



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE PROVOZU PROTOKOLU BITTORRENT

BITTORRENT PROTOCOL TRAFFIC DETECTION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

DANIEL FLOREK

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. LIBOR POLČÁK, Ph.D.

BRNO 2018

Zadání bakalářské práce

Řešitel: **Florek Daniel**

Obor: Informační technologie

Téma: **Detekce provozu protokolu BitTorrent
BitTorrent Traffic Detection**

Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s protokolem BitTorrent.
2. Nastudujte možnosti detekce protokolu BitTorrent.
3. Navrhněte detektor provozu BitTorrent v souboru ve formátu pcap a návrh konzultujte s vedoucím práce.
4. Návrh implementujte.
5. Implementaci otestujte na vhodných testovacích souborech, zaměřte se i na falešnou detekci (false positives).
6. Diskutujte možnosti rozšíření.

Literatura:

- Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang. 2004. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web (WWW '04)*. ACM, New York, NY, USA, 512-521.
- Raymond Siulai Wong, Teng-Sheng Moh, and Melody Moh. 2012. Efficient Semi-supervised Learning BitTorrent Traffic Detection - An Extended Summary. In: *Distributed Computing and Networking (ICDCN 2012)*. Lecture Notes in Computer Science, vol 7129. Springer, Berlin, Heidelberg.
- Joo V. Gomes, Pedro R. M. Inácio, Manuela Pereira, Mário M. Freire, and Paulo P. Monteiro. 2013. Detection and classification of peer-to-peer traffic: A survey. *ACM Comput. Surv.* 45, 3, Article 30 (July 2013), 40 pages.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese

<http://www.fit.vutbr.cz/info/szz/>

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Polčák Libor, Ing.**, UIFS FIT VUT

Datum zadání: 1. listopadu 2017

Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
602 00 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá detekciou BitTorrentovej komunikácie uloženej v súbore formátu pcap. Navrhol a implementoval som program, ktorý dokáže komunikáciu detekovať primárne na základe hĺbkovej analýzy paketov, ktorej výsledky využíva k detekcii IP adries a ich portov, podiehajúcich sa na komunikácií BitTorrent. Táto detekcia je voliteľne rozšíriteľná o detekciu pomocou analýzy tokov, čím sa môže dozieliť zvýšeného počtu výsledkov, no vzrastá šanca na chybné označenie.

Abstract

This thesis deals with a topic of BitTorrent protocol detection within a pcap file. I managed to design and implement a tool based on deep packet inspection which can detect IP addresses and their ports that were involved in a BitTorrent communication. This detection is extendable with flow analysis which may lead into more results but at the same time in a higher chance of false positives. Therefore this kind of detection is just optional.

Klíčová slova

Peer-to-Peer, detekcia, sieťová komunikácia, pcap, BitTorrent

Keywords

Peer-to-Peer, detection, internet traffic, pcap, BitTorrent

Citace

FLOREK, Daniel. *DETEKCE PROVOZU PROTOKOLU BITTORRENT*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Libor Polák, Ph.D.

DETEKCE PROVOZU PROTOKOLU BITTORRENT

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pana Ing. Libora Poláka Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Daniel Florek
16. kvetna 2018

Poděkování

Chcel by som poďakovať môjmu vedúcemu bakalárskej práce, pánovi Liborovi Polákovi, že mi poskytol odbornú pomoc.

Obsah

1	Úvod	3
2	Odchytávanie internetovej premávky	4
2.1	Základné metódy odchytávania paketov	4
2.1.1	Odchytávanie na koncovom zariadení	4
2.1.2	Zrkadlenie Portov (Port mirroring)	5
2.1.3	Rozbojovanie (Hubbing out)	5
2.1.4	Machine-in-the-middle	6
2.1.5	Odchytávanie využitím TAP	6
2.2	Packet sni ng technológie	6
2.2.1	Libpcap	7
2.2.2	Tcpdump	7
2.2.3	Wireshark	7
3	Peer-to-Peer architektúra sietí a protokol BitTorrent	8
3.1	Peer-to-Peer siete (P2P)	8
3.2	Protokol BitTorrent	9
3.2.1	Pojmy súvisiace s protokolom BitTorrent	10
3.2.2	Princíp fungovania siete BitTorrent	10
3.2.3	Torrentový súbor	11
3.2.4	Tracker server	12
3.2.5	Peer protocol	13
3.2.6	uTorrent Transport Protocol (uTP)	14
3.2.7	Distributed Hash Table (DHT)	15
4	Klasifikácia internetovej komunikácie	16
4.1	Klasifikácia pomocou ísel portov	16
4.2	Klasifikácia pomocou analýzy obsahu paketov (Deep packet inspection)	16
4.3	Klasifikácia pomocou heuristicky založených metód (analýza tokov)	17
4.4	Active Crawlers	18
4.5	False positives a False negatives	18
4.6	Snort	18
5	Návrh detekcie protokolu BitTorrent	20
5.1	Detekcia komunikácie s tracker serverom	20
5.2	Detekcia Peer Protocolu	21
5.3	Detekcia protokolu BitTorrent pomocou analýzy sie ových tokov	21

5.4	Pomer využívania UDP a TCP transportného protokolu v komunikácií Bit-Torrent	22
5.5	Šifrovanie komunikácie BitTorrent	24
6	Implementácia nástroja	26
6.1	Nástroj pre detekciu komunikácie BitTorrent	26
6.1.1	Rozpoznávanie komunikácie	26
7	Testovanie	30
7.1	Asová a pamäťová náročnosť nástroja	30
7.2	Presnosť detekčného nástroja	32
7.2.1	Testovanie na referenčných súboroch	33
7.2.2	Testovanie na koncovom zariadení	33
7.2.3	Testovanie v odchyte siete VUT	35
7.2.4	Testovanie false positives	36
8	Záver	37
	Literatura	38
A	Obsah CD	40

Kapitola 1

Úvod

Peer-to-Peer (P2P) architektúry sietí si postupom času získali svoju popularitu. Využívajú sa v službách ako VoP2P, distribuovanie multimedialneho obsahu, kryptomenách, ale hlavne pri zdieľaní súborov. Tieto služby generujú veľké množstvo internetovej komunikácie, čo najviac pociťujú poskytovatelia internetového pripojenia tým, že im zahlcujú prenosové pásmo, čo má za následok rôzne menšie, či väčšie technické problémy v sieťach a tým pádom nutné investície do rozširovania kapacity a zdokonaľovania infraštruktúry sietí. Druhým problémom týchto architektúr je fakt, že sa v rámci nich vo veľkom distribuujú citlivé informácie, prípadne diela podliehajúce autorským zákonom. Preto je potrebné mať takéto siete pod kontrolou a vedieť ich vhodne regulovať.

Jeden z protokolov využívajúci P2P architektúru sietí, ktorému sa venuje aj náplň tejto práce, sa nazýva BitTorrent. Tento protokol poskytuje rýchly a efektívny prenos súborov medzi ubovoňým počítačom a zariadením. Čím viac zariadení súbor s ňou má, alebo ho už má stiahnutý, tým rýchlejšie sa súbor distribuuje ďalším zariadeniam.

V rámci práce som implementoval nástroj v jazyku Python, ktorý dokáže v pcapovom súbore detekovať IP adresy a porty zapojené do komunikácie BitTorrent. Rozdeluje ich na usvedčené a podozrivé. Usvedčené IP adresy a porty boli odhalené na základe analýzy obsahov paketov. Podozrivé IP adresy a porty na druhú stranu boli označené na základe analýzy tokov, prípadne iných heuristik, ktoré nedokážu s určitou presnosťou protokol odhaliť. Nástroj navyše dokáže z určitých paketov vyextrahovať infohash torrentov, ktoré nijak alej nevyužíva, ale ukladá ich do výstupného súboru, odkiaľ je možné ich následne manuálne či automatizované spracovanie.

Jadro tejto práce tvorí šesť kapitol. Prvá kapitola sa venuje téme odchyťovania internetovej komunikácie do pcapového súboru. Druhá kapitola sa zaoberá teóriou P2P sietí a protokolu BitTorrent. V tretej kapitole sú popísané rôzne metódy klasifikácie internetových komunikácií. Štvrtá kapitola upravuje klasifikačné metódy k tomu, aby boli schopné detekovať protokol BitTorrent. Piata kapitola popisuje implementáciu detekčného programu. Posledná, šiesta kapitola sa venuje testovaniu hardwarovej náročnosti a presnosti implementovaného programu.

Kapitola 2

Odchytávanie internetovej premávky

K tomu, aby bolo možné zistiť prítomnosť určitých protokolov v internetovej premávke, je potrebné premávku najskôr za určitý časové obdobie odchytiť, a následne ju prehľadať za účelom nájdenia (resp. nenájdenia) hľadaných protokolov. K tomu slúžia metódy odchyťovania paketov, popísané v sekcii 2.1 a softwarové technológie používané pri odchyťovaní paketov popísané v sekcii 2.2.

2.1 Základné metódy odchyťovania paketov

Paket je dátová jednotka, ktorá sa prenáša medzi zdrojovým a koncovým zariadením a pozostáva z riadiacich dát, ktoré obsahujú informácie potrebné pre smerovanie paketu v sieti internet, tvoria ich napríklad IP adresy zdrojového a koncového zariadenia, zdrojový a koncový port a podobne. Okrem toho paket tvoria užívateľské dáta, tiež známe pod názvom *payload* [16]. Pod internetovou premávkou, prípadne internetovou komunikáciou sa môžu v tejto práci chápať všetky pakety prechádzajúce cez určitý sieťový uzol za určité časové obdobie.

Komunikáciu je možné odchyťovať v bode medzi dvomi sieťovými uzlami. Všetky spôsoby odchyťovania internetovej premávky majú jednu spoločnú vlastnosť, že sa komunikácia musí najskôr zduplikovať a jej kópia sa odchyťí a následne spracuje na monitorovacom zariadení. Monitorovacie zariadenie, alebo aj sniffer, býva primárne počítač, ale môže to byť aj iné zariadenie, na ktorom beží nástroj zvaný *packet sniffer* alebo aj *packet sniffing software*, ktorý odchyťáva každý paket, ktorý sa objaví na zvolenom sieťovom rozhraní monitorovacieho zariadenia.

V podsekciiach 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5 budú popísané najzákladnejšie a najjednoduchšie spôsoby zapojenia monitorovacieho zariadenia tak, aby dokázalo odchytiť komunikáciu medzi dvomi sieťovými uzlami.

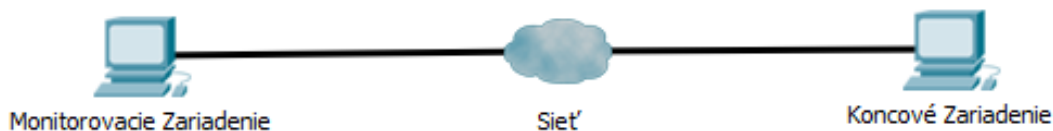
2.1.1 Odchyťovanie na koncovom zariadení

Odchyťovanie na koncovom zariadení je najjednoduchšou metódou odchyťovania paketov. Jedno z komunikujúcich zariadení je zároveň aj monitorovacie zariadenie a odchyťáva všetku svoju odchádzajúcu a prichádzajúcu komunikáciu [7]. Vyobrazenie tohoto zapojenia je možné vidieť na obrázku 2.1.

Metóda odchyťovania na koncovom zariadení je výhodná napríklad pre programátorov internetových aplikácií, aby mohli kontrolovať komunikáciu, ktorú prijíma a odosiela

ich aplikácia, prípadne pre odchytenie komunikácie za účelom skúmania správania rôznych protokolov. Nevýhodou tejto metódy však je, že nedokáže odchytiť nijakú širšiu komunikáciu, ako napríklad komunikáciu celej podsiete, pokiaľ by sa nemonitorovalo na každom koncovom zariadení v podsieti.

Metódu odchyťovania paketov na koncovom zariadení som použil k odchyteniu testovacích dát, využitých v podsekcii 7.2.2.

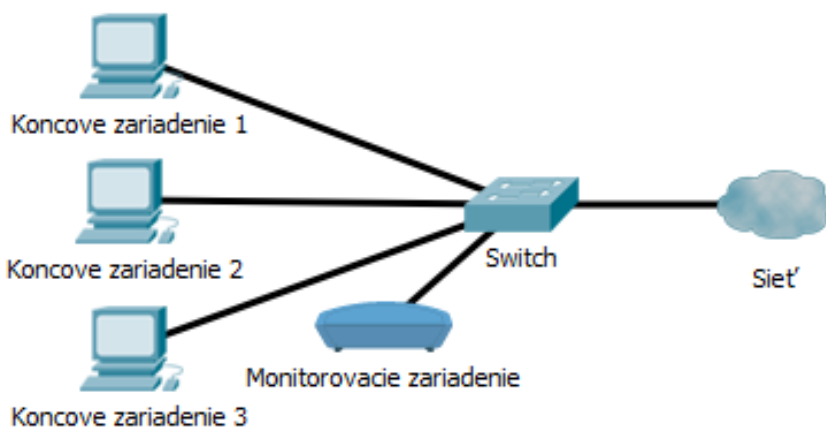


Obrázek 2.1: Koncové zariadenie je zároveň monitorovacie zariadenie

2.1.2 Zrkadlenie Portov (Port mirroring)

Pri tejto metóde sa využíva monitorovací režim na prepínači, ktorý umožňuje zrkadlenie všetkej komunikácie prechádzajúcej cez jeden, alebo viacero portov prepínača, na jeden monitorovací port. K tomuto monitorovaciemu portu sa následne pripojí monitorovacie zariadenie. Príklad zapojenia port mirroring je možné vidieť na obrázku 2.2.

Toto zapojenie však má jednu nevýhodu a tou je maximálna fyzická priepustnosť portu. Všetky porty na prepínači sú plne duplexné, čo znamená, že komunikujú zároveň oboma smermi. V prípade zrkadlenia viacerých portov zároveň môže súbežne prejsť maximálnu fyzickú priepustnosť monitorovacieho portu, čím sa môžu zabrať a strácať pakety, prípadne sa spomalí funkcionálnosť celého zariadenia [18].



Obrázek 2.2: Monitorovacie zariadenie je pripojené do monitorovacieho portu na prepínači

2.1.3 Rozbožovanie (Hubbing out)

Medzi dvoma zariadeniami sa pripojí rozbožovač, ktorý vďaka tomu, že všetko, čo príde na vstup ktoréhokoľvek jeho portu odošle na všetky ostatné porty, sa stará o duplikovanie komunikácie. Do rozbožovača sa pripojí monitorovacie zariadenie, ktoré odchyťáva všetku

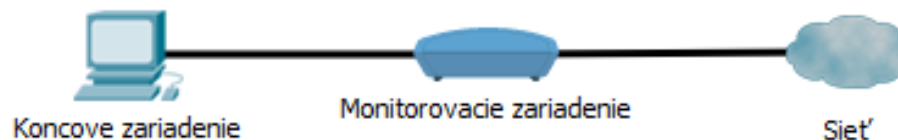
komunikáciu, ktorú mu rozbojva odošle na sieťové rozhranie. Vo väčšine prípadov sa však z full duplex komunikácie stáva half duplex. Nejedná sa o najideálnejší spôsob odchyťovania komunikácie, no v prípade, že prepínač nepodporuje zrkadlenie portov to v niektorých situáciách býva jediné možné riešenie odchyťovania komunikácie [7, 18]. Príklad zapojenia hubbin gout sa nachádza na obrázku 2.3.



Obrázek 2.3: Monitorovacie zariadenie je pripojené do rozbojva

2.1.4 Machine-in-the-middle

Monitorovacie zariadenie je zapojené sériovo medzi dvoma sieťovými uzly a odchyťáva komunikáciu. Podmienkou tohoto zapojenia je, že monitorovacie zariadenie musí obsahovať dve sieťové karty, ktoré sa využívajú ako transparentný bridge. Jedna karta prepája zariadenie **A** s monitorovacím zariadením a druhá karta prepája monitorovacie zariadenie so zariadením **B**. Pre odchytenie všetkej komunikácie stačí odchyťávať iba na jednom z týchto sieťových rozhraní [7]. Zapojenie je zobrazené na obrázku 2.4.



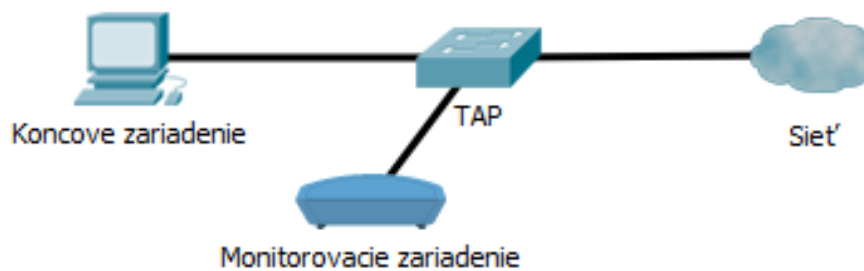
Obrázek 2.4: Monitorovacie zariadenie je sériovo pripojené medzi dvoma sieťovými body.

2.1.5 Odchyťovanie využitím TAP

Pri tejto metóde zapojenia sa využíva hardwarové zariadenie zvané TAP (tzv. Test Access Point, alebo Test Access Port, prípadne Terminal Access Point). TAP obsahuje aspoň dva sieťové porty, ktorými sa pripojí medzi dvoma sieťovými uzly, no samotné TAP nedokáže sieťovú komunikáciu odchyťávať, dokáže ju iba duplikovať a vyviesť na monitorovacie porty, ku ktorým sa pripojí monitorovacie zariadenie [7]. Vyobrazenie zapojenia TAP sa nachádza na obrázku 2.5.

2.2 Packet sniffing technológie

Packet sniffing technológie sú programy, alebo knižnice, ktoré bežia na monitorovacom zariadení a využívajú sa k odchyťovaniu jednotlivých paketov na jeho sieťových rozhraniach.



Obrázek 2.5: Monitorovacie zariadenie je pripojené k TAP

V podsekcii 2.2.1 je popísaná knižnica jazyka C/C++, *Libpcap*. V ďalších dvoch podkapitolách 2.2.2 a 2.2.3 sú popísané dva najznámejšie programy v oblasti odchyťovania paketov, Tcpdump a Wireshark. Všetky tri spomínané technológie dokážu odchytenú komunikáciu následne uložiť do súboru vo formáte pcap (skratka pre packet capture), ktorých analýzou sa zaoberá náplň tejto bakalárskej práce.

2.2.1 Libpcap

Libpcap je open source knižnica distribuovaná pod licenciou BSD poskytujúca rozhranie pre odchyťovanie paketov. Vytvorili ju výskumníci z Lawrence Berkley National Laboratory v roku 1994. Zámer autorov bol vytvoriť API nezávislé na platforme, bez toho, aby boli potrebné systémové moduly pre odchyťovanie paketov. Libpcap bol pôvodne vytvorený pre použitie v jazykoch C/C++, no už existujú varianty pre iné programovacie jazyky ako Python, Java, Ruby a iné. Funguje na väčšine UNIX-ových systémov. Pre platformu Windows existuje portovaná verzia WinPcap. V dnešnej dobe ju vyvíja tím, ktorý sa stará aj o vývoj programu Tcpdump [9].

2.2.2 Tcpdump

Tcpdump je open source packet sniffer a analyzátor obsahov paketov, ktorý beží v príkazovom riadku a nemá žiadne grafické užívateľské rozhranie. Distribuovaný je rovnako ako aj knižnica Libpcap pod BSD licenciou. Funguje na väčšine UNIXových systémov a implementovaný je pomocou knižnice Libpcap. Pakety dokáže filtrovať na základe obsahu z jednotlivých vrstiev paketov. Na platformu Windows existuje portovaná verzia s názvom WinDump, využívajúca knižnicu WinPcap.

2.2.3 Wireshark

Wireshark, rovnako ako aj Tcpdump je open source packet sniffer a analyzátor distribuovaný pod licenciou GNU GPLv2, ktorý obsahuje grafické užívateľské rozhranie. Je crossplatformový, takže funguje pod UNIX-ovými systémami aj systémami Windows. Má taktiež aj terminálovú verziu bez GUI s názvom Tshark. Oproti Tcpdump poskytuje pohodlnejšie pracovné rozhranie a väčší spôsob filtrácie a analýzy dát.

Kapitola 3

Peer-to-Peer architektúra sietí a protokol BitTorrent

Táto kapitola sa venuje teórii potrebnej k celistvému pochopeniu fungovania protokolu BitTorrent. V sekcii 3.1 sú v krátkosti popísané Peer-to-Peer siete, pod ktoré protokol BitTorrent spadá. Nasledujúca sekcia 3.2 sa venuje už samotnému protokolu BitTorrent.

3.1 Peer-to-Peer siete (P2P)

P2P siete nevyužívajú tradičný klient-server spôsob komunikácie, pri ktorej sa všetci klienti pripájajú na jeden centrálny server. V prípade P2P sa jednotlivé zariadenia podieajú na komunikácii nazývajú uzlami a zastávajú rolu servera a zároveň aj klienta. Všetky uzly v sieti sú si rovné a komunikujú navzájom medzi sebou. Vďaka tejto decentralizácii odpadá záťaž na jeden centrálny server, pretože sa rozdeľuje medzi všetkými účastníkmi siete. Väčšina týchto sietí však využíva aspoň jeden server, ktorý zohráva pomocnú úlohu. Uzly tento server kontaktujú a vyžadujú si od neho informácie potrebné pre nadviazovanie spojení s ďalšími uzlami [17]. Grafické porovnanie klient-server architektúry a P2P architektúry je možné vidieť na obrázku 3.1.

P2P siete majú široké pole využitia, ako napríklad real-time komunikácia (VoP2P), streaming audia a videa (P2PTV), ale pravdepodobne najrozšírenejšie využitie je zdieľanie súborov. Za zakladateľa a služieb využívajúcich P2P princíp zdieľania súborov sa považuje Napster. Podstatou tejto služby bolo zdieľanie hudby a rôznych audio súborov. Napster ešte nevyužíval koncepty plného decentralizovania, ktoré sú známe dnes, ale využíval centralizované adresáre. Vďaka týmto adresárom mal správca služby možnosť kontrolovať a predchádzať porušovaniu autorských práv, no nekonal tak, vďaka čomu bola služba americkou justíciou uznaná vinnou z porušovania autorských práv, čím definitívne skončila. Následne začala vznikať druhá generácia P2P protokolov pre zdieľanie súborov, ako napríklad Gnutella, FastTrack, či samotný BitTorrent, ktoré nevyužívajú žiadne centrálné adresáre, ale súbory sa distribuujú iba medzi príslušnými uzlami.

P2P siete trpia určitými funkčnými problémami. Prvý problém je nutná prítomnosť uzlov pre distribuovanie určitého obsahu. V prípade, že sa uzol odpojí, stráca sa aj obsah. Tento problém nastáva hlavne v momente, keď daný obsah vlastní iba malé množstvo uzlov. Druhým problémom sú užívatelia, ktorí P2P sieť využívajú k získaniu rôzneho obsahu, ale neprispievajú rovnakou mierou k ďalšej distribúcii tohoto obsahu. Posledným problémom je množstvo rizikového softwaru distribuovaného v rámci týchto sietí, ako malware a spyware.

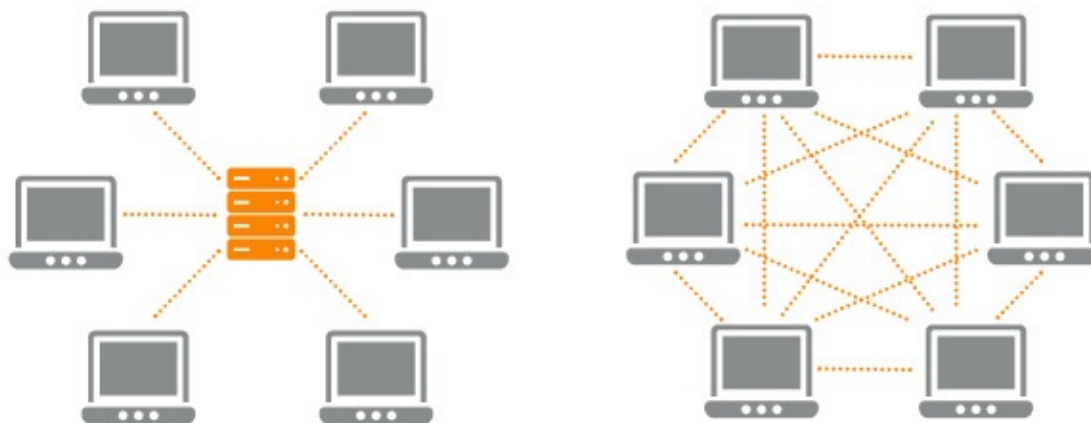
Legálnosť je ďalším pojmom spájajúcim sa s P2P sieťami pre zdieľanie súborov. Samotná myšlienka nelegálna nie je, no väčšina týchto systémov neimplementuje žiadne spôsoby ochrany licencovaného obsahu, vďaka čomu sa vo veľkej miere využívajú práve pre nezákonnú distribúciu takéhoto obsahu.

S rastúcou popularitou P2P aplikácií sa vytvorili debaty o ich dopade na výkon internetových sietí. V rokoch 2006 až 2009 sa odhadovalo, že P2P siete samotné zaberali približne 50-70% celkovej internetovej premávky.

Z perspektívy poskytovateľa internetových služieb aj malé množstvo používateľov dokáže zahŕňať značnú časť siete, prípadne aj celú sieť. Spôsobené je to hlavne tým, že P2P služby generujú veľké množstvo obojsmerných spojení. Z tohoto dôvodu sú poskytovatelia internetového pripojenia nútení využívať rôzne spôsoby prevencie zahlcovania sietí P2P službami [3].

Medzi spôsoby prevencie zahlcovania sietí P2P službami patrí:

- Obmedzenie šírky prenosového pásma pre každého používateľa.
- Blokovanie P2P premávky.
- Prioritizovanie P2P komunikácie medzi uzlami v lokálnej sieti.
- Investície do vybavenia s väčšou šírkou prenosového pásma a routerov s vyššou kapacitou.



Obrázek 3.1: Grafické vyobrazenie architektúr klient-server (vľavo) a P2P (vpravo)

3.2 Protokol BitTorrent

BitTorrent (skrátene BT) je protokol navrhnutý Bramom Cohenom v roku 2001 fungujúci na princípe P2P výmeny súborov, vďaka čomu sú dátové prenosy rozkladané medzi všetky zariadenia podieľajúce sa na distribúcii dát a nie je vyvíjaná zväčšená záťaž na jeden centrálny distribuovaný server. Keďže dokáže ušetriť hardwarové prostriedky, tak ho využívajú rôzne spoločnosti pri distribúcii ich obsahu. Príkladom môže byť spoločnosť Wargaming s ich hrou World of Tanks, kde klientská aplikácia dokáže nové aktualizácie sťahovať práve pomocou protokolu BitTorrent.

3.2.1 Pojmy súvisiace s protokolom BitTorrent

Predtým, než budú bližšie popísané princípy, na ktorých protokol BitTorrent funguje, je potrebné zadefinovať základné pojmy a výrazy, ktoré s týmto protokolom úzko súvisia a budú sa vyskytovať alej v tejto práci:

- **Peer** - Alebo aj uzol. Je zariadenie s ahujúce alebo odosielaajúce dáta pomocou protokolu BitTorrent.
- **Piece** - Alebo aj blok dát. Celkové dáta sú rozdelené na menšie ásti, povä šine po 256KiB, ktoré si uzly vymie ajú.
- **Swarm** - Alebo aj roj. Skupina všetkých peerov, ktorí s ahujú, alebo už vlastnia kompletne dáta.
- **Seeder** - Peer, ktorý vlastní kompletne dáta a už iba poskytuje dáta alej ostatným peerom. Daný peer "seeduje".
- **Leecher** - Peer, ktorý s ahuje dáta. Môže poskytovať tie bloky dát, ktoré už vlastní.
- **Initial Seeder** - Peer, ktorý s poskytovaním dát začal.
- **Tracker** - Alebo aj tracker server. Pomocný server, ktorý uchováva informácie o swarme. Môže komunikovať cez TCP (HTTP/HTTPS) protokol, alebo UDP protokol.
- **BitTorrentový klient** - Alebo aj BitTorrent klientská aplikácia, i skráteno BT klient. Je klientská aplikácia, ktorá dokáže s ahovať súbory pomocou protokolu BitTorrent.
- **Torrentový súbor** - Súbor obsahujúci metadáta o súboroch, ktoré sa zdieľajú a informácie k tomu, aby sa dokázala naviazať komunikácia medzi peermi. Má príponu .torrent.

3.2.2 Princíp fungovania siete BitTorrent

Užívateľ, ktorý chce zdieľať dáta vytvorí torrentový súbor, ktorý obsahuje metadáta o súboroch, ktoré chce zdieľať, a uloží ho privátne alebo verejne na internet. Najčastejšie to bývajú weby uchováajúce torrentové súbory. Príkladom takýchto stránok môžu byť thepiratebay.org, rarbg.to, i český cztorrent.net. Tento užívateľ sa nazýva aj *initial seeder*. Koncový užívateľ, ktorý chce tieto dáta získať, si z internetu stiahne spomínaný torrentový súbor, ktorý otvorí vo svojom BT klientovi. Následne klientská aplikácia súbor prečíta, vytiahne z neho potrebné informácie a pokúsi sa pomocou HTTP, HTTPS, alebo UDP protokolu kontaktovať tracker, ktorého URL adresa je obsiahnutá v torrentovom súbore. Klientská aplikácia mu odošle informácie o svojom uzle. Tracker si tieto informácie zaznamená, a späť odošle zoznam uzlov, ktoré si už dané dáta navzájom vymieľajú, takzvaný *swarm*. Následne už uzol kontaktuje jednotlivé uzly zo zoznamu sám, a komunikujú spolu cez *peer protocol*.

Keď sa uzol pripojí k swarmu, stáva sa z neho *leecher*. To znamená, že od ostatných uzlov v swarme si postupne vyžiada jednotlivé bloky dát, z ktorých následne vyskladá výslednú podobu dát. Okrem toho môže ostatným uzlom poskytovať tie bloky dát, ktoré už stiahol. Keď uzol stiahne všetky bloky dát a vyskladá z nich výslednú podobu s ahovaných dát, stáva sa *seedrom* a poskytuje tieto dáta ďalším uzlom, ktoré proces s ahovania ešte neukončili [5].

3.2.3 Torrentový súbor

Torrentový súbor je súbor s príponou .torrent, ktorého obsah je bencodovaný slovník (dictionary) a obsahuje metadáta potrebné pre začatie procesu a samotný proces sťahovania súborov, ktoré torrentový súbor popisuje. Bencoding je špeciálny typ kódovania, ktoré využíva pre ukládanie a prenos štruktúrovaných dát protokol BitTorrent [5].

Obsah torrentového súboru tvoria:

- **Announce** - URL tracker serveru. Môže obsahovať aj list viacerých serverov.
- **Info** - Slovník, ktorého obsah závisí na tom, či popisuje jeden, alebo viac súborov. Z jeho obsahu sa generuje SHA1 info hash.
 - **Name** - V prípade jedného súboru obsahuje meno súboru, v prípade viacerých súborov obsahuje názov priečinka, v ktorom sa súbory nachádzajú.
 - **Piece length** - Veľkosť jedného bloku. Pôvodný súbor, poprípade súbory sú rozdelené na menšie bloky, ktorých veľkosť sa uloží do tejto položky. Najväčšie to býva 256KiB.
 - **Length** - Veľkosť súboru. Táto položka je obsiahnutá iba v prípade jedného súboru.
 - **Files** - V prípade, že torrentový súbor popisuje viac súborov, obsahuje list slovníkov.
 - **Path** - Cesta k súboru.
 - **Length** - Veľkosť súboru.
 - **Pieces** - Konkatenácia 20 bajtových SHA1 hash reťazcov vygenerovaných pre jednotlivé bloky (kúsky) zdieľaných dát. Slúžia ku kontrole správnosti obsahov stiahnutých blokov.

Spomínaný obsah popisuje torrentový súbor v prípade najzákladnejšieho BitTorrentového prenosu bez akýchkoľvek rozšírení. V priebehu času sa vyvinulo pre protokol BitTorrent niekoľko rozšírení, napríklad možnosť decentralizovanej distribúcie zoznamov peerov pomocou Distributed Hash Table (DHT), možnosť viacerých trackerov, HTTP seeds a iné.

Na výpise 3.1 sa nachádza príklad dekódovaného obsahu torrentového súboru.

```
{
  'announce': 'http://tracker.si te1.com/announce',
  'info':
  {
    'name': 'directoryName',
    'piece length': 262144,
    'files':
    [
      {'path': ['111.txt'], 'length': 111},
      {'path': ['222.txt'], 'length': 222}
    ],
    'pieces': <binary SHA1 hashes>
  }
}
```

Výpis 3.1: Nebencodovaný torrentový súbor obsahujúci metadáta dvoch súborov a jeden tracker.

3.2.4 Tracker server

Po preítaní torrentového súboru BT klientom sa klient pokúsi kontaktovať tracker, prípadne viacero trackerov obsiahnutých v súbore. Komunikácia so servermi môže prebiehať pomocou protokolov HTTP, HTTPS a UDP. Komunikácia závisí od implementácie tracker serveru. Spôsob, akým je server implementovaný, prípadne port, na ktorom server na úroveň klientská aplikácia zistí pod a jeho URL [5].

Príklady URL trackerov:

- `http://tracker.net/announce.php` - HTTP tracker . Nešifrovaná komunikácia.
- `https://tracker.net/announce.php` - HTTPS tracker . Šifrovaná komunikácia.
- `udp://tracker.net:6969/` - UDP tracker na úrovni na porte 6969. Nešifrovaná komunikácia.

Komunikácia s HTTP trackerom spoíva v tom, že uzol serveru odošle takzvaný *announce* dotaz, ktorý sa skladá z HTTP príkazu "GET /announce.php?", kde uzol ako argumenty doplní nasledujúce informácie:

- **info hash** - 20 bajtový SHA1 hash vygenerovaný z obsahu kú a 'info' bencodovaného torrentového súboru.
- **peer id** - 20 bajtový reazec, ktorým sa daný uzol identifikuje.
- **port** - port, na ktorom uzol bude na úroveň .
- **uploaded** - ve kos dát v bajtoch, ktorú uzol zatiaľ odoslal.
- **downloaded** - ve kos dát v bajtoch, ktorú uzol zatiaľ stiahol.
- **left** - ve kos dát v bajtoch, ktorú uzol ešte musí stiahnuť , aby mal kompletne dáta.
- **event** - je voliteľný argument, môže nadobúdať hodnoty *started*, *completed* alebo *stopped*. Predstavuje stav daného torrentu u daného uzlu.

Po obdržaní týchto údajov si tracker zaznamená tento uzol do informácií o swarme a odpovedá bencodovaným zoznamom uzlov vo vyžiadanom swarme [5].

Odpove serveru obsahuje nasledujúce informácie o jednotlivých uzloch v swarme:

- **peer id** - 20 bajtový reazec jednoznačne identifikujúci daný uzol.
- **port** - číslo portu, na ktorom uzol na úroveň .
- **IP adresa** - IP adresa uzlu.

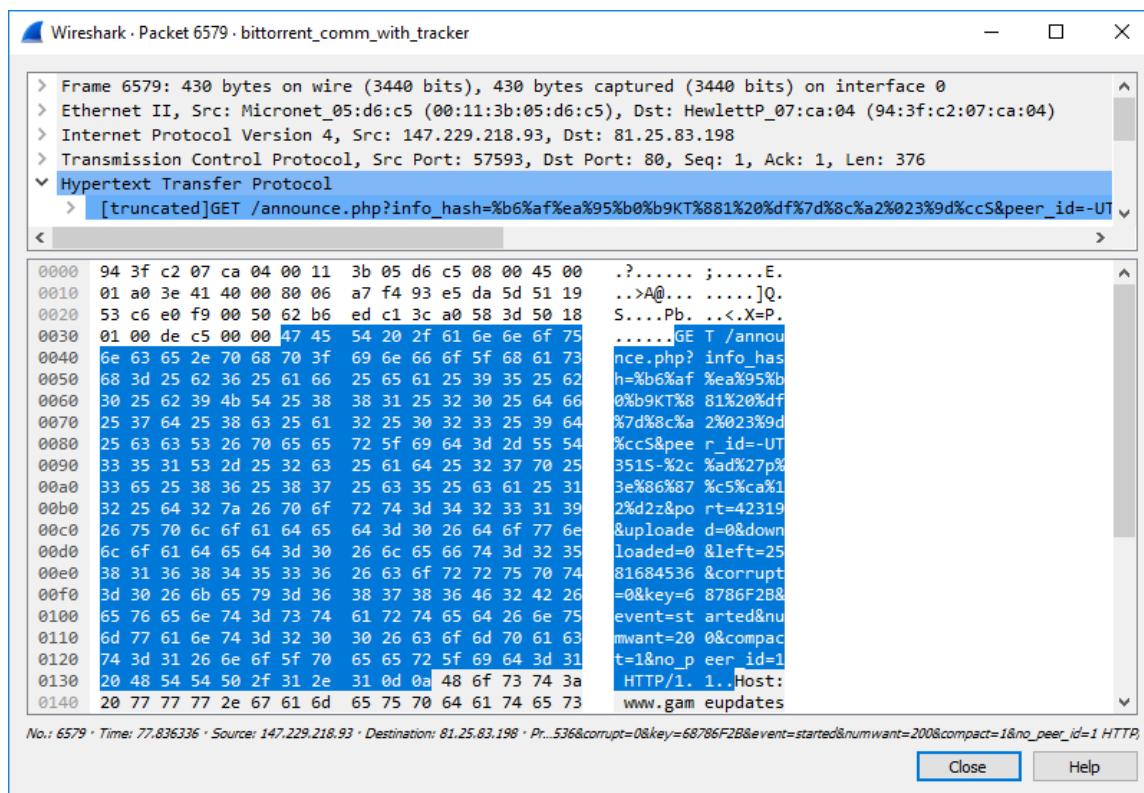
V prípade HTTPS trackeru prebieha komunikácia rovnakým spôsobom, ale v šifrovanej podobe.

UDP komunikácia s trackerom vyzerá podobne, ale jednotlivé údaje sú trackeru prenášané v binárnej podobe a ešte pred *announce* dotazom musí prebehnúť fáza *connect*, v ktorej tracker vygeneruje pre klienta *connection id*, ktorým klient následne oznámi *announce*, alebo *scrape* dotaz [20].

Okrem požiadavky *announce* môže peer kontaktovať tracker aj s inou požiadavkou, tzv. *scrape*, ktorú tvorí takisto HTTP príkaz v tvare "GET /scrape.php?", ktorý ako argument

používa iba samotný info hash daného torrentu a tracker na odpovie jednoduchou správou, ktorá sa môže líšiť od implementácie, no poväčšine obsahuje iba informácie o počte seederoch, leecherov a peerov v danom swarme. Klient na základe týchto informácií prehodnotí, či si od serveru vyžiada nový zoznam uzlov, alebo nie, čím sa môže ušetriť prenosové pásmo. Túto funkcionálnosť však nemusí mať implementovanú každý tracker, pretože sa jedná iba o rozšírenie [22].

Príklad odchyteného announce dotazu v programe Wireshark sa nachádza na obrázku 3.2.



Obrázek 3.2: Odchytený HTTP announce dotaz v programe Wireshark

3.2.5 Peer protocol

Po tom, čo peer od trackeru obdrží zoznam s informáciami peerov v swarme, začne nadväzovať s ostatnými peerami spojenie, aby sa mohol dopytovať, ktoré bloky dát už majú stiahnuté, a následne ich od nich vyžiadať a stiahnuť. Táto komunikácia sa vykonáva pomocou takzvaného *peer protokolu*, ktorý pracuje nad TCP aj UDP transportnými protokolmi. Je symetrický, takže správy odosielané ktorýmkoľvek smerom majú rovnakú štruktúru.

Ako prvé sa vykoná handshake medzi dvomi peerami. Najprv jeden druhému odošle správu handshake (alebo aj handshake message). Druhý peer skontroluje správnosť údajov v prijatej správe a v prípade správnosti spätne odošle vlastnú správu handshake. V opačnom prípade neodpovedá [5].

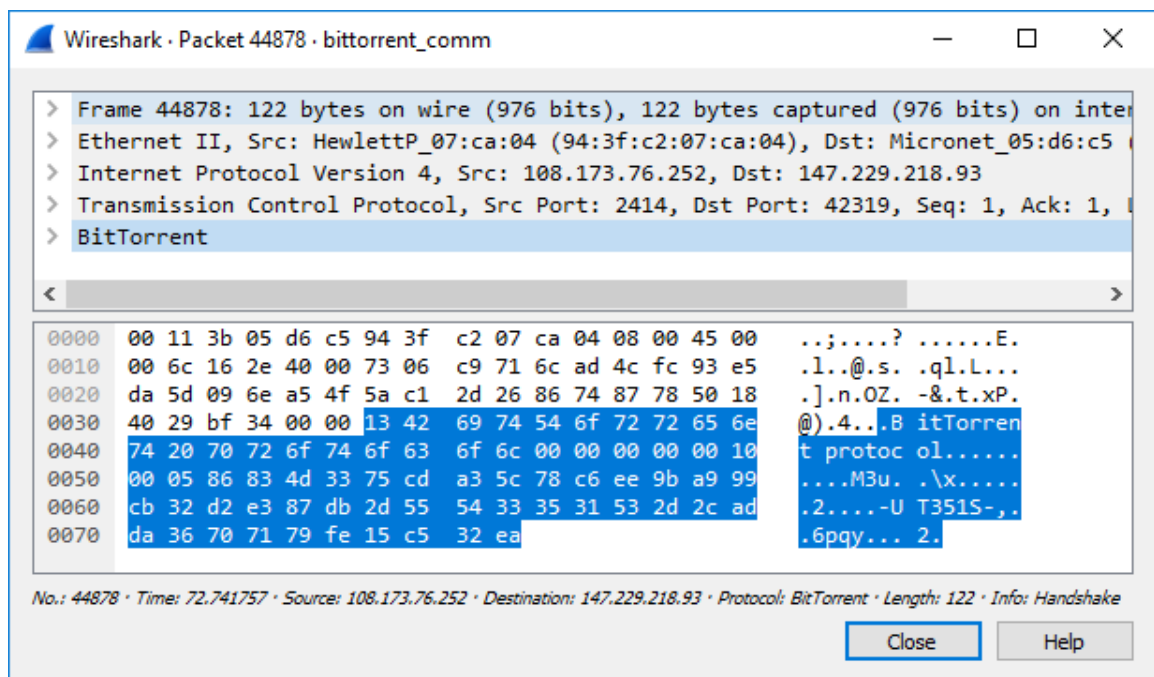
Handshake message obsahuje nasledujúcu štruktúru:

- **Bajt heximálnej hodnoty 0x13 (19 decimálne)** - Indikuje začiatočnú správu a zároveň začiatok nasledujúceho reťazca.

- Reťazec "BitTorrent protocol" - Názov použitého protokolu.
- Rezervovaných 8 bajtov.
- 20 bajtový SHA1 info hash.
- 20 bajtové Peer ID.

Po nadviazaní spojenia si peeri medzi sebou za nú zasiela informácie o blokoch dát, ktoré už vlastnia, prípadne o ktoré majú záujem pomocou *Peer messages*. Po tom, o sa dohodnú, zahajujú proces samotnej výmeny blokov [5].

Na obrázku 3.3 sa nachádza odchytený paket obsahujúci Handshake message.



Obrázek 3.3: Odchytená TCP handshake message v programe Wireshark

3.2.6 uTorrent Transport Protocol (uTP)

uTorrent Transport Protocol je transportný protokol postavený nad protokolom UDP, ktorý bol vyvinutý k tomu, aby nenarušoval internetové spojenia, ale naplno využíval vo né prenosové pásmo. Vzh adom k tomu, že BitTorrentová komunikácia povä šine beží na pozadí, mala by ma nižšiu prioritu, ako napríklad prehliadanie internetu, internetové hovory, i iné interaktívne prenosy. Pri využití pôvodnej TCP implementácie však BitTorrent rýchlo zap a odosielať frontu, ím oneskoruje a zbrz uje prakticky všetky ostatné prenosy. Navyiac, TCP rovnomerne distribuuje šírku prenosového pásma, takže ím viac spojení aplikácia má, tým vä šiu šírku prenosového pásma dostane. Vzh adom k tomu, že protokol BitTorrent sa vyzna uje ve kým množstvom spojení, získava zna nú as prenosového pásma pre seba.

uTP rieši tieto problémy tým, že využíva ve kos odosielať fronty sie ovejkarty ako ur itý modulátor odosielať dát. Ke nie je linka zahľtená, využíva ju naplno, ke však linku potrebujú iné procesy, reguluje množstvo odosielať dát v prípade potreby až skoro na nulu.

Správy Peer protokol využívajúce rozšírenie uTP majú rovnakú štruktúru, ktorá bola popísaná v podsekcii 3.2.5, no tieto dáta začínajú až za hlavičkou uTP, ktorá má veľkosť 20 bajtov [14].

3.2.7 Distributed Hash Table (DHT)

Distributed Hash Table je ďalšie rozšírenie protokolu BitTorrent. Vzniklo za účelom väčšej decentralizácie, aby nebolo potrebné využívať centrálny tracker, ale aby sa každý peer stal trackerom. Ide o takzvané "trackerless torrenty". Pod pojmom *uzol* sa v tomto prípade myslí DHT uzol, ktorým môže byť každý peer, ktorý má BT klienta implementujúceho chovanie DHT uzlu a má toto chovanie povolené.

Každý DHT uzol má vygenerované 20 bajtové globálne unikátne node ID. Medzi týmito node ID sa vykonáva vzdialenostná metrika pomocou logického XOR-u s node ID iných uzlov, alebo info hashmi. Čím nižšia je hodnota výsledku, tým je vzdialenosť menšia.

Keď chce peer zistiť zoznam peerov v swarme pomocou DHT, vykoná sa postupne vzdialenostná metrika medzi všetkými node ID uloženými vo vybudovanej DHT tabuľke a info hashom torrentu a vyberú sa tie uzly, ktorých výsledok bol najnižší. Tieto uzly sa následne kontaktujú so žiadosťou o zoznam peerov s ňou súvisiacim daným torrentom. Ak kontaktovaný uzol má informácie o požadovanom swarme, vráti zoznam peerov vo vyžiadanom swarme dotazujúcemu uzlu. V opačnom prípade vyhľadá v svojom zozname DHT uzlov tie uzly, ktorých node ID sú najbližšie k info hashu torrentu, a vráti ich dotazujúcemu uzlu. Dotazujúci uzol postupne kontaktuje všetky tieto uzly, až kým nezíska požadované informácie [13].

BitTorrentový klient si postupne buduje a rozširuje svoju smerovaciu DHT tabuľku, no pre prípad, že klient ešte nijakú nemá, sa musia do torrentového súboru definovať uzly, ktoré môže peer kontaktovať za účelom získania zoznamu peerov pre daný torrent [13]. K vybudovaniu pôvodnej DHT tabuľky využívajú niektorí BT klienti bootstrap servery, ktoré im sprostredkujú existujúce DHT uzly. Príkladom bootstrap serveru môže byť server `router.bittorrent.com` na úvahu na porte 8991 [15].

V dnešnej dobe DHT skôr sekunduje tracker serverom, aby sa chovanie fungovalo, aj v prípade, že tracker server už nefunguje, alebo má výpadok [13].

Kapitola 4

Klasifikácia internetovej komunikácie

Táto kapitola je venovaná všeobecným metódam a postupom využívaným pri klasifikácii internetových komunikácií z odchytenej internetovej premávky. alej popisuje existujúci nástroj určený na detekciu internetových komunikácií, Snort.

4.1 Klasifikácia pomocou ísel portov

Klasifikácia pomocou ísel portov je jeden z najstarších a najjednoduchších prístupov detekcie určitej internetovej komunikácie. Táto metóda sa zakladá na tom, že mnoho služieb využíva preddefinované a dobre známe porty (tzv. Well-known ports) z rozsahu 0-1023.

Pri detekcii pomocou ísel portov sa sleduje internetová premávka a porovnávajú sa ísla portov z paketov so všeobecne známymi íslami portov. Táto metóda je veľmi jednoduchá na implementáciu a takisto je aj výpočetne nenáročná, no nastáva však pri nej jeden zásadný problém a to ten, že množstvo dnešných internetových služieb síce majú všeobecne známe porty, ale mnoho implementácií daných služieb sa ich už nedrží a používajú vlastné ísla portov, ktoré sú často krát náhodné, alebo dokonca sám užívateľ si môže nastaviť porty, ktoré chce využívať, čím sa stáva táto metóda z veľkej časti nepresná. Niektoré služby sa takisto môžu maskovať ako iné služby tým, že využijú určitý port z rozsahu well-known ports [10].

Zo spomenutých dôvodov sa táto metóda klasifikácie považuje za zastaralú a v dnešnej dobe už nevyužívanú.

4.2 Klasifikácia pomocou analýzy obsahu paketov (Deep packet inspection)

Analýza obsahu paketov je jedna z najpoužívanejších a najefektívnejších metód detekcie internetovej premávky. Funguje na princípe prehadzovania obsahu aplikatívnej vrstvy paketu a hľadania určitých signatúr charakteristických pre hľadane protokoly. Vytvorí sa databáza signatúr pre jednotlivé protokoly, ktorá sa prehadzuje a v prípade zhody je možné presne určiť, o aký protokol a komunikáciu sa jedná. Databáza sa však musí s prípadnými zmenami protokolov neustále udržiavať aktualizovaná, aby sa protokoly mohli spoľahlivo a presne detekovať.

Problém s touto metódou nastáva v dvoch prípadoch. Prvý prípad je, že sa dá použiť iba na nešifrovanú komunikáciu, pretože keď je obsah paketov šifrovaný, nedokážu sa nájsť jednotlivé signatúry, čím sa stáva aj táto metóda neefektívna a prakticky nepoužiteľná. Druhý problém, ktorý môže táto metóda vytvárať je z právneho hľadiska, pretože sa analyzuje obsah aplikatívnej vrstvy jednotlivých paketov, čo môže v niektorých krajinách byť považované za porušovanie súkromia užívateľov.

Metóda analýzy obsahu paketov môže mať aj potenciálne nevýhody, ako napríklad zvýšená výpočtová a časová náročnosť v prípade veľkého množstva prehadzovaných paketov, preto je dobré vyberať iba vhodné vzorky na preskúmanie (napríklad podľa IP adresy, veľkosti daného paketu, či portov), aby sa aspoň čiastočne táto náročnosť znížila. Druhou nevýhodou môže byť aj neustála potreba udržiavať aktualizovanú databázu signatúr, pretože časom vznikajú nové protokoly a niektoré staré sa môžu mierne upravovať. Poslednou nevýhodou môže byť fakt, že táto metóda vykonáva klasifikáciu konkrétnych protokolov a nešpecifikuje sa na triedy protokolov. Nedokáže klasifikovať napríklad všetky P2P protokoly všeobecne, pokiaľ by neexistovala databáza signatúr všetkých P2P protokolov, čo by značne navyšovalo celkovú náročnosť detekčného nástroja [10]. V prípade tejto práce sa jedná skôr o výhodu, keďže sa zameriava na detekciu jedného protokolu.

4.3 Klasifikácia pomocou heuristicky založených metód (analýza tokov)

Aj keď je metóda klasifikácie na základe obsahu paketov popísaná v sekcii 4.2 presná, existujú prípady, kedy nie je vhodná. Preto sa vyvinuli rôzne metódy detekcie, ktoré sa nezakladajú na obsahu aplikatívnej vrstvy, ale fungujú primárne na analýze IP a transportnej vrstvy paketov. Navyše väčšina z nich neklasifikuje iba na základe jedného momentálneho paketu, ale priebežne si budujú dodatočné štatistické informácie, napríklad o tokoch. Nad paketom, alebo nad dátami o toku sa postupne vykoná konečná množina heuristík a na základe ich výsledkov sa rozhodne, či daný paket alebo tok spadá alebo nespadá pod určitú triedu protokolov. Využívajú sa heuristiky jednoduchšie, zložitejšie, prípadne rôzne ich kombinácie [10, 21].

Medzi primárne sledované vlastnosti paketov týmito metódami patria:

- IP adresy, zdrojové aj cieľové
- Využitie UDP alebo TCP protokolu
- Číslo portov
- Veľkosť paketov
- Rýchlosť prenosu
- Množstvo spojení
- Vzťahy medzi tokmi
- Množstvo neúspešných spojení (ICMP)
- Dĺžka toku

Hlavnou výhodou týchto metód je, že dokážu odhaliť šifrovanú komunikáciu a neporušujú práva na súkromie používateľa internetu. Trpia však nižšou presnosťou v porovnaní s metódou klasifikácie pomocou analýzy obsahu paketov a ich výsledky by sa mali brať iba ako podozrenie, nie ako presný jednoznačný výsledok.

Potenciálnou nevýhodou môže byť fakt, že dokážu detekovať triedy protokolov s rovnakým, alebo podobným chovaním a nedokážu detekovať jeden konkrétny protokol. V prípade, že je potrebné detekovať jeden konkrétny protokol, zaručene vznikajú takzvané false positives, ktoré sú bližšie popísané v sekcii 4.5 [10, 21].

4.4 Active Crawlers

Predchádzajúce spôsoby detekcie boli všetky pasívne. Nad zozbieranými dátami sa vykonali určité algoritmy a heuristiky za účelom získania štatistických, prípadne iných výsledkov, no negenerovali žiadnu dodatočnú internetovú premávku.

Aktívna klasifikácia využíva program, ktorý sa tvári ako klientská aplikácia implementujúca sledovaný protokol, ktorého úlohou je získavanie dát od jednotlivých zariadení aktívnym dotazovaním a následné spracovanie odpovedí od dotazovaných zariadení.

Aktívne metódy sa nevyužívajú ku klasifikácii ako takej, využívajú sa skôr za účelom získania dodatočných informácií, napríklad kontaktovaním P2P centrálného serveru falošnými dotazmi, alebo kontaktovaním jednotlivých uzlov v P2P sieti. Môže sa využiť napríklad k tomu, aby sa zistil kompletný zoznam uzlov v určitom swarme.

Nevýhodou aktívnej klasifikácie je, že sama vytvára internetovú premávku a dokáže spracovávať iba dáta, ktoré sama získa [10].

4.5 False positives a False negatives

Pri klasifikácii internetovej komunikácie sa môže stať, že isté pakety, alebo istý tok klasifikátor chybné vyhodnotí a označí ako podozrivé, no v skutočnosti podozrivé nie sú. Vznikajú *false positives*.

V prípade, že klasifikátor komunikáciu neoznačí ako podozrivú, aj keď by označená mala, vznikajú *false negatives* [8].

False negatives a false positives sa týkajú hlavne heuristicky založených metód, pretože detekcia iba na základe chovania určitého protokolu v sieti nikdy nebude úplne presná, pretože takéto chovanie môže vykazovať niekoľko protokolov, takže false positives prípadne false negatives vo väčšej či menšej miere budú vznikať vždy. Množstvo vzniknutých false positives či false negatives závisí od presnosti využitých heuristik [10].

4.6 Snort

Príkladom programu určeného na detekciu v internetovej premávke je *Snort*. Snort je open source aplikácia distribuovaná pod licenciou GNU GPLv2, pôvodne vytvorená Martinom Roeschom, ktorá slúži na odchyťovanie paketov, ich analýzu, ale aj ako detektor na základe obsahu paketov. V dnešnej dobe ho vlastní spoločnosť Cisco.

Snort dokáže pracovať v troch režimoch:

- **Sniffer mode** - číta pakety z internetového rozhrania a následne ich vypisuje na terminálový výstup.

- **Packet logger mode** - Zapisuje pakety do súboru.
- **Network Intrusion Detection System (NIDS) mode** - Vykonáva nad internetovou premávkou detekciu na základe pravidiel.

Detekcia sa vykonáva na základe zadaných pravidiel. Pravidlá si môže používateľ napísať sám, prípadne sú k programu mesa ne výrobcom distribuované oficiálne referenčné pravidlá. Registrovaní užívatelia na stránke výrobcu ich dostanú zdarma, no užívatelia, ktorí si platia mesa né predplatné, dostávajú tieto pravidlá o 30 dní skôr. Týmito pravidlami je možné program nakonfigurovať tak, aby v prípade nájdenia určitých znakov alebo signatúr v pakete o tom užívatel a informoval, alebo aby paket zapísal do samostatného súboru, prípadne rôzne ďalšie akcie, ktoré si užívatel zadefinuje.

Nevýhodou Snortu je, že nedokáže vykonávať klasifikáciu na základe tokov, ani nedokáže z paketov vyextrahovať určité informácie, ako napríklad v prípade tejto práce info hashu.

Vo výpise 4.1 je príklad pravidla pre program Snort, ktorý užívatel a informuje vždy, keď v TCP pakete detekuje komunikáciu s trackerom popísanú v podsekcii 3.2.4. Na aplikácii vrstve h adá re azce 'GET', '/announce', 'info_hash=' a 'event=started'. V prípade nájdenia spomínaných re azcov sa vypíše správa 'P2P BitTorrent announce request'.

Vo výpise 4.2 je príklad pravidla detekujúceho Peer protokol, ktorý je popísaný v podsekcii 3.2.5. Na aplikácii vrstve TCP paketu, ktorého koncový port je z rozsahu 6881 až 6889 sa h adá hexadecimálna hodnota 13 nasledovaná re azcom 'BitTorrent protocol'. V prípade nájdenia tejto signatúry sa vypíše správa 'P2P BitTorrent transfer'.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "P2P BitTorrent
announce request"; flow: to_server, established; content: "GET"; depth: 4;
content: "/announce"; distance: 1; content: "info_hash="; offset: 4;
content: "event=started"; offset: 4; classtype: policy-violation;
sid: 2180; rev: 2;)
```

Výpis 4.1: Príklad pravidla pre detekciu komunikácie s trackerom

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 6881:6889 (msg: "P2P BitTorrent
transfer"; flow: to_server, established; content: "|13|BitTorrent
protocol"; depth: 20; classtype: policy-violation; sid: 2181; rev: 2;)
```

Výpis 4.2: Príklad pravidla pre detekciu TCP Peer protocol handshake message

Kapitola 5

Návrh detekcie protokolu BitTorrent

V tejto kapitole upravím spôsoby klasifikácie popísané v kapitole 4 a definujem signatúry pre hbkovú analýzu tokov tak, aby vhodne detekovali protokol BitTorrent. alej som vykonal dve štúdie, kde som sa zameral na pomer TCP a UDP spojení a pomer šifrovanej a nešifrovanej komunikácie protokolu BitTorrent.

5.1 Detekcia komunikácie s tracker serverom

Na základe znalostí o komunikácií s tracker serverom, ktoré sú popísané v sekcii 3.2.4, som navrhol signatúry, ktoré budú komunikáciu BitTorrent vyh adáva v rámci HTTP paketov a tie sú:

- **GET /announce** - dotaz klienta žiadajúci zoznam peerov v swarme.
- **GET /scrape** - dotaz klienta žiadajúci informácie o swarme.

Detekcia na základe spomínaných re azcov bude úspešná iba v prípade, že komunikácia s trackerom prebiehala nešifrovaná, v opa nom prípade signatúry nebude možné nájs . Ke bude detekovaný takýto paket, vyextrahuje sa z jeho payloadu unikátny info hash torrentu, ktorý sa uloží pre prípadné použitie. alej sa z payloadu vyextrahuje aj port, na ktorom daný peer na úva a tento port sa spojí so zdrojovou IP adresou paketu. Všetka alšia komunikácia i už TCP alebo UDP obsahujúca kombináciu danej zdrojovej IP adresy s vyextrahovaným íslom portu sa ozna í za komunikáciu BitTorrent.

Z dôvodu ve kého množstva generovaných paketov protokolom BitTorrent nie je výhodné vyh adáva vyššie popísané signatúry v každom TCP pakete, ale je potrebné vybra vhodné vzorky. Vzh adom k tomu, že som nedokázal nájs nijakú štúdiu, ktorá by sa zaoberala ve kos ou paketov generovaných protokolom BitTorrent, vykonal som takúto štúdiu sám. Analýzu som vykonal v mnou odchytených BitTorrentových komunikáciách a v pcapovom súbore vytvorenom v rámci siete VUT, ktorá obsahuje odchyt BitTorrentového seedboxu. Pomocou programu Wireshark som v odchytených súboroch sledoval HTTP pakety obsahujúce komunikáciu s trackerom. Z bližšieho skúmania paketov som zistil, že ve kos paketov obsahujúcich dotaz announce sa pohybuje v rozmedzí 400-500 bajtov v prípade IPv4. V prípade IPv6 nieko ko paketov mierne presahovalo ve kos 500 bajtov. Ve kos paketov požiadavky scrape sa vždy pohybovala v rozmedzí 200-300 bajtov. Preto som stanovil rozsah 200-600 bajtov pre podozrenie z tracker komunikácie. Potenciálne by mohli vznika

false negatives v prípade, že by bola ve kos paketu mimo zadaný rozsah, ale z preskúmaných približne 900 tracker správ takáto situácia nenastala ani raz.

5.2 Detekcia Peer Protocolu

Ná základe znalostí o Peer protokole popísaných v podkapitole 3.2.5, aj v tomto prípade som definoval signatúru, ktorá sa nachádza v rámci správ handshake, ktorou je možné Peer protokol detekovať na základe obsahu paketov a tou je hexadecimálna hodnota 0x13 (decimálne 19), ktorá reprezentuje za iatok handshake správy, a zároveň d ŕžku nasledujúceho re azca, nasledovaná samotným re azcom "BitTorrent protocol", takže *0x13BitTorrent protocol*.

Samozrejme, ako aj v sekcii 5.1 aj v tomto prípade platí, že komunikácia nesmie byť šifrovaná, inak sa signatúra nedetekuje. V prípade nájdenia danej signatúry sa zdrojová IP adresa spolu so zdrojovým portom a koncová IP adresa spolu s koncovým portom ozna ia za usved ené z BitTorrentovej komunikácie a všetky ďalšie TCP a UDP pakety obsahujúce aspo jednu zo spomínaných dvoch kombinácií IP adresa-Port sa taktiež ozna ia za usved ené.

Naviac z payloadu paketu sa extrahuje 20 bajtov, pri TCP medzi 29 až 49 bajtom, ktoré obsahujú info hash. Pri UDP sa tento info hash nachádza až medzi 49 až 69 bajtom obsahu, z dôvodu 20 bajtovej uTP hlavi ky na za iatku handshake správy. Tento spôsob zvyšuje šancu na získanie info hashu v prípade, že tracker komunikácia prebiehala cez HTTPS, alebo cez UDP.

Aby sa nepreh adávali všetky pakety, aj v tomto prípade je treba zvoli vhodné vzorky na preskúmanie pomocou h bkovej analýzy obsahu. Rovnakým testovaním, ako bolo popísané v sekcii 5.1 som analyzoval ve kos TCP a UDP paketov obsahujúcich Peer protocol handshake správu. Z niekoľkých tisícov analyzovaných handshake paketov som zistil, že ich ve kos sa pohybuje v rozsahu od 100 do 200 bajtov v prípade TCP aj UDP, aj v prípade IPv4 a IPv6. Preto som stanovil rozsah pre podozrenie z Peer protocol handshake komunikácie na 100 až 200 bajtov.

5.3 Detekcia protokolu BitTorrent pomocou analýzy sie o vých tokov

V predchádzajúcich sekciiach 5.1 a 5.2 boli predstavené spôsoby detekcie protokolu BitTorrent analyzujúce obsah paketov. No tie sú neefektívne, keď je obsah paketov šifrovaný, pretože v nich nedokážu vyh ada potrebné signatúry. Rovnako sa môže sta , že sa pakety obsahujúce dané signatúry neodchyti vôbec, pretože odchyťovanie za alo v neskoršej fázi BitTorrent komunikácie. Preto je potrebné tieto varianty detekcie rozšíri o spôsob, ktorým je možné protokol detekovať bez potreby analýzy jeho payloadu. Preto som na základe analyzovaného chovania protokolu BitTorrent navrhol isté heuristiky, ktoré dokážu protokol BitTorrent detekovať z jeho chovania v sieti.

Heuristiky detekcie založené na chovaní protokolu BitTorrent:

- **Sledovanie čísel portov** - íslo zdrojového aj koncového portu pri komunikácii BitTorrent je väčšie ako 1023.
- **Rovnaké číslo portu pre TCP aj UDP komunikáciu** - Peer používa rovnaké íslo portu, ktoré odošle trackeru pre TCP aj UDP spojenia.

- **Veľké množstvo spojení z jedného UDP portu** - Klient z jedného UDP portu vä šieho ako 1023 kontaktuje ve ký po et iných zariadení takisto na úvajúcich na porte vä šom ako 1023.
- **Veľké množstvo TCP portov** - Pri TCP komunikácií sa pre každého kontaktovaného peera vytvára vlastný port, takže je možné sledova zvýšený po et TCP portov vä ších ako 1023, ktoré komunikovali na port vä ší ako 1023.
- **Čísla TCP a UDP portov z rozsahu 6881 až 6999** - Tieto porty sú prakticky "well-known ports" pre protokol BitTorrent [10].
- **Predchádzajúce odhalenie** - V prípade, že IP adresa už bola predtým odhalená pomocou niektorej z metód h bkovej analýzy paketov, tak je ve ká pravdepodobnos , že všetka alšia komunikácia z tejto IP adresy a portu vä šieho ako 1023 na port vä ší ako 1023 bude komunikácia BitTorrent.

Nevýhodou tohoto spôsobu detekcie je, že takéto správanie môže vykazova nieko ko rôznych P2P protokolov, takže v prípade, že bude v jednom pcapovom súbore prítomných viac protokolov s chovaním podobným BitTorrentu, budú nastáva false positives. Druhým problémom môžu by zase peeri, ktorí sa zamaskujú pod íslo portu z rozsahu 0-1023, ktorí sa nedetekujú vôbec (false negatives). Aby sa predišlo false positives, bude tento spôsob detekcie vo výslednej implementácií volite ný.

5.4 Pomer využívania UDP a TCP transportného protokolu v komunikácií BitTorrent

Pri zoznamovaní sa s protokolom BitTorrent som vykonal sledovanie využitia transportných protokolov UDP a TCP za ú elom zistenia pomeru týchto transportných protokolov v bežných užívateľských podmienkach, aby som vedel, akú komunikáciu mám v nástrojii o akáva ako dominantnú.

Testovanie som vykonal odchytom BitTorrentovej komunikácie s rôznymi nastaveniami BT klienta. Vytvorených bolo celkovo osem pcapových súborov. Prvé štyri súbory obsahujú BitTorrentovú komunikáciu s pôvodnými nastaveniami BT klienta, o znamená povolené rozšírenie uTP. V alších štyroch pcapových súborech sa s ahoval ten istý obsah, ako v prvých štyroch súborech s tým rozdielom, že rozšírenie uTP bolo zakázané. Ako testovací klient som využil Transmission, ktorý je predinštalovaný na opera nom systéme Ubuntu. Odchytený bol iba proces samotného s ahovania dát, nie ich následné seedovanie.

Obsah odchytených pcapových súborov je nasledovný:

- pcap 1: s ahovaný súbor ubuntu-18.04-live-server-amd64.iso, uTP povolené
- pcap 2: s ahovaný súbor ubuntu-18.04-desktop-amd64.iso, uTP povolené
- pcap 3: s ahovaný súbor ubuntu-17.10.1-server-i386.iso, uTP povolené
- pcap 4: s ahovaný súbor ubuntu-17.10.1-desktop-amd64.iso, uTP povolené
- pcap 5: s ahovaný súbor ubuntu-18.04-live-server-amd64.iso, uTP zakázané
- pcap 6: s ahovaný súbor ubuntu-18.04-desktop-amd64.iso, uTP zakázané
- pcap 7: s ahovaný súbor ubuntu-17.10.1-server-i386.iso, uTP zakázané

- pcap 8: s aho vaný súbor ubuntu-17.10.1-desktop-amd64.iso, uTP zakázané

Tabu ka 5.1 obsahuje pomery UDP a TCP spojení v prvých štyroch pcapových súboroch s rozšírením uTP povoleným.

Testovaný súbor	Po et BT paketov	Pomer TCP komunikácie	Pomer UDP komunikácie
pcap 1	1191750	2.7%	97.3%
pcap 2	3184687	5.2%	94.8%
pcap 3	1205753	1.6%	98.4%
pcap 4	2308940	2.5%	97.5%

Tabu lka 5.1: Pomer UDP a TCP paketov pri zapnutom rozšírení uTP

V tabu ke 5.2 sa nachádzajú pomery v súboroch so zakázaným rozšírením uTP.

Testovaný súbor	Po et BT paketov	Pomer TCP komunikácie	Pomer UDP komunikácie
pcap 5	1059260	99.1%	0.9%
pcap 6	2314702	99.87%	0.13%
pcap 7	927713	99.83%	0.17%
pcap 8	1862356	99.81%	0.19%

Tabu lka 5.2: Pomer UDP a TCP paketov pri vypnutom rozšírení uTP

alej som sa zameral na to, ktorý transportný protokol je predvolený na najpoužívanějších BitTorrentových klientoch. Dôvodom tohoto výskumu je môj názor, že bežný používate l BitTorrentovej aplikácie nie je dostato ne znalý problematike transportných protokolov v rámci protokolu BitTorrent nato ko, aby mal potrebu tieto nastavenia meni , o mi potvrdilo aj nieko ko udí z môjho okolia, ktorí pravidelne protokol BitTorrent využívajú. Tým pádom, ke sa zistí, ktorý protokol v týchto aplikáciách implicitne dominuje, tak sa ur í, ktoré z predchádzajúcich dvoch testovaní je možné považova za relevantnejšie.

Zoznam najpoužívanějších klientov som získal z lánku na portáli lifehacker.com, ktorý obsahoval aj anketu, v ktorej itatelia mohli hlasova za BT klienta, ktorý používajú. Do hlasovania sa celkovo zapojilo 13823 itate ov [12]. V tabu ke 5.3 je možné vidie jednotlivých klientov, ich percentuálne zastúpenie itate mi a informáciu o tom, i daný klient má alebo nemá predvolene povolené rozšírenie uTP. K týmto klientom som navyiac dodal ešte dvoch alších, ktorých som sám poznal, ale do hlasovania sa nedostali.

Vzh adom k tomu, že každý z testovaných BT klientov mal od inštalácie protokol uTP povolený, bude sa pri implementácií detek ného nástroja bra v úvahu fakt, že komunikácia BitTorrent v odchytených súboroch môže prebieha obomi transportnými protokolmi v nestálych pomeroch. Množstvo TCP a UDP komunikácie pri s aho vaní pomocou protokolu BitTorrent ve mi závisí od nastavenia BT klientov jednotlivých peerov v swarme. V prípade, že bude ma vä šina peerov rozšírenie uTP povolené, bude v tomto swarme dominova UDP komunikácia. V opa nom prípade bude vä šina spojení prebieha pomocou protokolu TCP.

Názov BT klienta	Zastúpenie itate mi	uTP povolené
uTorrent	42.47%	ÁNO
Transmission	21.23%	ÁNO
qBitTorrent	18.75%	ÁNO
Deluge	10.7%	ÁNO
Tixati	6.85%	ÁNO
Vuze	-	ÁNO
BitTorrent	-	ÁNO

Tabulka 5.3: Pomery využitia jednotlivých klientov itate mi a informácia o predvolenom využití rozšírenia uTP

5.5 Šifrovanie komunikácie BitTorrent

Pred implementáciou detekného nástroja som vykonal ešte jednu štúdiu, pri ktorej som zisoval množstvo šifrovanej tracker komunikácie. ale som sa zameril na šifrovanie Peer protokolu.

Šifrovanie komunikácie pri trackeroch závisí od toho, cez aký protokol prebieha. S HTTP a UDP servermi sa komunikuje nešifrovane a s HTTPS servermi šifrovane. Preto, som sledoval výskyt týchto serverov analýzou torrentových súborov. Testy boli vykonané nad 1230 súbormi. 1130 súborov mi bolo poskytnutých kolegom Martinom Grnárom, ktorý pre jeho bakalársku prácu napísal skript, ktorý automatizovane prehľadáva indexovacie servery a s ňou obsahuje torrentové súbory [11]. ďalších 100 vzoriek som manuálne stiahol z asi 15 najznámejších torrentových webov. Nad týmito vzorkami som spustil skript v jazyku Python, ktorý v nich skúmal množstvo HTTP, HTTPS a UDP trackerov. V tabu ke 5.4 sa nachádza zastúpenie jednotlivých typov serverov v súboroch a v tabu ke 5.5 je množstvo súborov s aspo jedným HTTPS trackerom a súborov bez HTTPS trackeru.

	HTTP trackery	HTTPS trackery	UDP trackery
Počet	1795	25	1817
Pomer	49%	1%	50%

Tabulka 5.4: Množstvo HTTP, HTTPS a UDP trackerov obsiahnutých v súboroch

	obsahuje HTTPS tracker	neobsahuje HTTPS tracker
Počet	1218	12
Pomer	99%	1%

Tabulka 5.5: Súbory, ktoré obsahovali aspo jeden HTTPS tracker a ktoré neobsahovali ani jeden HTTPS tracker

Z predchádzajúcich zistení vyplýva, že komunikácia s trackermi vo väčšine prípadov šifrovaná nie je, no nemali by sa tieto výsledky veľa globalizovať. Príkladom môžu byť torrentové súbory stiahnuté z najznámejšieho českého torrentového webu cztorrent.net (zároveň funguje aj ako privátny tracker), ktoré takmer všetky obsahujú aspo jeden HTTPS

tracker. Takisto sa môže stať, že aj keď daný torrentový súbor HTTPS tracker obsahuje, tak ho klientská aplikácia nevyužije, prípadne bude tento server nefunkčný a využije sa alternatívny, nešifrovaný HTTP alebo UDP server.

Keď sa týka šifrovania Peer protokolu je už zistenie množstva šifrovania trochu zložitejšie. Šifrovanie Peer protokolu primárne nevzniklo za účelom zabezpečenia jeho obsahu, ale ako spôsob obchádzania traffic shapingu poskytovateľov internetového pripojenia, pretože niektorí komunikáciu BitTorrent značne obmedzujú [2, 1, 19]. Táto komunikácia však nemá žiadne oficiálne rozšírenie týkajúce sa jej šifrovania. Šifrovanie plne závisí od tvorca BT klienta, či ho do svojej aplikácie zahrnie a akým spôsobom ho implementuje. Rozdielne implementácie môžu spôsobiť nekompatibilitu medzi jednotlivými BitTorrentovými klientami. Dokonca Bram Cohen, samotný zakladateľ protokolu BitTorrent, šifrovanie kritizuje [6]. Preto sa niektorí tvorcovia klientských aplikácií spojili k tomu, aby vytvorili vzájomnú kompatibilitu šifrovania svojich klientov [2]. Navyše, aby sa predišlo nekompatibilitě existujúcich klientov, tak väčšina z nich má ako jedno z možných nastavení preferovanie šifrovania, čo znamená, že BT klient môže s iným klientom, komunikovať šifrované v prípade, že majú kompatibilnú implementáciu šifrovania. V opačnom prípade bude ich komunikácia nešifrovaná. Avšak existuje možnosť šifrovanie vynútiť, čo môže mať za následok značné spomalenie sťahovania obsahu, pretože sa počet peerov v swarme obmedzí len na tých, so zapnutým a kompatibilným šifrovaním [19].

Mieru šifrovania Peer protokolu je ťažké odhadnúť, alebo odtestovať. Všetko závisí od peerov v swarme. Môžu nastať prípady, že všetci peeri budú mať šifrovanie povolené a ich BT klienti budú kompatibilní, takže komunikácia bude šifrovaná úplne. Rovnako sa môže stať, že v swarme bude mať väčšina peerov šifrovanie zakázané, alebo ich klienti budú nekompatibilní a tým pádom ich komunikácia nebude šifrovaná skoro vôbec.

Kapitola 6

Implementácia nástroja

V tejto kapitole bude popísaná už samotná implementácia detekcie nástroja, ktorý vykonáva detekciu na základe navrhnutých spôsobov detekcie popísaných v kapitole 5. Nástroj som implementoval v jazyku Python. Implementované boli dve verzie. Prvá využívala knižnicu pre spracovanie paketov *Scapy*. Tá však nebola z dôvodu časovej náročnosti vyhovujúca, preto bola vytvorená druhá varianta za pomoci knižnice *dpkt*. Popísaná bude iba druhá varianta, no v závere nej testovacej fázi budú obe verzie navzájom porovnané.

6.1 Nástroj pre detekciu komunikácie BitTorrent

Úlohou tohoto nástroja je offline analýza z pcapového súboru a detekcia IP adries a portov podozrivých z využívania protokolu BitTorrent pomocou heuristického analyzovania paketov doprevádzanej analýzou tokov. Taktiež sa z tracker a peer komunikácií extrahujú info hashe, ktoré sa dajú pri ďalšom manuálnom, či automatizovanom spracovaní ďalej využiť. Vstupom programu je súbor formátu pcap.

Vstupné parametre programu:

- -f filename [povinný]- súbor filename vo formáte pcap.
- -s [voliteľný] - povolí detekciu pomocou analýzy tokov.
- -n IP_network/network_mask [voliteľný] - vyfiltruje výstup programu iba na IP adresy, ktoré obsahujú zadany prefix. V prípade, že sa maska nezadá, program funguje ako filter konkrétnej adresy. Funguje iba pre IPv4
- -o outputfile [voliteľný] - výstup sa uloží do súboru outputfile vo forme XML.
- -d [voliteľný] - povolí výpis po nájdených IP, portov a info hashov. Iba na terminál.

V prípade, že sa nezadá outputfile, bude výstup vypísaný na terminál.

6.1.1 Rozpoznávanie komunikácie

Základom programu je čítanie jednotlivých paketov zo súboru vo formáte pcap. Pakety, ktoré neobsahujú TCP, alebo UDP protokol sú automaticky odfiltrované. Ešte pred klasifikáciou algoritmom sa odtestuje, či niektorá z IP adries spolu s jej portom z paketu už nebola v minulosti usvedčená na základe obsahu paketu z BitTorrent komunikácie. V prípade že áno, označí sa jej koncový bod rovnako za usvedčený z BitTorrent komunikácie a

prejde sa k ítaniu ďalšieho paketu. V prípade, že sa kombinácia IP adres a portov v pakete vyskytuje prvý krát, predá sa paket klasifika nému algoritmu, ktorý na základe typu transportného protokolu a ve kosti paketu ur í vzorky, ktoré odtestuje na prítomnos signatúr charakteristických pre protokol BitTorrent. Tento algoritmus je zakreslený diagramom na obrázku 6.1.

V prípade, že sa jedná o TCP paket, ktorý má ve kos 200 až 600 bajtov, testuje sa na prítomnos HTTP komunikácie s trackerom (pod a sekcie 5.1). H adajú sa v jeho payloade re azce *GET /announce* a *GET /scrape*. V prípade, že sa potvrdí prítomnos announce dotazu, vytiahne sa z jeho náplne info hash torrentu a port, na ktorom daný uzol na úva. Vzh adom k tomu, že vytiahnutý info hash je vo formáte urlencoded, tak sa musí najskôr dekódova a previes na tla ite nú podobu hexadecimálnych hodnôt. A až následne sa vo formáte re azca uloží do ur enej štruktúry. Ke že boli nájdené konkrétne charakteristiky protokolu BitTorrent, môže sa zdrojová IP adresa paketu spolu s extrahovaným portom z payloadu uloží do štruktúry uchováajúcej odhalené zariadenia. V prípade scrape dotazu sa uloží iba info hash, pretože port sa v týchto správach neprenáša.

Ak sa nájde TCP paket s ve kos ou 100 až 200 bajtov, testuje sa na prítomnosť Peer protokolu (pod a sekcie 5.2). Ako signatúra sa h adá hodnota *19 (0x13 hex)* nasledovaná re azcom *BitTorrent protocol*. V prípade, že bola signatúra nájdená, IP adresy zdrojového aj koncového zariadenia spolu s ich portami sa uložia do štruktúry uchováajúcej odhalené zariadenia, pretože znovu je možné s istotou ur í , že sa jedná o h adanú komunikáciu. Navyiac sa vyextrahuje info hash, ktorý sa nachádza medzi 29 až 49 bajtom payloadu daného paketu. Vzh adom k tomu, že je info hash v binárnej podobe, prevedie sa na tla ite nú podobu hexadecimálnych hodnôt a v prípade, že sa ešte predtým nevyskytol, sa uloží do štruktúry uchováajúcej info hashe.

UDP paket s ve kos ou 100 až 200 bajtov sa takisto testuje na prítomnosť Peer protokolu a prebieha úplne rovnakým spôsobom, ako pri TCP pakete, s jedným rozdielom, že ke sa signatúra detekuje, tak sa info hash vyextrahuje medzi 49 až 69 bajtom payloadu. Tento posun nastáva kvôli uTP hlavi ke, ktorej ve kos je 20 bajtov.

Ak sa nedetekuje ani jedna z predchádzajúcich signatúr a je povolený prepína -s, zavolá sa algoritmus, ktorý vykonáva detekciu na základe správania protokolu BitTorrent a buduje štruktúru uchováajúcu informácie o tokoch. Tento algoritmus je zakreslený v diagrame na obrázku 6.2. Algoritmus odtestuje, i je v danom pakete zdrojový aj koncový port vä ší ako 1023 a ak áno, zistí sa, i aspo jeden z portov paketu nie je z intervalu 6881 až 6999. V prípade že áno, uložia sa IP adresy a porty ako podozrivé, pretože daný rozsah spadá do 'well-known ports' protokolu BitTorrent. V opa nom prípade sa odtestuje, i aspo jedna z IP adres paketu už nebola odhalená na základe obsahu paketov. Ak áno, IP adresy a porty paketu sa uložia ako podozrivé. Ak paket nesp a ani jednu z predchádzajúcich kritérií, uložia sa pre obidve IP adresy a ich porty koncový bod. Na konci, ke sa už pre íta celý pcapový súbor sa vykoná finálna analýza týchto dát vybudovaných analýzou tokov a všetky IP adresy, ktoré komunikovali z UDP portu na aspo 30 rôznych koncových IP adres, alebo majú vytvorených aspo 30 TCP portov sa ozna ia ako podozrivé z komunikácie BitTorrent. Podozrivé preto, lebo na základe analýzy tokov nie je možné s presnos ou ur í , že sa naozaj jednalo o komunikáciu BitTorrent a nie o iný P2P protokol. Táto analýza je voliteľná, pretože je ve ká šanca vzniku false positives a false negatives.

Nakoniec sa vytvorí XML výstup programu. V prípade, že bol zadán prepína -n a bola ur ená adresa, i prefix, na ktoré sa má program zamera , tak sa do výstupu zapíšu informácie iba o tých IP adresách, ktoré vyhovujú zadanému filtru. Príklad výstupu programu je možné vidie vo výpise 6.1

Trieda packetClass

Táto trieda slúži pre ukladanie potrebných dát z paketu, ktorý bol načítaný zo súboru, pretože samotná knižnica dpkt neponúka veľa pohodlnú prácu s paketmi. Obsahuje atribúty source IP, destination IP, source Port, destination Port, payload a metódy, pre získavanie a nastavovanie týchto atribútov.

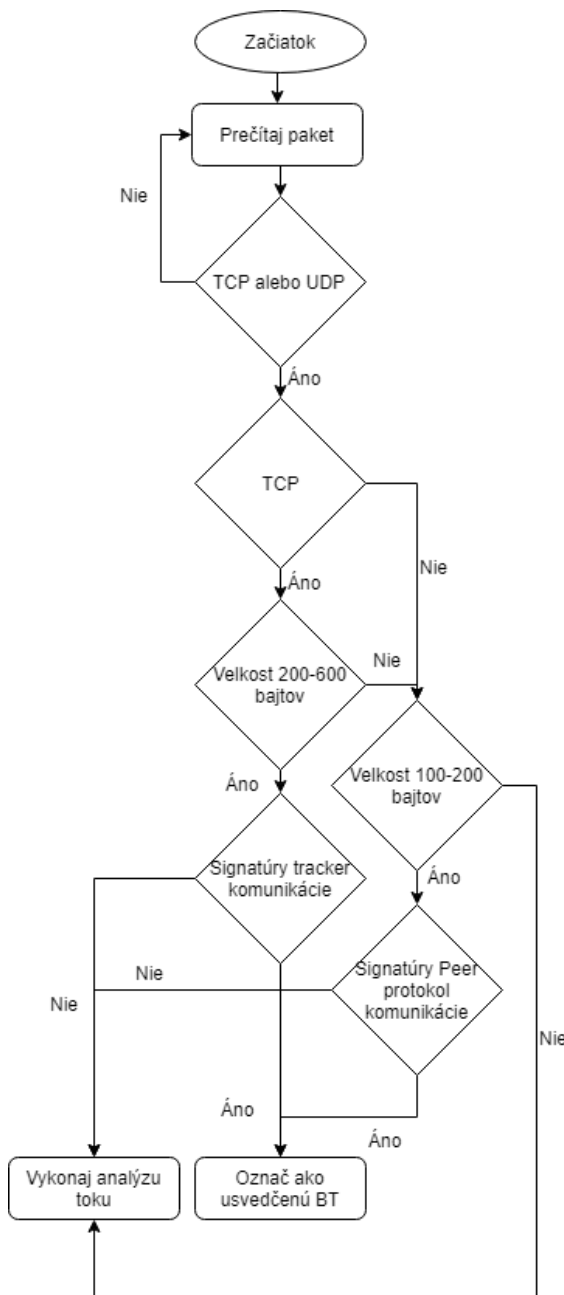
Štruktúry convinced a suspected

Štruktúra convinced obsahuje IP adresy a porty, ktoré boli odhalené pri tracker, alebo peer komunikácií, takže s istotou boli zapojené do komunikácie BitTorrent.

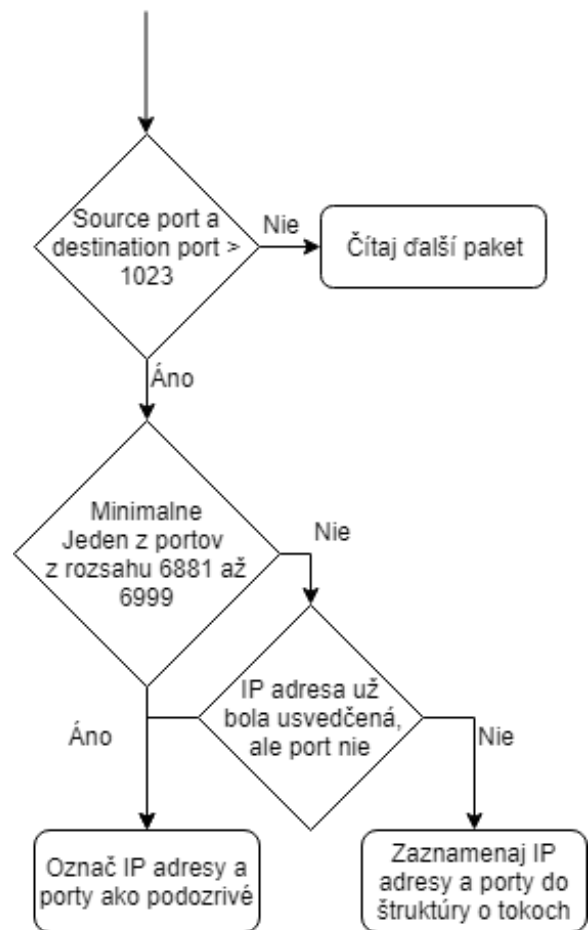
Štruktúra suspected sa zostaví nad vybudovanými tokmi, ktoré sa budujú postupne z paketov, ktorých neodhalila detekcia na základe obsahu paketov. Sú to IP adresy a porty, o ktorých nie je možné s istotou povedať, že komunikovali pomocou protokolu BitTorrent, no vykazujú také správanie. Jedná sa iba o odhad.

```
<?xml version="1.0" ?>
<Root>
  <Convinced>
    <IP name="194.109.162.159">
      <TCP>
        <Port>16881</Port>
      </TCP>
    </IP>
    <IP name="202.88.252.50">
      <TCP>
        <Port>18231</Port>
      </TCP>
    </IP>
    <IP name="213.122.214.127">
      <TCP>
        <Port>3860</Port>
        <Port>3861</Port>
      </TCP>
      <UDP>
        <Port>41730</Port>
      </UDP>
    </IP>
  </Convinced>
  <Info_Hashes>
    <Hash>0164fe7ef1105c57764170edf603c439d64214f1</Hash>
  </Info_Hashes>
</Root>
```

Výpis 6.1: Příklad XML výstupu obsahující iba usvedčené zariadenia



Obrázek 6.1: Diagram popisujúci zjednodušený algoritmus detekcie pomocou obsahu paketov



Obrázek 6.2: Diagram popisujúci zjednodušený algoritmus detekcie pomocou analýzy tokov

Kapitola 7

Testovanie

Aby sa overili charakteristiky, chyby, i presnos implementovaného nástroja, bolo potrebné vykona jeho dôkladné testovanie. Testovanie sa delilo na testy asovej a pamä ovej náro - nosti a testy presnosti detekcie.

7.1 asová a pamä ová náro nos nástroja

Pri tomto testovaní sa nijakým spôsobom ani okrajovo netestovala presnos , ale sledovala sa asová náro nos , vy aženie procesora a pamä ové nároky implementovaného programu. Výsledky boli následne porovnané s náro nos ou prvej verzie detek ného programu, ktorá využívala knižnicu Scapy. Pre testovanie sa využil laptop so systémom Ubuntu 16.04 LTS s procesorom I5-5200U, 256GB diskom SSD a 8GB RAM.

Pre testovanie sa použilo pä pcapových súborov rôznych ve kostí, s rôznymi po tami paketov. S ahovaný obsah v jednotlivých súboroch bol následovný:

- pcap 1: s ahovaný program winrar, zapnuté uTP, ve kos pcapu 200 MB
- pcap 2: s ahovaný ubuntu server, zapnuté uTP, ve kos pcapu 800 MB
- pcap 3: odchyt siete VUT obsahujúci seedbox, zapnuté uTP, ve kos pcapu 2000 MB
- pcap 4: s ahované 4 rôzne ubuntu distribúcie, vypnuté uTP, vypnuté šifrovanie, ve - kos pcapu 3000 MB
- pcap 5: s ahované 4 rôzne ubuntu distribúcie, vypnuté uTP, vynútené šifrovanie, ve - kos pcapu 3300 MB

Program s knižnicou dpkt som otestoval najskôr bez analýzy tokov a následne som rovnaké testovanie vykonal s analýzou tokov zapnutou, aby som zistil pamä ovú a asovú náro nos oboch variánt programu. Výstup programu bol vždy uložený do súboru. Výsledky testov pre asovú a maximálnu pamä ovú náro nos , vy aženie procesoru a rýchlos spracovania programu bez zapnutej detekcie pomocou analýzy tokov sú uložené v tabu ke 7.1. Výsledky programu pri povolenej detekcií pomocou analýzy tokov sú uložené v tabu ke 7.2.

Rovnaké testy sa následne vykonal na programe využívajúcom knižnicu Scapy a asová náro nos sa relatívne porovнала s asovou náro nos ou nástroja využívajúceho knižnicu dpkt s povolenou analýzou tokov uloženou v tabu ke 7.2. Výsledky tohoto testovania sú uložené v tabu ke 7.3.

Súbor	Počet Paketov	Vyťaženie procesoru	Spotrebovaná pamäť (max)	čas spracovania	Rýchlosť spracovania
pcap1	213950	100%	16MB	10.2s	20MB/s
pcap2	244979	100%	17MB	12s	66.7MB/s
pcap3	3478037	100%	22.5MB	174s	11.5MB/s
pcap4	2608610	100%	17.2MB	136s	22.5MB/s
pcap5	2863094	100%	16.9MB	98s	33.7MB/s

Tabuľka 7.1: Výsledky testov náročnosti programu bez zapnutej analýzy tokov

Súbor	Počet Paketov	Vyťaženie procesoru	Spotrebovaná pamäť (max)	čas spracovania	Relatívny asový nárast
pcap1	213950	100%	16.7MB	10.8s	5.9%
pcap2	244979	100%	17MB	12.4s	3.3%
pcap3	3478037	100%	23.5MB	179s	3%
pcap4	2608610	100%	17.4MB	136s	0%
pcap5	2863094	100%	17.1MB	101s	3%

Tabuľka 7.2: Výsledky testov náročnosti programu so zapnutou analýzou tokov

Podľa vykonaných testov sa zistilo, že pri zapnutej analýze tokov sa spotrebuje o 3 až 6 percent viac času a od 1 do 4,5 percenta viac pamäťových prostriedkov, než varianta bez analýzy tokov.

Súbor	Počet Paketov	Vyťaženie procesoru	Spotrebovaná pamäť (max)	čas spracovania	asový rozdiel
pcap1	213950	100%	40MB	100s	+909%
pcap2	244979	100%	35MB	140s	+1129%
pcap3	3478037	100%	70MB	1500s	+838%
pcap4	2608610	100%	54MB	1200s	+882%
pcap5	2863094	100%	50MB	860s	+851%

Tabuľka 7.3: Výsledky testov náročnosti programu využívajúceho knižnicu Scapy

Na výsledkoch testov je možné vidieť, že program využívajúci knižnicu Scapy spotrebuje viac, ako dvojnásobok pamäťových prostriedkov, čo verzia s knižnicou dpkt a spracováva dáta v niektorých prípadoch viac, než desaťkrát dlhšie. Problémom knižnice Scapy je pomalé načítanie a dekódovanie jednotlivých paketov, ktoré samotné skonsumuje až okolo 95% spotrebovaného času. Tretí testovaný súbor dokázala prečítať a spracovať až za 25 minút, čo ju úplne nepoužiteľnou pre väčšie pcapové súbory. Časové nedostatky knižnice Scapy boli hlavným dôvodom pre zmenu knižnice a prepísanie celého programu s využitím knižnice dpkt. Zmyslom tohoto porovnania je poukázať na možné nevýhody a problémy, ktoré môžu vzniknúť po zvolení nevhodných technológií.

V súčasnom stave by sa dal program zrýchliť drobnými optimalizáciami. Druhé možné riešenie, ako by sa dal program zrýchliť stále v rámci jazyka Python je napísať modul pre načítanie a dekódovanie pcapových súborov, ktorý by z nich extrahoval iba tie dáta, ktoré sú potrebné pre detekciu BitTorrentovej komunikácie a žiadne iné. Posledná, najefektívnejšia možnosť, ako zrýchliť detekciu je prepísať program do niektorého nižšieho programovacieho jazyka ako C, C++.

7.2 Presnosť nástroja

Odstestovať presnosť nástroja bolo ťažké hlavne z toho dôvodu, že neexistuje spôsob, ktorým by sa dala vykonať analýza testovacích dát úplne presne. Samozrejme platí fakt, že čím menšia sieť sa dá odchytať, tým presnejšia analýza sa dá vykonať.

K analýze testovacích dát existujú tri najzákladnejšie metódy:

- Ručná analýza
- Analýza iným klasifikátorom
- Ground Truth Verification Frameworky

Ručná analýza môže byť presná v prípade, že sa vykoná dôkladne a na rozumnom množstve dát. Je však pomalá, pretože vyžaduje viac času na spracovanie, než automatizované prístupy.

Analýza iným, už existujúcim klasifikátorom funguje na tom princípe, že o akékoľvek výsledky testovania stanoví existujúci klasifikátor. S tým však môžu nastávať isté problémy. Vzhľadom k tomu, že existujúci klasifikátor tiež určite využíva heuristiky analýzy obsahu paketov a analýzy tokov, môže byť v určitých prípadoch nepresný, napríklad keď sú pakety šifrované. Potom v prípade zrovnávania výsledkov existujúceho klasifikátora a testovaného klasifikátora v prípade nezhody je ťažké určiť, či je chyba v existujúcom, alebo testovanom klasifikátore [10].

Frameworky Ground Truth Verification fungujú podobne ako klasifikátory, ale miesto nejakého určitého výsledku sa vytvoria výstupy, ktoré sú prehľadné a prispôbené pre pohodlnejšiu ručnú analýzu. Na základe ručnej analýzy sa vytvoria o akékoľvek výsledky. Príkladom takéhoto frameworku môže byť "Ground Truth Verification System" vytvorený v rámci práce [4].

Ako metódu overenia výsledkov som zvolil existujúci klasifikátor spolu s metódou ručnej analýzy pcapových súborov. Ako existujúci klasifikátor som využil program nDPI, ktorý bol postavený na základe zrušeného projektu OpenDPI, ktorý prevzal tím vývíjajúci systém ntop. Program nDPI je klasifikátor na základe obsahu paketov, ktorý je doplnený ďalšími malými heuristikami k zvýšeniu jeho presnosti. Výsledky programu nDPI v procese testovania slúžia ako minimum toho, čo by mal implementovaný program detekovať. Ručná analýza, na druhú stranu slúži ako maximum, ku ktorému by sa mal program blížiť, ale nad touto hranicou už sú isté false positives.

Testovanie presnosti bolo rozdelené do štyroch častí podľa toho, do akej miery je možné vytvoriť ručnú analýzu pcapových súborov a tú následne porovnať s výstupmi programu:

1. Detekcia v referenčných pcapových súboroch dostupných na wiki stránkach programu Wireshark, ktoré obsahujú iba nešifrovanú komunikáciu BitTorrent a žiadnu inú. Pri tomto testovaní sa skúma funkcia hĺbkovej analýzy paketov, čo znamená, že všetky prítomné IP adresy by mali byť nájdené bez využitia prepínača -s.
2. Detekcia v rámci pcapových súborov odchytených na mojom koncovom zariadení.
3. Detekcia v pcapovom súbore, ktorý bol odchytený v sieti VUT a obsahuje v sebe seedbox. Súbor mi bol poskytnutý vedúcim bakalárskej práce.
4. Detekcia v súboroch, v ktorých by potenciálne mohli vzniknúť false positives. Sú to súbory neobsahujúce žiadnu BitTorrentovú komunikáciu. Tieto testy sú primárne určené na odtestovanie presnosti analýzy tokov.

7.2.1 Testovanie na referenčných súboroch

V tejto podkapitole popíšem detekciu, ktorú som vykonal v krátkych referenčných súboroch, vo ne dostupných na wiki stránkach programu Wireshark, ktoré obsahujú niečo ko názorných BitTorrentových paketov.

Obsah súborov je nasledovný:

- pcap 1: názorná TCP peer protokol komunikácia medzi dvomi peermi, nešifrovaná komunikácia
- pcap 2: komunikácia medzi siedmymi peermi plus HTTP tracker komunikácia, nešifrovaná komunikácia

Test pre súbor pcap 1 bol jednoduchý a oba programy dokázali bez problému nájs všetky o akávané IP adresy, porty aj info hash. Tabu ka 7.4 obsahuje výsledky detekcie.

Typ analýzy	Počet IP adries	Počet TCP portov	Počet UDP portov	Počet info hashov
Ru ná analýza	2	2	0	1
nDPI	2	2	0	1
Výstup programu	2	2	0	1

Tabulka 7.4: Výsledok testu pre referenčný súbor pcap 1

Druhý test pre súbor pcap 2 prebehol takisto v poriadku. Výstup programu nDPI obsahoval o jednu IP adresu navyše, no nejedná sa však o false positive, ale program nDPI do detekcie IP adries a portov zah ŕa aj trackery s ktorými implementovaný program nepo ŕíta. Výsledok testu je možné vidie v tabu ke 7.5.

Typ analýzy	Počet IP adries	Počet TCP portov	Počet UDP portov	Počet info hashov
Ru ná analýza	7	14	0	1
nDPI	8	15	0	1
Výstup programu	7	14	0	1

Tabulka 7.5: Výsledok testu pre referenčný súbor pcap 2

Výsledok tohoto testovania je ve mi pozitívny. Nájdene boli všetky o akávané IP adresy so všetkými ich portami. Navyše boli vyextrahované aj o akávané info hashe. Tento test však nemá nijakým spôsobom ani vzdialene simulova využitie nástroja v skuto ných podmienkach. Zameriava sa na testovanie funk nosti a presnosti implementovaných algoritmov vykonávajúcich detekciu na základe obsahu nešifrovaných paketov.

7.2.2 Testovanie na koncovom zariadení

Toto testovanie som vykonal nad šiestimi pcapovými súbormi, ktoré boli odchytené na mojom koncovom zariadení. Použitý bol BitTorrentový klient Transmission a vä šina ostatných služieb na zariadení bola vypnutá, aby sa zabránilo chybným detekciám false positives. Obsahy súborov boli nasledovné:

- pcap 1: 4 s ahoované linuxové distribúcie (4 info hashe), vypnuté uTP, preferované šifrovanie

- pcap 2: 4 s ahované linuxové distribúcie (4 info hashe), vypnuté uTP, vynútené šifrovanie
- pcap 3: 4 s ahované linuxové distribúcie (4 info hashe), zapnuté uTP, preferované šifrovanie
- pcap 4: 4 s ahované linuxové distribúcie (4 info hashe), zapnuté uTP, vynútené šifrovanie
- pcap 5: S ahovaná distribúcia ubuntu server, zapnuté uTP, vynútené šifrovanie
- pcap 6: S ahovaný program WinRAR, zapnuté uTP, vynútené šifrovanie

Výsledky testov pre pcap 1 sú pozitívne. V prípade implementovaného programu vznikol jeden false negative, pretože HTTP tracker komunikoval na porte 6969. Jedná sa iba o false positive portu, nie IP adresy, pretože tracker server fungoval aj ako peer. Výsledky testu pre pcap 1 sa nachádzajú v tabuľke 7.6.

Typ analýzy	Počet IP adries	Počet TCP portov	Počet UDP portov	Počet info hashov
Ručná analýza	677	2971	291	4
nDPI	670	2809	289	4
Výstup programu	677	2972	291	4

Tabuľka 7.6: Výsledok testu pre pcap 1

Pri súbore pcap 2 program nDPI o sa týka detekcie TCP zaostával za implementovaným nástrojom. Navyše program nDPI detekoval o 2 info hashe viac, než sa skutočne s ahovalo, čím v jeho prípade vznikli false positives. Výsledky tohoto testovania sú uložené v tabuľke 7.7.

Typ analýzy	Počet IP adries	Počet TCP portov	Počet UDP portov	Počet info hashov
Ručná analýza	507	2114	192	4
nDPI	449	1632	188	6
Výstup programu	503	2113	189	4

Tabuľka 7.7: Výsledok testu pre pcap 2

V súbore pcap 3 sa nachádzalo viac UDP komunikácie. Program nDPI trochu zaostával v detekcii TCP aj UDP. Navyše program nDPI detekoval 5 false positive info hashov. Výsledky testu sú uložené v tabuľke 7.8.

Typ analýzy	Počet IP adries	Počet TCP portov	Počet UDP portov	Počet info hashov
Ručná analýza	1396	1230	1536	4
nDPI	1316	1105	1480	9
Výstup programu	1385	1225	1530	4

Tabuľka 7.8: Výsledok testu pre pcap 3

Pri súbore pcap 4 program nDPI znovu hodne zaostával hlavne o sa týka detekcie TCP portov. Taktiež jeho detekcia znovu obsahovala false positives pri info hashoch. o sa týka implementovaného nástroja detekovalo sa o 7 TCP portov viac, než sa predpokladalo ru nou analýzou. Výsledky detekcie sú uložené v tabu ke 7.9.

Typ analýzy	Po et IP adries	Po et TCP portov	Po et UDP portov	Po et info hashov
Ru ná analýza	1932	3899	2017	4
nDPI	1863	2944	1968	8
Výstup programu	1926	3906	2016	4

Tabulka 7.9: Výsledok testu pre pcap 4

Súbor pcap 5 obsahoval kompletne šifrovanú komunikáciu Peer protokolu. Jediná nešifrovaná komunikácia bola medzi peerom a HTTP serverom. V tomto prípade testu mal program nDPI presnejšiu detekciu UDP portov, ale detekcia TCP portov bola znovu v porovnaní s implementovaným nástrojom znate ne slabšia.

Typ analýzy	Po et IP adries	Po et TCP portov	Po et UDP portov	Po et info hashov
Ru ná analýza	213	72	226	1
nDPI	210	15	222	1
Výstup programu	207	72	216	1

Tabulka 7.10: Výsledok testu pre pcap 5

Súbor pcap 6 obsahoval kompletne šifrovanú komunikáciu aj medzi trackerom aj medzi peermi. Na výsledku testu je vidno, že ani jeden z testovaných programov si s TCP komunikáciou moc neporadil. V Prípade programu nDPI znovu vznikli false positives pri info hashoch, troch IP adresách a dvoch UDP portoch. Výsledky tohoto testovania sa nachádzajú v tabu ke 7.11

Typ analýzy	Po et IP adries	Po et TCP portov	Po et UDP portov	Po et info hashov
Ru ná analýza	118	56	118	0
nDPI	121	9	120	3
Výstup programu	118	2	118	0

Tabulka 7.11: Výsledok testu pre pcap 6

7.2.3 Testovanie v odchyte siete VUT

alšie testovanie programu som vykonal v súbore odchyteného v rámci siete VUT. O tomto súbore moc toho známe nie je. Jediná známa informácia je, sa v om nachádza odchytený seedbox na IP adrese 147.229.12.246.

Implementovaný nástroj dokázal seedbox detekova spolu s jeho 2064 TCP portami a 335 UDP portami. Celková detekcia súboru bola zhrnutá do tabu ky 7.12. Implementovaný program navyiac dokázal odhali takmer všetku komunikáciu, ktorá bola ozna ená ru nou analýzou odchyteného súboru. Program nDPI je v analýze vä šieho súboru dos neefektívny, pretože implementovaný program dokázal odhali až vyše dvojnásobok TCP portov

a takmer dvojnásobok UDP portov. Implementovaný nástroj však vyextrahoval menej info hashov. Vzhľadom k tomu, že nie je známy bližší obsah odchyty internátnej siete, je ťažko povedať, či nDPI detekoval false positives, alebo implementovaný nástroj niektoré info hashy nedetekoval (false negatives). Z výstupu je taktiež zjavné, že implementovaný program vykonal nieko ko zlých detekcií, no tie boli spôsobené analýzou tokov.

Typ analýzy	Počet IP adries	Počet TCP portov	Počet UDP portov	Počet info hashov
Ručná analýza	5851	63889	4022	0
nDPI	2129	22815	2151	174
Výstup programu	5883	62821	3992	109

Tabulka 7.12: Výsledok testu pre pcap siete VUT

7.2.4 Testovanie false positives

Testovanie false positives prebiehalo rovnako, ako v prípade koncového zariadenia s tým rozdielom, že testované súbory neobsahovali žiadnu BitTorrentovú komunikáciu.

Obsahy súborov boli nasledovné:

- pcap 1 - prehliadanie webu + HTTP sťahovanie
- pcap 2 - Sledovanie Twitch.tv a Youtube
- pcap 3 - Hranie hry Counter Strike Global Offensive (využíva UDP server s portom väšším ako 1023)

Pri všetkých troch vstupných súboroch platilo to isté, že ani jeden z programov by nemal detekovať komunikáciu BitTorrent. Vzhľadom k tomu, že pre všetky tri vstupné súbory boli rovnaké výstupy, boli zhrnuté do jednej tabuľky 7.13.

Typ analýzy	Počet IP adries	Počet TCP portov	Počet UDP portov	Počet info hashov
Ručná analýza	0	0	0	0
nDPI	0	0	0	0
Výstup programu	0	0	0	0

Tabulka 7.13: Výstup programu pre pcap 1, pcap 2 a pcap 3

Kapitola 8

Záver

V rámci tejto práce sa podarilo splniť cieľ a vytvoriť detekčný nástroj v jazyku Python, ktorý dokáže z pcapového súboru detekovať IP adresy a porty, ktoré sa podieľajú po as doby odchyty súboru na BitTorrentovej komunikácii. Navyše dokáže z komunikácií s tracker servermi a medzi samotnými peerami získať info hashov s obsahom.

Existujú možnosti, ktorými by sa dal nástroj zrýchliť a zdokonaľiť. Prvou možnosťou je vytvoriť určité optimalizácie momentálneho kódu v jazyku Python. Druhou, efektívnejšou možnosťou je prepísanie programu do nízkoúrovňového jazyka ako C, či C++.

Presnosť nástroja veľmi závisí od miery šifrovania komunikácie BitTorrent. V prípade, že užívatelia v odchytenom pcapovom súbore nevyužívajú šifrovanú variantu protokolu, tak bude presnosť detekcie vysoká. V prípade, že sa komunikácia šifruje, závisí veľmi od toho, či sa jedná o TCP komunikáciu, alebo uTP komunikáciu. Hlavné v prípade TCP komunikácie tvorí šifrovanie veľký problém, pretože peeri si vytvárajú veľké množstvo TCP portov a detekovať všetky z nich iba na základe analýzy tokov je obtiažne a môže vzniknúť veľa false positives. Keď sa týka UDP je situácia priaznivejšia, vzhľadom na to, že peeri využívajú poväčšinou iba jeden UDP port na komunikáciu so všetkými vzdialenými peerami, tak je omnoho jednoduchšie odhaliť túto komunikáciu či už šifrovanú, tak aj nešifrovanú. V prípade šifrovanej varianty protokolu závisí veľmi aj na tom, či sa v súbore popri protokole BitTorrent využíval aj iný P2P protokol. V prípade, že nie, false positives by veľmi vznikali nemali. V opačnom prípade však bude pravdepodobne analýza tokov pomerne neúplná.

Na túto bakalársku prácu by sa dalo nadviazať vytvorením modulu pre tento nástroj, prípadne samostatným programom, ktorý by prešiel XML výstup detekčného programu a následne na internete vyhľadal informácie o uložených info hashoch, ako počet súborov, názvy súborov, veľkosť celkového obsahu popísaného daným info hashom a tak podobne. Prípadne už spomínané prepísanie tohoto programu do jazyka C/C++ a tým zvýšenie rýchlosti detekcie a jeho prípadné rozšírenie. Rovnako by sa dal výstup nástroju využiť pre určité štatistické účely, napríklad analýzu najčastejšie sa vyskytujúcich čísel portov, ktoré protokol BitTorrent využíva.

Literatura

- [1] Avoid traffic shaping. [Online; navštíveno 30.04.2018].
URL http://wiki.vuze.com/w/Avoid_traffic_shaping
- [2] BitTorrent protocol encryption. [Online; navštíveno 30.04.2018].
URL https://en.wikipedia.org/wiki/BitTorrent_protocol_encryption
- [3] Buford, J. F.; Yu, H.; Lua, E. K.: *P2P Networking and Applications*. Morgan Kaufmann Publishers, 2009, ISBN 978-0-12-374214-8.
- [4] Canini, M.; Li, W.; Moore, A. W.; aj.: GTVS: Boosting the Collection of Application Traffic Ground Truth. In *Traffic Monitoring and Analysis*, editace M. Papadopouli; P. Owezarski; A. Pras, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ISBN 978-3-642-01645-5, s. 54–63.
- [5] Cohen, B.: The BitTorrent Protocol Specification. [Online; navštíveno 30.04.2018].
URL http://www.bittorrent.org/beps/bep_0003.html
- [6] Cohen, B.: Obfuscating BitTorrent. Leden 2006, [Online; navštíveno 30.04.2018].
URL <https://bramcohen.livejournal.com/29886.html>
- [7] Ethernet capture setup. [Online; navštíveno 30.04.2018].
URL <https://wiki.wreshark.org/CaptureSetup/Ethernet>
- [8] False positives and false negatives. [Online; navštíveno 30.04.2018].
URL https://en.wikipedia.org/wiki/False_positives_and_false_negatives
- [9] Garcia, L. M.: Programming with Libpcap - Sniffing the Network From Our Own Application. *ročník 3, číslo 2*, 2008: s. 38–46, ISSN 1733-7186.
- [10] Gomes, J. a. V.; Inácio, P. R. M.; Pereira, M.; aj.: Detection and Classification of Peer-to-peer Traffic: A Survey. *ACM Comput. Surv.*, ročník 45, číslo 3, červenec 2013: s. 30:1–30:40, ISSN 0360-0300, doi:10.1145/2480741.2480747.
URL <http://doi.acm.org/10.1145/2480741.2480747>
- [11] Grnář, M.: *Detekce seedboxu v síti BitTorrent*. Bakalářská práce, Vysoké učení technické v Brně, Božetěchova 2, Brno, 2018.
- [12] Henry, A.: Most Popular BitTorrent Client: uTorrent. Květen 2015, [Online; navštíveno 30.04.2018].
URL <https://lifelifehacker.com/5813348/five-best-bit-torrent-applications/1705622513>

- [13] Loewenstern, A.; Norberg, A.: DHT Protocol. [Online; navštíveno 30.04.2018].
URL http://www.bittorrent.org/beps/bep_0005.html
- [14] Norberg, A.: uTorrent Transport Protocol. [Online; navštíveno 30.04.2018].
URL http://www.bittorrent.org/beps/bep_0029.html
- [15] Norberg, A.: DHT Bootstrap Update. Prosinec 2013, [Online; navštíveno 30.04.2018].
URL <https://engineering.bittorrent.com/2013/12/19/dht-bootstrap-update/>
- [16] Network packet. [Online; navštíveno 30.04.2018].
URL https://en.wikipedia.org/wiki/Network_packet
- [17] Peer-to-Peer. [Online; navštíveno 30.04.2018].
URL <https://en.wikipedia.org/wiki/Peer-to-peer>
- [18] Sanders, C.: *Analyza sítí a řešení problému v programu Wireshark*. Computer Press, 2012, ISBN ISBN 978-80-251-3718-5.
- [19] Setting up your client to work with your network. [Online; navštíveno 30.04.2018].
URL <http://help.utorrent.com/customer/portal/articles/1753715>
- [20] van der Spek, O.: UDP Tracker Protocol for BitTorrent. [Online; navštíveno 30.04.2018].
URL http://www.bittorrent.org/beps/bep_0015.html
- [21] Xia, T.; Song, R.: A Method of P2P Traffic Identification on Internet Based on the Deep Flow Inspection. In *2009 International Conference on Communication Software and Networks*, Feb 2009, s. 777–780, doi:10.1109/ICCSN.2009.123.
- [22] Zelinskie, J.: Tracker Protocol Extension: Scrape. [Online; navštíveno 30.04.2018].
URL http://www.bittorrent.org/beps/bep_0048.html

Príloha A

Obsah CD

Priložené CD obsahuje nasledujúce súbory:

- Bakalársku prácu vo formáte PDF
- Zdrojové súbory tejto bakalárskej práce
- Detekčný nástroj v jazyku Python
- PDF dokumentáciu detekčného nástroja