Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Ústav aplikované informatiky

Detekce protokolu BitTorrent v reálném síťovém provozu prostřednictvím protokolu IPFIX/NETFLOW

Bakalářská práce

Luděk Scholz

Školitel: Ing. Jan Fesl Ph.D.

České Budějovice 2020

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Luděk Scholz
(jméno, příjmení, tituly)

Obor – zaměření studia: Aplikovaná informatika

Katedra/ústav PřF JU, kde bude práce vypracována a obhájena: UAI
Školitel: Ing. Jan Fesl Ph.D.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF JU: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Detekce protokolu BitTorrent v reálném síťovém provozu prostřednictvím protokolu IPFIX/NETFLOW

Úkoly práce:

- Vytvořit literární rešerši na téma maskování a detekce protokolů.
  1. Úvod do problematiky zneužití a maskování síťového provozu.
  2. Popis metod používaných pro detekci P2P systémů (heuristiky, statistické metody a metody strojového učení).
  3. Detailní popis architektury systému BitTorrent včetně popisu metod pro maskování provozu.

Cíle práce:

- Navrhnout metodiku pro detekci protokolu BitTorrent v reálém síťovém provozu, včetně jejího detailního popisu.
- Implementovat a otestovat detektor provozu, který bude založen na výše zmíněné metodice, včetně ověření úspěšnosti detekce vyplývající z reálných měření.

Základní doporučená literatura:

[1] Bashir, Ahmed & Huang, Changcheng & Nandy, Biswajit & Seddigh, Nabil. (2013). *Classifying P2P activity in Netflow records: A case study on BitTorrent.* IEEE International Conference on Communications. 3018-3023. 10.1109/ICC.2013.6655003.

[2] A. M. Gossett, I. Papapanagiotou and M. Devetsikiotis, "An apparatus for P2P classification in Netflow traces," *2010 IEEE Globecom Workshops*, Miami, FL, 2010, pp. 1361-1366.
doi: 10.1109/GLOCOMW.2010.5700160

[3] Shang-Chiuan Su, Yi-Ren Chen, Shi-Chun Tsai, and Yi-Bing Lin, "*Detecting P2P Botnet in Software Defined Networks,*" Security and Communication Networks, vol. 2018, Article ID 4723862, 13 pages, 2018. https://doi.org/10.1155/2018/4723862.

[4] Bashir, Ahmed. (2012). *Classifying P2P Activities in Netflow Records*. Ottawa, Ontario: Carleton University. ISBN: 978-0-494-93490-6

[5] Petráček, Martin. *Detekce anomálií založená na strojovém učení v reálném síťovém provozu.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

[6] Dillon, C. (2014). *Peer-to-Peer Botnet Detection Using NetFlow*. University of Amsterdam.

[7] Gomes, J. V., Inacio, P. R. M., Pereira, M., Freire, M. M., and Monteiro, P. P. (2013). *Detection and classification of peer-to-peer traffic: A survey.* ACM Comput. Surv. 45, 3, Article 30 (June 2013), 40 pages. DOI: http://dx.doi.org/10.1145/2480741.2480747

Financování práce .....

Školitel práce .....podpis: .....

U externích vedoucích fakultní garant práce .....podpis: .....

Garant oboru bak. studia (nepožaduje se u oboru biologie) .....podpis: .....

Vedoucí katedry/ústavu PřF JU, kde proběhne obhajoba.....podpis: .....

Případný souhlas vedoucího ústavu AV .....podpis: .....

V Českých Budějovicích dne .....Podpis studenta .....

Bibliografické údaje

Scholz, L., 2020: Detekce protokolu BitTorrent v reálném síťovém provozu prostřednictvím protokolu IPFIX/NETFLOW. [BitTorrent protocol detection with use of IPFIX, Bc. Thesis, in Czech] – Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Abstrakt

Hlavním cílem této bakalářské práce je tvorba metodiky pro detekci protokolu BitTorrent na základě analýzy síťového provozu prostřednictvím protokolu IPFIX. Součástí práce je i teoretická část s popisem základních pojmů a problémů. V druhé části je samotná metodika a její aplikace. Závěrem práce je otestování vytvořené metodiky.

Klíčová slova

bittorrent, torrent, peertopeer, p2p, ipfix, detekce, analýza, chování sítě, metodika

Abstract

The main goal of this bachelor thesis is to create a methodology for the detection of the BitTorrent protocol based on the analysis of network traffic with use of IPFIX. Theoretical part of the thesis describe basic concepts and problems. The second part is the methodology itself and its application. The conclusion of this thesis is the testing of the developed methodology.

Keywords

bittorrent, torrent, peertopeer, p2p, ipfix, detection, analysis, network behavior, metodology

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne 2.4.2020 Luděk Scholz

Poděkování

Děkuji vedoucímu mé bakalářské práce Ing. Janu Feslovi za vedení a odborné rady a své rodině za podporu při studiu.

Obsah

1 Úvod

Kvůli neustálému nárůstu velikosti dat a tím i nárůstu náročnosti na datový tok, bylo zapotřebí ulehčit serverům od jejich zátěže. Proto také vznikl nový protokol BitTorrent využívající odlišnou architekturu, a tou je P2P (z anglického Peer to Peer). Tato architektura umožňuje klientům nezávisle mezi sebou posílat data a tím ulehčit serveru od zátěže. Tato metoda se rychle rozšířila a díky možnosti dosáhnout mnohonásobně větší rychlosti datového přenosu se P2P sdílení souborů stalo velmi oblíbeným nástrojem pro nelegální distribuci obsahu.

BitTorrent je protokol velmi agresivní, navazuje stovky, až tisíce spojení v krátkém časovém úseku, a tím zahlcuje síť. Proto také vznikla potřeba tento protokol v sítích odhalovat a v místech kde se zneužívá i blokovat. Avšak detekce již není jen otázkou zablokování konkrétních portů, nebo sledování obsahu paketů, protože programy využívající BitTorrent se snaží stále zlepšovat v maskování provozu, ať už se jedná o šifrování, nebo například změny portů.

První část práce, tedy teoretická část, je zaměřena na téma detekce a maskování BitTorrentu, kde v úvodu je popsán samotný protokol BitTorrent, jeho dopady na síť a také protokol NetFlow/IPFIX. Dále jsou zde popsány metody používané pro maskování BitTorrentu, ale i metody pro samotnou detekci.

Druhá část práce, tedy praktická část, je zaměřená na tvorbu vlastní metodiky na detekci protokolu BitTorrent. Metodika se skládá ze třech částí. Z odchytu dat, zpracování dat a následné analýzy dat. Samotná analýza dat je naprogramovaný vlastní detektor provozu, využívající zachycená data. Tento detektor je vytvořen na základě zkoumání reálných dat, v souvislosti se znaky chování BitTorrentu. Součástí praktické části je i otestování detektoru na reálných datech.

2 Úvod do problematiky

Pro lepší pochopení celé problematiky je zapotřebí popsat základní pojmy, vysvětlit mechanismy fungování a popsat některé systémy související s tématem.

2.1 Architektury sítě

Architektura „Klient-Server“

Tato architektura je jedním ze základních a hlavních kamenů síťové technologie. Základ této architektury je rozvržení „klientských“ zařízení nekomunikujících mezi sebou ale navazující komunikace se serverem.

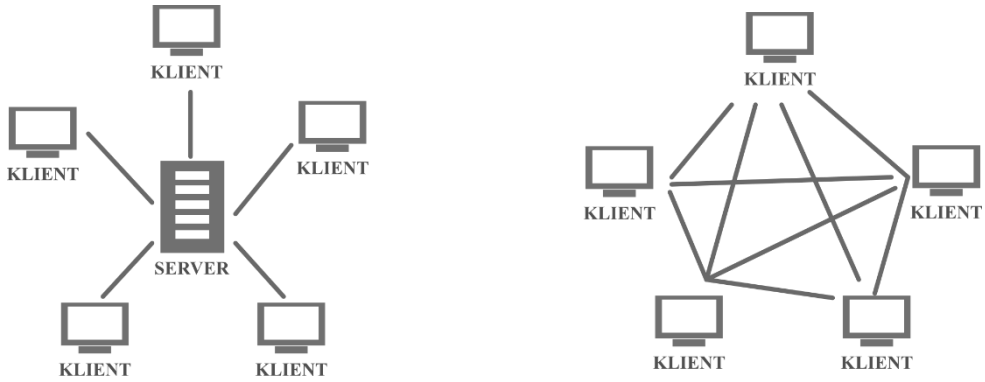
Nejčastěji se s touto technologií můžeme setkat při použití protokolů HTTP nebo SMTP, které jsou používány při klasickém prohlížení webových stránek a posílání emailů. Tato architektura bývá také často využívána ve firemních aplikacích, kde je potřeba centralizovat systém a ulehčit koncovým stanicím od zátěže.

Architektura „Klient-Klient“

Také nazývaná jako peer-to-peer nebo zkráceně jen P2P, je topologie sítě kde klienti nenavazují připojení na server, ale připojují se navzájem mezi sebou. V tomto modelu se rozprostírá zátěž na všechny klienty. Díky tomu se mohou snížit náklady pro provozovatele aplikací.

Rychlosti stahování dat v modelu P2P obvykle dosahují řádově výších hodnot při nižších provozních nákladech. Proto je také architektura P2P v oblibě pro nelegální sdílení obsahu. Architekturu P2P využívá řada aplikací s využitím různých protokolů, ať už se jedná o zkoumaný BitTorrent, nebo Bitcoin či Windows Update. [1, 2] Aplikace Skype také využívala P2P, ale v současnosti se Skype přesouvá z P2P na Cloudové řešení. [3] P2P využívá také řada dalších

aplikací které slouží pro stahování velkých objemů dat, které však v posledních letech P2P opouštějí, hlavně z důvodu stížností na extrémní zatížení sítí právě díky architektuře P2P.



Obrázek 1 - Rozdílné architektury sítě

Vlevo architektura klient-server, vpravo architektura klient-klient [Zdroj: Autor]

2.2 Protokoly sítě

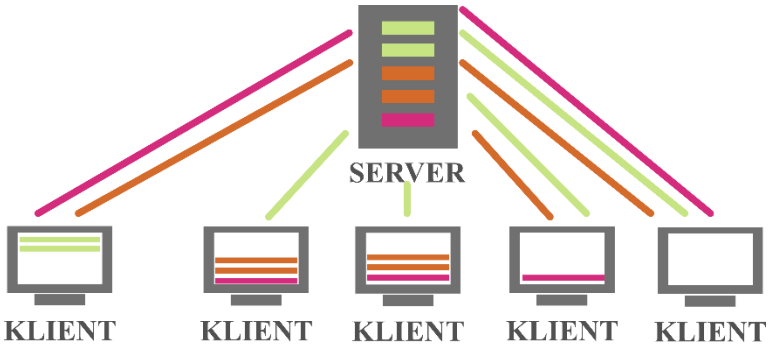
Síťový protokol je standart, jímž se řídí přenos dat mezi dvěma síťovými prostředky. Jedná se o souhrn pravidel upřesňující komunikaci pro dvě rozdílná zařízení. Nejčastěji se můžeme setkat s internetovými protokoly rodiny TCP/IP, mezi které patří například IP (Internet protocol), TCP (Transmission control protocol) nebo UDP (Universal datagram protocol). Dalšími protokoly, se kterými se setkáváme denně, jsou například HTTP (Hypertext transfer protocol) využívaný pro weby, FTP pro přenos souborů či SSH (Secure shell) jako náhrada za nezabezpečený telnet.

2.3 BitTorrent

Protokol BitTorrent je založený na modelu sítě architektury „Klient-Klient“ neboli P2P. Což se dá pochopit jako decentralizovaný model, kdy data stahujeme od ostatních uživatelů, nezávisle na serveru.

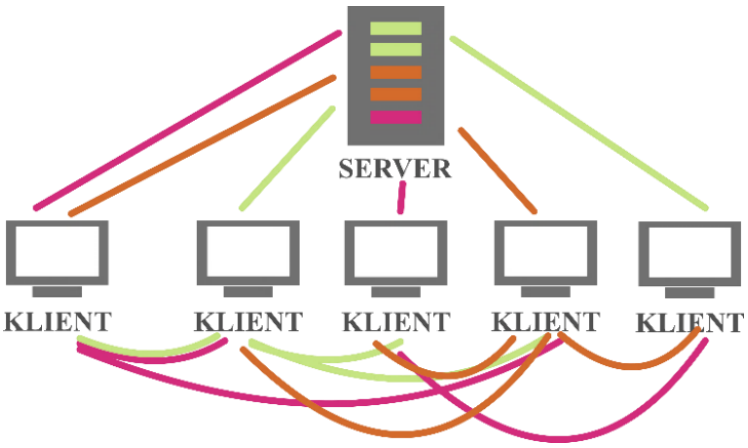
Mezi nejznámější aplikace využívající protokol BitTorrent a také aplikace testované v praktické části této práce, patří BitTorrent, uTorrent, QBitTorrent nebo biglyBT. Tyto aplikace využívají při základním nastavení porty z rozsahu 6881-6889 a port 6969. [4]

Jak již bylo zmíněno, BitTorrent je často spojován s nelegální činností, ale je také mnoho případů kdy BitTorrent je využíván pro distribuci obsahu s volnou licencí. Například řada linuxových distribucí je poskytována ve formě torrentu. Tyto torrenty linuxových distribucí byly později využity pro praktickou část a jsou součástí přílohy.



Obrázek 2 - Stahování souborů ze serveru

V případě stahování ze serveru musí samotný server obsloužit všechny klienty rovný dílem, tím se zvyšuje náročnost na konektivitu serveru. [Zdroj: Autor]



Obrázek 2 - Stahování souborů pomocí P2P

V případě stahování P2P se serveru uleví a zátěž se rozprostře mezi stanice, které mezi sebou sdílejí data, která již mají. [Zdroj: Autor]

2.3.1 Definice pojmů spojených s BitTorrentem

- *Peer* je jakýkoliv účastník síťové komunikace.
- *Seed* je účastník síťové komunikace který již má stáhnutý celek a poskytuje ho ostatním.
- *.torrent* je přípona souboru obsahující informace potřebné pro navázání síťového provozu. [5]
- *Magnet Link* je alternativa oproti souborům s příponou .tracker, umožnuje stahování napřímo z odkazu [6]
- *Torrent klient* je aplikace umožňující načtení informací o torrentu a následné stažení dat.
- *Tracker* je server starající se o uložení IP adres všech peerů. Při zahájení stahování tracker odešle určitou část peerů torrentovému klientovi a ten naváže připojení a zahájí stahování.
- *Swarm* je prostředí se skupinou uživatelů připojených ke stejnému torrentu
- *DHT síť* je alternativa k trackeru. V originále „Distributed hash table (DHT) network“

2.3.2 Klient

Pro správné fungování je nejprve nutné nainstalovat torrentového klienta, pro příklad lze uvést uTorrent. Torrentový klient se stará o načtení a následné zpracování a stahování souborů od ostatních peerů. Klient načte data buďto ze souboru s příponou .torrent, nebo novějším způsobem z odkazu „Magnet Link“. Po načtení potřebných informací, je klient schopen od různých uživatelů stahovat různé části stahovaného souboru a díky tomu maximalizovat možnou rychlost. Torrentový klient má na starost také skládání stahovaných částí do jednoho celku.

2.3.3 Tracker

Při stahování torrentů za použití trackeru jsou data o peerech stažena z centrálního serveru, který také pomáhá s koordinací stahování. Komunikace mezi klientem a trackerem probíhá na jednoduché úrovni na protokolu HTTP, klient odešle požadavek a tracker odpoví s IP adresou a portem na který se klient připojí, aby komunikoval s trackerem a ostatními peery. Na rozdíl od DHT sítě může tracker odmítnout žádost, tedy může být i nakonfigurován jako privátní.

2.3.4 Magnet Link a DHT síť

Náhradou trackeru se snaží být technologie magnet linku a DHT. Na rozdíl od trackeru zde není server, na kterém by byly uložené IP adresy peerů. O vše se stará decentralizovaná síť mezi peery. Zjednodušeně řečeno, když klient načte údaje z magnet linku, zeptá se sousedů v síti, jestli nemají požadované informace, pokud ne, sousedé se zeptají dalších sousedů. Díky tomu je celá síť nezávislá na jednotlivcích. [7, 8] Další výhodou oproti trackeru, je anonymita. Tracker již z principu fungování musí ukládat informace o klientech, kdo a co stahuje, u DHT se informace neukládá, pouze v případě zapnutého klienta se informace sdílí do sítě DHT v podobě unikátních ID.[9]

2.4 Dopady BitTorrentu

Jedním z hlavních důvodů nutnosti vyhledávat provoz BitTorentu v síti je jeho agresivita. BitTorrent si nejprve zabere celou dostupnou šířku pásma, ale poté vytlačuje ostatní účastníky síťového provozu. BitTorrent zároveň vytváří veliké množství navázaných připojení v krátkém časovém okně, a kvůli tomu zaplňuje velice rychle tabulky NAT které využívají i ostatní účastníci provozu. Do tabulky se musí zaznamenávat IP adresy i porty pro tyto krátké připojení.

Druhým, neméně důležitým důvodem, je obliba BitTorrentu mezi uživateli šířící nelegální obsah. Podle řady studií je většina obsahu nalezeného na torrentech ilegálního původu. [10] Avšak i přesto je tato technologie využívána i pro legální účely.

2.5 NetFlow / IPFIX

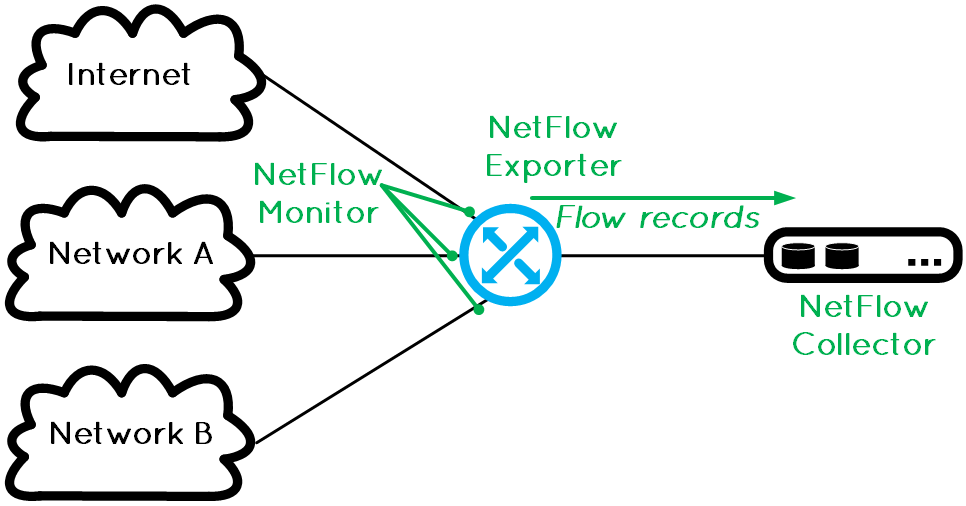
S neustálým narůstáním datových toků, se stávají metody založené na analýze celých paketů, méně efektivní. S nárůstem dat se zvyšuje i nárok na výpočetní výkon. Proto je zapotřebí snížit počet zkoumaných prvků, Jedním z řešení je agregace paketů do jednotlivých celků, do tzv. datových toků. Jedno z řešení využívající agregaci paketů je NetFlow od Cisca, který dal základ pro vznik dalších podobných řešení.

2.5.1 IPFIX

Ve snaze sjednotit a standardizovat toto řešení vznikla skupina Internet Protocol Flow Information Export working group. Následně vznikl dokument RFC 3917 definující požadavky a specifika IPFIX. Samotný IPFIX je založen na NetFlow verzi 9. [11]
Existují i další protokoly založené na stejném fungování, vytvořené pro účely firem, jako například NetStream, který je využíván zařízením od společnosti Huawei.

2.5.2 Architektura NetFlow / IPFIX

Pro funkčnost samotného NetFlow / IPFIX je zapotřebí určitá infrastruktura. Datové toky vznikají ze síťové komunikace, proto je zapotřebí zachytit všechnu probíhající komunikaci.



Obrázek 3 - Příklad architektury NetFlow/IPFIX [22]

Flow exporter

První článkem je Netflow exporter**.** To může být samotný router schopný exportovat NetFlow/IPFIX záznamy, nebo takzvaná sonda. To může být samostatné hardwarové zařízení, například sonda „flowmon“ od společnosti „Flowmon Networks“, nebo jiné softwarové řešení. V obou případech je nutné data ze síťové komunikace nějakým způsobem nasměrovat do sondy, Nejlepším řešením je zrcadlení portů na switchi. Vše, co probíhá, se kopíruje a posílá samostatnou linkou k sondě, ať už se jedná o fyzické zařízení nebo software. Díky tomu není omezována konektivita a nejsou zahlcována zařízení zpracováním datových toků.

Kolektor

Kolektor je samostatný systém starající se o přijímaní dat z exporteru a jejich následné zpracování, ať už se jedná o zálohování, upravování, nebo odesílání dál.

Sám kolektor se může starat i o vyhodnocování dat, a být tedy posledním článkem systému, nebo odesílat data na další vyhodnocující systémy.

3 Maskování BitTorrentu

V této kapitole se zaměříme na maskování provozu BitTorrentu, ať už z pohledu uživatele, který chce zůstat anonymní, nebo z pohledu samotného protokolu a jeho skrytí před detekcí a následnou blokací.

3.1 Maskování BitTorrentu pro uživatele

BitTorrent jako takový není sám o sobě nelegální, jedná se pouze o technologii šíření dat. Zda BitTorrent uživatel využívá legálně nebo nikoliv lze říct podle obsahu přenášených dat. V každé zemi může být nelegální obsah specifikovaný jinak, ale zjednodušeně řečeno se jedná o obsah, ke kterému nemáte autorská práva, nebo licenci. Jako příklad si můžeme uvést filmy, seriály, hudbu nebo počítačový software. Dalším nelegálním obsahem jsou zakázané materiály, nebo materiály jinak závadné. Jako příklad si můžeme uvést dětskou pornografii.

Někteří poskytovatelé internetu se snaží BitTorrent z těchto důvodů blokovat. Ať už blokací portů, nebo například omezením propustnosti připojení pro své klienty.

Proto řada uživatelů hledá způsob, jak zamaskovat svou aktivitu na BitTorrentu před ostatními. A to ať už se jedná o poskytovatele připojení, nebo případně i samotnými společnostmi vlastnící práva, nebo před orgány činnými v trestním řízení.

3.1.1 Pojmy anonymita a pseudoanonymita

Ve světě internetu se využívají tyto dva pojmy, anonymita a pseudoanonymita. O úplné anonymitě se můžeme bavit v situaci kdy není způsob ke zpětnému dohledání chování uživatele na internetu. O úplné anonymitě se teoreticky můžeme bavit o využití TORu. Zatímco úplná anonymita je velice těžko dosažitelná, dosáhnout pseudoanonymitu je o mnoho snazší.

V případě pseudoanonymity se jedná o anonymní chování, kdy ale existuje alespoň jedna strana, která zná původce chování, v případě VPN se jedná o poskytovatele. Ten se může zavázat o neposkytování informací, ale anonymita uživatelů závisí pouze na něm.

3.1.2 VPN

Nejjednodušší možností, jak dosáhnout pseudoanonymity, je využití VPN (Virtual private network), což volně v překladu znamená virtuální soukromá síť. Jedná se o tunel mezi klientem a poskytovatelem VPN. Pro vnější pozorovatele se tedy jakákoliv vaše aktivita tváří jako aktivita vycházející od poskytovatele VPN. Jediná strana, která zná, kdo je původcem chování, je samotný poskytovatel VPN, jedná se tedy o pseudoanonymní přístup. Nevýhodou, kromě neúplné anonymity, může být zpomalená rychlost připojení, či výpadky služby. Nevýhodou také může být zpoplatnění služeb u kvalitnějších poskytovatelů VPN.

3.1.3 Proxy

Jako druhá možnost je využití proxy serveru. Na rozdíl od VPN, proxy pouze zamaskuje IP adresu klienta, koncovému uživateli. Tedy pokud je sledováno, z jakých IP adres se daný torrent stahuje, IP adresa klienta zde nebude uvedena, bude nahrazena IP adresou proxy serveru. Tak jako u VPN i zde se jedná o pseudoanonymní přístup a také zde může docházet k výpadkům služby, kolísaní rychlosti, ale také správce proxy serveru může odposlouchávat veškerou komunikaci procházející skrz něj.

3.2 Maskování samotného protokolu

Je odhadováno že jedna třetina provozu na síti je P2P, proto vznikla potřeba u poskytovatelů připojení a správců sítě tento provoz detekovat a poté omezovat rychlosti, případně i kompletně odstavit provoz. Pro BitTorrent tím vznikl problém, a proto stále vznikají nové způsoby, jak BitTorrent zamaskovat.

3.2.1 Změny portů

Nejsnazší a nejrychlejší způsob detekce BitTorrentu je podle portů. Proto se klientské aplikace využívající BitTorrent adaptovali a jsou schopny využívat náhodně generované porty, nebo dokonce i sám uživatel si může zvolit porty které chce využívat.

3.2.2 Šifrování komunikace

Na rozdíl od zašifrované komunikace na protokolu HTTPS, BitTorrent si nedává za úkol šifrovat data pro soukromí uživatelů, ale hlavním cílem je zamaskovat samotný protokol. Toto šifrování mění hlavičku, případně i tělo paketu. Díky tomu je nemožné pro detekci využívat inspekci paketů v jakékoliv formě.

3.3 Problém BitTorrentu přes TOR

Jako ideální řešení se na první pohled jeví nápad využít BitTorrent přes TOR. Tedy přes síť, díky které se klient může stát opravdu anonymní. Zde se však objevuje veliký problém, moderní trackery BitTorrentu využívají protokol UDP, TOR na druhé straně podporuje pouze TCP. Proto v případě, kdy uživatel nastaví proxy server TORu, například v programu uTorrent, program kompletně ignoruje nastavení proxy. Poté vzniká absurdní situace, kdy síť TOR sice dokonale anonymizuje provoz, ale program na straně klienta přesto odesílá jeho skutečnou IP adresu, sice anonymní sítí, ale tracker i ostatní peerové ve swarmu jsou schopni zjistit jeho identitu. [12]

4 Detekce BitTorrentu

Jak již v minulé kapitole bylo zmíněno, BitTorrent je velmi agresivní a je dobré ho mít pod kontrolou. Z tohoto důvodu je nutné BitTorrent detekovat. Pro detekci BitTorrentu již existuje několik metod. Ať už se jedná o nejzákladnější, které lze snadno obejít, nebo o pokročilejší vyžadující analýzu velikého množství dat.

4.1 Detekce podle portů

Detekce podle portů je jedna z nejjednodušších metod a patří k prvním. Většina aplikací využívající síťovou komunikaci mají předefinované porty, které využívají. Jak již bylo zmíněno v kapitole 2.3, BitTorrent komunikuje na portech rozsahu 6881-6889 a na portu 6969. [4] Díky tomu můžeme poměrně snadno a bez většího zatížení strojů vytvořit filtr na tyto porty, kdy se sleduje provoz v síti a porovnávají se porty se seznamem. Bohužel BitTorrent se vůbec nemusí držet těchto portů a může fungovat na portech, buďto náhodně vygenerovaných, nebo i zvolených samotným uživatelem. I v případě, kdy bychom odhalili konkrétně využívané porty BitTorrentem, za několik minut může klient využívat úplně jiné porty. Díky tomu je samotná detekce portů nepoužitelná.

4.2 Detekce založená na DPI (deep packet inspection)

Detekce pomocí analýzy paketů je jedna z nejpoužívanějších metod. Její princip je prohledávání paketů a hledání určitých signatur dle předem daného seznamu. Díky tomu jsme schopni přesně určit o jaký protokol se jedná. Zároveň jsme ale nuceni neustále seznam aktualizovat, hlídat změny ve specifikacích protokolu a upravovat naše signatury. Jinak by náš filtr brzy ztratil účinnost.

Inspekce paketů má ovšem několik nevýhod. Mezi nejzávažnější je jistě nemožnost detekce při šifrovaném BitTorrentu, protože šifrovaný BitTorrent již neobsahuje signatury dle našeho seznamu. V této situaci se stává inspekce paketů neúčinná a díky tomu nepoužitelná.

Druhou nevýhodou je její náročnost na výpočetní výkon a tím i zvýšená časová náročnost. Na rozdíl od jednoduchého filtrování portů, je zde potřeba každý paket detailně prozkoumat. Se

zvyšováním provozu na síti se zvyšuje i počet paketů nutných k prozkoumání. Proto je dobré zkoumat pakety pouze splňující již předem daná jednoduchá pravidla. Například velikost paketů, nebo rozsah portů, abychom například nezkoumali zbytečně pakety systémových aplikací. Další prvek zvyšující náročnost na výpočetní výkon je fakt, že tato metoda vykonává detekci protokolů, tedy může detekovat jednotlivé protokoly fungující na principu P2P, ale ne P2P jako celek. V případě, kdybychom chtěli detekovat všechna P2P spojení, je nutné mít seznam všech protokolů využívající P2P a jejich signatury a díky tomu by byl celý proces ještě více náročný na výpočetní výkon.

Další nevýhodou může být samotný princip fungování, kdy by se v některých zemích dalo klasifikovat jako narušení soukromí uživatelů.

4.3 Detekce programem Snort

V této kapitole se zaměříme na detekci pomocí jednoduché analýzy paketů ale i složitější signatury za pomocí programu snort.

4.3.1 Detekce komunikace s trackerem

Pokud bychom byly schopni sestavit a udržovat seznam trackerů a webů poskytující torrenty, byly bychom schopni odchytnout pokus o komunikaci přes BitTorrent ještě před jejím začátkem. Jak v kapitole 2.3 popisujeme komunikaci, před samotnou komunikací na protokolu BitTorrent je třeba zajistit potřebné informace z trackeru nebo DHT. Tato prvotní komunikace s trackerem probíhá na protokolu HTTP, proto je možné programem snort odchytit požadavek GET který je cílený na server z naší černé listiny trackerů a zároveň hlavičku obsahující pro nás klíčová slova, jako je přípona „ .torrent“.

Tato metoda je již v dnešní době zbytečná, a to hlavně díky technologii DHT a magnet linku, kdy k zahájení torrentu není potřeba komunikace s trackerem ani stahování torrentu. Dalším důvodem, proč se tato metoda nepoužívá, je její nejednoznačnost, i když určí, že klientská stanice komunikuje s trackerem, neurčí a nenajde přímo komunikaci na protokolu BitTorrent.

4.3.2 Detekce zahájení spojení (handshake)

Předešlá metoda byla schopna odhalit komunikaci související s BitTorrentem, ale samotný protokol BitTorrent nikoliv. Podle specifikace BitTorrentu musí každý peer ve swarmu před zahájením spojení poslat handshake určitého tvaru. Díky tomu je možné sestavit pravidlo pro snort k detekci tohoto „pozdravu“. [13] Tato metoda by mohla být velmi efektivní, avšak u této metody je bohužel také problém. V případě, kdy klient svou komunikaci šifruje, je prakticky nemožné tento handshake odhalit.

4.4 Způsoby detekce

I v případě, kdy máme určité znaky, či data, podle kterých můžeme BitTorrent detekovat, musíme zvolit způsob, jak určovat hranici, kdy se o BitTorrent jedná a kdy ne.

4.4.1 Heuristika

Metoda detekce pomocí heuristické analýzy spočívá v zjednodušení rozhodování, pomocí předem daných rozhodovacích stromů. [15, s. 3] Tento způsob rozhodování bývá rychlejší než klasické algoritmy, ale za cenu ztráty přesnosti. Tato metoda se běžně využívá například v antivirovém softwaru pro detekci nových mutací virů. [16]

4.4.2 Statistické metody

Jedná se o metody využívající principy statistické matematiky. Jedná se o obor, kdy získáváme informace z numerických dat. Praxi statistických metod můžeme rozdělit do třech etap:

- **Získávání dat**, v této části je zapotřebí získat vhodná data s dobrou výpovědní hodnotou.
- **Analýza dat**, v této části analyzujeme získaná data, organizujeme je a popisujeme. Tato část se někdy nazývá „popisná statistika“. [17]
- **Statistické usuzování,** je část kdy již ze zorganizovaných dat usuzujeme závěry o širším vzorku dat. Neprovádíme jen závěry, ale i zhodnocení, jak je závěr spolehlivý. [18]

4.4.3 Strojové učení

Jedná se o učení stroje, které se automaticky v průběhu času upravuje a zlepšuje. Učení probíhá na základě vytvoření matematického modelu pomocí učících setů. [19] Strojové učení je založené na principech statistických metod, ale na rozdíl od nich, je výsledek spíše předpovědí, než dedukcí ze známých parametrů jako je to u statistických metod.[20] Strojové učení může najít uplatnění při zkoumání velkých datových setů, dataminingu, nebo například i v běžném životě kdy se reklamní systémy na internetu učí jaká reklama je pro uživatele nejlepší.

4.5 Případy false positive a false negative

False positive a negative jsou případy kdy je špatné vyhodnocení. U false positive se jedná o vyhodnocení chování, kdy je určen provoz jako protokol BitTorrent i když se o BitTorrent vůbec nejedná.

Na druhé straně false negative je vyhodnocení BitTorrentu jako neškodného, tudíž selhání pravidla. Avšak toto vyhodnocení se většinou týká detekce za pomocí analýzy chování sítě, kde nemáme napevno daná pravidla, jak se má systém chovat, a proto budou v tomto případě false negative a false positive vznikat vždy.

5 Metodika založená na analýze síťových toků

Pro vytvoření a vyzkoušení vlastní metodiky byla zvolena možnost detekce za pomocí analýzy chování sítě s využitím protokolu IPFIX. Tato metoda není závislá na parametrech protokolu BitTorrent, ale zaměřuje se na samotné chování protokolu. Díky tomu lze detekovat i maskovaný provoz BitTorrentu.

5.1 Požadavky na systém

Hlavním úkolem systému je detekce protokolu BitTorrent. Je však zapotřebí, aby byl protokol odhalen i v jeho šifrované podobě. Systém musí fungovat na základě analýzy záznamů IPFIX. Úkolem je minimalizovat případy false negative, tedy neoznačení BitTorrentu.

Z hlediska technologického, je systém vyvíjen na platformě Windows 10, s možností migrace na jiné platformy. V úvahu připadá OS Linux. Systém jako celek musí být použitelný i mimo vlastní testovací prostředí, tedy například i v případech jiného sestavení sondy a kolektoru, za dodržení podmínek pro strukturu dat.

5.2 Analýza problémů

Nejprve je zapotřebí určit signifikantní znaky chování BitTorrentu podle kterých budeme protokol detekovat. Z předešlé kapitole popisující chování BitTorrentu víme základní rysy chování. Důležitým faktorem při vybírání signifikantních znaků je ale naše samotné omezení dat, které lze získat. Z tabulky níže lze vyčíst možné parametry které získáváme z reportu IPFIX kolektoru. Z IPFIX je možné získat i jiné parametry, ale v tomto konkrétním případě jsou uvedené použitelné parametry i po anonymizaci dat.

Název parametru	
iana:octetDeltaCount	Celková velikost přenesených dat v toku
iana:packetDeltaCount	Celkový počet paketů v toku
iana:protocolIdentifier	Užitý protokol
iana:sourceTransportPort	Užitý port zdroje toku
iana:sourceIPv4Address	IP adresa zdroje toku
iana:destinationTransportPort	Užitý port příjemce toku
iana:destinationIPv4Address	IP adresa příjemce toku
iana:flowStartMilliseconds	Časové razítko začátku toku
iana:flowEndMilliseconds	Časové razítko konce toku

Tabulka 1 – Parametry získané pomocí IPFIX [Zdroj: Autor]

Pro určení bude použita kombinace signifikantních znaků chování s omezením na UDP protokol s primárním využití parametrů jako jsou velikost, počet paketů, a identifikační čtveřice: ip adresa zdroje, port zdroje, ip adresa příjemce a port příjemce.

5.2.1 Detekce různých spojení

První část je test na počet různých spojení. Když uživatel využívá klasické služby na sdílení obsahu, jako například online uložiště, nebo online disk, stahuje data z jednoho serveru. I v případě, kdy uživatel stahuje z vícero zdrojů, obvykle se zdroje pohybují v řádech jednotek. Ovšem v případě protokolu BitTorrent se navazuje připojení na desítky možných stanic. Zkoumáním bylo zjištěno, že tento test je nejlépe zaměřit na počet datových toků navázaných s různými stanicemi v určitém časovém úseku v poměru k počtu datových toků navázaných na „stejné místo“. Díky tomu lze dobrým způsobem odhalit agresivitu BitTorrentu při navazování mnoha desítek spojení., kdy výsledná hodnota je v rozmezí absolutní hodnoty 1 až 0, kdy 1 je vysoká pravděpodobnost BitTorrentu.

5.2.2 Detekce krátkých spojení

Druhá část je test na počet různých krátkých spojení. BitTorrent se neustále snaží navazovat nová spojení s dalšími klienty. Ne všichni jsou však dostupní, a proto je připojení velmi krátké, ať už z časového hlediska, nebo velikostí přenesených dat. Mimo to také BitTorrent může využívat krátké spojení formou broadcastu při hledání, za užití DHT. Zkoumáním několika testovacích množin bylo určeno že i tento test se zaměřuje na počet rozdílných krátkých datových toků, v poměru k počtu všech krátkých datových toků, kdy výsledná hodnota je také v rozmezí absolutní hodnoty 1 až 0 jako předešlý test.

5.2.3 Detekce spojení s velkým datovým tokem

Třetí část je test na velké spojení. BitTorrent slouží pro přenášení dat, proto při jeho využívání bývají přenášena data ve větším měřítku. Tento test se tedy zaměřuje na síťové toky s větším počtem přenesených dat. Stejně jako v předešlých testech bylo praktickým zkoumáním zjištěna závislost. A to mezi počtem unikátních velkých spojení a počtem celkových spojení bez omezení velikosti. Výsledná hodnota je jako v předešlých testech v rozmezí absolutní hodnoty 1 až 0 jako předešlý test.

5.3 Návrh systému

5.3.1 Architektura systému jako celku

Aplikace sama o sobě nebude schopna fungování. Pro správné fungování je zapotřebí vytvořit prostředí kde se data budou generovat, ať už jen jako testovací, nebo reálný provoz, následně data zachytávat, zpracovávat a odesílat do aplikace na analýzu. Právě aplikace na analýzu těchto dat je primární část této bakalářské práce. Lze tedy po stránce architektury systém rozdělit do třech hlavních částí.

1) Odchyt dat
2) Zpracování dat
3) Analýza dat

Všechny tři části budou běžet na různých zařízeních, ať už z důvodu možnosti simulace reálného provozu, kdy budou data odchytávána na různých zařízeních v různých provozech, nebo z důvodu požadavků na různé operační systémy.

5.3.2 Odchyt dat

Vzhledem k finanční náročnosti na hardwarovou sondu, i náročnosti zprovoznění softwarové sondy, je odchyt dat rozdělen do dvou kroků, kdy první krok je pro snazší odchyt dat zjednodušen.

Prvním krokem je samotné získání dat, kdy pro jednoduchost byl zvolen program Wireshark. Ten dokáže zachytit provoz na jakékoliv stanici, kde je spuštěný. Díky jednoduchosti programu Wireshark můžeme provádět odchyty na velkém počtu různých stanic.

Druhou částí je samotná sonda. Ta v tomto sestavení získává již dříve zachycená data vyexportovaná programem Wireshark. Data se pro sondu tváři jako reálný provoz a díky tomu snadno generuje IPFIX záznamy. Nevýhodou této metody je ztráta, nebo znehodnocení některých parametrů, jako například časová razítka toků, z důvodu, že data jsou sondě předkládána v jeden moment bez časového rozestupu jako v reálném provozu. Při tvorbě metodiky bylo nutné k této skutečnosti přihlížet.

Jako sonda je zamýšlená softwarová sonda nProbe. Tato sonda je v omezeném použití k dispozici bez komerční licence. Omezení spočívá v limitu exportu záznamů.

5.3.3 Zpracování dat

Pro zpracování dat je zapotřebí zprovoznit kolektor. Zamýšlený kolektor je IPFIXcol2 od společnosti CESNET. Tento kolektor je opensource s kódem k dispozici na githubu. Kolektor se bude starat od přebírání dat ze sondy, anonymizaci dat a následný export do formátu čitelného pro aplikaci provádějící analýzu dat.

5.3.4 Analýza dat

Třetí, poslední část systému, je samotná aplikace pro analýzu IPFIX záznamů a vyhodnocování BitTorrent protokolu v provozu. V tomto případě nebude využit žádný dostupný software, ale aplikace bude naprogramována. Jako programovací jazyk bude zvolen Python verze 3.8. Výhodou je, že Python je multiplatformní moderní jazyk a díky tomu bude možné aplikaci provozovat pod systémem Windows i Linux. Vyvíjení aplikace bude probíhat na operačním systému Windows 10 pro 64bit verze. Aplikace bude schopna číst externí data z formátu JSON a následně exportovat své výsledky do souboru. Případně zobrazovat přímo jako output v konzoli.

5.4 Implementace systému

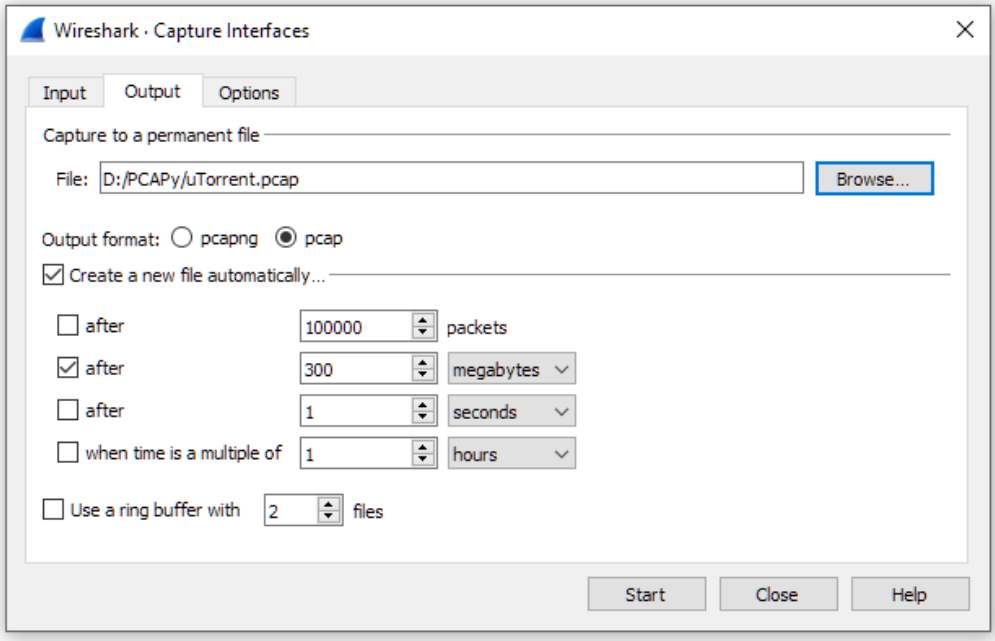
Příprava architektury aplikace jako celku

Aby bylo možné záznamy síťových toků vyhodnocovat, je zapotřebí tyto záznamy nejprve získat a následně zpracovat. Pro obě akce byly zvoleny již funkční aplikace.

5.4.1 Aplikace pro sběr dat

Pro sběr dat je možné použít buďto sondu ve formě hardwarového zařízení, nebo „virtuální“ sondu ve formě aplikace na fyzickém zařízení. Pro účely této práce byla zvolená softwarová sonda nProbe na operačním systému Windows 10 v kombinaci s aplikací wireshark. Sonda nProbe je schopna generovat NetFlow záznamy, nebo také IPFIX záznamy, které také byly zvoleny pro tuto práci.

Pro odchyt dat je tedy zvolen program wireshark, kdy odchytává všechny pakety probíhající na určeném síťovém rozhraní. Pro testování bylo v mém případě zvoleno rozhraní ethernet5, ethernet1 nebo eth1. Avšak u každého zařízení se síťové rozhraní může jmenovat jinak. Na druhé stránce nastavení, je zvolen způsob ukládání. Data se ukládají do souborů s příponou .pcap o velikosti 300mb každý. Tento odchyt byl prováděn na několika stanicích.



Obrázek 5 - Nastavení wiresharku [Zdroj: Autor]

Následně byly soubory obsahující nachytané pakety předloženy sondě nProbe, pro kterou se pakety tvářily jako reálný provoz. Z těchto odchytů je sonda nProbe schopna vygenerovat IPFIX záznamy, které následně předává dál. Pro nastavení sondy slouží příkazový řádek. Pro toto konkrétní nastavení byl použit příkaz:

```
nprobe /c -i 2.pcap --collector 192.168.0.107:4739 --flow-version 10
```

- „nprobe /c -i“ v příkazu vyvolává sondu v režimu čtení dat ze souboru.
- „2.pcap“ v příkazu určuje soubor pro načtení.
- „—collector 192.168.0.107:4739“ v příkazu určuje cestu ke kolektoru. Tedy jeho IP adresu a za dvojtečkou jeho port.
- „--flow-version 10“ určuje verzi protokolu pro export. V tomto případě je to IPFIX v10. V případě špatného nastavení verze by nebyl schopen kolektor data číst.

5.4.2 Aplikace pro zpracování dat

Po sběru dat je potřeba data zpracovat do čitelného formátu a také veškerá data anonymizovat. Pro tyto účely slouží kolektor. Jako kolektor byla vybrána aplikace IPFIXcol2 od CESNETu. Jedná se o modulární kolektor, v tomto případě běžící na operačním systému Ubuntu. Celý kolektor se skládá ze tří hlavních modulů.

Prvním je tzv. „input plugin“ starající se o způsob získávání dat – IPFIX záznamů. V tomto případě byl zvolen plugin pro naslouchání na TCP portu.

Druhým modulem je tzv. Intermediate plugin, který se stará o zpracování dat. V tomto případě byl zvolen plugin anonymizující data.

Posledním modulem je tzv. output plugin starající se o výstupní formát dat. Pro vývoj a testování aplikace byl zvolen modul pro export dat ve formátu snadno čitelném, a to formát JSON. Pro budoucí pokračování ve vývoji je možné využít i plugin pro export na TCP port, a tedy přímo na aplikaci pro analýzu dat.

Pro nastavení kolektoru slouží konfigurační soubor, který se při spuštění kolektoru nastaví jako aktuální konfigurace. Pro spuštění kolektoru slouží příkaz:

```
ipfixcol2 -c <config_file>
```

- „<config_file>“ zde určuje cestu ke konfiguračnímu souboru.

Konfigurační soubor se skládá ze sekcí podle modulů kolektoru. Celý konfigurační soubor je součástí přílohy. Konfigurační soubor je založený na příkladu, který je k dispozici v dokumentaci kolektoru na githubu. První část konfigurace je import, tedy naslouchání portu:

```
<inputPlugins>
    <input>
      <name>UDP input</name>
      <plugin>udp</plugin>
      <params>
        <localPort>4739</localPort>
        <localIPAddress>192.168.0.107</localIPAddress>
        <connectionTimeout>600</connectionTimeout>
        <templateLifeTime>1800</templateLifeTime>
        <optionsTemplateLifeTime>1800</optionsTemplateLifeTime>
      </params>
    </input>
  </inputPlugins>
```

Parametry input pluginu jsou v tomto případě: port, IP adresa, timeout a lifespan.
Druhou částí konfigurace je zpracování. Zde můžeme data anonymizovat. V příloze této práce jsou anonymizovaná data pomocí této konfigurace:

```
  <intermediatePlugins>
    <intermediate>
      <name>Flow anonymization</name>
      <plugin>anonymization</plugin>
      <params>
        <type>truncation</type>
      </params>
    </intermediate>
  </intermediatePlugins>
```

Poslední část konfigurace je export dat, tedy export do formátu JSON. Konfigurace se skládá ze specifikace exportního pluginu a následné konfigurace samotného outputu – cesty k souboru. Zde je zvolen pro export plugin „json“ nastavený v parametru „plugin“ a pro output je nastavena cesta v parametru „path“.

```
<outputPlugins>
  <output>
    <name>JSON output</name>
    <plugin>json</plugin>
    <params>
      <tcpFlags>formatted</tcpFlags>
      <timestamp>formatted</timestamp>
      <protocol>formatted</protocol>
      <ignoreUnknown>true</ignoreUnknown>
      <ignoreOptions>false</ignoreOptions>
      <nonPrintableChar>true</nonPrintableChar>


      <outputs>
        <file>
          <name>Store to files</name>
          <path>/temp/ipfixcol/flow/%Y/%m/%d/</path>
          <prefix>json.</prefix>
          <timeWindow>300</timeWindow>
          <timeAlignment>yes</timeAlignment>
        </file>
      </outputs>
    </params>
  </output>
</outputPlugins>
```

5.4.3 Vývojové prostředí

Vývojové prostředí je software používaný pro usnadnění psaní a orientaci v kódu. Pro vývoj aplikace bylo zvoleno vývojové prostředí ATOM IDE. ATOM IDE je vývojové prostředí, které kromě pythonu je možné použít i na řadu dalších programovacích jazyků. Veliká výhoda je i nativní integrace se službou GitHub, díky které je možné zálohování a verzování kódu přímo z IDE. Pro zálohování bylo tedy zvoleno privátní varianty repositáře na GitHubu. ATOM je k dispozici zdarma pro veškeré hlavní operační systémy. (21)

6 Samotná aplikace pro analýzu dat

Samotná aplikace pro analýzu načítá data ze souborů JSON vyexportovaných z kolektoru IPFIXcol2 každých 5 minut. Aplikace je však schopna zpracovávat data i s menším nebo větším časovým rozsahem. Aplikace k detekci využívá i poměr mezi podezřelým spojením a zdravým spojením, proto v případě, kdy by byla ke zpracování množina dat s velkým časovým úsekem, například v řádech hodin a provoz BitTorrentu by byl pouze v řádech minut, je vysoká pravděpodobnost takzvaného případu „false negative“ kdy i provoz BitTorrentu by byl označený jako zdravý. Ve výchozím nastavení, kdy se data načítají po kratších časových úsecích je však aplikace velice účinná.

Outputem aplikace je výsledek třech různých testů pro každou komunikující stanici nalezenou v síti. Výsledky se zobrazují v konzoli, nebo se ukládají s časovým razítkem testu do logového souboru.

V následujících odstavcích je popsána struktura programu, jeho třídy, metody, funkce a nejhlavnější funkčnost.

6.1 Třídy

Aplikace obsahuje dvě hlavní třídy, a jejich parametry a metody. První třída je samotný objekt síťového proudu s názvem „Flow“. Tento objekt se vytváří z každého záznamu z reportu z kolektoru IPFIX. Každý objekt této třídy obsahuje tyto parametry:

Parametr	**Název proměnné**	**Parametr**	**Název proměnné**
Velikost	size	Počet paketů	pakets
Protokol	protocol	Kontrolní bit	controlBit
IP zdroje	ipSource	Port Zdroje	portSource
IP příjemce	ipDest	Port příjemce	portDest
Čas začátku toku	finTimeStart	Čas konce toku	finTimeEnd

Tabulka 2 – Proměnné ve třídě Flow [Zdroj: Autor]

Druhou třídou je objekt samotné stanice s názvem „Station“. Každý objevený bod v síti je převedený do stanice. Díky tomu jde rozeznat jaká stanice má jaké přiřazené proudy. Jednotlivé paramtery objektu jsou:

Parametr	**Název proměnné**
IP adresa	ip
Odchozí proudy v poli	flowsOut
Příchozí proudy v poli	flowsIn

Tabulka 3 – Proměnné ve třídě Station [Zdroj: Autor]

Doplňující parametry jsou pro jednotlivé výsledky testů, tyto parametry jsou při inicializaci stanice nastavené na hodnotu „0“. Následně se pomocí vlastních metod pro změnu parametru již vytvořeného objektu (tzv „seter“) mění na výsledek analýzy. Tyto parametry jsou:

Parametr	**Název proměnné**
Výsledek detekce různých odchozích spojení	testOut
Výsledek detekce různých příchozích spojení	testIn
Výsledek detekce různých krátkých spojení	testSmall
Výsledek detekce spojení s velkým datovým tokem	testBig

Tabulka 4 – Proměnné s výsledky ve třídě Station [Zdroj: Autor]

6.2 Funkce

Kromě tříd a jejich metod program obsahuje také několik funkcí. Funkce je část kódu, která lze volat názvem. Díky tomu se náš kód stává přehlednější a efektivnější. Funkce se sama o sobě bez zavolání nikdy nespustí. Funkce může, ale nemusí, vyžadovat parametry a po dokončení funkce může vracet i několik hodnot. V programu jsou využité funkce s parametry i bez parametrů.

Funkce loader(file)

První funkcí je „loader“ s jedním parametrem „file“. Tato funkce se stará o otevření souboru specifikovaného parametrem „file“, a jeho následné načtení jako slovníku JSON. Samotné načítání souboru musí být ošetřeno na chybu, kdy by soubor nešel otevřít. V tu chvíli funkce vrátí hodnotu „False“ a ukončí se. Stejným případem je i ošetření situaci kdy by nešel soubor načíst jako JSON, funkce se ukončí a vrátí hodnotu „False“.

Po úspěšném načtení souboru do formátu JSON je vytvořeno několik proměnných, do kterých jsou uložena data přímo načtená z jednotlivých záznamů z JSON souboru (viz „Tabulka 1 – parametry získané pomocí IPFIX“). Zde dochází k jednomu problému, a to, pokud nebude nalezena IPv4 adresa, tak funkce využije IPv6 adresu. Následuje vytvoření objektu „Flow“ za pomoci získaných dat.

Součástí této funkce je také přidání jednotlivých proudů ke stanicím. V listu stanic je tedy hledána stanice s IP adresou totožnou s jednou z IP adres, ať už odchozí, nebo příchozí, z námi vytvořeného proudu pomocí objektu „Flow“. Pokud je stanice nalezena, proud je přiřazen do jednoho ze dvou polí příslušné stanice. Pokud je stanice příjemce proudu, je objekt „Flow“ přiřazen do pole „flowsIn“. Pokud je stanice odesílatelem proudu, je přiřazen do pole „flowsOut“. Avšak pokud není stanice v seznamu nalezena, je nejprve vytvořena nová a následně se proces opakuje. Tato funkce po úspěšném dokončení vrací hodnotu „True“,

Funkce runTests()

Aby každá testovací funkce nemusela zvlášť projíždět každou stanici, je zde funkce runTests(), která má za úkol cyklem projet postupně všechny stanice ze seznamu stanic, který byl vytvořen funkcí „loader“ a pro každou stanici vyvolat následující testy.

Funkce difFlowTest(station)

První testovací funkcí je funkce s názvem „difFlowTest“ s parametrem „station“. Tato funkce vytvoří dvě pole, „connIn“ a „connOut“ kam ukládá jednotlivé IP adresy protějších stanic ve spojení. Díky tomu lze poté vyfiltrovat duplikáty a určit poměr mezi počtem všech odchozích a unikátních odchozích spojení, respektive poměr mezi počtem všech příchozích a unikátních příchozích spojení. Výsledná čísla jsou následně pomocí vlastní metody třídy „Station“ zanesena jako výsledek pro danou stanici do jejích parametrů „testOut“ a „testIn“.

Funkce smallFlowTest(station)

Druhou testovací funkcí je funkce s názvem „smallFlowTest“ s parametrem „station“. Tato funkce stejně jako předchozí funkce porovnává poměr mezi všemi toky a mezi unikátními toky. V tomto případě je ale test omezen na velikost proudu pod 300 b. Výsledek je uložen pomocí vlastní metody třídy „Station“ pro danou stanici do jejího parametru „testSmall“.

Funkce bigFlowTest(station)

Poslední testovací funkcí je funkce s názvem „bigFlowTest“ s parametrem „station“. Tato funkce má za úkol najít unikátní odchozí toky s velikostí vetší než 100 kb, respektive unikátní příchozí toky vetší než 100 kb. Následně počet těchto unikátních toků vydělí celkovým počtem příchozích i odchozích toků stanice. Výsledek je uložen pomocí vlastní metody třídy „Station“ pro danou stanici do jejího parametru „testBig“.

Funkce analyze()

Poslední funkce programu je „analyze“, tato funkce je bez parametru. Účelem je spuštění funkce „runTests“ načtení všech výsledků testů, zpracování a následný výpis do konzole s exportem do souboru. Zkoumáním byly zjištěny mezní hodnoty pro BitTorrent u každého rozhodování. Pokud výsledné číslo testu je větší než 0.2, jedná se pravděpodobně o BitTorrent.

Cyklus konzole

Pro snadné spouštění jednotlivých analýz a testování je zde jednoduchý algoritmus, cyklus, čekající na input od uživatele. Po zadání cesty k souboru se spouští funkce „loader“ pokud funkce doběhne bez chyby, tedy vrátí hodnotu „True“, spustí se funkce „analyze“ která již spouští další navazující funkce a testy.

6.3 Problém dat z kolektoru

Kolektor v současném nastavení exportoval špatně naformátované soubory. Python knihovna na čtení JSON objektů soubory nedokázala načíst. Proto byl vytvořen jednoduchý script, také v pythonu, pro opravu těchto souborů. Jedná se o přidání hranaté závorky na začátek souboru, přidání čárky za každý záznam proudu a naposledy přidání hranaté závorky pro zavření JSONu. V příloze práce jsou již všechny JSON soubory opravené, není tedy třeba je nijak upravovat a lze je rovnou zkoumat detektorem BitTorrentu.

6.4 Požadavky aplikace

Pro správné fungování aplikace je nutné mít nainstalovaný python verze 3.8. Aplikace je testována na OS Windows 10 64 bit, ale měla by být funkční i pod jiným OS v případě, kdy bude pro detektor připravené stejné prostředí.

Pro správnou funkci je vyžadováno, aby lokální instalace pythonu obsahovala tyto knihovny:

- os
- json
- sys
- datetime
- dateutil.parser
- ipaddress

6.5 Použití aplikace

Jedná se o konzolovou aplikaci, tedy o aplikaci bez grafického rozhraní. V příloze této práce jsou k dispozici dvě verze aplikace. První verzí je testovací pro snazší testování velkého počtu vzorků, a druhá verze určená pro spouštění s parametry.

Testovací verze aplikace se jménem „detektor_konzole.py“ se spouští bez parametru. Po spuštění je již automaticky nastavený rozsah zkoumaných IP adres pro testování přiložených testovacích množin. Následně již můžeme psát cestu k souboru JSON obsahující záznamy IPFIX. Aplikace poté vypíše stanice, u kterých byl potencionálně odhalen BitTorrent.

Druhá verze aplikace se jménem „detektor_parametr.py“, se spouští z příkazového řádku s užitím parametru. Parametr je cesta k souboru k analýze. Tato verze aplikace slouží pro automatizaci, kdy není potřeba uživatel, aby pokaždé spouštěl testovací verzi a uváděl cestu k souboru.

Příkaz pro spuštění detektoru může vypadat například takto:

```
detektor_parametr.py záznamy\t1.json
```

Tento příkaz spustí detektor, který projde „t1.json“ soubor ve složce „záznamy“. Pokud bude BitTorrent odhalen, následné vypíše výsledek do logu.

Výsledek testů lze najít v logu, který je automaticky generovaný do podsložky „/logs“ ve složce spuštěného programu. Detektor třídí výsledky dle IP adres. Pokud se jedná o IP adresu odpovídající rozsahu adres, je nález vypsaný do konzole a zapsaný do logu "/logs/log_filtered_xx_xx_xx.txt" kde xx v názvu odpovídá časovému razítku. Pokud IP adresa stanice neodpovídá rozsahu, je nález zapsaný do logu "/logs/log_all_xx_xx_xx.txt"

Když by program neměl právo na vytvoření složky nebo souboru, vypíše se chyba i výsledek testu do konzole.

7 Testování

Pro ověření funkčnosti je zapotřebí systém otestovat. K tomu slouží testovací množiny. Všechny mé testovací množiny lze rozdělit do těchto dvou skupin:

- Stanice bez BitTorrentu
- Stanice s BitTorrentem

7.1 Stanice bez BitTorrentu

Případy v této skupině jsou stanice kde se provozoval běžný „domácí“ provoz bez užití BitTorrentu. Výsledkem testu bude, zda a v jakém množství dochází k případům „false positive“. Jedná se o několik různých domácích stanice a jejich dva různé případy využití sítě:

- Případ 1 – Běžná síťová aktivita. Prohlížení zpráv na internetu, email, chat.
- Případ 2 – Náročnější síťová aktivita. Stahování videa, sledování YouTube a podobně.

Výsledky testů jsou uvedeny v tabulkách níže. Správný výsledek testů by v této skupině měl být negativní. V opačné případě se jedná o případ „false positive“ kdy je označen pozitivně případ kdy se o BitTorrent nejedná.

	Detekce různých odchozích spojení	**Detekce různých krátkých spojení**	**Detekce spojení s velkým příchozím datovým tokem**
Běžná síťová aktivita	Negativní	Negativní	Negativní
Náročnější síťová aktivita	Negativní	Negativní	Negativní

Tabulka 5 – Výsledky testování bez BitTorrentu [Zdroj: Autor]

7.2 Stanice s BitTorrentem

V této skupině je testováno, zda program dokáže odhalit BitTorrentový provoz, test je tedy zaměřen na tzv. případy „false negative“ kdy je chybně označen případ s BitTorrentem jako případ bez BitTorrentu. Stejně jako v předešlé skupině, i zde se jedná o několik různých domácích stanic a jejich různé případy:

- Případ 1 – Klient uTorrent
- Případ 2 – Šifrovaný klient uTorrent
- Případ 3 – Šifrovaný klient qBitTorrent
- Případ 4 – Klient biglyBT
- Případ 5 – Klient BitTorrent
- Případ 1 – Klient eMule

	Detekce různých odchozích spojení	**Detekce různých krátkých spojení**	**Detekce spojení s velkým příchozím datovým tokem**
Klient uTorrent	Pozitivní	Pozitivní	Převážně pozitivní
Šifrovaný klient uTorrent	Pozitivní	Pozitivní	Negativní
Klient qBitTorrent	Pozitivní	Pozitivní	Převážně pozitivní
Klient biglyBT	Pozitivní	Pozitivní	Negativní
Šifrovaný klient BitTorrent	Pozitivní	Pozitivní	Negativní
Klient eMule	Pozitivní	Pozitivní	Převážně pozitivní

Tabulka 6 – Výsledky testování s BitTorrentem [Zdroj: Autor]

Z výsledků v tabulce lze vyčíst, že první dvě metody pro detekci jsou velice účinné, metoda detekce pomocí velkých souborů je částečné použitelná pouze u nešifrovaného provozu, proto tento test je pouze doplňkový.

7.3 Seznam testovacích dat

Níže je tabulka obsahující rozdělení dat do dvou množin. Provoz bez BitTorrentu a provoz s BitTorrentem. Tato tabulka slouží pro lepší orientaci při vlastním zkoušení přiloženého programu.

Název souboru s provozem bez BitTorrentu.	Název souboru s provozem s BitTorrentem.
b1.json	t1.json
b2.json	t2.json
b3.json	t3.json
m1.json	mt1.json
m2.json	mt2.json
m3.json	mt3.json
j1.json	bit1.json
j2.json	bit2.json

Tabulka 7 – Názvy souborů testovacích množin [Zdroj: Autor]

7.4 Případy „False positive“

Vzhledem k principu detekce na základě znaků chování, je možnost vzniku případů false positive, tedy případu, kdy je za BitTorrent označen provoz, kdy znaky chování naplňují definici BitTorrentu.

To lze u některých protokolů odfiltrovat například ignorováním určitých portů, nebo IP adres. Bohužel toto řešení, kdy omezíme detektor BitTorrentu, vytváří bezpečnostní mezeru a dává šanci pro vznik případů false negative, kdy právě díky označení některých portů jako bezpečné nastává problém. V tomto případě je BitTorrent schopný svůj provoz zamaskovat pod porty, které jsou námi označené jako bezpečné.

Jinou možností je samotná analýza ostatních protokolů chovající se podobně jako BitTorrent a hledání odlišných znaků. Díky tomu by se četnost false positive mohla snížit a přesnost detekce BitTorrentu naopak zvýšit. To však vyžaduje detailní analýzu ostatních protokolů.

Závěr

Cíle této práce byly uvedení do problematiky, popis stávajících metod pro detekci BitTorrentu a následný výzkum konceptu možnosti detekce pomocí IPFIX.

V teoretické části jsem vysvětlil pojmy nutné k pochopení celé problematiky, nahlédli jsme na problém z obou stran pohledu, ať už ze strany detekce, nebo naopak ze strany maskování.

Praktická část práce je poté zaměřená na samotnou tvorbu metodiky pro detekci protokolu BitTorrent a její následnou aplikaci. Metodiku jsem vytvořil na základě protokolu IPFIX a vlastního zkoumání desítek množin zachycených dat s provozem BitTorrentu. Metoda je založená na detekci pomocí analýzy chování protokolu BitTorrent. Ze zkoumání jsem zjistil tři možné testy chování, které jsou závislé na samotné definici protokolu BitTorrent. Detekce různých odchozích spojení, detekce různých krátkých spojení a detekce spojení s velkým příchozím datovým tokem. První dva testy se jeví jako velmi účinné, třetí test, ačkoliv dokáže částečně detekovat nešifrovaný provoz BitTorrentu, ztrácí v případě šifrování účinnost. Metodika byla aplikována a testována s pomocí sondy nProbe a kolektoru IPFIXcol2, starající se o odchyt dat, v kombinaci s vlastním programem napsaném v Pythonu.

Do budoucna je několik možností vhodných pro další vývoj. Uvažuji o automatizaci celého procesu, více provázat všechny tři prvky systému. Tedy sondu, kolektor a program. S tím souvisí i možnost upravit program pro použití téměř v reálném čase a napojení na hardwarovou sondu s možností exportu více informací o jednotlivých datových tocích. Díky tomu by se systém mohl stát daleko účinnější a využívat i jiné parametry protokolu IPFIX, které v současném sestavení softwarové sondy a odchytu dat pomocí wiresharku, nejsou k dispozici.

8 Seznam použité literatury

[1] *Bitcoin.org* [online]. c2009-2020 [cit. 2019-08-20]. Dostupné z: https://bitcoin.org/en/developer-reference#type-2-hierarchical-deterministic-hd-wallets

[2] *Support.microsoft.com* [online]. Redmond, c2020 [cit. 2019-08-20]. Dostupné z: https://support.microsoft.com/cs-cz/help/4468254/windows-update-delivery-optimization-faq

[3] *Blogs.skype.com* [online]. c2019 [cit. 2019-08-18]. Dostupné z: https://blogs.skype.com/news/2016/07/20/skype-the-journey-weve-been-on/

[4] *www.cisco.com: Cisco SCA BB Protocol Reference Guide* [online]. San Jose, c1992-2020 [cit. 2019-08-18]. Dostupné z: https://www.cisco.com/c/en/us/td/docs/cable/serv_exch/serv_control/broadband_app/protocol_ref_guide/protocol_ref_guide/01_p2p.html

[5] *www.utorrent.cz: µTorrent uživatelský manuál* [online]. c2005-2020 [cit. 2019-08-20]. Dostupné z: http://www.utorrent.cz/help/manual/

[6] *Www.vuze.com: All About Magnet Links and Torrent Files* [online]. San Mateo, c2020 [cit. 2019-08-20]. Dostupné z: https://www.vuze.com/about-torrents/magnet-links

[7] LOEWENSTERN, Andrew a Arvid NORBERG. *Www.bittorrent.org: DHT Protocol* [online]. 2008 [cit. 2019-08-20]. Dostupné z: https://www.bittorrent.org/beps/bep_0005.html

[8] WANNER, Richard. *Detecting Torrents Using Snort* [online]. 2008, 28.11.2008 [cit. 2019-10-13]. Dostupné z: https://www.sans.org/reading-room/whitepapers/detection/detecting-torrents-snort-33144

[9] LOEWENSTERN. *Getright.com: GetRight® and BitTorrent™'s DHT Network* [online]. c2020 [cit. 2019-08-20]. Dostupné z: http://getright.com/dhtnetwork.html

[10] CHENG, JACQUI. *Arstechnica.com: Only 0.3% of files on BitTorrent confirmed to be legal* [online]. New York, 2010 [cit. 2019-08-20]. Dostupné z: https://arstechnica.com/tech-policy/2010/07/only-03-of-files-on-bit-torrent-confirmed-to-be-legal/

[11] QUITTEK, Juergen, Tanja ZSEBY, Benoit CLAISE a Sebastian ZANDER. *Requirements for IP Flow Information Export (IPFIX)* [online]. Swinburne University, 2004 [cit. 2019-08-20]. Dostupné z: https://tools.ietf.org/html/rfc3917

[12] *Blog.torproject.org: Bittorrent over Tor isn't a good idea* [online]. April 29, 2010. Seattle, c2020 [cit. 2019-08-20]. Dostupné z: https://blog.torproject.org/bittorrent-over-tor-isnt-good-idea

[13] COHEN, Bram. *Www.bittorrent.org: The BitTorrent Protocol Specification* [online]. April 29, 2010. 2008 [cit. 2019-08-20]. Dostupné z: https://www.bittorrent.org/beps/bep_0003.html

[14] COHEN, Bram. *Github.com: Collector configuration* [online]. April 29, 2010. Redmond, c2020 [cit. 2019-08-20]. Dostupné z: https://github.com/CESNET/ipfixcol2/blob/master/doc/sphinx/configuration.rst#example-configuration-files

[15] PEARL, Judea. *Heuristics: Intelligent search strategies for computer problem solving*. Addison-Wesley Publishing Company, 1984. ISBN 0-201-05594-5.

[16] It-slovnik.cz: Heuristika. *It-slovnik.cz* [online]. c2002-2020 [cit. 2020-01-14]. Dostupné z: https://it-slovnik.cz/pojem/heuristika

[17] HENDL, Jan. *Přehled statistických metod zpracování dat: analýza a metaanalýza dat*. Vyd. 2., opr. Praha: Portál, 2006. ISBN 80-7367-123-9. [cit. 2020-05-15].

[18] Descriptive and Inferential Statistics. *Statistics.laerd.com* [online]. Lund Research, c2018 [cit. 2020-05-15]. Dostupné z: https://statistics.laerd.com/statistical-guides/descriptive-inferential-statistics.php

[19] MITCHELL, Tom M. Machine learning. Boston: McGraw-Hill, 1997. ISBN 0070428077. [cit. 2020-05-14]

[20] STEWART, Matthew. *The Actual Difference Between Statistics and Machine Learning* [online]. 2019, 25.3.2019 [cit. 2020-05-14]. Dostupné z: https://towardsdatascience.com/the-actual-difference-between-statistics-and-machine-learning-64b49f07ea3

[21] Atom.io. Atom [online]. c2020 [cit. 2020-05-13]. Dostupné z: https://atom.io/

[22] *Cisco Flexible Netflow configuration* [online]. In: . Jerome Tissieres, c2017-2020 [cit. 2019-12-21]. Dostupné z: https://aboutnetworks.net/cisco-flexible-netflow-configuration/

9 Seznam obrázků a tabulek

Obrázky:

Tabulky:

10 Seznam příloh

Příloha 1 – Program a testovací množiny
Příloha 2 – Konfigurační soubor kolektoru