SMTP Simple Mail Transfer Protocol

SPADE Statistical Packet Anomaly Detection Engine

SQL Structured Query Language

SW Software

TCP Transmission Control Protocol

TP True Positive

UDP User Datagram Protocol

Příloha B

Obsah přiloženého CD

- /
  - `Text` ..... text diplomové práce
    - `DP_Andera.pdf` ..... diplomová práce ve formátu PDF
    - `DP_Andera_tex` ..... diplomová práce ve formátu TEX
  - `Anomaly Detection Python` ..... Programová část
    - `Zdrojové kódy` ..... zdrojové kódy
      - `Isolation Forest/KDDCUP99` ..... zdrojové kódy aplikace IF na KDDCUP99
      - `Local Outlier Factor/KDDCUP99` ..... zdrojové kódy aplikace LOF na KDDCUP99
      - `kddcup99_basicDataProcessing` ..... zdrojové kódy základního zpracování
      - `Huawei` ..... zdrojové kódy aplikace na reálná data
        - `dataHuawei` ..... exportovaná data ve formátu csv
        - `img` ..... obrázky vizualizace sítě
      - `Pcap Data` ..... reálná data v původním formátu