FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Detekce skrytých kanálů používající DNS over TLS
Student:	Bc. Lukáš Melcher
Vedoucí:	Ing. Karel Hynek
Studijní program:	Informatika
Obor / specializace:	Počítačová bezpečnost
Katedra:	Katedra informační bezpečnosti
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Seznamte se s problematikou monitorování síťového provozu pomocí síťových toků. Analyzujte možnosti vytvoření skrytého kanálu skrz protokol DNS over TLS. Zaměřte se na jeho kvalitativní charakteristiky jakými jsou propustnost, stabilita spojení a ztrátovost paketů. Dále prozkoumejte projevy skrytých komunikačních kanálů na síťové úrovni (na úrovni síťových paketů a IP toků) a možnosti jak podobné skryté komunikační kanály detekovat. Navrhněte algoritmus rozpoznávající skryté kanály používající DNS over TLS, je možné se inspirovat článkem [1]. Vytvořte softwarový prototyp detektoru, který bude schopen zpracovávat reálný síťový provoz. Softwarový prototyp otestujte na vytvořených datech i datech poskytnutých vedoucím závěrečné práce.

[1] T. Cejka, et al. "Stream-wise detection of surreptitious traffic over DNS," In proceedings of CAMAD2014.

Elektronicky schválil/a prof. Ing. Róbert Lórencz, CSc. dne 29. listopadu 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Detekce skrytých kanálů používající DNS over TLS

Bc. Lukáš Melcher

Katedra informační bezpečnosti
Vedoucí práce: Ing. Karel Hynek

27. dubna 2022

Poděkování

Rád bych poděkoval vedoucímu své diplomové práce Ing. Karlu Hynkovi za vedení a cenné rady. Rovněž děkuji vedoucímu výzkumného pracoviště, kde práce vznikala Ing. Tomášovi Čejkovi, Ph.D. a dalším kolegům z laboratoře monitorování síťového provozu.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 27. dubna 2022 . . . . . .

Odkaz na tuto práci

Melcher, Lukáš. *Detekce skrytých kanálů používající DNS over TLS*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Ochrana soukromí uživatelů v online světě je často diskutovaným tématem. Nešifrovaný a při odposlechu čitelný je však protokol DNS. Jako řešení bylo navrženo několik šifrovaných alternativ. Přesto i s používáním těchto protokolů souvisí bezpečnostní rizika. Tato práce analyzuje zašifrované DNS a primárně se zaměřuje na riziko tunelování pomocí DNS over TLS. Dále poskytuje přehled kvalitativních charakteristik poskytovatelů DoT a diskutuje vhodnost jejich využití v konfiguracích pracovních stanic.

Hlavním výstupem je návrh a implementace prototypu detektoru tunelovaného provozu v DoT. Jeho úspěšnost je v závěru podrobena kritice a jsou diskutovány možnosti vylepšení.

Klíčová slova DoT, DNS přes TLS, soukromé DNS, ochrana soukromí, strojové učení, skryté kanály, exfiltrace dat, botnety

Abstract

Privacy protection of users in the online world is a frequently discussed topic. However, the DNS protocol is unencrypted and easy readable while sniffing. Several encrypted alternatives have been proposed as a solution. Nevertheless, there are security risks associated with the use of these protocols. This work analyzes encrypted DNS and primarily focuses on the risk of tunneling using DNS over TLS. It also provides an overview of the qualitative characteristics of DoT providers and discusses the suitability of their use in workstation configurations.

The main output is the design and implementation of a prototype of DoT tunneled traffic detector. In the end, its success is criticized and possibilities for improvement are discussed.

Keywords DoT, DNS over TLS, DNS privacy, privacy protection, machine learning, covert channels, data exfiltration, botnets

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Ochrana soukromí uživatelů na internetu a v digitálním prostředí obecně se stává předmětem mnoha diskuzí a více či méně objektivních článků, ať už v akademické či laické sféře. Ve společnosti narůstá povědomí o potřebě chránit své soukromí i v online světě, rovněž se stává obecně známým riziko zneužívání informací — i v souvislosti s pokusy o vydírání.

Jedním z protokolů, které dlouho zůstávaly nešifrované a při odposlechu čitelné, pozůstávalo DNS. Právě DNS ale bylo terčem odposlechů, a snadno zneužíváno. Jako příklad lze zmínit projekt americké NSA, která využívala ke sledování lidí program MORECOWBELL [1]. Dalším rizikem pro soukromí může být sledování online aktivit poskytovatelem internetu (nebo ISP z *Internet Service Provider*), které je v některých státech naprosto legální [2]. Následně mohou ISP prodávat odposlechnutá data reklamním společnostem.

Na tyto a další případy zneužívání uživatelských dat reaguje komunita vývojem nových, bezpečnějších protokolů. Tato práce zmiňuje tři přístupy, DNS over HTTPS [3], DNS over TLS [4] a DNS over QUIC [5]. Se zaváděním těchto šifrovaných alternativ, které podrobněji rozebírá oddíl 1.1 jsou však spojena některá rizika a komplikace. Odposlech DNS provozu používají komponenty antivirových systémů pro ochranu před škodlivým softwarem. Pokud je nějaké doménové jméno identifikováno jako IoC (*Indicator of Compromise* — indikátor kompromitace), AV zamezí jeho přeložení a tím významně omezuje možnosti útoku.

Podobně systémy rodičovské ochrany (*parental control systems*) sledují DNS a případně chrání děti blokováním části DNS provozu. Implementované jsou například i v novějších routerech [6, 7]). V nich je často možné nastavovat jednoduchá pravidla pro filtraci DNS dotazů. Tedy, pokud dotaz bude obsahovat *zakázané slovo*, nebude vyhodnocen a stránka se bude jevit jako nevyhledatelná. Je samozřejmé, že stačí využít například VPN, nebo šifrované DNS pro vyhnutí se takové restrikci, ale pro navrhnuté účely často podobný filtr postačuje.

Všimněme si, že v případě využití nějakého z výše uvedených protokolů umožňujících zašifrování DNS dotazů dochází, alespoň do určité míry, k centralizaci jinak značně decentralizovaného systému DNS. Poskytovatel soukromého DNS má pak přístup ke všem vyhledáváním daného uživatele. Je potom věcí důvěry a nastavení politik ochrany soukromí spotřebitele, jak jsou takto získaná data využívána (tedy problém potenciálního prodeje osobních dat se přesouvá od běžných ISP k poskytovatelům šifrovaného DNS).

Co se týče rizik, tak se již objevily nástroje, které umí tunelovat (viz oddíl 1.2) data pomocí DoH [8]. Tohoto rizika si všímá i akademická veřejnost [9]. Autor zde uvádí, že data byla exfiltrována (podrobněji zde 1.2.1.1) i když byly využívány „aktuální firewally, počítače s aktuálními antivirovými produkty a sofistikovaná webová proxy“[1]

Tunely realizované pomocí šifrovaného DNS tedy představují reálný bezpečnostní problém. Tato práce si klade za cíl vyvinout prototyp detektoru tunelů v DNS over TLS, což je jeden z typů šifrovaného DNS (viz oddíl 1.1.2). Aby bylo možné takového cíle dosáhnout, byla provedena analýza šifrovaných DNS i tunelování pomocí nešifrovaného DNS. Práce také seznamuje s možnostmi monitoringu síťového provozu. Sledování provozu je stavěno před velké výzvy, zvlášť za narůstajícího množství šifrovaného provozu na úkor nešifrovaného [10]. Byly rovněž analyzovány různé metody strojového učení i nástroje pro tunelování provozu pomocí DNS.

Jako součást praktické části byla vytvořena měřící soustava a na ní byly dále testováni jednotliví poskytovatelé DoT i různé tunelovací nástroje. Pro poskytovatele DoT byly naměřeny kvalitativní charakteristiky a diskutovány výsledky s uvážením situace ze stránky systémového administrátora. Rovněž byly pomocí ní vytvořeny datové sady, jež jsou jedním z výstupů práce, a které byly využity k datové analýze. Analýza byla provedena za účelem nalezení specifických znaků provozu obsahujícího tunely. Tyto znaky byly použity dále k vytvoření prototypu klasifikátoru založeného na pravidlech.

V závěrečné části je shrnut vývoj prototypů klasifikátorů DoT tunelovaného provozu. Byl navržen deterministický klasifikátor založený na znacích nalezených při datové analýze. Jako další byl vytvořen prototyp klasifikátoru založený na strojovém učení. Oba prototypy byly otestovány, byla diskutována jejich přesnost i možnosti vylepšení.

[1] Překlad autora. V originále: *These DNS exfiltration tests were performed using up-to-date firewalls, workstations with up-to-date antivirus products, and sophisticated web proxies.*

Kapitola 1

Analýza

V této části práce seznamuje s teoretickým základem pro pochopení náplně i širších souvislostí a důsledků, jež z ní plynou. Nejprve bude rozebrán samotný princip protokolu DNS a jeho moderních, šifrovaných verzí. Dále budou shrnuta rizika, která s využíváním DNS přicházejí. Bude využit pohled uživatele, útočníka i administrátora komplexnějšího systému.

1.1 Domain Name System

Protokol DNS [11] je jedním ze základních protokolů, které umožňují internetovým technologiím fungovat tak, jak jsou uživatelé zvyklí. Takřka každý uživatel počítače či chytrého mobilního zařízení ho využívá každý den.

Cílem DNS je překlad IP adres serverů, které jsou špatně zapamatovatelné pro běžné uživatele na doménová jména. Proces z pohledu klienta spočívá v odeslání dotazu na nastavenou adresu serveru a očekávání odpovědi, kterou po doručení zpracuje. Server, na který je požadavek doručen zkontroluje vlastní paměť, a pokud má dotazovanou doménu uloženu, odešle odpověď. Pokud informaci o doménovém jménu uloženu nemá, odešle dotazy na další DNS servery a jakmile od některého získá odpověď, přepošle ji klientovi. Pokud žádnou odpověď nezíská, odešle klientovi chybovou hlášku o nenalezení dotazované domény.

1.1.1 DNS over HTTPS

DoH je zkratkou pro anglický výraz *DNS over HTTPS*, tedy přenos DNS dotazů pomocí protokolu HTTPS. Specifikován je tento protokol v RFC 8484 [3]. Výhodou je jak velká rozšířenost protokolu HTTPS, tak jeho osvědčenost v praxi. Navíc služby DoH jsou v podstatě REST API, takže i práce s ním se snadno realizuje.

Důsledkem využití tohoto protokolu je zašifrování komunikace a znesnadnění přístupu k obsahu při odposlechu komunikace útočníkem. Další vlast-

```
GET /dns-query?dns=AAABAAABAAAAAAAAB2V4YW1wbGUDY29tAAABAAE HTTP/1.1
Host: dns.example.com
Accept: application/dns-message

POST /dns-query HTTP/1.1
Host: dns.example.com
Accept: application/dns-message
Content-Type: application/dns-message
Content-Length: 29
00 00 01 00 00 01 00 00 00 00 00 00 07 65 78 61
6d 70 6c 65 03 63 6f 6d 00 00 01 00 01
```

Obrázek 1.1: Ukázka DoH wireformat dotazu a odpovědi [15]

ností tohoto protokolu je „splynutí“ DoH s běžným HTTPS provozem. Odhalení přítomnosti DoH je navíc netriviální úkol. Pokud je jako *resolver* DoH využíván nějaký obecně známý, lze pro rozpoznání využít jeho IP adresu. V opačném případě DoH skutečně splývá s provozem a pro jeho rozpoznání se využívá mimo jiné strojové učení [12].

Úplná anonymita a skrytí dat je však stále předmětem výzkumu. Zajímavý je kupříkladu přístup využití statistické analýzy. Zvláště pak využití DoH bez správného zarovnání (paddingu) může způsobit poměrně úspěšnou analýzu odposlechu provozu [13].

Podle dokumentace Google Public DNS lze použít dotazování DoH dvěma způsoby [14]:

1. DNS Wireformat[2] zapouzdřený v HTTP dotazu, kdy se používá HTTP metod GET a POST

2. JSON — Komunikace probíhá ve formátu JSON pomocí HTTP

DoH je nyní poměrně široce podporováno v operačních systémech, webových prohlížečích i v Open-Source DNS resolverech [17] [18].

1.1.2 DNS over TLS

DoT je zkratkou pro anglický výraz *DNS over TLS*, tedy přenos DNS dotazů zašifrovaných pomocí protokolu TLS. Podle příslušného RFC [4] má vyhrazený port TCP 853.

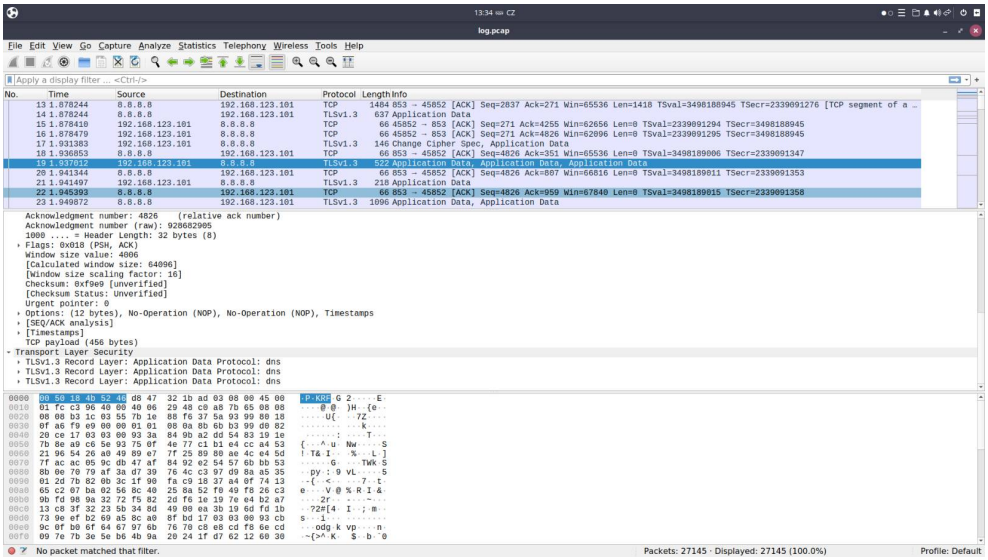
Technicky jde tedy pouze o zapouzdření DNS dotazů do TLS protokolu. Velkou výhodou oproti DoH je tedy daleko nižší režie — zatímco u DoT je třeba jen zapouzdření, u DoH je zapouzdřováno i HTTP.

DoT je podporováno v mobilních zařízeních se systémem Android od verze 9 (Pie) [19]. Další podporu má u lokálních DNS forwarderů (Stubby [20]) — a je tedy možné ho nastavit i u některých routerů (toho bylo využito i v praktické části této práce). „Velké“ webové prohlížeče v současné době DoT nepodporují.

[2]Wireformat je standardní formát dotazu DNS [11].

```
{
  "Status": 0,  // NOERROR - Standard DNS response code (32 bit integer).
  "TC": false,  // Whether the response is truncated
  "RD": true,   // Always true for Google Public DNS
  "RA": true,   // Always true for Google Public DNS
  "AD": false,  // Whether all response data was validated with DNSSEC
  "CD": false,  // Whether the client asked to disable DNSSEC
  "Question":
  [
    {
      "name": "apple.com.",  // FQDN with trailing dot
      "type": 1              // A - Standard DNS RR type
    }
  ],
  "Answer":
  [
    {
      "name": "apple.com.",    // Always matches name in the Question section
      "type": 1,               // A - Standard DNS RR type
      "TTL": 3599,             // Record's time-to-live in seconds
      "data": "17.178.96.59"   // Data for A - IP address as text
    },
    {
      "name": "apple.com.",
      "type": 1,
      "TTL": 3599,
      "data": "17.172.224.47"
    },
    {
      "name": "apple.com.",
      "type": 1,
      "TTL": 3599,
      "data": "17.142.160.59"
    }
  ],
  "edns_client_subnet": "12.34.56.78/0"  // IP address / scope prefix-length
}
```

Výpis kódu 1.1: Příklad odpovědi na DNS dotaz pomocí JSON DoH formátu [16]



Obrázek 1.2: Ukázka zachyceného DoT provozu v programu Wireshark

1.1.3 DNS over QUIC

DoQ je zkratkou pro anglický výraz *DNS over QUIC* a v současné době je jeho specifikace ve stavu RFC Draft [5]. Je zde znovu využit stejný systém jako u DoT. DNS dotaz ve wireformátu se odesílá uvnitř Quic spojení. V současné době má vyhrazený port 853 UDP. Samotná specifikace je ale zatím pouze ve stavu návrhu.

Podle García et al. [21] není použití DoQ v současné době příliš běžné. Podobně nástrojů (klientů) není k dispozici mnoho [17].

1.2 Tunelování pomocí DNS

Podle Čejky et al. [22] bývá vzhledem ke kritické důležitosti DNS povolen na mnoha systémech, i pokud mnoho jiných protokolů podléhá různým restrikcím. Tento fakt naneštěstí nahrává útočníkům, kteří zneužívají vlastností DNS.

Wang et al. [23] definuje DNS tunelování jako „typ tunelovací techniky založené na protokolu DNS, který využívá proces dotazování DNS a zapouzdřuje data v DNS dotazech a odpovědích k vytvoření vlastního tunelu pro přenos komunikace mezi odesílatelem a příjemcem“. [3]

Nástroje pro DNS tunelování jsou zpravidla implementované pomocí klient — server architektury. Jejich soupis podle [23] je v tabulce 1.1.

[3] Překlad autora. V originále [23]: „ The DNS tunnel is a type of tunnel technique, based on the DNS protocol, which utilises the DNS query process and encapsulates the data in the DNS query/response package to build a proprietary tunnel for transmission communication

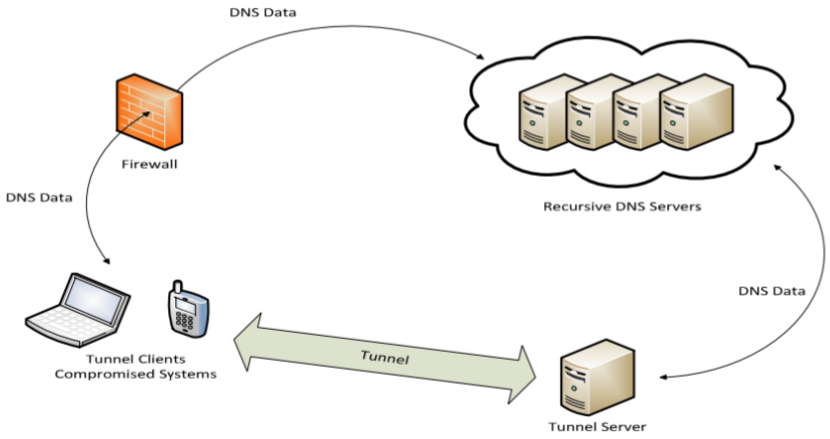
Tabulka 1.1: Přehled DNS tunelovacích nástrojů podle [23]

Vrstva	**Nástroj**	**Kódování**	**Typy dotazů**
IP	NSTX	Base64	TXT
	Dnscat2	Hexadecimal	A, AAAA, CNAME, MX, TXT
	Iodine	Base32/64/128	A, CNAME, NULL, MX, TXT, SRV
	TUNs	Base32/64	CNAME
TCP	Dns2tcp	Base64	TXT, KEY
	OzymanDns	Base32/64	A, TXT
	Heyoka	Binary	TXT

```
RX: client 172.217.45.133, type 1, name ytbg5h.ns1.tunel.gq
TX: client 172.217.45.133, type 1, name ytbg5h.ns1.tunel.gq, 48 bytes data
RX: client 74.125.187.132, type 1, name ytbg5o.ns1.tunel.gq
TX: client 74.125.187.132, type 1, name ytbg5o.ns1.tunel.gq, 48 bytes data
```

Obrázek 1.3: Ukázka DNS dotazu při tunelování pomocí nástroje iodine

Prostor pro skrytá data není příliš velký. Podle dokumentace [11] je maximální délka doménového jména 253 bajtů včetně oddělovače. Délka každé značky („label“) je 63 bajtů. Odděleny jsou tečkou. Doménové jméno může být tvořeno až 37 různými znaky [24]. Nástroje pro kódování obvykle využívají Base32 či Base64 kódování [23] a některé data šifrují, jiné je posílají jen zakódované.



Obrázek 1.4: Schéma tunelování DNS [25]

Útočník může postupovat následovně (viz obrázek 1.4):

1. Zaregistruje si doménu. Není potřeba žádná „exkluzivní“ doména — stačí náhodná s neobvyklou příponou, kterou lze získat zdarma (což na druhou stranu může přispět k jeho odhalení).

between the sender and receiver. “

2. Zřídí si server s veřejnou IP adresou — to lze velmi snadno a rychle pomocí moderních cloudových řešení.
3. Na tomto serveru nainstaluje serverovou část softwaru pro tunelování.
4. Pokud následně na zařízení oběti zakóduje data, která potřebuje odeslat ze zařízení do DNS dotazů, může je začít odesílat na běžné DNS servery a odtamtud se rekurzivními dotazy dostanou na útočníkův server, kde jsou dekódovány a útočník může odpovídat DNS odpověďmi. Existuje i metoda přímého připojení, která ale může být snadněji detekovaná.

Výhodou DNS tunelování je to, že jak již bylo zmíněno v oddílu 1.1, provoz protokolu DNS bývá jen málokdy filtrován a je tedy větší pravděpodobnost, že projde skrz firewall oběti[22].

1.2.1 Modely hrozeb

Práce se zaměřuje na zneužití DNS k tunelování při modelech hrozeb exfiltrace dat (1.2.1.1), C&C (1.2.1.2) a skrytého browsingu (1.2.1.3). Tyto modely jsou dále postupně rozebrány.

1.2.1.1 Exfiltrace dat

Pro pochopení problematiky je nastíněn následující scénář: Osobní počítač oběti je infikován špionážním virem, který sleduje jeho aktivity, sbírá informace a především se pomocí keyloggeru dostane k přihlašovacím údajům k internetovému bankovnictví.

Tento „úspěch“ počítačového viru by byl útočníkovi ale úplně k ničemu, pokud by se mu nepodařilo sesbíraná data odeslat z počítače oběti a následně zneužít. Právě odeslání dat ze zařízení oběti bude označováno jako exfiltrace dat.

Je třeba rovněž zvážit objem odesílaných dat. V tomto příkladu bude asi stačit velmi krátký řetězec. V případě, že bude chtít virus exfiltrovat důvěrné informace o klientech nějaké firmy po napadení firemního serveru, již takových dat mohou být klidně řádově GB až desítky GB v případě malého podniku, v případě velké korporace bude odhad pravděpodobně větší.

Při exfiltraci takového objemu dat bude třeba daleko robustnějšího tunelu. Podstatné budou především jeho kvalitativní charakteristiky jako je propustnost, ztrátovost nebo stabilita.

1.2.1.2 C&C

Podle Martina Korce [26] využívá malware *Command and Control* (také C&C nebo C2) jako běžný způsob ke správě botnetů a dává mu možnost ovládat nakažené zařízení. Jako definici botu (jednotce botnetu) ve své práci [26] uvádí, „že jde o program, který dokáže zneužít zranitelnost PC, IoT nebo síťového

prvku k infiltraci bez vědomí uživatele, šířit se na jiná zařízení, je schopen jednat samostatně, přijímat příkazy vzdáleně pomocí definovaného protokolu a provádět přijaté příkazy, exfiltrovat požadovaná data. [4] K odesílání příkazů do zařízení potřebuje útočník udržovat spojení z napadeného zařízení na server.“ [5].

Dalším scénářem, který je v této diplomové práci uvažován jako možné zneužití DoT tunelů je tedy přenos příkazů pro sítě C&C. Detekce takových příkazů je součástí antivirové ochrany a proto je v zájmu útočníka skrývat tuto komunikaci, jak je to jen možné. Za povšimnutí stojí, že na rozdíl od mnoha scénářů exfiltrace informací, zde není potřeba odesílat velké objemy dat — a tedy útočníkům stačí skrytý kanál s daleko skromnějšími parametry.

Dietrich et al. [27] uvádí příklad botnetu *Feederbot*, který využívá DNS pro přenos příkazů. Autoři článku popisují způsoby detekce škodlivé komunikace včetně dělení vzorků provozu podle příslušnosti k jednotlivým „rodinám“. Autor této diplomové práce považuje za odůvodněnou obavu, že využití skrytého kanálu v šifrovaném DNS by mohlo ztížit podmínky detekce, jež prováděli Dietrich et al. [27].

1.2.1.3 Skrytý browsing

Může se stát, že chce uživatel skrýt svoje prohledávání internetu před jeho poskytovatelem, či se vyhnout jeho zpoplatnění — například v hotelech.

Znovu tedy půjde o přenášení velkého objemu dat (oproti normálnímu DNS provozu). Na rozdíl od exfiltrace dat lze ale předpokládat, že poměr mezi přijatými a odeslanými daty bude daleko vyrovnanější.

Návody pro snadnou realizaci DNS tunelování za účelem skrytého prohlížení webu jsou na internetu snadno dostupné. Lze nalézt konkrétní postupy i s vysvětlením principu [28], takže i laický uživatel může pomocí připravených nástrojů skrývat svou online aktivitu.

1.2.2 Obrana

Nejdříve bude uvedena úvaha nad obranou systému před DNS tunelováním. Pak budou diskutovány možnosti obrany proti DoT tunelování.

[4] Překlad autora. V originále: „Definition of bot is broad, many authors are using different definitions, but it can be defined as follows: Bot is piece of program, which is capable of using at least one of the following properties such as using vulnerabilities on PC, IoT or network devices to infiltrate without user's conscious, reproduce itself to other devices, it is able to act independently, capable to receive commands remotely using define protocol and execute these commands, ex-filtrate desired data. “

[5] Volný překlad autora. V originále [26]: „The usage of C2 architecture is a common way to have an infected bot under control and have the ability to control the machine. To send commands to the device, the attacker needs to maintain a connection from machine to the server.“

Dle autorů článku [23], je možné odlišit možnosti detekce na metody založené na sledování: 1. jednotlivých paketů, 2. celých datových toků (princip flow monitoringu rozebírá oddíl 1.3).

Jako doporučené znaky(*features*) z pohledu obsahu dotazu uvádí shrnující článek [23] tyto:

- **Velikost paketů** — Podle autora článku [29] je obvykle při exfiltraci soukromých dat potřeba odeslat velký objem dat, což se může promítnout na velikosti odesílaných DNS dotazů. Pokud dochází ke skrytému prohlížení webu, bude nejspíše velký objem dat odesílán i přijímán.
- **Poměr velikosti dotazu a odpovědi** — Jak zmiňuje i Wang et al. [23], pokud dochází k exfiltraci útočník bude pravděpodobně potřebovat odeslat data *ze* zařízení, ne *do* něj a odpovědi tedy budou pravděpodobně obsahovat například jen příkazy nebo potvrzení o přijetí. Proto se může objevit nepřirozený poměr mezi dotazy a odpověďmi. Naopak u C&C nebo skrytého browsingu se nemusí poměr jevit tolik podezřele.
- **Délka doménového či subdoménového jména** — Na základě datového kódování se původní doménové jméno nemění, dynamicky se mění subdoména, v které jsou zakódována data[23].
- **Počet subdomén a značek** [30] — Značky (*labels*) mohou mít až 63 bajtů. Pokud je třeba přenést pomocí DNS množství dat, existuje předpoklad, že bude značek více a že mohou obsahovat počet bajtů blízký povolenému maximu.
- **Počet či poměr speciálních znaků a čísel v doménovém jménu** [31] — Z principu jsou doménová jména vytvářena jako snadno zapamatovatelná. Pokud jde o doménové jméno naplněné zakódovanými daty (třeba v Base64), bude pravděpodobně obsahovat nestandardní počet/poměr takových znaků.
- **Poměr nejdelšího smysluplného řetězce v doménovém jménu** — Viz komentář výše. Zmiňuje mimo jiné Liu et al. [32].
- **Neobvyklé typy dotazů** — Pokud aktuální provoz vykazuje neobvyklou distribuci DNS typů (ať už oproti pozorovaného v internetu či naměřenému v konkrétní síti) může jít o ukazatel na existenci DNS tunelů. Uvádí Liu et al. [33], Wang et al. [23] ilustruje tabulkou s rozložením jednotlivých typů dotazů v provozu.
- **Specifické signatury** — Podrobné zkoumání tunelovacích nástrojů týmy expertů vyústilo v popsání specifických znaků v DNS dotazech. Takový znak například uvádí Van Horenbeeck [34]. Získání takových signatur je ale proces náročný na prostředky i na čas. Navíc útočníkovi může stačit využít jiný nástroj či aktualizovat stávající.

- **Cílový DNS server** — Za předpokladu využití přímého spojení mezi kompromitovaným klientem a útočníkovým DNS serverem je možné zaznamenat provoz směřující na neznámý DNS server, což může podle Wang et al. [23] zapříčinit překročení hranice stanovené v detekčních pravidlech. Jde ale o znak, který může útočník za některých okolností poměrně snadno překonat — stačí využít nepřímé spojení pomocí rekurzivního dotazování. Rovněž může útočníkovi pomoci podvržení adresy zdrojového zařízení.

Pokud budeme uvažovat znaky (*features*) založené na provozu (tedy ne na analýze samostatného paketu). Podle Wang et al. [23] zvláště systémy detekce, které nepracují v reálném čase staví na charakteristických znacích (*features*), které jsou založené na časovém kontextu přijímání a odesílání paketů s DNS provozem. V případě sledování jednotlivých paketů přichází systém o mnoho informací, jež mu kontext probíhajících datových toků dokáže nabídnout:

- **Počet DNS dotazů z IP adresy** — Jak je uvedeno u principu DNS tunelování 1.2, tak prostoru pro data není mnoho. Proto je pro přenesení většího objemu dat potřeba mnoho DNS dotazů — jejich neobvyklý počet může být podezřelý [35]. Útočník se nicméně může detekci bránit podvrhováním zdrojových IP adres.

- **Počet DNS dotazů na konkrétní doménu** — Podle Almusawi et al. [35] může být podezřelý také velký počet dotazů na jednu doménu. Útočník ale může provoz rozdělit na několik různých serverů.

- **Počet subdomén ke každé doméně** — Obvykle je k dispozici jen relativně nízký počet subdomén ke každé doméně. V případě tunelu může být podle Wang et al. [23] o mnoho řádů větší.

- **Časový odstup** — Situace, kdy není dotazovaný DNS záznam obsažen v cache hlavních DNS serverů a ty se musí pokaždé dotazovat až k útočníkově serveru je netypický a může být podle Liu et al. [33] důležitým znakem pro detekci.

- **Historie domény** [36, 35] — Domény již dříve zneužité pro tunelování můžeme detekovat na základě *blocklistu*. Avšak vytvoření nové domény je snadná záležitost a nelze se proto na *blocklisty* spoléhat.

- **Geografické umístění DNS serveru** [37] — Za předpokladu malé firmy či soukromé sítě by bylo zvláštní, kdyby se nějaké zařízení vytrvale dotazovalo DNS serveru například na jiném světadíle. Využitý předpoklad je však poměrně silný.

- **Osamocené DNS dotazy** [37] — Když se dotazujeme na IP adresu serveru, existuje logický předpoklad, že následně získanou IP adresu

využijeme a budeme se serverem interagovat. V případě DNS tunelování o žádnou interakci na základě odpovědi nestojíme a po DNS odpovědi pravděpodobně nebude následovat jinak očekávaný request na dotazovaný server.

- **Počet NXDOMAIN** [38] — I při běžném provozu můžeme očekávat NXDOMAIN odpovědi (to znamená, že překlad na IP adresu nebyl úspěšný). Přesto při tunelování lze očekávat jejich významně vyšší počet.

V obou případech můžeme použít metody založené:

1. **Na pravidlech**
2. **Na strojovém učení (ML)**

Všimněme si, že u mnoha znaků se objevují formulace jako *významně vyšší*, *velký objem*, *nepřirozený poměr*, *relativně nízký, neobvyklý, velký počet*,.... Tyto formulace poukazují na potřebu využití nějaké hraniční hodnoty, která určí míru „podezřelosti“ provozu.

V případě použití metody založených na pravidlech budeme takovou hodnotu pravděpodobně určovat na empirickém základě, či podle doporučení v odborné literatuře. Pokud je využito strojové učení, bude třeba dodat klasifikátoru data, na jejichž základě bude natrénován. Podrobněji je rozebráno v oddílu 1.4.

Byly zmíněny možnosti detekce DNS tunelování. Oproti běžnému DNS vykazuje DoT tunelování několik významných odlišností. Několik důležitých bodů:

- Chybí přístup k obsahu dotazu — ten je zašifrovaný.
- Rozpoznání DoT — DoT splývá s jinými typy provozu. Je to z toho důvodu, že protokol TLS je široce rozšířený. Pokud pomineme znaky jako jsou IP adresy známých poskytovatelů DoT a port 853 (který je daný specifikací [4]), pak je rozpoznání DoT netriviálním problémem. A útočník pravděpodobně nebude mít velké problémy vytvořit vlastní DoT resolver běžící na náhodném portu na zařízení s „neznámou“ IP adresou (neznámou z hlediska seznamů DoT resolverů).

Bezesporu funkčním řešením pro administrátora systému by bylo nastavení firewallů a dalších bezpečnostních politik, že zakáže šifrované DNS, na detekci DNS tunelů využije známé nástroje a zakáže dále jakýkoliv šifrovaný provoz — případně do klientských stanic doinstaluje vlastní certifikáty a bude veškerý provoz rozšifrovávat a kontrolovat. Existuje mnoho důvodů pro to, vyhnout se takovým praktikám, soukromím počínaje a výkonem systému konče. V této práci je dále předpokládáno, že administrátor DoT ponechá povolené.

Pak bude potřebovat detekci DoT tunelů mezi jinak legitimním DoT provozem (jako DoT provoz bude označen provoz na cílovém portu 853 přičemž je předpokládáno, že komunikace s útočníkovým DoT serverem běžícím na jiném portu bude zablokována na základě jiných bezpečnostních politik).

1.3 Monitoring datových toků

Existuje více možností monitoringu síťového provozu. Je možné je podle Hofstede et al. [39] dělit na:

Aktivní monitoring může probíhat například pomocí nástrojů jako je `ping` nebo `traceroute`.

Pasivní monitoring spočívá ve sledování provozu, který sítí prochází.

V této práci bylo využíváno jen pasivního monitoringu datových toků. Jednou z jeho možností je zachytávání a ukládání paketů k další analýze. Přestože tento přístup vykazuje některé výhody, nemusí být realizovatelný (či minimálně vhodný) na rozsáhlých, vysokorychlostních sítích. Příčinou je potřeba nákladného hardwaru, bez nějž není takový monitoring možný. Už pouze uložení provozu o objemu řádově ve stovkách gigabitů za sekundu může být pro infrastrukturu velkou výzvou. Zpracování těchto dat bude nejspíše vyžadovat další procesy náročné na prostředky. Rovněž může být problémem dlouhá doba analýzy a tím i komplikovaná pružná reakce, například na škodlivý provoz v síti.

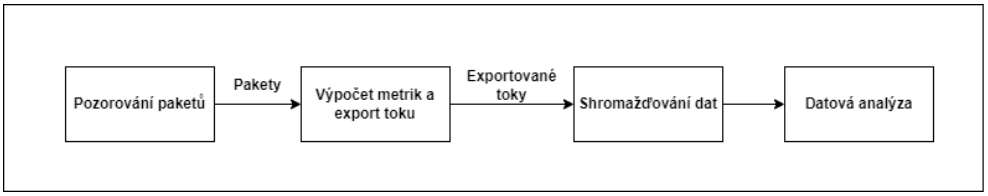
Dalším nevýhodným aspektem ukládání paketů může být ochrana soukromí uživatelů. Kromě toho je otázkou, kolik informací z takového uloženého provozu bude možné získat vzhledem k poměru šifrovaného a nešifrovaného objemu [10].

Logickým krokem je pak agregace paketů do množin podle systematických pravidel a následně extrakce dostupných informací o paketech do vyšších datových struktur. Výsledkem je tedy informace o datovém toku.

V rámci této práce byl síťový provoz monitorován právě jen na základě datových toků. Tato kapitola obsahuje základní přehled potřebných informací pro pochopení výhod a úskalí práce s datovými toky. Dále shrnuje základní vlastnosti exportéru IPFIXProbe, z nichž následně vyplývají některá opatření při zpracovávání zachycených dat a jejich vyhodnocování.

Datový tok z anglického *data flow* má velmi obecnou definici. Zdroj [39] [40] uvádí, že jde o „množinu IP paketů, které prochází pozorovacím bodem v síti v určitém časovém intervalu. Přitom platí, že všechny pakety patřící do toku mají množinu společných vlastností“[6]

[6] Překlad autora. V originále [40]: „a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties“



Obrázek 1.5: Fáze monitoringu síťových toků, ilustrace podle [39]

Klíče datového toku z anglického *flow keys* jsou předem definované vlastnosti datového toku. Jako příklad je možné uvést IP adresy, mezi kterými komunikace probíhá, čísla portů, MAC adresy...

Je možné se setkat s datovými toky

- Jednosměrnými
- Obousměrnými (*biflow*)

V rámci této práce bylo pracováno pouze s obousměrnými datovými toky (jednosměrné byly filtrovány).

1.3.1 Princip

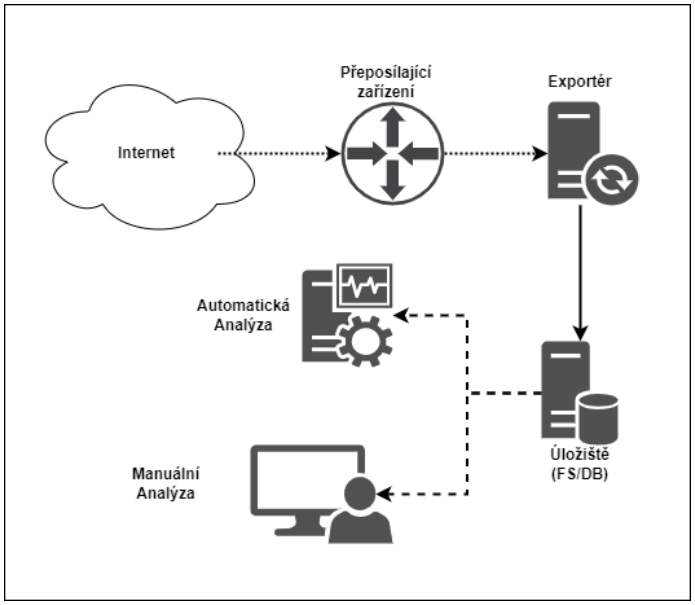
Jak uvádí Hofstede et al. [22] a ukazují obrázky 1.5 a 1.6, vytváření záznamů o datových tocích má několik fází. Jedná se o:

Pozorování paketů — probíhá na specifikovaném rozhraní či rozhraních. Pro monitoring je tedy třeba přístup k síťovým prvkům. Druhou možností zpravidla bývá zpracovat zaznamenaný datový tok (například ve formátu `pcap`). To s sebou však nese řadu nevýhod, jako jsou velikost záznamů (záznamy o datových tocích jsou pochopitelně výrazně menší, než záznamy provozu), ochrana soukromí uživatelů a v závislosti na provedení i zpoždění při analýze (viz úvod oddílu).

Výpočet metrik a export toku — zde dochází k samotnému vytváření záznamů o datovém toku a spojování informací o paketech pomocí *flow keys*.

Shromažďování dat — jak je vidět na obrázku 1.6 využívá se systémů s databázemi či jiným úložištěm, kam jsou data odesílána a dále skladována.

Datová analýza — může být automatizovaná i manuální. Netřeba zdůrazňovat, že manuální přístup je výrazně náročnější na prostředky a v některých případech i nepoužitelný, vzhledem k objemu dat. Zkombinovat možnosti je možné tak, že bude použita manuální analýza pouze výběru dat například tam, kde míra „jistoty“ automatizovaných nástrojů kolísá.



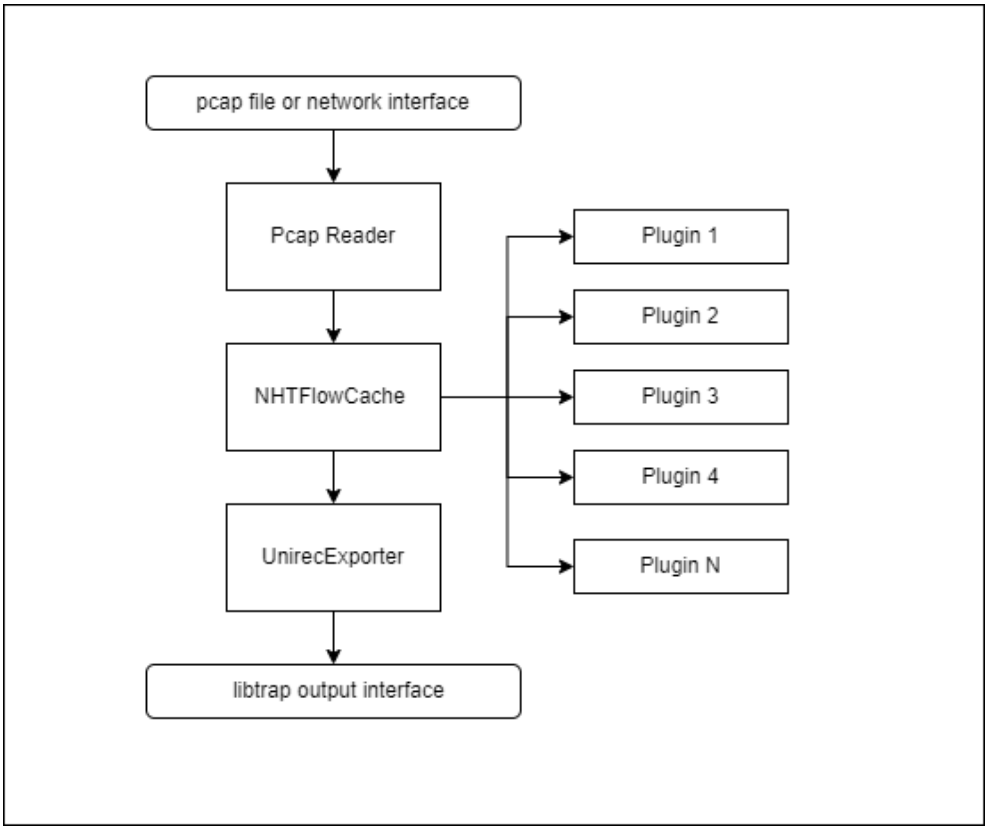
Obrázek 1.6: Možnosti monitoringu toků — architektura, ilustrace podle [39]

1.3.2 IPFIXProbe

IPFIXProbe [41] je open–source systém vyvíjený sdružením CESNET, který byl v této práci využit pro monitoring datových toků.

Přestože systém nabízí mnoho dodatečných rozšíření, při tvorbě této práce byla využívána jen základní část exportovaných atributů toku s rozšířením PSTATS, tedy **základní**:

1. DST_MAC – cílová MAC adresa
2. SRC_MAC – zdrojová MAC adresa
3. DST_IP – cílová IP adresa
4. SRC_IP – zdrojová IP adresa
5. BYTES – počet bajtů ve směru k cíli
6. BYTES_REV – počet bajtů ve směru ke zdroji
7. LINK_BIT_FIELD – identifikace exportujícího zařízení
8. TIME_FIRST – čas prvního zaznamenaného paketu
9. TIME_LAST – čas posledního zaznamenaného paketu
10. PACKETS – počet paketů ve směru k cíli
11. PACKETS_REV – počet paketů ve směru ke zdroji



Obrázek 1.7: Schéma architektury IPFIXProbe – ilustrace podle [41]

12. DST_PORT – cílový port
13. SRC_PORT – zdrojový port
14. PROTOCOL – trasnportní protokol
15. TCP_FLAGS – TCP flagy jednotlivých paketů ve směru k cíli
16. TCP_FLAGS_REV – TCP flagy jednotlivých paketů ve směru ke zdroji

a PSTATS (informace o prvních 30 nenulových paketech v toku):

1. PPI_PKT_LENGTHS – velikosti
2. PPI_PKT_TIMES – časy zaznamenání
3. PPI_PKT_DIRECTIONS – směry
4. PPI_PKT_FLAGS – TCP flagy

1.3.3 Komplikace

V souvislosti s monitoringem datových toků však vyvstávají některé výzvy pro nastavení systému i analýzu získaných dat.

Export datového toku je fáze, kdy jsou exportovány informace o toku do datové struktury jež ho představuje. Ta je následně předána do dalšího kroku procesu, jímž může být další zpracování, uložení či analýza. Určit, kdy k exportu má dojít, však může být problém. Nemusí totiž dojít k zachycení informace o ukončení toku. V takové situaci je logickým řešením timeout. Jeho nastavení je klíčové.

- Pokud je nastavený příliš krátký, pak dochází k dělení datových toků a ztrátě informací.
- Pokud je nastavený příliš dlouhý, ztrácí systém možnost pružně reagovat na škodlivý provoz. Rovněž narůstají požadavky na prostředky infrastruktury.

Zdroj a cíl toku může být velmi náročné určit, zejména pokud nedojde k zachycení počátku komunikace. V této práci bylo při analýze využíváno otáčení datových toků podle zdrojové IP. Tj. když byla cílová adresa z rozsahu lokálních IP adres, byl datový tok „otočen“.

Chybějící části toku je problém, ke kterému může docházet i jinak, než exportováním části neukončeného toku. Například v případě, kdy pakety neprocházejí sledovaným bodem v obou směrech bude výsledný tok pouze jednosměrný. V této práci byly takové toky filtrovány a nebyly použity.

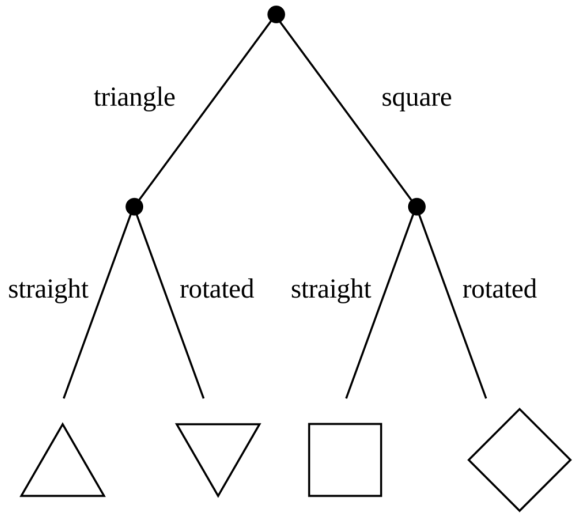
1.4 Strojové učení

Tento oddíl rozebírá teoretický základ týkající se strojového učení. Popisy algoritmů níže jsou pouze základní, protože podrobná analýza algoritmů ML nebyla předmětem této práce. Pro pochopení výsledků a využitých metod je nicméně potřebná alespoň jejich rámcová znalost.

1.4.1 Rozhodovací stromy

Rozhodovací stromy jsou sérií podmínek uspořádaných do stromové struktury. Klasifikace probíhá průchodem stromu od kořene po list. List představuje klasifikovanou třídu. Fáze tréninku — tedy vytváření — rozhodovacího stromu se liší v závislosti na využitém algoritmu. Často se využívá algoritmus C4.5, o kterém mluví mimo jiné Hssina et al. [42] a porovnává s algoritmem ID3. Pro realizaci byla využita verze zahrnutá v knihovně `sklearn`, která tento algoritmus realizuje podle Breiman et al. [43].

Vizualizace jednoduchého rozhodovacího stromu pro klasifikaci tvarů je na obrázku 1.8. Tvary jsou listy stromu, zatímco podmínky jsou reprezentovány uzly.



Obrázek 1.8: Ukázka rozhodovacího stromu [44]

1.4.2 K nejbližších sousedů

Algoritmus založený na K nejbližších sousedech [45] je založen na myšlence, že data jsou představována body v n–rozměrném prostoru, kde n je počet sledovaných vlastností (*features*). Klasifikovaná data jsou následně vyhodnocována na základě k nejbližších sousedů (***k*** ***n****earest* ***n****eighbors*).

Vzdálenost klasifikovaných dat od dat již uložených může být určována na základě různých metrik a rovněž může být použita k vyhodnocení — například kategorie bližších sousedů může být zvýhodněna oproti kategorii sousedů, kteří jsou ve větší vzdálenosti.

V případě velké trénovací datové sady může být vyhodnocování testovacích dat velmi náročné na čas, protože je třeba nalézt k nejbližších sousedů v celé trénovací sadě (pokud není během trénování redukována) a následně (někdy i pomocí vzdálenosti) dopočítat výslednou kategorii.

1.4.3 Random Forest

U rozhodovacích stromů hrozí snadné přetrénování a následné špatná klasifikace nově určovaných záznamů, jak uvádí Pal [46]. Proto se používají přístupy jako Random Forest a Extra Trees (které jsou známé taky jako extrémně randomizované stromy *Extreme Randomized Trees*).

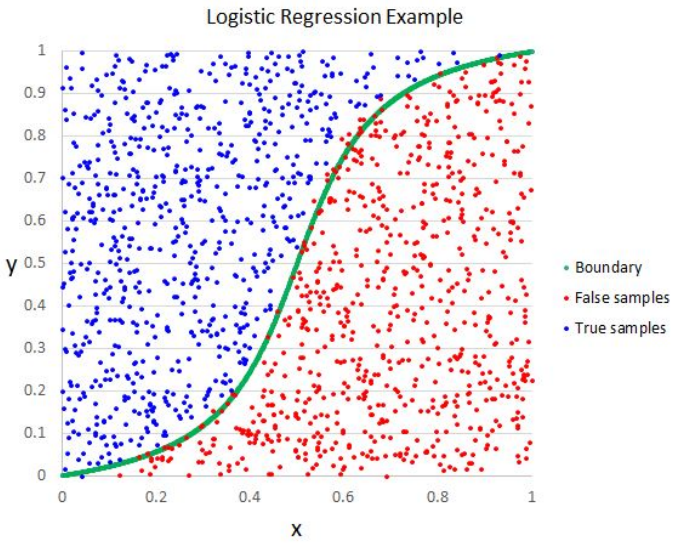
V průběhu trénování klasifikátoru Random Trees [47] dochází k vytvoření nových datových sad (*bootstraping*). Toto vytvoření je v podstatě „zamíchání“ vzorků s možností opakování záznamů. Pro každou sadu se vytváří podmnožina sledovaných charakteristických vlastností (*features*) a na každé této sadě je natrénován jeden rozhodovací strom. Výsledkem je množina rozhodovacích stromů.

Klasifikace pak probíhá tak, že se konkrétní záznam nechá vyhodnotit každým z rozhodovacích stromů a jejich výsledky jsou následně spojeny do jednoho (například tak, že když většina stromů klasifikuje data jako kategorii A, je tato kategorie označena jako výstup celého algoritmu). Náhodnost „lesa“ je pak dána dvěma faktory — 1) „zamícháním“ původní datové sady a vytvořením nových datových sad a 2) výběrem sledovaných charakteristických vlastností (*features*) pro každou datovou sadu.

1.4.4 Extra Trees

Extra Trees algoritmus [48] je velmi podobný algoritmu Random Forest. Vykazuje jen několik rozdílů:

- Každý rozhodovací strom je trénován na celé datové sadě — nepoužívá se zde *boostraping*.
- Je randomizované pořadí rozhodovacích pravidel ve stromě — na rozdíl od jiných verzí konstrukce rozhodovacích stromů, kde se použije pravidlo, které je *optimální* (rozdělí nejvíce záznamů). Problém, který se u tohoto běžně používaného přístupu objevuje, je, že optimalita se vyhodnocuje nad trénovací datovou sadou, což může přispět k přetrénování modelu.



Obrázek 1.9: Znázornění funkce algoritmu logistické regrese[50]

1.4.5 Logistická regrese

Pro snazší pochopení principu algoritmu logistické regrese může pomoci představit si jednotlivě naměřená data ve vícerozměrném prostoru. Dimenze prostoru je určena počtem sledovaných *features*. Cílem je nalézt takovou nadrovinu (či křivku nebo rovinu, pokud pracujeme s malým počtem *features*), která oddělí třídy vzorků. Její rovnice 1.1, kde i určuje číslo trénovacího vzorku, θ, σ odpovídají odhadům parametrů a n udává počet *features*.

Následně se pro testovaná data testuje, kde vůči nadrovině leží a podle toho je určována jejich třída, případně míra jistoty, že k dané třídě patří.

$$\begin{aligned} P(y^{(i)} = 1 | x^{(i)}; \theta) &= \sigma(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \cdots + \theta_n x_n^{(i)}) \\ &= \sigma(\theta^T x^{(i)}) = \frac{1}{1 + e^{-\theta^T x^{(i)}}} \\ \theta^T &= [\theta_0, \ldots, \theta_n] \\ x^{(i)} &= \begin{bmatrix} x_0^{(i)} \\ \ldots \\ x_n^{(i)} \end{bmatrix} \end{aligned} \tag{1.1}$$

1.1: Rovnice pro logistickou regresi s více *features* [49]

Kapitola 2

Analýza charakteristik DoT tunelů

Tato kapitola rozebírá stěžejní téma z pohledu existence rizik v reálném světě. Soustředí se na realizaci tunelování skrze DoT. Cílem tedy bylo ověřit, zda se naplní teoretická rizika zmiňovaná v předchozí kapitole a pokud ano, do jaké míry.

Rovněž pro detekci bylo nezbytné mít k dispozici ukázková data, bez nichž by nebylo možné vyčíst charakteristické znaky pro deterministický klasifikátor ani natrénovat klasifikátor založený na strojovém učení.

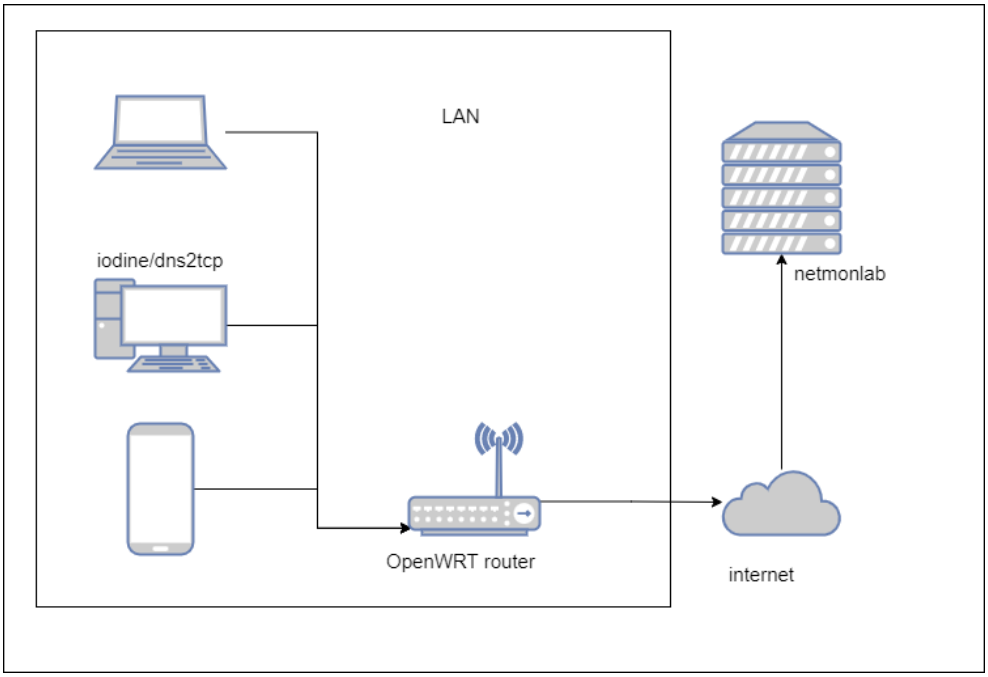
Kapitola informuje o využité metodologii, nástrojích a vysvětluje výběr konkrétních poskytovatelů DoT. V závěru seznámí s výsledky měření a diskutuje vhodnost využití jednotlivých resolverů pro reálné nasazení v sítích.

2.1 Měřící soustava

V době vzniku práce byla podpora DoT ve webových prohlížečích a dalších programech omezená. Proto bylo pro generování využito zaznamenávání datového provozu procházejího skrze měřící bod.

Docíleno toho bylo pomocí routeru se systémem OpenWrt [51] (TP-Link Archer AC1750), kterým byla oddělena část běžně využívané sítě. Na router byl nainstalován nástroj IPFIXProbe, o kterém je oddíl 1.3.2. Do routeru byl navíc doinstalován lokální DNS resolver — Stubby. Ten byl nakonfigurován tak, aby všechny nešifrované DNS dotazy převáděl na DoT. IPFIXProbe pak „odposlouchával“ na výstupním rozhraní routeru.

Systém IPFIXProbe byl během vzniku práce nastaven tak, že v případě, že v rámci toku nepřibyl žádný paket po dobu 65 sekund, pak byl tok vyexportován. Dále platilo, že i pokud byla v toku aktivita, byl vyexportován nejdéle po pěti minutách.



Obrázek 2.1: Schéma měřicí soustavy

Z routeru byla data odesílána na server `netmonlab.fit.cvut.cz`, kde byla uložena a následně analyzována. Tunelování probíhalo výhradně na zařízeních se systémem linux (případně na virtuálním zařízení s OS linux běžícím na hostujícím počítači s OS MS Windows 10).

Pro vytvoření datové sady byly využity tři programy umožňující tunelování `iodine`, `dns2tcp` a `dnscat2` — ty běžely na počítačích v segmentu sítě vytvořeném pomocí OpenWrt routeru. Jejich funkce a možnosti jsou rozebrány dále.

2.1.1 iodine

Iodine je často využívaný nástroj pro tunelování, který nabízí velmi podrobné ladící možnosti, uživatelsky přívětivé možnosti nastavení a snadný způsob použití.

Praktickou vlastností tohoto nástroje je, že po úspěšném spojení na linuxovém zařízení (na OS Windows nebylo testováno) doplní nové rozhraní (snadno možné vyvolat výpis pomocí příkazu `ifconfig`). Toto rozhraní má IP adresu z nastaveného rozsahu privátních IP adres. V této síti je kromě aktuálního zařízení ještě iodine server, pomocí kterého komunikace probíhá. Možnosti propojení více zařízení tímto způsobem nebyly zkoumány.

Díky této vlastnosti je poměrně snadné simulovat C&C i scénář exfiltrace dat. Bylo otestováno i nastavení jako `default gateway` za účelem zkoušení

skrytého prohlížení webu, ale takovou zátěž využívaná konfigurace nezvládla a spadla. Součástí navazujících prací může být taková realizace, která tuto zátěž zvládne. Ovšem při využití Stubby resolveru a rekurzivního spojení docházelo pravděpodobně k přetížení setupu. Tomu nasvědčují i občasné problémy s překladem doménových jmen i v průběhu běžného používání routeru.

Iodine dále podporuje možnost přímého připojení na iodine server. Tím je myšleno, že DNS dotazy, které obsahují zakódovaná data jsou odesílány přímo na zadanou IP adresu iodine serveru, případně si jí iodine jednou získá a dále se připojuje přímo. Druhou možností je připojení skrze rekurzivní dotazování, kdy je cílem dotazu vždy nejdříve spolehlivý DNS server, který se následně dotazuje na speciální iodine server. Druhá verze má pochopitelně horší vlastnosti, ale zato se při sledování síťového provozu neobjeví rozsáhlé datové toky mezi zařízením a nějakou neznámou stanicí na internetu (která by navíc mohla být zanedlouho označena jako škodlivá), ale (možná nepřirozeně rozsáhlý) provoz mezi stanicí a pravděpodobně důvěryhodným DNS serverem.

Lze využít i volitelné nastavení maximální velikosti DNS odpovědi, ve které jsou pochopitelně přenášená data. Díky tomu nemusí být provoz vytvořený systémem tolik nápadný — přestože neobvykle velké množství DNS dotazů i menší velikosti může být (a mělo by být) v síti detekováno a měl by být uvědoměn administrátor.

2.1.2 dns2tcp

Nástroj dns2tcp umožňuje navázat tcp spojení skrze DNS tunel. Na rozdíl od nástroje iodine vyžaduje nastavení takzvaného zdroje (*resource*), kterým může být spojení například pomocí ssh, ssl tunelu, pop3 nebo smtp [52]. V této práci bylo využíváno spojení pomocí ssh.

Pro porozumění výsledkům a komplikacím při měření je třeba zdůraznit důsledky, které tento přístup nese. Podívejme se na to očima útočníka. Vezměme si příklad, kdy chce pomocí vlastního TCP protokolu spravovat botnet síť. Při použití přístupu, který byl pro experimenty využit (zjednodušená situace, v reálné situaci by byl pravděpodobně využit jiný nástroj než dns2tcp — dost možná speciálně vyvinutý pro tento účel, menší a „nenápadnější“) je třeba tunelovat pomocí DNS a uvnitř něj pomocí SSH. Navíc je v případě DoT celý provoz zapouzdřen do TLS. Několikeré zapouzdření snižuje propustnost a alespoň v případě měření realizovaných v průběhu vytváření této práce rovněž snižovalo stabilitu takového skrytého kanálu.

2.1.3 dnscat2

Nástroj dnscat2 [53] je doporučován v mnoha zdrojích, měření pomocí dnscat2 v rámci této práce však nebylo úspěšné.

Tunely vytvořené pomocí tohoto systému se nedařilo navázat a pokud se již spojení vytvořit povedlo, tunely byly velmi nestabilní a nebyly využitelné pro praktické úkony.

Příčinnou mohla být nesprávná manipulace se závislostmi serverové části vytvořené v jazyce `Ruby`, jak ostatně zmiňuje autor [53].

Z těchto technických důvodů byla pozornost soustředěna na zbylé dva nástroje pro DNS tunelování.

2.2 Poskytovatelé DoT

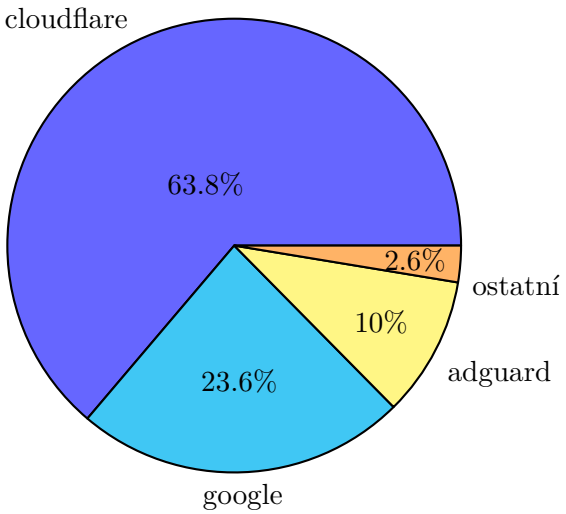
Přestože bylo DoT standardizováno relativně nedávno (v roce 2016 [4]), tak je dnes k dispozici poměrně široká nabídka poskytovatelů DoT.

Jak ukazuje oddíl 2.4, tak výkonové rozdíly mezi poskytovateli jsou při tunelování DoT více než značné.

Výběr DoT resolverů byl učiněn na základě experimentálně zjištěných, skutečně využívaných DoT adresách ze sítě CESNET. Bylo provedeno 24 hodinové měření na cílovém portu 853 a podle toho bylo vyhodnoceno, kteří poskytovatelé se používají. Rozložení je možné vidět na obrázku 2.2. Experimentálně zjištění poskytovatelé DoT byli doplněni některými poskytovateli z `https://dnsprivacy.org` [54]. Jejich URL a IP adresy jsou v tabulce 2.1.

Tabulka 2.1: Využití poskytovatelé a jejich IP adresy

dns.google.com	8.8.8.8
family-filter-dns.cleanbrowsing.org	185.228.168.168
anydns.cesnet.cz	195.113.144.21
dns.alidns.com	223.5.5.5
dot-de.blahdns.com	78.46.244.143
ore-dns.bitdefender.net	35.247.80.47
one.one.one.one	1.1.1.1
fdns2.dismail.de	159.69.114.157
dns9.quad9.net	9.9.9.9
dot1.applied-privacy.net	146.255.56.98
dns.nextdns.io	178.255.154.59
dns-family.adguard.com	94.140.15.16
dns.adguard.com	94.140.14.14
fra-dns.bitdefender.net	35.242.226.78
olo-fcf18d.dns.nextdns.io	193.9.112.136
dns3.digitalcourage.de	5.9.164.112
ore-dns.bitdefender.net	35.247.80.47
dns.sb	185.222.222.222



Obrázek 2.2: Graf rozdělení DoT provozu dle poskytovatelů v lednu 2022 na síti CESNET 2

2.3 Metody a postup měření

Sledovány byly tyto vlastnosti kanálu:

1. **Délka spojení** před pádem — lze tedy chápat i jako stabilitu
2. **Charakteristiky RTT** tedy „Round Trip Time“ — ty nám pomáhají udělat si představu o latenci spojení
3. **Ztrátovost**
4. **Propustnost**

Délka spojení byla měřena pomocí skriptu, který simuloval keep–alive spojení (tedy jen jakési hlášení klienta serveru, že funguje a je připraven vykonávat příkazy se současnou funkcí udržování spojení a tedy umožnění serveru odesílat případná data i do lokálního segmentu sítě). Z klienta odesílal řetězec *ping*, a server po jeho přijetí odeslal zpět klientovi řetězec *pong*. Klient vyčkal deset vteřin a postup opakoval.

Tento experiment simuloval C&C scénář. Bylo třeba ověřit, jak dlouho vydrží takový tunel — pokud nezvládne pracovat delší dobu při takto nízkém zatížení, existuje logický předpoklad, že se nehodí ani pro náročnější úlohy.

Propustnost byla měřena pomocí programu **scp** v OS Linux. Bylo zahájeno kopírování velkého textového souboru z klienta na server s běžícím tunelovacím nástrojem. Kopírování běželo alespoň půl hodiny, aby byl větší předpoklad, že se projeví případné politiky DNS resolvera týkající se omezování počtu DNS dotazů.

Ztrátovost a RTT charakteristiky byly měřeny pomocí programu `ping`, který ve výstupu zobrazoval požadované charakteristiky.

U dns2tcp nebylo `ping` možné použít. Tím, že bylo třeba provoz tunelovat skrze ssh tunel na konkrétním portu, nebylo možné použít nástroj založený na ICMP protokolu (jednoduše proto, že ICMP protokol neobsahuje koncept portů). Byly testovány nástroje nabízející podobnou funkcionalitu jako program `ping` založené na TCP protokolu, ale výsledky RTT charakteristik nebyly dostatečně kvalitní, aby umožňovaly publikaci.

Jako další měřící metoda byl z tohoto důvodu navržen postup, kdy bude na serveru i klientovi zapnut záznam provozu (pomocí `tcpdump`) a zachycen do souboru formátu `pcap`. V době záznamu bude probíhat tunelování DoT. Následně budou soubory analyzovány pomocí programu `Wireshark` a podle počtu duplikací paketů (jen DoT) bude vypočítána ztrátovost. K duplikaci paketů ale nedocházelo pouze z důvodu ztrátovosti, ale i kvůli různým cesty skrze infrastrukturu, rekurzivnímu dotazování serverů apod. Důsledkem toho bylo, že pakety byly duplikované jak na straně odesílatele (což se k měření hodilo), tak na straně příjemce (což se k měření nehodilo).

Za silného předpokladu, že duplikace na straně odesílatele je způsobena ztrátovostí (tj. že jediným důvodem duplikace paketů je to, že se paket ztratí a proto je odeslán znovu) a duplikace na straně příjemce je způsobena jinými faktory, je možné ztrátovost spočítat. Výpočet probíhal tak, že byl spočten počet paketů odeslaných, přijatých a následně získán jejich podíl. Před součtem přijatých paketů byla provedena deduplikace v souladu s předpokladem. Při měřeních vycházela ztrátovost u poskytovatele Google mezi pěti a šesti procenty, což jsou relativně věrohodná čísla (při srovnání s nástrojem iodine a jeho využití u stejného poskytovatele). Vzhledem k tomu, o jak silný předpoklad ale jde, není tento výsledek mezi ostatními uveden.

2.4 Výsledky a diskuze

Tento oddíl je nutné zahájit poznámkou, že naměřené výsledky nemusí plně odpovídat měření na jiné měřící soustavě. Viz oddíl 2.1, kde je seznámení se strukturou soustavy. Výkon soustavy závisí na více okolnostech - mimo jiné na výkonu routeru, který musí překládat všechny nepřeložené DNS dotazy a který v této konfiguraci rovněž prováděl export datových toků a jejich odesílání na `netmonlab.fit.cvut.cz`.

Tabulka s výsledky 2.2 obsahuje naměřená data u jednotlivých poskytovatelů DoT. Popisuje však poskytovatelů méně, než tabulka 2.1 — vynechává většinu z těch, u kterých tunelování vůbec nebylo možné spustit, případně u kterých tunel „spadnul“ tak rychle, že neměl praktické využití.

Nejzajímavějších výsledků bylo docíleno pomocí poskytovatelů Google a Dismail. U Dismailu se sice nepodařilo tunelovat skrze dns2tcp, ale u iodine bylo docíleno nejvyšší naměřené rychlosti. Smutným faktem ale je, že Google

je v současnosti jedním z nejpoužívanějších DoT resolverů (viz graf 2.2, obsahující data ze sítě CESNET 2) a přitom se v době tvoření této práce podařilo vytvořit při jeho použití DoT tunely s nezanedbatelnými vlastnostmi. Jak bude ale dále rozvíjeno v oddíle 3.3, vykazuje specifickou vlastnost — a to že po přenesení charakteristického objemu dat resetuje spojení.

Přímo zarážejícím faktem jsou naměřené hodnoty u poskytovatele CleanBrowsing. Využita pro měření totiž byla speciálně jeho verze, určená pro rodičovskou kontrolu — měla by tedy blokovat stránky s nevhodným obsahem a chránit před malwarem. Naměřená hodnota (0,9 KB) sice není vysoká a ztrátovost je značná, ale přesto by podle názoru autora mohla postačovat pro C&C scénář. Nechť si čtenář povšimne zejména velmi dobré stability tunelu.

Z bezpečnostního hlediska jsou výborným překvapením hodnoty naměřené u poskytovatele CloudFlare, už pro jeho rozšíření(viz graf 2.2). Tam se totiž vůbec tunel nepodařilo vytvořit — což je správně, protože verze 1.1.1.1 je určena pro rodiny.

Výše uvedené výsledky by mohli sloužit jako nápověda — doporučení chcete-li — pro administrátory sítí, jaké poskytovatele DoT povolit ve své vnitřní síti, a které naopak nepodporovat.

Je třeba připomenout, že výkon a vlastnosti se mohou v čase měnit s aplikováním nových politik a filtračních metod jednotlivých poskytovatelů DoT.

Tabulka 2.2: **Charakteristiky tunelů pro vybrané DoT poskytovatele** — Nástroj iodine/dns2tcp, Stabilita v min. před pádem, RTT charakteristiky v ms (kromě std)

	Nást.	Použ.	Stab.	RTT-min	RTT-avg	RTT-max	RTT-std	Ztrát.	Rychl.
google	**i**	**Ano**	>360	27,9	51,4	3593,1	147,4	2,50%	22 KB/s
	d	**Ano**	120	28,7	213,4	17 365,2	986,4	N	18,5 KB/s
cleanbrowsing	**i**	**C&C**	>360	50,6	1103,8	10 821,6	1489,8	16,80%	0,9 KB/s
	d	**Ne**	—	—	—	—	—	—	—
cesnet	**i**	**C&C**	>360	717,1	745,6	1900,5	111	74,40%	1,5 KB/s
	d	**Ne**	4	—	—	—	—	N	—
alidns	**i**	**C&C**	>360	161	211	8637,6	1192	57,60%	0,1 KB/s
	d	**Ne**	—	—	—	—	—	—	—
blahdns	**i**	**C&C**	50	70,7	618	5126,6	1006,5	23,60%	1 KB
	d	**Ne**	—	—	—	—	—	—	—
cloudflare	**i**	**Ne**	—	—	—	—	—	—	—
	d	**Ne**	—	—	—	—	—	—	—
dismail	**i**	**Ano**	>360	37,4	82	3345	258,6	2,60%	29 KB/s
	d	**Ne**	—	—	—	—	—	—	—
quad 9	**i**	**Ne**	2	—	—	—	—	—	—
	d	**Ne**	—	—	—	—	—	—	—
applied-privacy	**i**	**Ne**	4	531	1241,6	7161,4	1198,4	9,20%	—
	d	**Ne**	—	—	—	—	—	—	—

Kapitola 3

Detekce

V této kapitole bude čtenář seznámen s navrženými metodami detekce, jejich využitím a odchylkami a bude rovněž diskutována jejich použitelnost v praxi. Navrženy byly dva prototypy řešení:

1. **Deterministický klasifikátor** založený na pravidlech a znacích vybraných ze seznamů výše. Podrobněji v oddílu 3.3.
2. **ML klasifikátor** založený na strojovém učení. Bylo testováno více druhů ML (zmíněné v oddílu 1.4) s různými parametry. Podrobněji v oddílu 3.4.

3.1 Technologie

Oba klasifikátory byly vyvíjeny v prostředí Jupyter Notebook [55] (tedy v jazyce Python), které usnadnilo práci s daty a umožnilo rychlejší a pružnější reakce při analýze dat.

3.2 Datová sada

Vytvoření datových sad bylo nezbytnou součástí této práce. Byly nutné pro:

1. Rozbor a analýzu tunelovaného provozu za účelem nalezení charakteristických znaků a vlastností datových toků.
2. Trénink a vyhodnocení úspěšnosti detekce klasifikátoru založeného na strojovém učení.

3.2.1 Příprava dat

Pro zisk datové sady s vhodnými vlastnostmi byla využita filtrace — bylo vyžadováno, aby v datovém toku musely projít alespoň dva pakety oběma

směry. Smyslem tohoto opatření bylo, že z kratších toků není možné vyčíst relevantní vlastnosti.

Využité byly pouze obousměrné datové toky — jednosměrné byly odstraněny. Zejména u datové sady ze sítě CESNET 2 hrozilo, že přes monitorovací sondu přejde jen jedna strana toku — a taková data by nejen nepomohla při analýze, ale hlavně by při významnějším podílu v sadě mohla stát za špatnými odhady klasifikátoru využívajícího strojové učení.

3.2.2 Sady s DoT tunely

Během měření kvantitativních vlastností tunelů i mimo něj byla vytvářena datová sada s tunely. Princip spočíval ve vytvoření tunelu, spuštění skriptu generujícího datový provoz a následně monitoringu činnosti. Monitoring byl neoddělitelnou nutností, protože při využití většiny poskytovatelů DoT byly tunely nestabilní a naneštěstí relativně často „padaly“. V takovém případě bylo využíváno scénáře co nejrychlejšího opětovného navázání spojení.

Celkem byly vytvořeny tři skripty pro generování provozu:

1. Skript pro keep-alive scénář
2. Skript odesílající textový soubor regulovanou rychlostí 1 KB/s
3. Skript odesílající textový soubor regulovanou rychlostí 10 KB/s

Jak vyplývá z tabulky 2.2, tak pro druhý a třetí případ bylo možné využít jen DoT poskytovatele google a dismail.

Využité datové sady byly nakonec tyto:

- `google_iodine_ka`
- `google_iodine_one`
- `google_iodine_ten`
- `google_dns2tcp_ka`
- `google_dns2tcp_one`
- `google_dns2tcp_ten`
- `dismail_iodine_ka`
- `dismail_iodine_one`
- `dismail_iodine_ten`

Jak si čtenář sám jistě povšimnul, název se sestává ze jména poskytovatele DoT, dále z názvu využitého nástroje a končí metodou generování provozu. Pro jednodušší práci byly tyto sady i spojeny a byl z nich vyfiltrován jiný, než DoT provoz.

- `dot_tunnels`
  - Počet záznamů: 1970
  - DoT: 1970
  - Ostatní: 0

Za běhu skriptu bylo zařízení generující provoz dále používáno pro procházení webu — pro „zašumění“ dat běžným provozem a tedy zisk věrnějších dat.

Záznamy o datových tocích byly průběžně vytvářeny a odesílány na server `netmonlab.fit.cvut.cz`, odkud byly zpětně vyfiltrovány od ostatních záznamů a dále zpracovány do formátu `csv`, ze kterého byly později načítány pro datovou analýzu.

Vzhledem k časové náročnosti a nutnosti neustálého dozoru při vytváření sady, není tato sada příliš rozsáhlá — rozšíření datové sady je docela určitě cestou pro vylepšení detekčních možností klasifikátorů — zejména ML.

3.2.3 Sada bez tunelů

Generování datové sady bez tunelů spočívalo v aktivním využívání routeru, na kterém probíhal sběr záznamů o datových tocích. K datové analýze byla následně využita sada vytvořená v průběhu prosince 2021.

- `normal_traffic`
  - Počet záznamů: 863 772
  - DoT: 20 445 (jen legitimní)
  - Ostatní: 843 327

3.2.4 Sada dat ze sítě CESNET 2

Jako referenční sada byla využita sada generovaná z dat ze sítě sdružení CESNET. Tato data obsahující jen DoT provoz byla anonymizována a následně použita pro datovou analýzu konkrétních vlastností datových toků využívaných deterministickým klasifikátorem. Anonymizovaná, nefiltrovaná data byla dodána vedoucím práce.

Bylo předpokládáno, že v této datové sadě se nevyskytují DoT tunely a podle toho bylo nahlíženo na výsledky — viz výsledky měření 4.

Využívána byla sada obsahující filtrovaná data z ledna 2022 a dále sada vytvořená z dat z 15. a 16.12.2021 a z 10.1.2022, kdy byla tato data profiltrována, vyčištěna a zamíchána. Z těchto dat byl vybrán první půl milión řádek. Využívána byla přednostně před sadou z ledna kvůli rychlejšímu času zpracování a tedy zisku solidních výsledků při nižších nárocích na prostředky.

Konkrétně tedy sady:

- `ref_shuffled`
  - Počet záznamů: 500 000
  - DoT: 500 000
  - Ostatní: 0
- `ref_january`
  - Počet záznamů: 1 883 544
  - DoT: 1 883 544
  - Ostatní: 0

3.3 Deterministický klasifikátor

Koncepce deterministického klasifikátoru byla navržena tak, že samotný klasifikátor je funkce, která jako argument přijímá samostatný záznam o síťovém toku a jejím výstupem je vypočítané skóre. Funkce jako taková se sestává z mnoha jiných funkcí, kdy se každá zaměřuje na různé znaky. Je třeba však předeslat, že se jedná o heuristiku a dochází zde tedy k odchylkám.

Je pravdou, že „vystřižením“ toku z kontextu datového provozu ubývají některé možnosti detekce. Tento přístup byl navržen pro zjednodušení a rovněž pro snížení nároků na prostředky detekčního modulu.

Během měření a analyzování naměřených dat se podařilo nalézt některé charakteristické vlastnosti v datové sadě tvořené naměřenými tunely — vlastnosti, jež referenční sada (bez tunelů) nevykazovala. Níže jsou uvedeny:

Reset spojení se projevil u poskytovatele Google. Díky experimentům se ukázalo, že Google spojení resetuje ve většině případů na základě objemu přenesených dat (viz tabulka 3.1, kde je vidět množství TCP příznaků pro reset). Toho lze využít — pokud je poskytovatelem Google a tok vykazuje charakteristický přenesený objem, můžeme ho označit jako „podezřelý“ — už proto, že zmíněný objem není zanedbatelný a pokud je vyčerpán za relativně krátký čas, může nám být zajímavým vodítkem. Tento přístup může být pochopitelně problémový, například pokud budeme mít v síti rozsáhlý segment s NAT — zařízení v něm mohou generovat velké množství dotazů.

Velikost toku může být obecně vodítkem — zejména, pokud detekce probíhá například na domácí síti. Poskytovatel Dismail spojení neresetuje tak, jako Google, ale umožňuje přenos velkého množství dat. Některé toky tak vykazovaly velikost řádově megabajtů. S přihlédnutím k vlastnostem systému IPFIXProbe (podrovněji k vlastnostem v oddílu 1.3.2) tedy může jít i o více než 3 KB za sekundu — což pro DNS, potažmo DoT není typické. Znovu zde může dojít k chybám zapříčiněných NAT segmenty v síti.

Tabulka 3.1: Ukázka dekódovaných TCP flagů posledního z paketů v datové sadě vytvořené pomocí nástroje iodine a poskytovatele Google s provozem 10 KB/s

Flags	Počet
-APRSF	129
- - - - - - -	6
-AP-SF	6
-APR-F	2
-AP- - -	1
-AP-S-	1
-APRS-	1

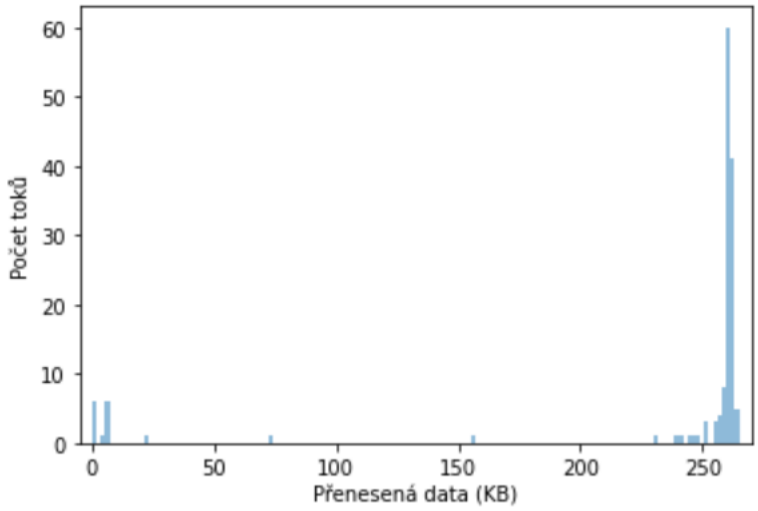
Velikost paketů nabízí obecně mnoho zajímavého. Pro toky, které měly délku větší než daný počet paketů jsem dopočítal nad datovými sadami konfidenční intervaly[7] pro velikost největšího paketu, nejmenšího paketu a některé kvantily. Díky tomu bylo možné stanovit hranice, mimo které bude tok označen jako podezřelý. Úvaha za tím tkví především v tom, že nástroje jako iodine uměle nastavují velikost odesílaných dotazů a takové rozdělení se při větším využití tunelu blíží až rovnoměrnému. Především pak ale nenajdeme v takových tocích výjimečně velké pakety, které se ale v „běžných“ tocích vyskytují — byť jich jsou třeba jen zlomky procent.

I zde pochopitelně narazíme na mnoho nesnází — například využití různého paddingu může takovou analýzu znesnadnit a stanovení konfidenčních intervalů může být závislé na konkrétní konfiguraci.

3.3.1 Analýza velikosti DoT toků — Google

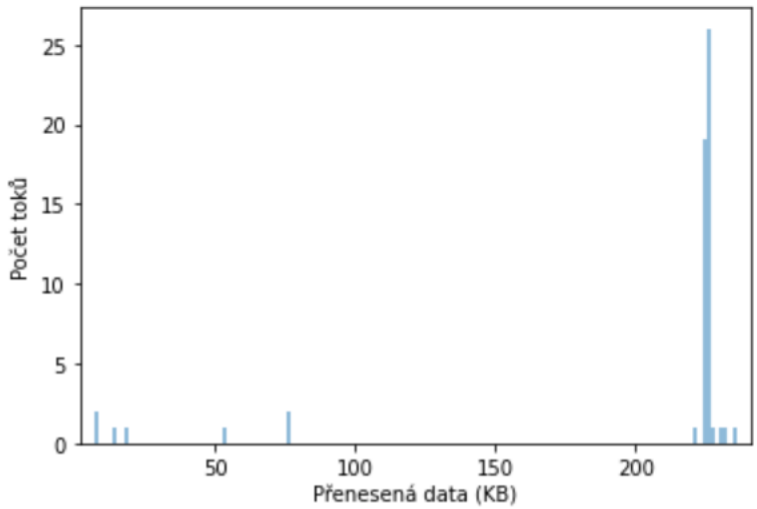
Velikost toku mnoho napovídá, ale u poskytovatele Google je zvláště charakteristická. To je pěkně vidět na histogramu 3.1.

[7] Byl zde využit předpoklad, že rozdělení paketů se bude s množstvím vzorků blížit normálnímu rozdělení.



Obrázek 3.1: Počet bajtů na tok v datové sadě google_iodine_ten (viz oddíl 3.2)

Za povšimnutí stojí velký peak těsně nad přenesenými 250 KB (viz graf 3.1). Pro detekci velmi zajímavý fakt. Bylo uvažováno, jak bude vypadat histogram jen u keep–alive scénáře. Ten je vidět na obrázku 3.2.



Obrázek 3.2: Počet bajtů na tok v datové sadě google_iodine_ka (viz oddíl 3.2)

Na základě identifikace poskytovatele Google a velikosti datových toků byla navržena součást deterministického klasifikátoru.

3.3.2 Analýza maximální velikosti paketů

V rámci ověřování a hledání zajímavých metrik pro určení ideálních charakteristik pro možnou klasifikaci byly zkoumány statistické vlastnosti zaznamenaných velikostí přenesených paketů. Konkrétně:

1. Velikost největšího paketu toku
2. Velikost nejmenšího paketu toku
3. Kvantil 25% velikostí
4. Kvantil 50% velikostí
5. Kvantil 75% velikostí

$$\left(\overline{X} - \frac{S_x}{\sqrt{n}} \cdot t_{n-1}(0,975), \overline{X} + \frac{S_x}{\sqrt{n}} \cdot t_{n-1}(0,975)\right) \tag{3.1}$$

3.1: Rovnice oboustranného konfidenčního intervalu pro $\alpha = 5\%$ a neznámý rozptyl [56]

Pro všechny hodnoty byl počítán oboustranný konfidenční interval s hladinou významnosti $\alpha = 5\%$ podle rovnice 3.1. V rovnici $\overline{X}$ odpovídá výběrovému průměru, S_X je výběrový rozptyl, n je počet vzorků a t_{n-1} značí t-rozdělení s $n-1$ stupni volnosti. Výsledky jsou v následujících tabulkách (tj. 3.2):

Tabulka 3.2 ukazuje, že velikost nejmenšího paketu se pro odlišení provozu s DoT tunelu od „čistého“ příliš nehodí. Jejich rozdíly jsou velmi malé. Zato však maximální paket vykazuje již velký potenciál.

Opravdu velké rozdíly jsou pak vidět u konfidenčních intervalů pro kvantily. V některých situacích je hodnota pro „čistý“ provoz více než 6x menší, než pro provoz s tunely.

Jak velikost největšího z paketů v toku, tak kvantily rozdělení jejich velikostí byly využity jako součásti deterministického klasifikátoru.

3.3.3 Analýza množství přenesených dat — obecně

Jako možný charakteristický znak pro toky obsahující DoT tunely se jevila možnost sledování přenesených dat. Z toho důvodu byla provedena analýza datové sady s tunely a rovněž datové sady bez DoT tunelů.

Jak je vidět na histogramech 3.3, objevuje se nemálo toků, které mají výrazně odlišný počet přenesených dat — vidět to můžeme na pravém histogramu 3.3, kde pozorujeme součet velikostí paketů v toku nad 200 KB.

Povšimněme si ale zejména statistických vlastností toků u poskytovatele `dismail` v tabulce 3.3.

Znovu je ale třeba zmínit, že datová sada s DoT tunely by byla třeba rozsáhlejší pro získání spolehlivějších výsledků.

Tabulka 3.2: Oboustranné konfidenční intervaly pro velikost minimálního a maximálního paketu a kvantilů velikostí paketů při $\alpha = 5\%$, kde $\overline{X}$ je výběrový průměr a $\overline{X} \pm h$ jsou hranice intervalu. Datové sady byly použity `ref_january` a dále `dot_tunnels`.

Nejmenší paket

Sada	$\overline{X}$	$\overline{X}$ - h	$\overline{X}$ + h
dot_tunnels	44,79	43,06	46,52
ref_january	45,63	45,54	45,72

Největší paket

Sada	$\overline{X}$	$\overline{X}$ - h	$\overline{X}$ + h
dot_tunnels	1383,89	1375,92	1391,86
ref_january	970,61	969,94	971,28

25% kvantil

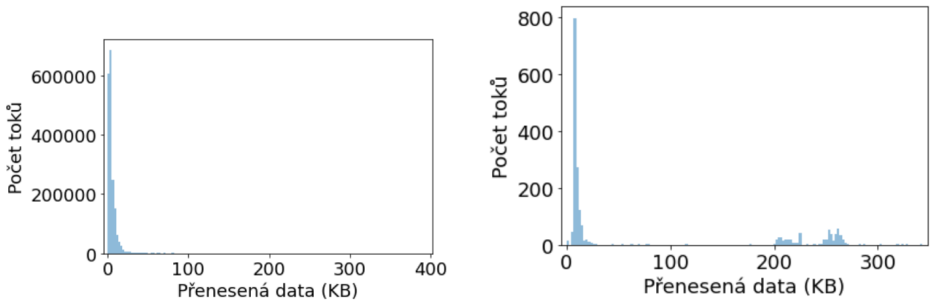
Sada	$\overline{X}$	$\overline{X}$ - h	$\overline{X}$ + h
dot_tunnels	104,64	97,68	111,60
ref_january	16,87	16,79	16,94

50% kvantil

Sada	$\overline{X}$	$\overline{X}$ - h	$\overline{X}$ + h
dot_tunnels	169,83	159,35	180,31
ref_january	32,14	31,99	32,29

75% kvantil

Sada	$\overline{X}$	$\overline{X}$ - h	$\overline{X}$ + h
dot_tunnels	201,50	188,26	214,74
ref_january	51,37	51,14	51,59



(a) Normální provoz (b) DoT tunel

Obrázek 3.3: Histogramy počtu bajtů v toku — omezeno na toky menší než 400 KB

Tabulka 3.3: Výpis statistických vlastností toků s DoT tunely při odesílání 10 KB/s s využitím poskytovatele dismail a nástroje iodine

počet	15
výb. průměr	2565 KB
st. odchylka	519,35 KB
minimum	2070 KB
kvantil 25%	2267,5 KB
kvantil 50%	2366 KB
kvantil 75%	2654 KB
maximum	3867 KB

3.3.4 Funkce

Jak už bylo řečeno výše, deterministický klasifikátor byl navržen jako modulární, přičemž každý modul (funkce) sleduje nějaké charakteristické kritérium v provozu a podle jeho splnění či částečného splnění přiřadí vzorku skóre.

Funkce sledují tyto charakteristické znaky:

Flag pro sledování resetu u Googlu, z toho důvodu monitoruje i IP poskytovatele DoT

Přenesená data — pokud je poskytovatelem Google, kvůli informacím, jež plynou z grafů 3.1 a 3.2

Přenesená data — pro jiné poskytovatele. Důvodem jsou zjištění například v tabulce 3.3

Kvantily velikosti paketů — jako konstanty je třeba využít hranice konfidenčních intervalů z datové analýzy — je využit jednostranný interval kvůli zjištění z datové analýzy (viz tabulka 3.2

Velikost největšího přeneseného paketu — je třeba hranice konfidenčního intervalu, opět postačoval jednostranný interval (viz tabulka 3.2)

Výhodou bylo, že se jednotlivé funkce daly odstraňovat a opětovně přidávat do prototypu klasifikátoru za účelem získání lepší úspěšnosti detekce či úpravou statistických vlastností (například potlačení detekce legitimního provozu jako škodlivého i za cenu nižší úspěšnosti detekce).

3.3.5 Realizace

Pseudokód výsledného prototypu deterministického klasifikátoru je na výpisu kódu 3.1.

Funkcionalita je nastavitelná pomocí konstant. Jde o **1.** konstanty *threshold*, které určují hranice pro detekci konkrétního znaku, jak zmiňuje oddíl 1.2.2.

```
def tunnel_clf(flow):
    return google_tunnel_clf(flow) + size_tunnel_clf(flow)

def size_tunnel_clf(flow):
    return flow_size_sign(flow) + quantile_sign(flow) + packet_size_sign(flow)

def google_tunnel_clf(flow):
    return is_google_dot(flow) * google_bytes_count_sign(flow) * flag_sign(flow)

def packet_size_sign(flow):
    if flow.packets < PACKET_COUNT_LIMIT:
        return 0
    if max(flow.packets) > MAX_PACKET_SIZE_THRESHOLD:
        return MAX_PACKET_SIZE_SCORE
    return 0

def quantile_sign(flow):
    if flow.packets < PACKET_COUNT_LIMIT:
        return 0
    quant = np.quantile(flow.packets, PACKET_SIZE_QUANTILE)
    if quant > LOWER_PACKET_SIZE_QUANTILE_THRESHOLD:
        return QUANTILE_SIGN_SCORE
    return 0

def flow_size_sign(flow):
    if flow.kilo_bytes_count > FLOW_SIZE_THRESHOLD:
        return FLOW_SIZE_SCORE
    return 0

def google_bytes_count_sign(flow):
    if flow.kilo_bytes_count < GOOGLE_DOT_SIZE_HIGHER_THRESHOLD
      and kilo_bytes_count > GOOGLE_DOT_SIZE_LOWER_THRESHOLD:
        return GOOGLE_BYTES_COUNT_SCORE
    return 0

def flag_sign(flow):
    if 'R' in (flow.tcp_flags or flow.tcp_flags_rev):
        return 1
    return 0
```

Výpis kódu 3.1: Pseudokód prototypu deterministického klasifikátoru.

Za **2.** jde o konstanty *score*, které umožňují nastavení významnosti jednotlivých znaků. Za **3.** pak systém využívá nastavení počtu paketů pro některé znaky a výběr kvantilu, na němž probíhá klasifikace.

V průběhu realizace byla zkoušena i úspěšnost klasifikátoru. V základní verzi však neposkytovala příliš dobré výsledky (data viz oddíl 4.2). Tím je myšleno, že přestože některé případy byly opravdu správně detekovány, nebylo jich mnoho a navíc existovalo velké procento „false positive“ případů — tedy případů kdy sice klasifikátor detekuje provoz jako obsahující DoT tunely, ale

přitom může jít o provoz plně legitimní. Problém byl například v tom, že mnoho toků obsahovalo nějaký opravdu velký paket a proto byly označeny za podezřelé.

I přes tento nezdar práce s deterministickým klasifikátorem nebyla ukončena a dále byly zkoušeny různé možnosti úpravy funkce. Jako výhodné se ukázalo být přestat používat funkce, které podezřelost provozu určovaly na základě přenesených dat v toku a které pracovaly s kvantily rozdělení velikostí paketů v toku (rovněž pak s velikostí největšího paketu). Více v oddílu 4.2.

3.4 ML klasifikátor

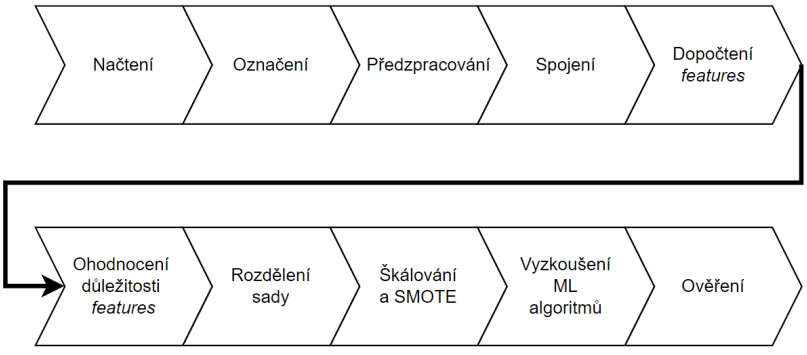
V této sekci bude shrnuto, jakým způsobem bylo využito metod strojového učení pro klasifikaci DoT tunelů.

Postup byl následovný:

Načtení dat Načteny byly datové sady, se kterými se dále pracovalo. Tedy jak sady bez DoT tunelů, tak s nimi.

Označení dat Data, o kterých bylo známo, že obsahují DoT tunely byly označeny pomocí pravdivostní hodnoty True, jinak jim byla nastavena hodnota False. Označování bylo prováděno na základě času, kdy byla data měřena — pokud v té době byl generován provoz DoT tunely, pak byl veškerý DoT provoz označen jako obsahující tunely. To s sebou samozřejmě nese určitou možnost nepřesností, pokud však budou omezené, měly by si s nimi ML algoritmy poradit.

Předzpracování datových sad Datové sady byly ještě připravovány pro lepší zpracování. Například docházelo k otáčení některých toků (v případě, že cílová IP adresa odpovídala lokální síti) a byly využívány jen toky, které měly v obou směrech alespoň dva pakety (tj. kdy ze zdroje byly



Obrázek 3.4: Schéma postupu při detekci ML klasifikátorem

vyslány dva pakety a naopak dva byly přijaty). Toto předzpracování bylo využito pro zlepšení vlastností datových sad a omezení důsledků některých vlastností kolektoru IPFIXProbe.

Spojení datových sad Spojením datových sad jsem získal jednu celistvou, se kterou jsem dále pracoval.

Dopočítání charakteristických vlastností Charakteristické vlastnosti jsou klíčové pro počítání s algoritmy strojového učení. Tyto vlastnosti, obvykle nazývané anglicky *features*, byly dopočítávány pomocí již vytvořené funkcionality Laboratoře monitorování síťového provozu — pomocí *Feature Exploration Toolkit* [57]. Důležitým faktem je, že strojové učení nemělo k dispozici IP, MAC adresy ani použité porty. Vzhledem k tomu, že využívány byly sady, které obsahovaly pouze DoT, neměl by se tento fakt projevit na úspěšnosti. Je ale třeba zdůraznit, že před využitím výsledného prototypu je třeba doplnit jednoduchou komponentu, která by první filtrovala z provozu jiná, než DoT data.

Ohodnocení důležitosti charakteristických vlastností bylo rovněž prováděno za pomoci již implementovaných funkcí. Výsledek je vidět na obrázku 3.5. Zajímavé je srovnání tohoto grafu s analýzou datových sad. I tam byly délky paketů a jejich statistické vlastnosti rozpoznány jako významné pro klasifikaci.

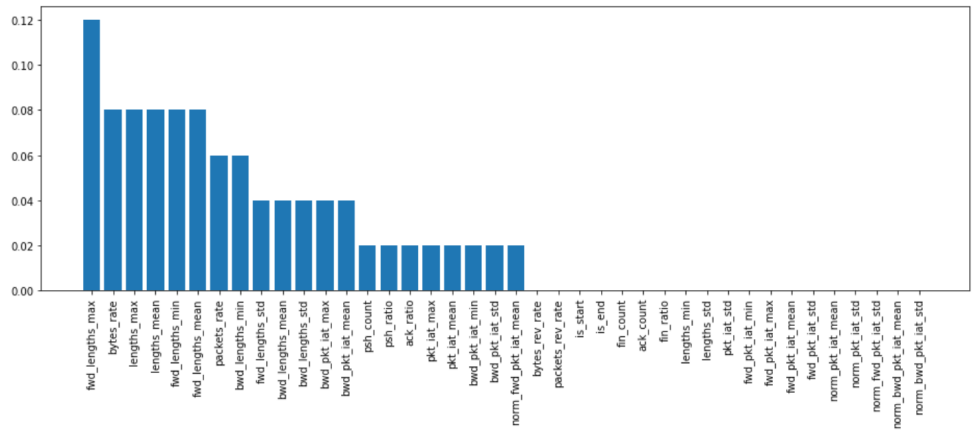
Rozdělení datové sady na trénovací a testovací bylo provedeno s parametrem 0,3 (tj. 70% dat se využilo pro trénink a na zbytku se ověřovala úspěšnost algoritmu).

Bylo využito „zamíchání“ datové sady pro pravděpodobnostní zajištění rozdělení dat s DoT tunely napříč sadami vytvořenými rozdělením.

Oprava nevyváženosti datových sad bylo zařazeno proto, že datové sady vykazovaly opravdu velkou nevyváženost mezi legitimním provozem a záznamem DoT tunelování. Proto bylo využito algoritmu SMOTE [58] (***S**ynthetic **M**inority **O**versampling **T**echnique*) připraveného v jazyce Python. Ten trénovací sadu doplňuje o vygenerované záznamy. Rovněž bylo využito škálování.

Vyzkoušení různých ML algoritmů s různými parametry pro nalezení optimálního proběhlo jak ručně (pro testovací účely), tak automatizovaně pomocí hrubé síly (*GridSearchCV*). Skóre jednotlivých algoritmů spolu s nejlepšími nalezenými parametry je k dispozici v tabulkách 4.1 a 4.2.

Vyhodnocení výsledků bylo prosté. Po získání výsledků pro všechny algoritmy byl vybrán ten s nejlepší úspěšností.



Obrázek 3.5: Ohodnocení důležitosti *features* pomocí klasifikátoru AdaBoost

Ověřování na kontrolní sadě je nezbytné s ohledem na riziko přetrénování modelu. Proto byla úspěšnost algoritmu dále ověřována na kontrolní sadě dat.

Celkem byly využity dva různé přístupy, kdy nejdříve byly jako trénovací a testovací sady využívány pouze autorem naměřené datové sady — tedy `dot_tunnels` a `normal_traffic`. Druhá možnost spočívala ve využití i části referenčních dat pro doplnění testovací a trénovací sady. Oba přístupy měly své výhody. Zejména dat s DoT tunelovaným provozem nebylo mnoho. Takže při připojení referenční sady byla získána širší datová základna a byly zaznamenány i typy provozu, které v domácím prostředí nebyly vygenerovány. Na druhou stranu již silně nevyvážená sada se stala ještě nevyváženější.

Jak bylo již zmíněno, tak klasifikátory strojového učení neměly přístup k IP, MAC adresám ani ke zdrojovému a cílovému portu. Důvodem byla především snaha o to, aby klasifikátor sledoval statistické vlastnosti toků a ne konkrétní adresy a porty. Navíc bude jednodušší přenositelnost modelu (pokud by byl model závislý na konkrétní IP adrese monitorovacího zařízení, pravděpodobně nebude fungovat optimálně, když se tato adresa změní). Jako možné vylepšení výsledků by bylo jistě možné zkombinovat tento přístup s přístupem deterministickým a pokud je například předpokládáno, že na adrese 1.1.1.1 poskytovatele Cloudflare není tunel možné vytvořit, rovnou taková data označit jako „čistá“. Tento přístup má ale jistou nevýhodu a to, že stěží zareaguje na nějaké komplexnější změny v chování serverů DoT poskytovatelů. Takže pokud by došlo k porušení funkcionality, která DoT tunelům zabraňuje, nebo pokud by nějaký hacker objevil nový způsob, jak takové funkce obejít, klasifikátor by takový tok automaticky vyřadil jako „čistý“, přestože jím není — a především by ani nedal možnost algoritmu strojového učení takový tok vůbec klasifikovat.

V experimentech byl nicméně volen takový cíl, aby byl klasifikátor schopen

rozeznat „čistý“ DoT provoz, od DoT provozu, který obsahuje tunely. Klasifikátor proto dostal datovou sadu, která byla již vyfiltrovaná — obsahovala pouze záznamy o tocích, jež využívaly port 853 [4]. Příklad natrénovaného rozhodovacího stromu je na obrázku 3.6. V práci se využívaly stromy hlubší, ale každopádně si lze povšimnout podmínek v uzlech a rovněž zdrojů nepřesností v listech. To je dobře vidět na listech 6–9 zleva. Výsledně klasifikované třídy jsou tvořeny i určitou částí špatně detekovaných vzorků.

3.4.1 Charakteristické vlastnosti

Pro správné fungování klasifikátorů bylo třeba získat charakteristické vlastnosti (*features*) na základě kterých detekce probíhá. Jak bylo zmíněno dříve, ty byly dopočítávány pomocí funkcionality již vytvořené [57]. V tabulce 3.4 je jejich přehled. Diplomová práce Ing. Daniela Uhříčka [57] jich uvádí více, nicméně v této práci byla využita jen jejich podmnožina a to z důvodu neúčinnosti (podle výpočtu významnosti) či korelovanosti s jinými *features*. Podobné *features* zpracovává dále nástroj `CICFlowMeter` [59], jehož funkcionalita však v této práci využita nebyla.

Celkově se dají kategorizovat do několika skupin [57], viz níže. Popis *features* je v tabulce 3.4:

1–4 obsahují informace o velikosti toku, jak v součtu, tak samostatně. Hodnoty jsou normalizované podle délky toku.

5–8 jsou statistiky TCP příznaků.

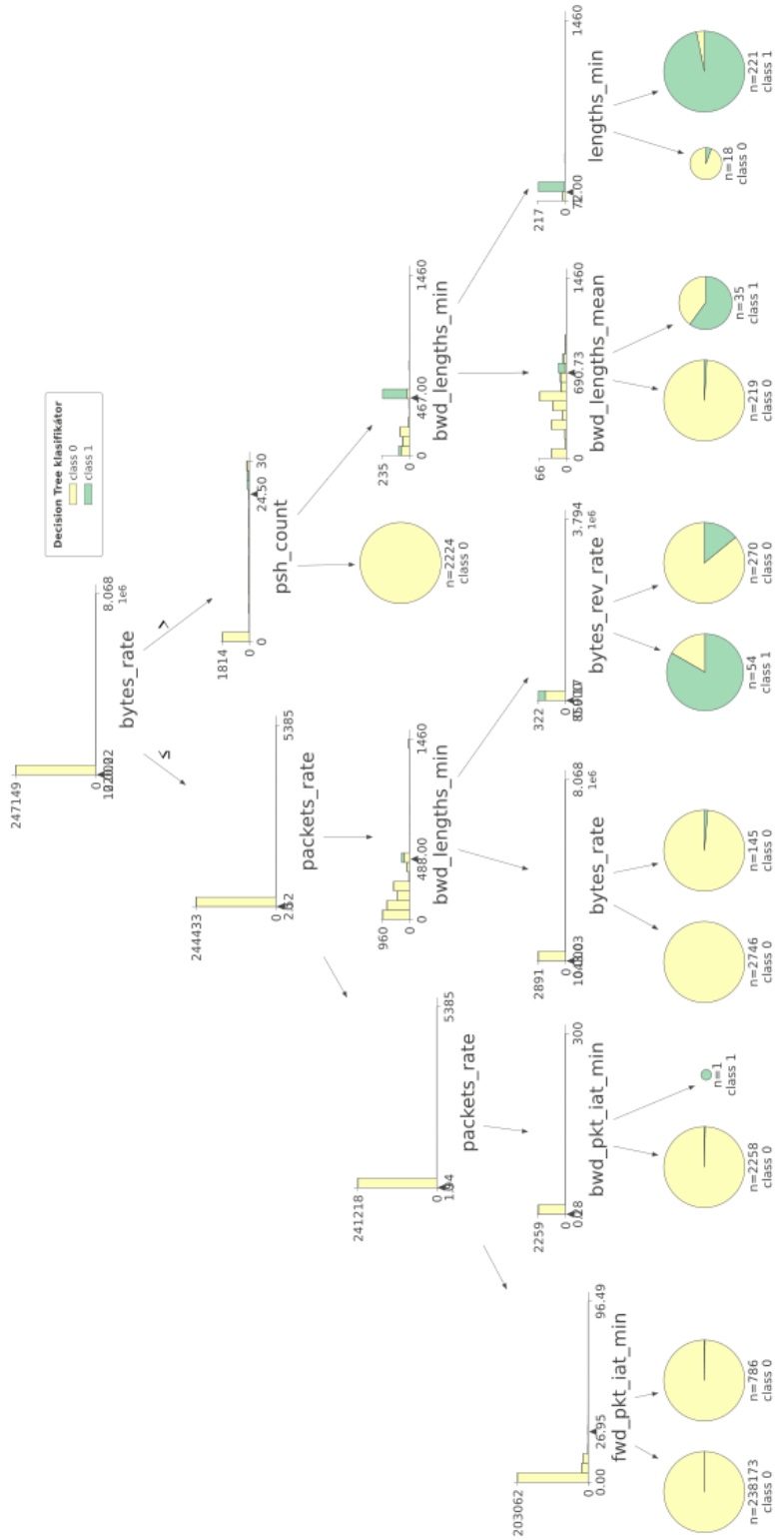
9–19 zahrnují statistiky o délkách paketů v toku.

20–30 informují o časových intervalech mezi pakety. Mohou nám pomoci například s detekcí nezvykle krátkých odstupů mezi dotazy (viz oddíl 1.2.2 a bod „Časový odstup“).

31–36 dále rozšiřují informace o předchozí skupině. Normalizují ale časy podle hranice 5 sekund. Čas nad pět sekund je charakterizován jako „dlouhý“, zatímco kratší, než 5 sekund jako „krátký“.

Tabulka 3.4: Charakteristické vlastnosti (*features*) využívané ML klasifikátorem

Číslo	**Feature**	**Zkratka**
1	Počet bajtů směrem k cíli	bytes_rate
2	Počet bajtů směrem ke zdroji	bytes_rev_rate
3	Počet paketů směrem k cíli	packets_rate
4	Počet paketů směrem ke zdroji	packets_rev_rate
5	Počet PSH	psh_count
6	Poměr PSH	psh_ratio
7	Počet ACK	ack_count
8	Poměr ACK	ack_ratio
9	Délka nejmenšího paketu	lengths_min
10	Délka největšího paketu	lengths_max
11	Průměr velikostí	lengths_mean
12	St. odchylka velikostí	lengths_std
13	Délka nejmenšího paketu ve směru k cíli	fwd_lengths_min
14	Délka největšího paketu ve směru k cíli	fwd_lengths_max
15	Průměr velikostí paketů ve směru k cíli	fwd_lengths_mean
16	St. odchylka velikostí ve směru k cíli	fwd_lengths_std
17	Délka nejmenšího paketu ve směru ke zdroji	bwd_lengths_min
18	Průměr velikostí paketů ve směru ke zdroji	bwd_lengths_mean
19	St. odchylka velikostí ve směru ke zdroji	bwd_lengths_std
20	Největší časový odstup mezi pakety	pkt_iat_max
21	Průměrný odstup mezi pakety	pkt_iat_mean
22	St. odchylka čas. odstupů mezi pakety	pkt_iat_std
23	Nejmenší odstup mezi pakety ve směru k cíli	fwd_pkt_iat_min
24	Největší odstup mezi pakety ve směru k cíli	fwd_pkt_iat_max
25	Průměrný odstup mezi pakety ve směru k cíli	fwd_pkt_iat_mean
26	St. odchylka odstupů mezi pakety ve směru k cíli	fwd_pkt_iat_std
27	Nejmenší odstup mezi pakety ve směru ke zdroji	bwd_pkt_iat_min
28	Největší odstup mezi pakety ve směru ke zdroji	bwd_pkt_iat_max
29	Průměrný odstup mezi pakety ve směru ke zdroji	bwd_pkt_iat_mean
30	St. odchylka mezi pakety ve směru ke zdroji	bwd_pkt_iat_std
31	Normalizované odstupy paketů – průměr	norm_pkt_iat_mean
32	Norm. odstupy paketů – stdev	norm_pkt_iat_std
33	Norm. odstupy paketů – průměr ve směru k cíli	norm_fwd_pkt_iat_mean
34	Norm. odstupy paketů – stdev ve směru k cíli	norm_fwd_pkt_iat_std
35	Norm. odstupy paketů – průměr ve směru ke zdroji	norm_bwd_pkt_iat_mean
36	Norm. odstupy paketů – stdev ve směru ke zdroji	norm_bwd_pkt_iat_std



Obrázek 3.6: Znázornění rozhodovacího stromu při nastavení maximální hloubky na čtyři úrovně

3.4.2 Realizace

Měření úspěšnosti klasifikace probíhalo v rámci prostředí `JupyterHub`, jak je zmíněno v úvodu kapitoly. Výstupem práce je však `Python` skript, který pracuje na příkazové řádce. Ve výpise pseudokódu 3.2 je vidět hlavní funkce prototypu klasifikátoru DoT tunelů. Tento skript je možné nakonfigurovat pomocí:

1. Upravením konstant
   - **DEFAULT_CLASSIFICATOR**
   - **DEFAULT_FEATURES**
   - **DEFAULT_OUTPUT**
2. Spuštěním s argumenty
   - **src** (povinný)
   - **dst**
   - **clf**
   - **features**

Pokud nejsou zadány nepovinné argumenty, použijí se přednastavené v konstantách. Argument **src** zadává zdrojový `csv` soubor, zatímco **dst** udává výstupní. Pomocí argumentu **clf** lze specifikovat cestu k vyexportovanému klasifikátoru. Pak je potřeba pomocí **features** načíst seznam využitých charakteristických vlastností (*features*). Pozor, skript provádí konverzi časů a generuje *features* pomocí *Feature Exploration Toolkit* [57], žádné jiné nepřidává a není možné další bez přepracování využívat. Je však možné nastavit jinou podmnožinu. Není zde připravena funkce nastavení *thresholdu*.

Výstupem skriptu je **1.** textový výstup obsahující postup klasifikace a počet vzorků klasifikovaných jako záznamy obsahující DoT tunely. A za **2.** je výstupem `CSV` dokument, který obsahuje upravené klasifikované záznamy včetně sloupce *pred*, který uvádí výsledek klasifikace.

```
def run(src, dst, clf, feature_file):
    df = load_source(src, feature_file)
    print('Data loaded. ')
    model = load_model(clf)
    print('ML model loaded. ')
    pred = model.predict(df)
    df['pred'] = pred.tolist()
    positive = pred.sum()
    print('Found ', positive
        , ' positive records. \nIt\'s ',
        positive*100/len(pred),
        '\% of all records. ')
    save_result(df, dst)
```

Výpis kódu 3.2: Pseudokód prototypu ML klasifikátoru.

Kapitola 4

Testování

V této kapitole bude shrnuta metodologie porovnání, výsledky a úspěšnost detekce jednotlivých klasifikátorů a rovněž proběhne diskuze odchylek a jejich možných příčin.

4.1 Metriky a postupy

V tomto oddílu bude shrnuto, jak bylo postupováno při testování prototypů klasifikátorů. Dále — a především — budou vysvětleny metriky, pomocí kterých byly algoritmy porovnávány.

Matice záměn je maticí, která ve sloupcích obsahuje skutečnou třídu záznamu, zatímco v řádcích jsou hodnoty předpovězené klasifikátorem. Matice záměn byly používány pro ukázku účinnosti klasifikátoru, protože obsahují více informací a je z nich možné vyčíst i které třídy byly zaměňovány za jiné včetně poměru. Vzhledem k tomu, že detekce cílila jen na dvě hodnoty a stačila tedy pravdivostní hodnota jsou matice rozměru 2×2.

Úspěšnost detekce je dána jako počet správně klasifikovaných záznamů ku počtu klasifikovaných záznamů (viz rovnice 4.1[8]). Mohlo by se stát, že klasifikátor, který by vše klasifikoval jako `True` by v sadě obsahující 99 % záznamů o správné hodnotě `True` vykazoval úspěšnost 99 %. Proto jsou také do práce vkládány matice záměn. Výpočet přesnosti probíhá na testovací sadě (jak bylo zmíněno v postupu v úvodu oddílu 3.4 o ML klasifikátoru, sady s tunelovaným i netunelovaným provozem jsou označeny, spojeny a „zamíchány“, aby byly následně rozděleny ve zvoleném poměru na sadu trénovací a testovací).

FP – ref značí počet falešně pozitivních záznamů (dále označované i jako „false positive“) — tedy záznamů o tocích, které jsou „čisté“, ale které

[8] TP = počet vzorků správně určených jako pozitivní, TN = počet vzorků správně určených jako negativní, T = počet pozitivních vzorků, F = počet negativních vzorků

klasifikátor označí jako toky obsahující DoT tunel. Testováno je na referenčních sadách, o kterých je předpokládáno, že obsahují pouze „čistý“ provoz. Tento předpoklad je založen na tom, že v rámci analýzy nebyly nalezeny žádné nástroje pro DoT tunelování a na tom, že DoT je relativně nová technologie. Přítomnost těchto tunelů v referenčních sadách však vyloučit nelze.

$$\frac{TP + TN}{P + N} \tag{4.1}$$

4.1: Rovnice pro výpočet přesnosti z matice záměny.

4.2 Detekce deterministickým klasifikátorem

Vzhledem k nadějným výsledkům datové analýzy z oddílu 3.3 byla očekávána solidní úspěšnost tohoto modelu. Tato očekávání se však nenaplnila.

Při spuštění detekce základní verze klasifikátoru na sadu `dot_tunnels` bylo detekováno pouhých 39% toků jako toky obsahující DoT tunel. Když byla klasifikace spuštěna na sadě `ref_shuffled`, a tedy bylo předpokládáno, že by žádné tunely detekovány neměly být, bylo získáno 11,4% detekovaných případů. Což je velmi značné množství „false positive“ ve výsledku.

Po odstranění funkce sledující největší paket v toku a kvantily se podařilo docílit celkem 0,032% detekovaných záznamů na referenční sadě (`ref_shuffled`), což je jako „false positive rate“ velmi dobrý výsledek. Se snížením špatně detekovaných záznamů se však snížila i úspěšnost správně detekovaných dat, a to na 23,7% při využití sady `dot_tunnels`. Jako „false positive“ se v experimentu ukázaly toky s poskytovatelem Google, kde byla splněna podmínka jak pro reset flag, tak pro velikost toku. Proto je otazné, zda nemohly být nějaké DoT tunely i v referenční sadě. Zbylá podmínka na počet přenesených dat v toku v referenční sadě nebyla splněna pro žádný z toků.

Ani tento výsledek není ideální a úspěšnost detekce ojedinělého tunelu v kontextu širšího provozu naneštěstí není vysoká. Přesto lze však považovat tento výsledek nejen za zajímavý pro velké poskytovatele internetových služeb, kteří zpracovávají velké objemy dat a nemají kapacitu na ověřování správnosti detekce. Těm by algoritmus na základě vytvořeného prototypu mohl pomoci. Navíc by ale při rozsáhlejším datovém výzkumu mohlo dojít k nalezení dalších charakteristických vlastností poskytovatelů DoT, které se projeví při tunelování skrze DoT, a tedy i k vylepšení klasifikátoru.

Je ale třeba zdůraznit, že v běžném provozu se může úspěšnost detekce lišit. Jeden z hlavních důvodů mohou být jiné poměry využívání poskytovatelů DoT — to je vidět na vytvoření datové sady `dot_tunnels` v oddíle 3.2.2 oproti referenčnímu provozu na grafu 2.2.

4.3 Detekce klasifikátorem strojového učení

Zde jsou shrnuty výsledky úspěšnosti detekce pomocí ML klasifikátoru. Cílem bylo dosáhnout lepších výsledků, než u relativně neúspěšného pokusu s deterministickým klasifikátorem.

4.3.1 Bez využití referenční sady

Experiment spočíval v trénování i testování klasifikátoru pouze na sadě vytvořené z autorem naměřených dat (kombinace sad `dot_tunnels` společně s `normal_traffic`). Teprve následně byla využita referenční sada pro ověření úspěšnosti detekce (využita byla část sady `ref_shuffled`, jejích 70% vybraných po promíchání záznamů) — pro zamezení přetrénování modelu a získání očekávatelné úspěšnosti.

Výsledky zobrazené v podobě maticí záměn je možné vidět na ilustraci 4.1. Pro výpočet byly využity parametry získané z výpočtu pomocí `GridSearchCV`. Ty jsou v tabulce 4.1.

4.3.2 S využitím referenční sady

Zde již byla využita část referenční sady za účelem doplnění typů provozu, které se v autorem generovaném nevyskytovaly, do datové sady. Konkrétně byl model natrénován a následně testován na kombinaci sad `dot_tunnels`, `normal_traffic` a `ref_shuffled`. Úspěšnost detekce byla následně ověřována na podmnožině sady `ref_january` (konkrétně na 70% — tedy 1 318 482 záznamech). Výsledky jsou v podobě matic záměn na ilustraci 4.2 a nejlepší parametry jsou v tabulce 4.2.

Výsledky se po využití části referenční sady výrazně zlepšily. Výpočet algoritmu KNN byl velmi pomalý (v porovnání s ostatními testovanými algoritmy — viz tabulka 4.3), což ale odpovídá očekávání. Jinak například algoritmus Random Forest dosáhl lepší úspěšnosti než 99,9% (na testovaných sadách), což je výsledek výrazně lepší, než bylo očekáváno. Jak je ale vidět z matic záměn, stále zůstává poměrně velké množství falešně pozitivních nálezů.

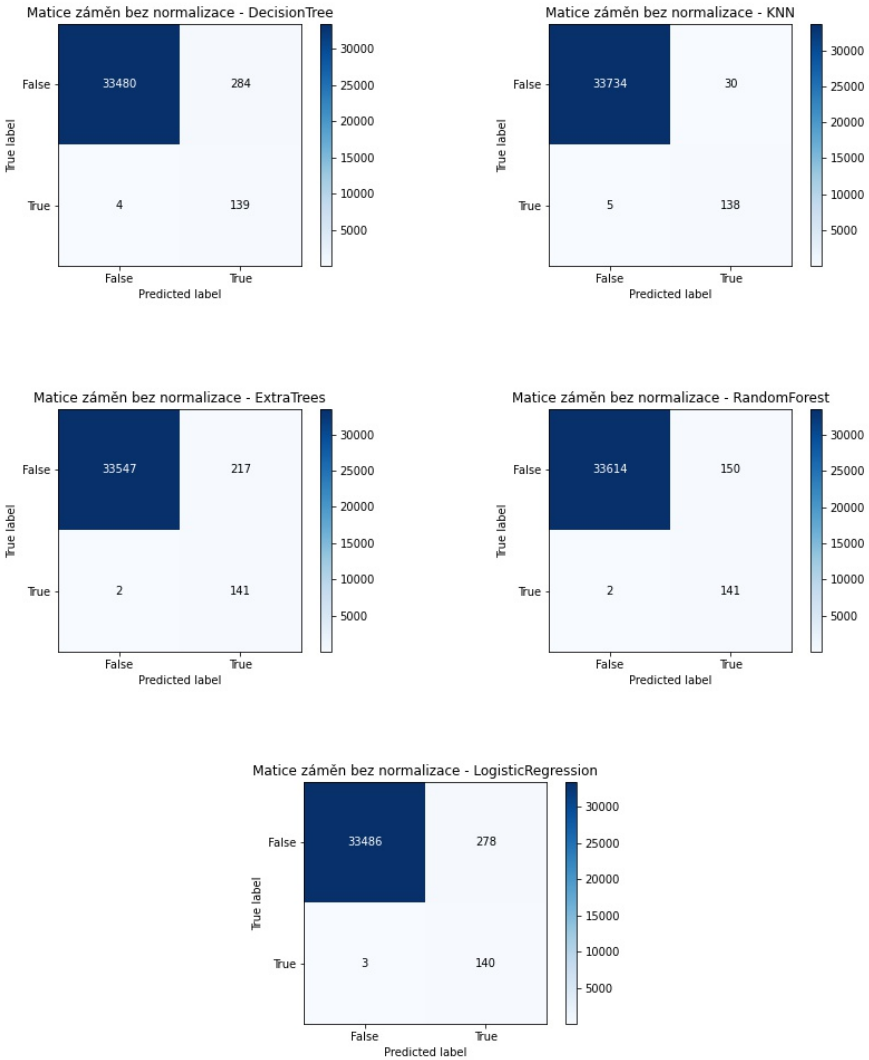
4.3.3 Snížení množství falešně pozitivních záznamů

Pokud má být klasifikátor nasazen v reálném provozu, je nezbytné uvažovat o množství falešně pozitivních záznamů. Pokud totiž půjde o síť, na níž je rozsáhlý provoz, a i kdyby klasifikátor na ní nasazený vykazoval podobně dobrou úspěšnost, jako při testování na využitých datových sadách, stejně by bylo velmi náročné kontrolovat záznamy označené jako obsahující tunely. A to z důvodu velkého objemu takových záznamů.

Proto byla v rámci této práce experimentováno i s funkcí klasifikátorů `predict_proba`. Ta místo pravdivostní hodnoty vrací hodnotu pravděpodobnostní (tedy $[0 - 1]$). Pak je možné tuto hodnotu převést pomocí *thresholdu* a

Tabulka 4.1: Nalezené nejlepší parametry ML algoritmů bez využití referenční sady (FP – ref je zkratka pro počet falešně pozitivní nálezy v sadě o 350 000 záznamech)

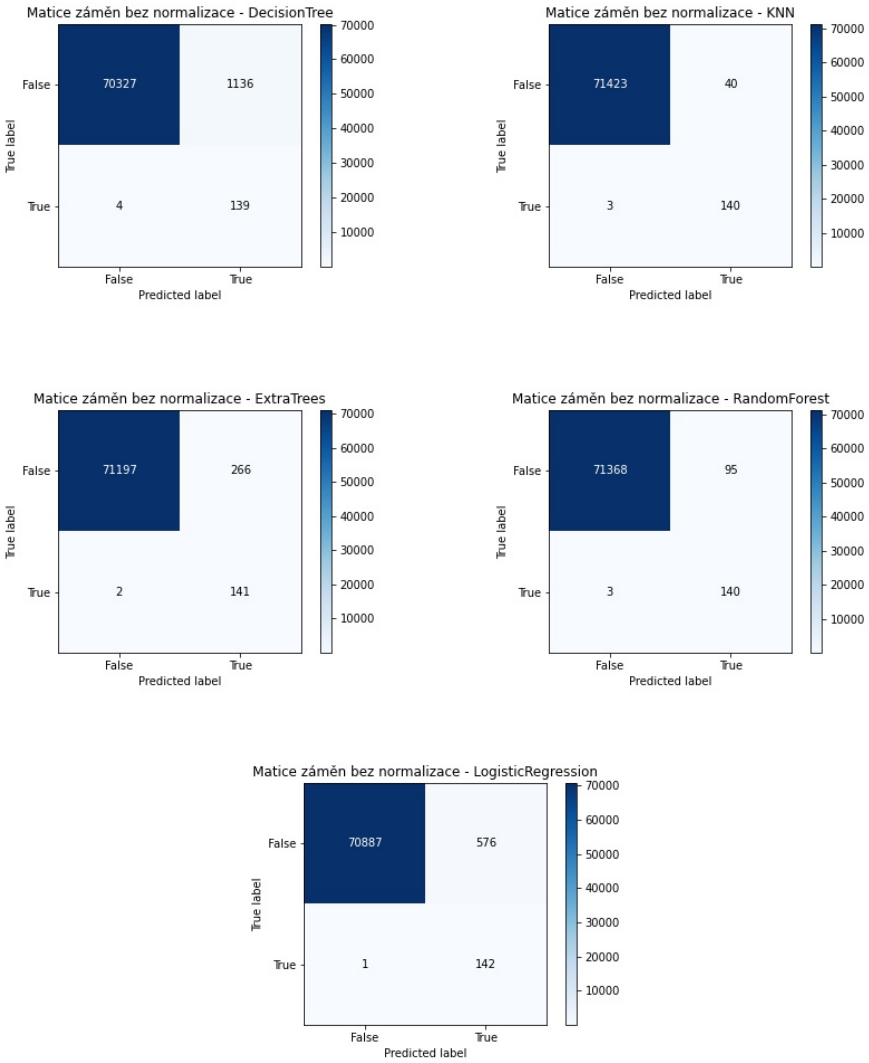
ExtraTrees	úspěšnost	0,99895
	FP – ref	8117
	kritérium	entropy
	max. hloubka	8
	max. features	auto
	min. vzorků v listu	0,1
	min. odděl. vzorků	2
	počet odhadů	7
RandomForest	úspěšnost	0,99875
	FP – ref	2644
	kritérium	entropy
	max. hloubka	5
	max. features	auto
	min. vzorků v listu	1
	min. odděl. vzorků	2
	počet odhadů	7
DecisionTree	úspěšnost	0,99263
	FP – ref	38 985
	kritérium	entropy
	max. hloubka	5
	max. features	auto
	min. vzorků v listu	1
	min. odděl. vzorků	2
	počet odhadů	7
LogisticRegression	úspěšnost	0,99710
	FP – ref	13 154
	c	1
	počet iterací	1000
KNN	úspěšnost	0,98934
	FP – ref	3117
	počet sousedů	6
	váhy	distance



Obrázek 4.1: Matice záměn u jednotlivých klasifikátorů při využití pouze vlastnoručně naměřených dat

Tabulka 4.2: Nalezené nejlepší parametry ML algoritmů s využitím referenční sady (FP – ref je zkratka pro počet falešně pozitivní nálezy v sadě o 1 318 482 záznamech), c je parametr regularizace

ExtraTrees	úspěšnost	0,998243%
	FP – ref	1281
	kritérium	entropy
	max. hloubka	9
	max. features	auto
	min. vzorků v listu	1
	min. odděl. vzorků	2
	počet stromů	7
RandomForest	úspěšnost	0,99914%
	FP – ref	574
	kritérium	entropy
	max. hloubka	5
	max. features	auto
	min. vzorků v listu	1
	min. odděl. vzorků	2
	počet stromů	7
DecisionTree	úspěšnost	0,99186%
	FP – ref	9691
	kritérium	entropy
	max. hloubka	5
	max. features	auto
	min. vzorků v listu	2
	min. odděl. vzorků	2
	počet odhadů	7
LogisticRegression	úspěšnost	0,99567%
	FP – ref	5281
	c	0,1
	počet iterací	1000
KNN	úspěšnost	0,98495%
	FP – ref	217
	počet sousedů	7
	váhy	distance

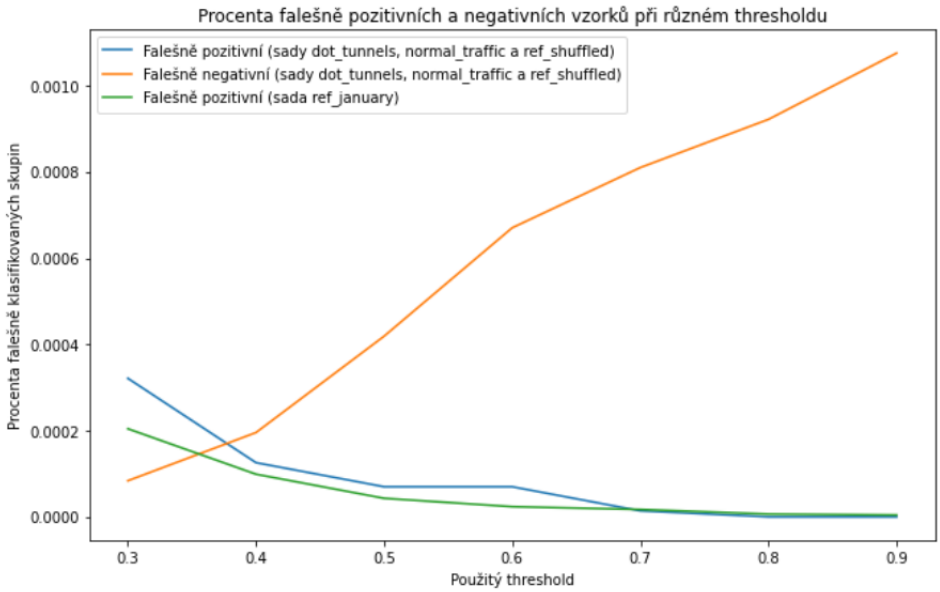


Obrázek 4.2: Matice záměn u jednotlivých klasifikátorů při využití části referenčních dat pro natrénování modelů ML

Tabulka 4.3: Rychlost klasifikace u prototypu trénovaného s využitím ref. sady v záznamech za sekundu. Klasifikace byla spuštěna v kontejnerizovaném JupyterHubu, který běžel na serveru s procesorem `Intel(R) Xeon(R) CPU E5-2609 v3 @ 1.90GHz`, 252 GB RAM a systémem Linux `Fedora 35`

ExtraTrees	378,92
RandomForest	379,93
DecisionTree	365,09
LogisticRegression	382,3
KNN	96,4

výrazu $P_x > threshold$. Pro Random Forest, který měl výborné výsledky, bylo proto zkoušeno více hranic (*thresholdů*). Výsledky lze vidět na grafu 4.3. Nastavení hranice si musí provést administrátor systému podle potřeb organizace a podle prostředků, které má k dispozici na vyhodnocení falešně pozitivních záznamů.



Obrázek 4.3: Změny počtu falešně pozitivních a negativních vzorků při změně thresholdu

4.4 Diskuze

Výsledky deterministického klasifikátoru se ukázaly jako daleko horší, než i jen základní verze algoritmů strojového učení. Proto autor prototyp deterministického klasifikátoru nedoporučuje k reálné detekci. Jeho základ, jímž jsou však některé specifické znaky DoT provozovatelů mají podle názoru autora potenciál stát se skutečným přínosem. Pro sestavení spolehlivého klasifikátoru na jejich bázi by ale bylo třeba tyto znaky ověřit na rozsáhlejší datové sadě a současně je doplnit o znaky další. Pak by detekce mohla být využitelná i pro praktické účely.

Jak již bylo zmíněno, z důvodu velké časové náročnosti sběru dat[9] (spolu s generováním probíhal déle, než 3 měsíce) nejsou výsledné datové sady obsahující DoT tunelovaný provoz příliš rozsáhlé. Z toho důvodu je třeba interpretovat výsledky ML klasifikátoru opatrně.

[9] Jako příklad lze uvést dvouhodinové generování dat u poskytovatele DoT `dismail`, kde byly vygenerovány pouhé nízké desítky záznamů o tocích — a bylo by jich daleko méně, kdyby exportér nedělil toky na části po pěti minutách.

Přesto však výsledky klasifikátorů vypadají velmi příznivě. A zejména, pokud by se podařilo před nasazením do konkrétní sítě získat vzorek „čistých“ dat přímo z té konkrétní sítě, bylo by možné dát klasifikátoru možnost se na těchto datech natrénovat. Vodítkem k tomu může být srovnání modelů strojového učení ve skupinách grafů 4.1 a 4.2. Procentuální změna úspěšnosti zde sice není tolik patrná z důvodu již výborné úspěšnosti bez tréninku na části referenční sady, zato je zřejmý rozdíl v absolutních počtech falešně pozitivních záznamů.

Důležitým poznatkem je také to, že se zdá, že u DecisionTree klasifikátoru pozorujeme přetrénování modelu. U ExtraTrees a RandomForest je chybovost na referenční sadě daleko nižší, jak i odpovídá teorii — kde modely ExtraTrees a RandomForest byly vytvořeny právě proto, aby se předcházelo přetrénování.

Překvapením se ukázal být model KNN. Na základě jeho principu existoval předpoklad, že bude jeho úspěšnost negativně ovlivněna výraznou nevyvážeností datových sad. Tyto předpoklady se však nepotvrdily, příčinou může být využité převzorkování a přeškálování pomocí knihovních funkcí `SMOTE` a `StandardScaler`. Delší doba výpočtu (viz tabulka 4.3) je vzhledem k rozdílné rozsáhlosti tréninkových sad pochopitelná.

Jako výhodné se ukázala možnost nastavení *thresholdu* (graf je zobrazení na obrázku 4.3). Díky možnosti přizpůsobení této konstanty je možné regulovat množství záznamů, které jsou klasifikovány jako falešně pozitivní. Organizace, která pak takový klasifikátor nasadí se může rozhodnout na základě dostupných prostředků, kolik bude detekovat falešně pozitivních vzorků. Za jejich nízké počty sice „zaplatí“ zvýšeným podílem vzorků falešně negativních, a velmi pravděpodobně i celkově nižší úspěšností detekce. Ale bude moci například manuálně analyzovat detekované vzorky, či omezí stížnosti klientů, pokud by aktivně blokovala legitimní provoz, který by byl klasifikován jako tunelování pomocí DoT.

Jedním z výstupů této práce je prototyp klasifikátoru založeného na strojovém učení realizovaný jako Python skript. Ten pracuje s CSV záznamy o datových tocích a na nich provádí detekci DoT tunelů. Tento prototyp využívá uložené natrénované klasifikátory. Při testování tohoto prototypu byly ale naměřeny významně horší výsledky (a to i méně, než 98 %). Důvodem může být neúplná kompatibilita verzí knihoven stojících za uložením a načtením modelů — na což poukazují rovněž zobrazená varování.

Závěr

DNS je jedním z protokolů, které stále nejsou šifrované. To představuje reálné riziko pro soukromí uživatelů. Se zaváděním šifrovaných alternativ jsou však spojeny některé komplikace a rizika. Tato práce si jako cíl vytkla vytvoření prozkoumání kvalitativních vlastností DoT resolverů a vytvoření prototypů klasifikátorů, které by byly schopné rozeznat tunelovaný DoT provoz od provozu bez tunelů.

V rámci této práce byla provedena analýza možností šifrování DNS provozu, tunelování pomocí DNS včetně různých scénářů provozu. Bylo vytvořeno stručné shrnutí možností monitoringu provozu s důrazem na monitoring datových toků. Nastudovány byly také základní principy metod strojového učení.

Podařilo se sestavit provozuschopnou měřicí sestavu, na které byly následně naměřeny kvalitativní charakteristiky DNS over TLS tunelů pro různé poskytovatele DoT a současně pro nástroje iodine a dns2tcp. Tyto výsledky byly vyhodnoceny a byla diskutována zjištění v kontextu administrace sítí a rozsáhlých informačních systémů.

Jedním z výstupů práce jsou také jak datové sady obsahující datové toky s DNS over TLS tunelovaným provozem, tak kontrolní sady generované na stejné síti. Na těchto sadách proběhla datová analýza a byly nalezeny zajímavé vlastnosti některých poskytovatelů DoT.

Byl vytvořen prototyp klasifikátoru jak ze zjištěných statistických vlastností tunelovaného provozu, tak i pomocí algoritmů strojového učení. Prototypy byly otestovány na vlastních i referenčních datových sadách. Dále byla rozvíjena a vylepšována úspěšnost těchto klasifikátorů. Výsledky byly vyhodnoceny a diskutovány v kontextu provozu na rozsáhlé síti.

Zadání této diplomové práce bylo splněno ve všech bodech. Osobně jsem rád, že jsem tuto práci mohl vypracovat. Nejen, že není ohledně DoT publikováno mnoho prací, ale především mě výzkum bavil a mohl jsem si významně rozšířit znalosti o datovou analýzu síťového provozu. Je plánováno, že hlavní části budou dále publikovány jako článek pro některou z vědeckých konferencí a doufám, že se práce stane základem pro navazující vědeckou činnost.

Bibliografie

1. GROTHOFF, Christian; WACHS, Matthias; ERMERT, Monika; APPELBAUM, Jacob. NSA's MORECOWBELL: knell for DNS. *Unpublished technical report*. 2017. Dostupné také z: `https://git.gnunet.org/bibliography.git/plain/docs/mcb-en.pdf`.
2. BRODKIN, Jon. *Senate votes to let ISPs sell your Web browsing history to advertisers*. 2022. Dostupné také z: `https://arstechnica.com/tech-policy/2017/03/senate-votes-to-let-isps-sell-your-web-browsing-history-to-advertisers/`.
3. HOFFMAN, Paul E.; MCMANUS, Patrick. *DNS Queries over HTTPS (DoH)* [RFC 8484]. RFC Editor, 2018. Request for Comments, č. 8484. Dostupné z DOI: `10.17487/RFC8484`.
4. *RFC 7858 - specification for DNS over Transport Layer Security (TLS)*. [B.r.]. Dostupné také z: `https://datatracker.ietf.org/doc/html/rfc7858`.
5. HUITEMA, Christian; DICKINSON, Sara; MANKIN, Allison. *DNS over Dedicated QUIC Connections*. Internet Engineering Task Force, 2022-02. Internet-Draft, draft-ietf-dprive-dnsoquic-10. Internet Engineering Task Force. Dostupné také z: `https://datatracker.ietf.org/doc/html/draft-ietf-dprive-dnsoquic-10`. Work in Progress.
6. *ASUSWRT: Asus česká republika*. [B.r.]. Dostupné také z: `https://www.asus.com/cz/Content/ASUSWRT/`.
7. *How to configure Parental Controls on the Wi-Fi Routers (case 1)?* [B.r.]. Dostupné také z: `https://www.tp-link.com/cz/support/faq/1531/`.
8. *Project DNSExfiltrator*. 2022. Dostupné také z: `https://github.com/Arno0x/DNSExfiltrator`.

9. HADDON, David A E; ALKHATEEB, Haider. Investigating Data Exfiltration in DNS Over HTTPS Queries. In: *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*. 2019, s. 212–212. Dostupné z DOI: 10.1109/ICGS3.2019.8688016.

10. *HTTPS encryption on the web*. 2022. Dostupné také z: https://transparencyreport.google.com/https/overview?hl=en.

11. *Domain names - implementation and specification* [RFC 1035]. RFC Editor, 1987. Request for Comments, č. 1035. Dostupné z DOI: 10.17487/RFC1035.

12. VEKSHIN, Dmitrii; HYNEK, Karel; CEJKA, Tomas. DoH Insight: Detecting DNS over HTTPS by Machine Learning. In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. Virtual Event, Ireland: Association for Computing Machinery, 2020. ARES '20. ISBN 9781450388337. Dostupné z DOI: 10.1145/3407023.3409192.

13. HYNEK, Karel; CEJKA, Tomas. Privacy Illusion: Beware of Unpadded DoH. In: *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2020, s. 0621–0628. Dostupné z DOI: 10.1109/IEMCON51383.2020.9284864.

14. *DNS-over-HTTPS (DoH) — Public DNS — Google Developers*. Google, [b.r.]. Dostupné také z: https://developers.google.com/speed/public-dns/docs/doh.

15. LU, Chaoyi; LIU, Baojun; LI, Zhou; HAO, Shuang; DUAN, Haixin; ZHANG, Mingming; LENG, Chunying; LIU, Ying; ZHANG, Zaifeng; WU, Jianping. An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come? In: 2019. ISBN 978-1-4503-6948-0. Dostupné z DOI: 10.1145/3355369.3355580.

16. *JSON API for DNS over HTTPS (DoH) — Public DNS — Google Developers*. Google, [b.r.]. Dostupné také z: https://developers.google.com/speed/public-dns/docs/doh/json.

17. *DNS Privacy Clients*. [B.r.]. Dostupné také z: https://dnsprivacy.org/dns_privacy_clients/.

18. NLNETLABS. *NLnetLabs/unbound: Unbound is a validating, recursive, and caching DNS resolver.* [B.r.]. Dostupné také z: https://github.com/NLnetLabs/unbound.

19. *Google Public DNS for your devices*. 2022. Dostupné také z: https://developers.google.com/speed/public-dns/docs/using#android.

20. *Project Stubby*. 2022. Dostupné také z: https://github.com/getdnsapi/stubby.

21. GARCÍA, Sebastián; HYNEK, Karel; VEKSHIN, Dmitrii; CEJKA, Tomas; WASICEK, Armin. *Large Scale Measurement on the Adoption of Encrypted DNS*. 2021.

22. CEJKA, Tomas; ROSA, Zdenek; KUBATOVA, Hana. Stream-wise detection of surreptitious traffic over DNS. In: *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. 2014, s. 300–304. Dostupné z DOI: 10.1109/CAMAD.2014.7033254.

23. WANG, Yue; ZHOU, Anmin; LIAO, Shan; ZHENG, Rongfeng; HU, Rong; ZHANG, Lei. A comprehensive survey on DNS tunnel detection. *Computer Networks*. 2021, roč. 197, s. 108322.

24. JOSEFSSON, Simon. *The Base16, Base32, and Base64 Data Encodings* [RFC 4648]. RFC Editor, 2006. Request for Comments, č. 4648. Dostupné z DOI: 10.17487/RFC4648.

25. YU, Bin; SMITH, Les; THREEFOOT, Mark; OLUMOFIN, Femi G. Behavior Analysis based DNS Tunneling Detection and Classification with Big Data Technologies. In: *IoTBD*. 2016.

26. KOREC, Martin. *Detekce malware pomocí identifikace periodického chování [online]*. 2018. Dostupné také z: https://theses.cz/id/mjm1at/. Diplomová práce. Masarykova univerzita, Fakulta informatiky Brno. SUPERVISOR: RNDr. Martin Drašar, Ph.D.

27. DIETRICH, Christian J.; ROSSOW, Christian; FREILING, Felix C.; BOS, Herbert; STEEN, Maarten van; POHLMANN, Norbert. On Botnets That Use DNS for Command and Control. In: *2011 Seventh European Conference on Computer Network Defense*. 2011, s. 9–16. Dostupné z DOI: 10.1109/EC2ND.2011.16.

28. GALOBARDES, Roger. *Learn how easy is to bypass firewalls using DNS tunneling (and also how to block it)*. 2022. Dostupné také z: https://medium.com/@galolbardes/learn-how-easy-is-to-bypass-firewalls-using-dns-tunneling-and-also-how-to-block-it-3ed652f4a000.

29. LIU, Chang; DAI, Liang; CUI, Wenjing; LIN, Tao. A Byte-level CNN Method to Detect DNS Tunnels. In: *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. 2019, s. 1–8. ISSN 2374-9628. Dostupné z DOI: 10.1109/IPCCC47392.2019.8958714.

30. WANG, Zheng. *Combating Malicious DNS Tunnel*. arXiv, 2016. Dostupné z DOI: 10.48550/ARXIV.1605.01401.

31. AHMED, Jawad; GHARAKHEILI, Hassan Habibi; RAZA, Qasim; RUSSELL, Craig; SIVARAMAN, Vijay. Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks. *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2019, s. 649–653.

32. LIU, Chang; DAI, Liang; CUI, Wenjing; LIN, Tao. A Byte-level CNN Method to Detect DNS Tunnels. In: *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*. 2019, s. 1–8. Dostupné z DOI: 10.1109/IPCCC47392.2019.8958714.

33. LIU, Jingkun; LI, Shuhao; ZHANG, Yongzheng; XIAO, Jun; CHANG, Peng; PENG, Chengwei. Detecting DNS Tunnel through Binary-Classification Based on Behavior Features. In: *2017 IEEE Trustcom/BigDataSE/ICESS*. 2017, s. 339–346. Dostupné z DOI: 10.1109/Trustcom/BigDataSE/ICESS.2017.256.

34. VAN HORENBEECK, Maarten. *Detection of DNS tunneling*. 2022. Dostupné také z: https://www.daemon.be/maarten/dnstunnel.html#detect.

35. ALMUSAWI, Ahmed; AMINTOOSI, Haleh. *DNS Tunneling Detection Method Based on Multilabel Support Vector Machine*. 2022. Dostupné také z: https://www.hindawi.com/journals/scn/2018/6137098/.

36. KARA, A. Mert; BINSALLEEH, Hamad; MANNAN, Mohammad; YOUSSEF, Amr; DEBBABI, Mourad. Detection of malicious payload distribution channels in DNS. 2014, s. 853–858. Dostupné z DOI: 10.1109/ICC.2014.6883426.

37. DO, Van; ENGELSTAD, Paal E.; FENG, Boning; THANH, Do. Detection of DNS Tunneling in Mobile Networks Using Machine Learning. In: 2017, s. 221–230. ISBN 978-981-10-4153-2. Dostupné z DOI: 10.1007/978-981-10-4154-9_26.

38. NUOJUA, Viivi; DAVID, Gil; HÄMÄLÄINEN, Timo. DNS Tunneling Detection Techniques – Classification, and Theoretical Comparison in Case of a Real APT Campaign. In: GALININA, Olga; ANDREEV, Sergey; BALANDIN, Sergey; KOUCHERYAVY, Yevgeni (ed.). *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Cham: Springer International Publishing, 2017, s. 280–291. ISBN 978-3-319-67380-6.

39. HOFSTEDE, Rick; ČELEDA, Pavel; TRAMMELL, Brian; DRAGO, Idilio; SADRE, Ramin; SPEROTTO, Anna; PRAS, Aiko. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials*. 2014, roč. 16, č. 4, s. 2037–2064. Dostupné z DOI: 10.1109/COMST.2014.2321898.

40. AITKEN, Paul; CLAISE, Benoît; TRAMMELL, Brian. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [RFC 7011]. RFC Editor, 2013. Request for Comments, č. 7011. Dostupné z DOI: 10.17487/RFC7011.

41. *Project IPFIXProbe - GitHub*. 2022. Dostupné také z: https://github.com/CESNET/ipfixprobe.

42. HSSINA, Badr; MERBOUHA, Abdelkarim; EZZIKOURI, Hanane; ERRITALI, Mohammed. A comparative study of decision tree ID3 and C4.5. *(IJACSA) International Journal of Advanced Computer Science and Applications*. 2014, roč. Special Issue on Advances in Vehicular Ad Hoc Networking and Applications. Dostupné z DOI: 10.14569/SpecialIssue.2014.040203.

43. BREIMAN, Leo; FRIEDMAN, Jerome H; OLSHEN, Richard A; STONE, Charles J. *Classification and regression trees*. Routledge, 2017.

44. *Decision Tree - Overview*. 2022. Dostupné také z: https://corporate%5C-financeinstitute.com/resources/knowledge/other/decision-tree/.

45. PETERSON, L. E. K-nearest neighbor. *Scholarpedia*. 2009, roč. 4, č. 2, s. 1883. Dostupné z DOI: 10.4249/scholarpedia.1883. revision #137311.

46. PAL, M. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*. 2005, roč. 26, č. 1, s. 217–222. Dostupné z DOI: 10.1080/01431160412331269698.

47. BREIMAN, Leo. *1 RANDOM FORESTS–RANDOM FEATURES*. 1999.

48. GEURTS, Pierre; ERNST, Damien; WEHENKEL, Louis. Extremely randomized trees. *Machine learning*. 2006, roč. 63, č. 1, s. 3–42.

49. *Multiple Logistic Regression Explained*. 2022. Dostupné také z: https://mlcorner.com/multiple-logistic-regression-explained-for-machine-learning/.

50. *A Short Introduction – Logistic Regression Algorithm*. 2022. Dostupné také z: https://helloacm.com/a-short-introduction-logistic-regression-algorithm/.

51. *Welcome to the OpenWrt Project*. 2022. Dostupné také z: https://openwrt.org.

52. *Project dns2tcp - GitHub*. 2022. Dostupné také z: https://github.com/alex-sector/dns2tcp.

53. *Project dnscat2 - GitHub*. 2022. Dostupné také z: https://github.com/iagox86/dnscat2.

54. *DNS over TLS Resolvers*. 2022. Dostupné také z: https://dnsprivacy.org/public_resolvers/#dns-over-tls-dot.

55. *Project Jupyter*. 2022. Dostupné také z: https://jupyter.org.

56. TURČIČOVÁ, Marie. *Intervaly spolehlivosti*. 2022. Dostupné také z: https://www2.karlin.mff.cuni.cz/~turcic/Konfidencni_intervaly.pdf.

57. DANIEL, Uhříček. *Detekce IoT malware v počítačových sítích*. 2021. Dipl. pr. České vysoké učení technické v Praze. Vypočetní a informační centrum.

58. *SMOTE - Imbalanced Learn Documentation*. 2022. Dostupné také z: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html.

59. LASHKARI, Arash Habibi. *Ahlashkari/CICFlowMeter: CICFlowmeter-v4.0 (formerly known as iscxflowmeter) is an ethernet traffic bi-flow generator and analyzer for anomaly detection that has been used in many cybersecurity datsets such as Android adware-general malware dataset (CICAAGM2017), IPS/IDS Dataset (CICIDS2017), Android Malware Dataset (CICANDMAL2017) and distributed denial of service (CICDDOS2019)*. [B.r.]. Dostupné také z: https://github.com/ahlashkari/CICFlowMeter.

Příloha A

Seznam použitých zkratek

AV Antivirus

C&C Command and Control

DoH DNS over HTTPS

DNS Domain Name System

DoT DNS over TLS

DoQ DNS over QUIC

JSON JavaScript Object Notation

ICMP Internet Control Message Protocol

IoC Indicator of Compromise

IoT Internet of Things

IP Internet Protocol

IPFIX IP Flow Information Export

ISP Internet Service Provider

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

MS Microsoft

NAT Network Address Translation

OS Operating System

QUIC Quick UDP Internet Connections

RTT Run Trip Time

SSH Secure Shell

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP Transmission Control Protocol

VPN Virtual Private Network

Příloha **B**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `datasets` ..... vytvořené datové sady
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS