FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Softwarový modul pro rozpoznání VPN v síťovém provozu
Student:	Bc. Martin Čtrnáctý
Vedoucí:	Ing. Tomáš Čejka, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačové systémy a sítě
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Seznamte se s problematikou monitorování síťového provozu na úrovni paketů a tzv. síťových toků.
Nasbírejte vzorky provozu vybraných Virtual Private Network (VPN) nástrojů (např. OpenVPN, Cisco AnyConnect) v podobě zachycených paketů a rozšířených síťových toků.
Proveďte analýzu zachyceného provozu a zaměřte se na charakteristické vlastnosti, které je možné využít pro identifikaci VPN spojení.
Navrhněte algoritmus pro detekování VPN spojení v síťovém provozu.
Dle návrhu vytvořte softwarový prototyp, který je schopen zpracovávat provoz z reálné sítě.
Navržené řešení otestuje, zaměřte se na přesnost rozpoznání a četnost falešně pozitivních výsledků.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Pavel Tvrdík, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 17. prosince 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Softwarový modul pro rozpoznání VPN v síťovém provozu

Bc. Martin Čtrnáctý

Katedra počítačových systémů
Vedoucí práce: Ing. Tomáš Čejka, Ph.D.

28. května 2020

Poděkování

Děkuji veducímu práce Ing. Tomáši Čejkovi, Ph.D. za odborné rady a metodické vedení diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 28. května 2020 . . . . . .

Odkaz na tuto práci

Čtrnáctý, Martin. *Softwarový modul pro rozpoznání VPN v síťovém provozu.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato diplomová práce se zabývá možnostmi detekce VPN v síťovém provozu, neboť VPN lze zneužít k obcházení kontroly provozu či exfiltraci dat. Práce na základě zachycených vzorků VPN komunikace navrhuje a testuje různé metody detekce VPN. Následně dané detekční metody implementuje, diskutuje jejich výhody a nevýhody a požadavky na jejich správné fungování. Nakonec testuje schopnosti detekce jednotlivých detekčních metod a porovnává je.

Klíčová slova VPN, monitoring, detekce, IP toky, OpenVPN, DPI

Abstract

This thesis deals with the possibilities of VPN detection in a network traffic because VPN can be misused to bypass traffic control or for data exfiltration. Based on captured samples of VPN communication, the thesis designs and tests various VPN detection methods. The thesis then implements the detection methods, discusses their advantages and disadvantages and the requirements for their proper functioning. Finally, the thesis tests the detection capabilities of individual detection methods and compares them.

Keywords VPN, monitoring, detection, IP flow, OpenVPN, DPI

Obsah

Úvod **1**

1 Rešeršní část **3**
1.1 VPN 3
1.2 IP toky 6
1.3 Nástroje 7
1.4 Charakteristika VPN provozu 7
1.5 OpenVPN 11
1.6 Existující řešení 18

2 Analýza a návrh **21**
2.1 Testovací data 21
2.2 Detekce OpenVPN analýzou paketů 23
2.3 Detekce AnyConnect analýzou paketů 25
2.4 Detekce s využitím IP toků z Cisco Joy 26
2.5 Detekce s využitím IP toků z NEMEA modulu 31
2.6 Metody hodnocení 32
2.7 Detekce s využitím strojového učení 33

3 Realizace **37**
3.1 Detekce OpenVPN analýzou paketů 37
3.2 Detekce AnyConnect analýzou paketů 39
3.3 Detekce s využitím IP toků z Cisco Joy 39
3.4 Detekce s využitím IP toků z NEMEA modulu 42
3.5 Detekce s využitím strojového učení 43

4 Testování a výsledky **45**
4.1 Detekce OpenVPN analýzou paketů 45
4.2 Detekce AnyConnect analýzou paketů 46
4.3 Detekce s využitím IP toků z Cisco Joy 46

4.4 Detekce s využitím IP toků z NEMEA modulu 48
4.5 Detekce s využitím strojového učení 48
4.6 Porovnání jednotlivých metod 49

Závěr **51**

Literatura **53**

A Seznam použitých zkratek **55**

B Obsah přiloženého CD **57**

Seznam obrázků

1.1 Schéma komunikace bez VPN a s využitím VPN 4
1.2 Navazování spojení OpenVPN 17

2.1 Schéma detekce OpenVPN 24
2.2 Schéma detekce AnyConnect 25

3.1 Komunikace mezi moduly při použití jednosměrných IP toků . . . 42
3.2 Komunikace mezi moduly při použití obousměrných IP toků 43

Seznam tabulek

1.1 OpenVPN zpráva typu *p_control* 13
1.2 OpenVPN zpráva typu *p_data* 14
1.3 TLS první část 15

2.1 Úspěšnost detekce v závislosti na časové agregaci 27
2.2 Porovnání charakteristik jednosměrných a obousměrných IP toků . 28
2.3 Napočítané charakteristiky VPN vs. běžný provoz Cisco Joy 31
2.4 Napočítané charakteristiky VPN vs. běžný provoz NEMEA 32

4.1 Testování Cisco Joy 47
4.2 Testování NEMEA 48
4.3 Testování strojového učení 1 49
4.4 Testování strojového učení 2 49

Úvod

VPN umožňuje zapouzdření dat a jejich zaslání šifrovaným kanálem přes síť do cílové destinace. Toho lze využít k legitimním účelům, jako například zabezpečení připojení v sítích, kde může existovat riziko podvržení, zabezpečení vzdáleného přístupu na prostředky v lokální síti nebo propojení dvou lokálních sítí přes jinou nezabezpečenou síť. Existuje ovšem i možnost využít VPN k účelům, které mohou být považovány z pohledu správce sítě za nežádoucí, jako například obcházení blokace stránek, obcházení kontroly provozu, exfiltrace dat či šíření nelegálního obsahu.

Práce se proto zabývá možnostmi detekce VPN v síťovém provozu. Zkoumá a analyzuje vlastnosti VPN protokolů a charakteristiky samotného VPN provozu za účelem navržení vhodných detekčních mechanismů, k čemuž využívá nasbírané vzorky dat. Na základě analýzy je navrženo, implementováno a otestováno pět různých detekčních programů využívajících různé způsoby detekce.

Prvním používaným detekčním způsobem je detekce na základě analýzy jednotlivých paketů, kdy se detektor snaží odhalit inicializaci VPN spojení. Tento způsob detekce používají dva vyvinuté detekční programy. Jeden je zaměřený na protokol OpenVPN, druhý na protokol Cisco AnyConnect.

Druhým používaným detekčním způsobem je agregace charakteristik provozu z IP toků a jejich porovnávání s charakteristikami VPN provozu. Pro tento způsob detekce je potřeba velké množství vhodně klasifikovaných testovacích dat, ze kterých bude možné charakteristiky VPN provozu odvodit. Detekci pomocí charakteristik využívají tři detekční programy. Jeden používá rozšířené IP toky z exportéru Cisco Joy, druhý používá základní IP toky z exportéru z NEMEA systému. Třetí program používá rozšířené IP toky z Cisco Joy, ale k detekci využívá rozhodovací metodu naučenou pomocí strojového učení.

Cílem práce je vyhodnocení a porovnání různých metod detekce VPN, proto práce diskutuje výhody a nevýhody těchto metod, stejně jako požadavky na jejich správné fungování. Na závěr práce je provedeno testování schopnosti detekce detekčních programů a porovnání jednotlivých metod. Přínosem této diplomové práce je i detailní analýza komunikace známých VPN nástrojů na paketové úrovni jejich komunikačních protokolů.

Kapitola 1

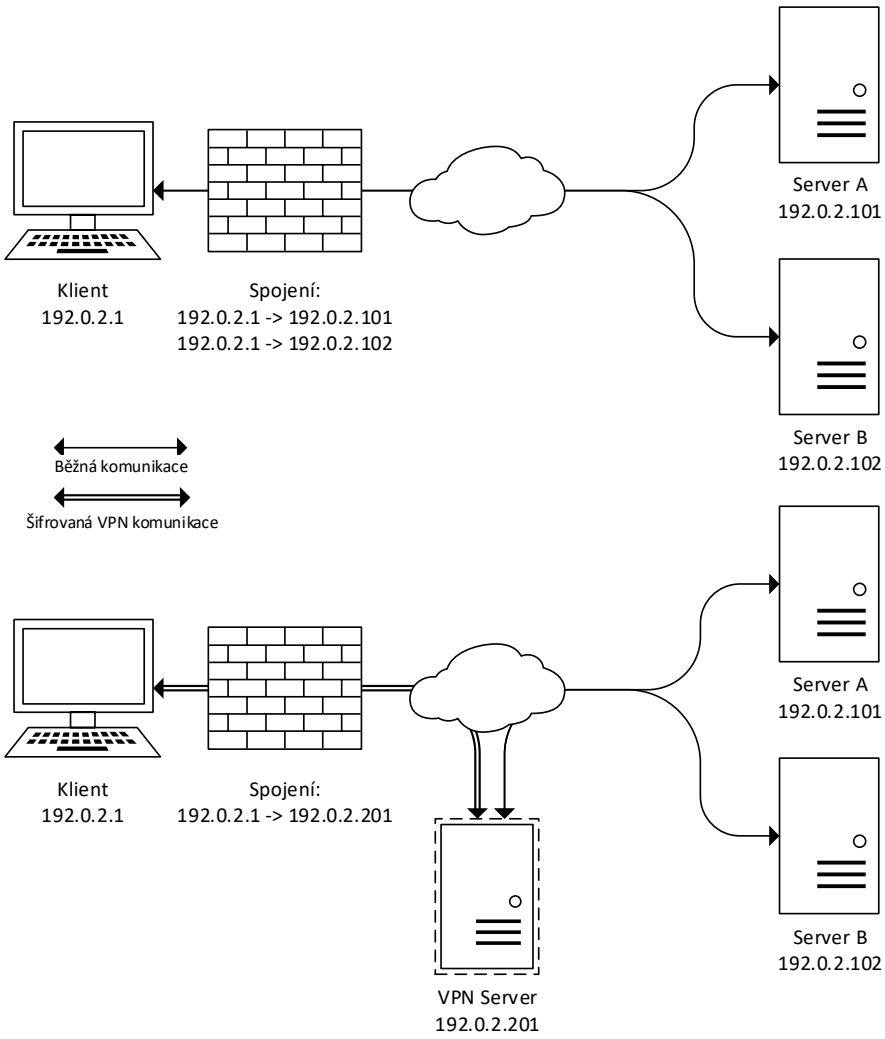
Rešeršní část

1.1 VPN

VPN neboli virtuální privátní síť umožňuje zapouzdření dat a jejich zaslání šifrovaným kanálem přes síť do cílové destinace. Příkladem může být paket směřující z počítače jedna na počítač dva, který je přímo na počítači jedna nebo na jiném síťovém prvku po cestě zašifrován a zapouzdřen do protokolu vyšší vrstvy. Takto pokračuje až do bodu, kde je opět dešifrován a poslán jako běžný provoz. Bodem, kde je komunikace dešifrována, může být cílový počítač nebo jiný síťový prvek po cestě paketu. Dle potřeby je možné přes tento tunel směrovat všechnu nebo pouze část komunikace, stejně jako lze šifrovat komunikaci pouze mezi dvěma počítači nebo několika podsítěmi na obou stranách, které mohou obsahovat mnoho počítačů. Rozdíl mezi komunikací s VPN a běžným provozem ilustruje Obrázek 1.1.

1.1.1 Využití

Možností pro využití VPN je několik. Vzhledem k tomu, že veškerá komunikace procházející přes VPN tunel je šifrována, lze ji využít pro zabezpečení připojení v sítích, kde může existovat riziko podvržení nebo odposlechnutí komunikace. Například uživatel připojený k veřejné WiFi, který potřebuje nahrát důvěrný soubor na FTP úložiště. Spustí VPN a vytvoří šifrovaný tunel na server, odkud je již komunikace k serveru bezpečná. Uživatel potom všechna data posílá tímto šifrovaným tunelem a jeho komunikace nemůže být podvržena nebo odposlechnuta. Tento model je možné a vhodné využít například i u firemních zařízení, která je potřeba používat mimo firemní síť. Lze je nastavit tak, že ihned po zapnutí se připojí šifrovaným VPN tunelem k firemnímu serveru a jakákoliv jiná komunikace bude blokována. Tím lze zabránit podvržení nebo odposlechnutí komunikace na lokální síti a zároveň veškerá komunikace může procházet firemním firewallem, takže může být kontrolována, i když je zařízení připojené přes jinou než firemní síť. Tím je možné zabránit například



Obrázek 1.1: Schéma komunikace bez VPN a s využitím VPN

odposlechnutí komunikace po nešifrovaném protokolu nebo podvržení DNS záznamu.

VPN lze využít také ke skrytí komunikace před ISP. Pokud uživatel vytvoří VPN tunel na vzdálený server a bude do tohoto tunelu směrovat veškerou svoji komunikaci, bude komunikace z pohledu navštívených serverů pocházet ze vzdáleného serveru, a nikoliv od klienta. Tím pádem je VPN tunel možné využít také k tomu, aby komunikace vypadala, jako by přicházela z IP adresy příslušící jinému státu, čímž je možné obejít geoblokaci určitých služeb. Protože je komunikace šifrována, je možné zneužít tunel také jako způsob, jakým zabránit správci sítě získávat informace o tom, kam a jaká data posíláme.

Další možností využití je vzdálený zabezpečený přístup na prostředky v lokální síti, které nechceme vystavovat do internetu. Například administrátor potřebuje mít přístup z domova do management rozhraní serveru či obchodník potřebuje u klienta přístup k firemnímu datovému úložišti. V takovém případě uživatel spuštěním VPN vytvoří šifrovaný tunel do firemní sítě a poté může přistupovat na prostředky firemní sítě, jako kdyby byl do této sítě přímo připojený.

VPN se také využívá k propojení dvou lokálních sítí přes internet nebo jinou nezabezpečenou síť. Jako příklad můžeme uvést dvě pobočky, u nichž chceme, aby zaměstnanci jedné pobočky mohli přistupovat na lokální prostředky druhé pobočky. V takovém případě se vytvoří VPN tunel, který bude šifrovat data mezi těmito pobočkami. Tento typ VPN bývá označován jako site-to-site VPN.

VPN tedy zajišťuje důvěrnost šifrováním informací tak, že nikdo nepovolaný nemůže komunikaci přečíst. Šifrováním a ověřováním také znemožňuje komunikaci podvrhnout, čímž zajišťuje integritu komunikace. Použitím VPN lze také zpřístupnit například lokální zdroje určité sítě, takže zajišťuje i dostupnost.

1.1.2 Možnosti zneužití

Vzhledem k technické podstatě VPN je možné ji využít i k účelům, které by mohly být z pohledu správce sítě považovány za nežádoucí. Prvním důvodem může být obcházení bezpečnostních politik. Pokud například zaměstnavatel blokuje přístup na některé weby, které nesouvisí s výkonem práce, pak je možné použitím VPN přístup na takovéto weby získat.

Druhým důvodem může být obcházení kontroly provozu. Pokud správce sítě kontroluje, jaký provoz přichází a odchází ze sítě, použitím VPN je možné výrazně snížit schopnost správce sítě rozpoznat, jaký provoz VPN spojením prochází a kam míří. Toho lze zneužít například k exfiltraci dat, k čemuž bývají často zneužívány i jiné metody než VPN tunely [1], [2].

1.2 IP toky

Vzhledem k tomu, že analýza jednotlivých paketů je velmi náročná jak na výkon, tak na kapacitu paměti, používají se k analýze síťového provozu i jiné přístupy. Jedním z nich jsou takzvané IP toky neboli flow, které vytváří agregované statistické údaje o zachyceném provozu na síti [3]. Tyto IP toky je pak možné analyzovat a skladovat snadněji než kompletní záznam paketového provozu na síti. IP toky jsou definovány jako sekvence paketů se shodnou pěticí údajů: zdrojová IP adresa a port, cílová IP adresa a port a číslo protokolu. IP tok potom obsahuje údaje o času prvního a posledního paketu, počtu přenesených paketů a bajtů a další údaje v závislosti na implementaci exportéru IP toků.

V souvislosti s IP toky rozlišujeme také takzvaný aktivní a neaktivní timeout. Neaktivní timeout určuje, po jaké době, kdy neproběhne komunikace v rámci jednoho pěticí parametrů definovaného spojení, dojde k uzavření IP toku. Při další komunikaci mezi danou pěticí již bude vytvořen nový IP tok. Aktivní timeout potom určuje, po jaké době dojde k uzavření IP toku, přestože komunikace v rámci jednoho pěticí parametrů definovaného spojení stále probíhá. Aktivní timeout je potřeba pro to, aby se informace o spojení mohly dostávat do programů, které je zpracovávají dříve, než je celé spojení dokončeno. Standardně bývá používáno pět minut pro aktivní timeout a půl minuty pro neaktivní timeout.

Cisco Joy

Cisco Joy je volně dostupný software pro exportování IP toků ze síťové komunikace [4]. Jeho vstupem může být buď přímo síťové rozhraní, na kterém bude zachytávat právě probíhající komunikaci, nebo soubor s již zachycenou komunikací. Výstupem je pak soubor v JSON formátu obsahující agregované informace o síťovém toku. Kromě běžných informací, jako je počet paketů, jejich velikosti a čas komunikace, umí Cisco Joy vyexportovat také takzvané rozšířené IP toky, které obsahují navíc informace o DNS dotazech, TLS spojeních, entropii dat, TCP příznacích a další.

NEMEA

NEMEA je modulární systém pro analýzu síťového provozu založený na zpracovávání IP toků [5]. Skládá se z mnoha nezávislých modulů, které jsou propojeny přes komunikační rozhraní, a každý z modulů má svůj vlastní úkol. Komunikace mezi moduly se provádí předáváním zpráv, kde zprávy obsahují IP toky, výstrahy, statistiky nebo předzpracovaná data.

NEMEA obsahuje také vlastní exportér IP toků nazvaný *flow_meter*, který dokáže generovat IP toky ze souboru obsahujícího zachycený provoz nebo zachytávat provoz na síťovém rozhraní. Exportér IP toků je ve formě jednoho

z modulů, takže vygenerované IP toky jsou dále předávány pomocí výstupního komunikačního rozhraní.

1.3 Nástroje

V rámci práce byl používán modul pro strojové učení scikit-learn a knihovna pro zachytávání paketů libpcap.

Scikit-learn

Scikit-learn je Python modul pro strojové učení postavený na knihovně SciPy a je distribuován na základě 3-bodové BSD licence [6]. Projekt vznikl v roce 2007 jako projekt Google Summer of Code a v současné době je udržován týmem dobrovolníků. Jednou z nabízených možností je klasifikace objektu pomocí binárního rozhodovacího stromu, která je v práci testována jako způsob možné detekce VPN.

Strojové učení může být pro detekci v síti velmi užitečné [7], je k němu však potřeba velké množství dobře klasifikovaných trénovacích dat, protože jinak nemůže podávat dobré výsledky.

Libpcap

Knihovna libpcap poskytuje rozhraní pro monitorování sítě na úrovni paketů [8]. Umožňuje sběr síťových statistik, monitorování zabezpečení, ladění sítě atd. V práci je použita k zachytávání síťového provozu na síťových rozhraních a ke zpracovávání již zachycených dat v podobě PCAP souborů.

V práci byla tato knihovna využita pro zachytávání paketů v implementacích detektorů, které zpracovávají jednotlivé pakety.

1.4 Charakteristika VPN provozu

VPN provoz je vzhledem ke své povaze určitým způsobem charakteristický. Část práce se proto bude snažit odhalovat VPN provoz na základě jeho charakteristických vlastností, které se liší od standardního provozu.

Tyto charakteristiky budou počítány z IP toků pro každou IP adresu, která se zúčastnila komunikace. To například znamená, že pokud se přenese deset paketů z IP adresy A na IP adresu B, je potřeba započítat těchto deset paketů jak do charakteristik IP adresy A, tak do charakteristik IP adresy B.

Pro účely detekce je také potřeba mít pokud možno všechnu komunikaci, která s danou IP adresou proběhla. Nelze detekovat VPN na základě charakteristiky provozu, pokud máme pouze zlomek z celkového provozu dané IP adresy. Z toho vyplývá, že je potřeba získávat data z provozu na hranici lokální sítě a omezit množinu IP adres, pro kterou jsou charakteristiky počítány, na IP adresy z lokální sítě. Dalším důvodem pro detekci na hranici lokální sítě

je překlad adres. Pokud budeme detekovat VPN na základě charakteristiky provozu až za překladem adres, nedokážeme rozlišit provoz podle jednotlivých uživatelů a jasné charakteristiky VPN provozu jednoho uživatele se schovají do běžného provozu ostatních uživatelů.

Průměrný počet paketů v IP toku

Počet paketů v IP toku je závislý na tom, jak moc se v časovém úseku daného IP toku mezi pěticí zdrojová IP adresa a port, cílová IP adresa a port a číslo protokolu komunikovalo. Běžný uživatel například při načtení webové stránky nejdříve pošle DNS dotaz na DNS server, potom položí požadavek na server s webovou stránkou, zjistí, že stránka využívá obsah z jiného serveru, a tak položí dotaz na další server, jehož obsah je ve webové stránce. Poté uživatel klepne na odkaz na jinou stránku, který ho přesměruje opět na jiný server, ten načítá obsah z dalšího serveru atd. To znamená, že většinou je použito mnoho IP toků s menším množstvím paketů, na rozdíl od komunikace s využitím VPN, kde je veškerý provoz zabalen a poslán tunelem, takže na síti je vidět pouze komunikace s VPN serverem. Zároveň je potřeba neustále spojení udržovat a kontrolovat, i když se zrovna žádná data nepřenáší. To ve výsledku znamená, že komunikace za použití VPN generuje zpravidla výrazně více paketů v rámci jednoho IP toku, než je běžné bez použití VPN.

Počet unikátních portů

Počet unikátních čísel portů, které jsou v rámci komunikace s určitou IP adresou použity, jsou odlišné v situaci, kdy je VPN použita, od situace, kdy VPN použita není. Ve většině případů, kdy operační systém vytváří nové spojení, vygeneruje nové číslo zdrojového portu.

Pokud VPN není aktivní, znamená to, že pro každý požadavek každé aplikace je použito nové číslo zdrojového portu. Pokud VPN aktivní je, operační systém sice stále pro každé nové spojení vygeneruje nový zdrojový port, ale poté je komunikace zabalena do tunelu, takže se vznik nového zdrojového portu na zachycené VPN komunikaci neprojeví.

Počet unikátních IP adres

Při standardní komunikaci komunikuje většinou počítač s větším množstvím IP adres, a to například s velkým množstvím různých webových serverů, serverem pro odchozí a příchozí poštu, serverem pro aktualizace, NTP serverem, servery pro odesílání různých dat o využití atd. Při použití VPN se komunikuje zpravidla pouze s VPN serverem, takže je množství unikátních IP adres, se kterými sledovaná IP adresa komunikovala, výrazně nižší než bez použití VPN.

DNS dotazy

Pro překlad DNS názvů na IP adresy se používá DNS dotazů. Ty jsou proto běžnou součástí komunikace, například při procházení webu. V případě použití VPN jsou pak tyto dotazy zabaleny do tunelu, takže pro monitorovací systém nejsou pozorovatelné. Problémem však může být použití interního DNS serveru (typicky ve firmách), kdy se stanice dotazují interního serveru, který se v případě, že nezná odpověď, doptá do internetu. Výsledkem potom je, že všechny DNS dotazy odchází do internetu právě z tohoto interního DNS serveru a nejsou součástí komunikace do internetu jednotlivých stanic.

Průměrný čas jednoho IP toku

Při běžné komunikaci bez použití VPN převažují krátké IP toky. Většina komunikace, jako je DNS dotaz, načtení webové stránky, stažení pošty atd., trvá obvykle maximálně několik vteřin. Při běžném používání bez VPN mají IP toky většinou krátkou dobu trvání. Při použití VPN naopak trvá IP tok obvykle výrazně déle, protože spojení VPN je udržováno po dlouhou dobu a prochází jím veškerá komunikace z daného počítače.

Počet SYN flags

Při použití protokolu TCP se pro vytvoření nového spojení odesílá paket s příznakem SYN. Tento příznak lze jednoduše agregovat do IP toků. Otevírání nových TCP spojení se děje při běžné komunikaci bez použití VPN velmi často, avšak při použití VPN je provoz zabalen do tunelu, takže takovéto pakety na síti nejsou časté. To znamená, že bez použití VPN lze těchto paketů zachytit výrazně větší množství než při použití VPN.

Průměrná entropie

Vyšší entropie neboli míra neuspořádanosti dat v paketech může naznačovat použití šifrování. Ačkoliv i data bez použití VPN mohou být šifrována, při použití VPN jsou šifrována vždy.

Počet TLS spojení

TLS je protokol používaný v mnoha aplikacích pro šifrovanou komunikaci a díky tomu, že je známa jeho implementace, lze ho v síťovém provozu jednoduše detekovat. Dnes už ho používá většina webových stránek pro zabezpečení komunikace klienta se serverem, což znamená, že v běžné komunikaci bez použití VPN se objevuje velké množství těchto nově navazovaných TLS spojení. V případě použití VPN je veškerý provoz tunelován a počet navazovaných TLS spojení detekovatelných na síti je nízký nebo nulový.

OpenVPN *data*

Protože OpenVPN je jedna z velmi často používaných implementací VPN, je vhodné snažit se detekovat i konkrétně tento protokol. Každý paket OpenVPN obsahuje nezašifrovanou hodnotu, takzvaný opcode, která definuje typ zprávy protokolu. Může být proto počítána statistika, kolik procentuálně zpráv mělo na pozici, která je v případě OpenVPN vyhrazena pro opcode, hodnotu označující zprávu typu *data*. Zpráva typu *data* je nejčastěji posílaná zpráva protokolu OpenVPN, a proto je vysoké procento paketů, které vypadají jako OpenVPN zpráva typu *data*, znakem, že zachytávaný provoz může být OpenVPN provoz.

Poměry komunikace s jednotlivými IP adresami

Jelikož se při provozu tunelovaném přes VPN komunikuje primárně s VPN serverem, znamená to, že většina komunikace probíhá pouze s jednou IP adresou. Při běžné komunikaci bez použití VPN naopak komunikace typicky probíhá na velké množství různých IP adres, protože komunikuje mnoho aplikací s mnoha zdroji. Pokud tedy spočítáme, kolik procent komunikace proběhlo s jakou IP adresou, bude v případě použití VPN jedna z IP adres inklinovat k tomu, že s ní proběhla většina komunikace, a ostatní se budou blížit nule, zatímco při běžném provozu bez VPN bude provoz rozdistribuován mezi jednotlivé IP adresy rovnoměrněji.

Walsh-Hadamardova transformace

Walsh-Hadamardova transformace umožňuje odhalit určitou periodicitu v datech. Vzhledem k tomu, že u VPN lze očekávat používání stále stejných protokolů nižších vrstev, podobné hlavičky většiny paketů a opakující se pakety kontrolující spojení, budou pakety VPN vykazovat větší podobnost než pakety běžného provozu bez VPN.

Ukázka implemetace Walsh-Hadamardovy transformace v Cisco Joy [9]. Pro každé 4 bity je napočítáno spektrum, které je na konci výpočtu zprůměrováno.

```
wht_process_four_bytes (wht_t *wht, const uint8_t *d) {
x[0] = d[0] + d[2];
x[1] = d[1] + d[3];
x[2] = d[0] - d[2];
x[3] = d[1] - d[3];
wht->spectrum[0] += (x[0] + x[1]);
wht->spectrum[1] += (x[0] - x[1]);
wht->spectrum[2] += (x[2] + x[3]);
wht->spectrum[3] += (x[2] - x[3]);
}
```

Kontrola vůči blacklistu

K detekci VPN je možné využít blacklist obsahující seznam IP adres VPN serverů a porovnávání komunikujících IP adres s tímto blacklistem. Taková detekce je výrazně ovlivněna kvalitou a aktuálností daného blacklistu. Výhodu je, že v případě kvalitních dat se jedná o velmi dobrý indikátor schopný detekovat velké procento VPN spojení. Nevýhodou je, že sehnat takový seznam není jednoduché, navíc na spoustě IP adres může běžet více služeb, nejen VPN server.

Vždy si také může kdokoliv zprovoznit svůj vlastní VPN server na krátkou dobu a pouze na jedno použití, kde je potom jeho možnost zachycení blacklistem téměř nulová. V práci byl využit volně dostupný blacklist spravovaný komunitou [10].

Reverzní DNS záznamy

Použití VPN lze určit i tak, že budeme předpokládat, že IP adresa, na které DNS server běží, má v názvu řetězec „vpn“. Je tedy možné pokládat reverzní DNS dotazy na IP adresy, které se účastnily komunikace, a vyhledávat v nich tento řetězec. Problémem je delší doba potřebná na provedení reverzního DNS dotazu a také možnost falešných detekcí, například webů, které mají řetězec „vpn“ v názvu.

1.5 OpenVPN

OpenVPN je velmi populární software, který umožňuje vytváření šifrovaných VPN spojení. Jedná se o volně dostupný software včetně zdrojového kódu, publikovaný pod licencí GNU General Public License, který je k dispozici pro mnoho platforem.

Díky otevřenému zdrojovému kódu je možné zjistit, jak přesně protokol OpenVPN funguje a jaká je struktura přenášených dat. Proto lze OpenVPN detekovat inspekcí jednotlivých paketů snadněji než uzavřené protokoly.

OpenVPN umožňuje vytvářet tunely pomocí protokolů TCP i UDP. Standardně se používá protokol UDP, neboť je efektivnější a spolehlivost přenosu dat může být zajišťována zapouzdřenými přenášenými protokoly. Pokud se při přenosu ztratí zapouzdřený paket TCP, TCP si vyžádá jeho opakování a není třeba, aby VPN zajišťovala spolehlivost přenosu dat. TCP se hodí například tam, kde je spojení přes UDP blokováno.

K šifrování dat i kontrolních zpráv využívá OpenVPN knihovnu OpenSSL. K výměně klíčů používá protokol TLS. Autentizaci lze provádět pomocí předem sdílených klíčů, certifikátů nebo uživatelského jména a hesla. Při inicializaci spojení je možné vynutit kontrolu a podepisování paketů pomocí HMAC. Poté server naváže komunikaci pouze s takovým klientem, který zaslal řádně

podepsaný paket, a na nepodepsané pakety vůbec neodpoví. To službu ochrání v případě pokusu o DoS útok nebo skenování portů.

Server dokáže při připojení na straně klienta vynutit různá nastavení, jako například používání konkrétních DNS, nastavení směrování atd. To je výhodné zejména pro klienty, kteří nemusí například kvůli změně adresního rozsahu v síti serveru měnit svoji konfiguraci.

OpenVPN může pracovat buď v režimu N:1 (klient - server), nebo 1:1 (point-to-point tunel). Spojení je možné vytvořit buď pomocí TUN adaptérů, zajišťujících propojení sítí na úrovni síťové vrstvy, nebo pomocí TAP adaptérů, zajišťujících propojení na úrovni linkové vrstvy. Volba adaptéru závisí na konkrétních potřebách propojení, vždy je však potřeba, aby na obou stranách tunelu byly použity adaptéry stejného typu.

1.5.1 Protokol OpenVPN

OpenVPN využívá vlastní protokol nad protokoly TCP či UDP. Protokol rozlišuje několik typů zpráv, které mají různou strukturu a datové položky [11]. Typ zprávy je v hlavičce každého OpenVPN paketu rozlišován pomocí takzvaného opcode (uvedený vždy v závorce u typu zprávy). Typy zpráv zasílaných v OpenVPN mohou být:

p_control_hard_reset_client_v1 (1) reset spojení klientem a zapomenutí předchozího stavu,

p_control_hard_reset_server_v1 (2) reset spojení serverem a zapomenutí předchozího stavu,

p_control_soft_reset_v1 (3) změna klíče,

p_control_v1 (4) paket řídící spojení (obvykle TLS),

p_ack_v1 (5) potvrzení paketů řídících spojení,

p_data_v1 (6) šifrovaná data,

p_data_v2 (9) šifrovaná data s id klienta,

p_control_hard_reset_client_v2 (7) reset spojení klientem a zapomenutí předchozího stavu, metoda získání klíče ≥ 2,

p_control_hard_reset_server_v2 (8) reset spojení serverem a zapomenutí předchozího stavu, metoda získání klíče ≥ 2,

p_control_hard_reset_client_v3 (10) reset spojení a zapomenutí předchozího stavu, metoda získání klíče ≥ 2, klientský TLS klíč.

Metoda získání klíče jedna přímo odvozuje klíč pomocí náhodných bitů získaných z RAND funkce OpenSSL. Metoda získání klíče dva používá informace z náhodných klíčů obou stran a spojuje je pomocí TLS PRF funkce. Jako výchozí se používá metoda získání klíče dva.

Zpráva typu *p_control*

Zpráva tohoto typu je určená k řízení spojení. Při přijetí jsou tyto zprávy zpracovávány odlišnou logikou než zprávy obsahující data. Používají se například pro inicializaci či reset spojení nebo pro změnu klíče.

Tabulka 1.1: OpenVPN zpráva typu *p_control*

	Bity							
Bajty	0–7		8–15	16–23			24–31	
0–3	*packet len. (16)*			*opcode (5)*	*key_id (3)*		*session_id (64)* →	
4–7	← *session_id (64)*→							
8-11	← *session_id (64)*						*hmac (128)* →	
12–15	← *hmac (128)* →							
16–23	← *hmac (128)* →							
24–31	← *hmac (128)* →							
32–39	← *hmac (128)*						*hmac (160)* →	
40–47	← *hmac (160)*						*packet-id (32)* →	
48–55	← *packet-id (32)*						*packet-id (64)* →	
56–63	← *packet-id (64)*						*time_t (32)* →	
64–71	← *time_t (32)*						*p_ack len (8)*	
72–79	*p_ack packet-id array (32)*							
80–87	*p_ack remote session_id (64)* →							
88–95	← *p_ack remote session_id (64)*							
96–103	*message packet-id (32)*							
104–112	*payload* →							

Schéma zprávy typu *p_control* zobrazuje Tabulka 1.1 a obsahuje následující položky:

packet length (2 bajty) délka zprávy (pouze u TCP),

opcode (5 bitů) id typu zprávy,

key_id (3 bity) id klíče,

session_id (8 bajtů) id relace,

hmac (16 nebo 20 bajtů) HMAC autentizace (nepovinné),

packet-id (4 nebo 8 bajtů) id paketu (nepovinné, ochrana proti replay útokům),

time_t (4 bajty) timestamp (nepovinné),

p_ack packet_id array length (1 bajt) délka pole s id paketů k potvrzení,

p_ack packet-id array (4 bajty) id paketu, který se potvrzuje (opakuje se tolikrát, kolik paketů je potvrzováno),

p_ack remote session_id (8 bajtů) id potvrzované relace (pokud je něco potvrzováno),

message packet-id (4 bajty) id paketu,

payload (n bajtů) TLS data.

Zprávy typu *p_control* jsou z důvodu zajištění spolehlivosti potvrzovány. Potvrzování se provádí buď v další zprávě typu *p_control*, nebo samostatnou zprávou typu *P_ACK*, jejíž struktura je podobná zprávě typu *p_control*, oproti které postrádá TLS data (payload) a vlastní id paketu (message packet-id).

Zpráva typu *p_data*

Zpráva typu *p_data* přepravuje samotná šifrovaná data. Protože je těchto zpráv v komunikaci největší množství, používá se v nich místo kompletního id relace (*session_id*) pouze kratší verze id klíče (*key_id*).

Tabulka 1.2: OpenVPN zpráva typu *p_data*

	Bity				
Bajty	0–7	8–15	16–23		24–31
0–3	*packet length (16)*		*opcode (5)*	*key_id (3)*	*peer_id (24)* →
4–7	← *peer_id (24)*		*packet-id (32)* →		
8-11	← *packet-id (32)*		*packet-id (64)* →		
12–15	← *packet-id (64)*		*payload* →		

Zprávu typu *p_data* ukazuje Tabulka 1.2 a obsahuje následující položky:

packet length (2 bajty) délka zprávy (pouze u TCP),

opcode (5 bitů) id typu zprávy,

key_id (3 bity) id klíče,

peer_id (3 bajty) id klienta (nepovinné),

packet-id (4 nebo 8 bajtů) id paketu (nepovinné, ochrana proti replay útokům),

payload (n bajtů) inicializační vektor, šifrovaná data, volitelně HMAC.

1.5.2 Protokol TLS

Protokol TLS je kryptografický protokol pro zabezpečenou komunikaci přes síť [12]. Je používán v mnoha aplikacích, například při procházení webu, posílání emailů či VoIP, kde zabraňuje odposlouchávání či falšování zpráv. Vzhledem k charakteru zasílaných zpráv lze protokol rozdělit na první a druhou část. V první části je určen typ zprávy, verze protokolu a délka zprávy. V druhé části jsou zasílány parametry příslušné danému typu zprávy. Pro účely detekce OpenVPN je vhodné sledovat i dění v protokolu TLS verze 1.2, který OpenVPN používá pro inicializaci klíčů šifrované komunikace.

Tabulka 1.3: TLS první část

	Bity			
Bajty	0–7	8–15	16–23	24–31
0–3	*record type (8)*	*version (16)*		*length (16)* →
4–7	← *length (16)*	*payload* →		

Schéma první části TLS protokolu ukazuje Tabulka 1.3 a obsahuje tyto položky:

record type (1 bajt) typ zprávy,

version (2 bajty) verze protokolu,

length (2 bajty) délka zprávy.

Typ zprávy lze rozpoznat podle *record type* (typ zprávy), který může nabývat následujících hodnot:

change_cipher_spec (20) signalizuje začátek šifrování komunikace,

alert (21) oznamuje problémy nebo nestandardní situace,

handshake (22) zřizuje zabezpečené spojení,

application_data (23) přenáší samotná data.

Podle toho, jaký typ zprávy obsahuje první část protokolu, se liší obsah druhé části protokolu.

Protokol *handshake*

Protokol *handshake* je druhá část protokolu, která obsahuje nejvíce možných typů zpráv. Slouží ke zřízení zabezpečeného spojení. Typ zprávy je určen položkou o velikosti jednoho bajtu, která následuje ihned za položkami první části protokolu. Může posílat zprávy následujících typů:

hello_request (0) restart zřizování zabezpečeného spojení serverem,

client_hello (1) inicializace zřizování zabezpečeného spojení klientem, podporované šifry a komprese,

server_hello (2) odpověď serveru na inicializaci, vybraná šifra a komprese,

certificate (11) zpráva obsahující řetězec certifikátů,

server_key_exchange (12) parametry algoritmu výměny klíčů od serveru,

certificate_request (13) požadavek klientského certifikátu serverem,

server_done (14) dokončení výběru šifry a komprese serverem,

certificate_verify (15) zaslání podepsaných dat privátním klíčem pro ověření certifikátu,

client_key_exchange (16) parametry algoritmu výměny klíčů od klienta,

finished (20) dokončení zřizování zabezpečeného spojení.

Pro účely detekce je zajímavá zpráva *client_hello*, kde klient sděluje podporované šifry a komprese, což lze využít k určité identifikaci klienta.

Protokol *change_cipher_spec*

Protokol *change_cipher_spec* je typ druhé části protokolu, který má pouze jednu zprávu oznamující začátek šifrování. Komunikace následující po této zprávě je již šifrována.

Protokol *alert*

Protokol *alert* je typ druhé části TLS a je používán pro oznamování problémů nebo nestandardních situací. Má pouze dva bajty, kdy první indikuje závažnost chyby a druhý její přesný druh dle stanoveného číselníku.

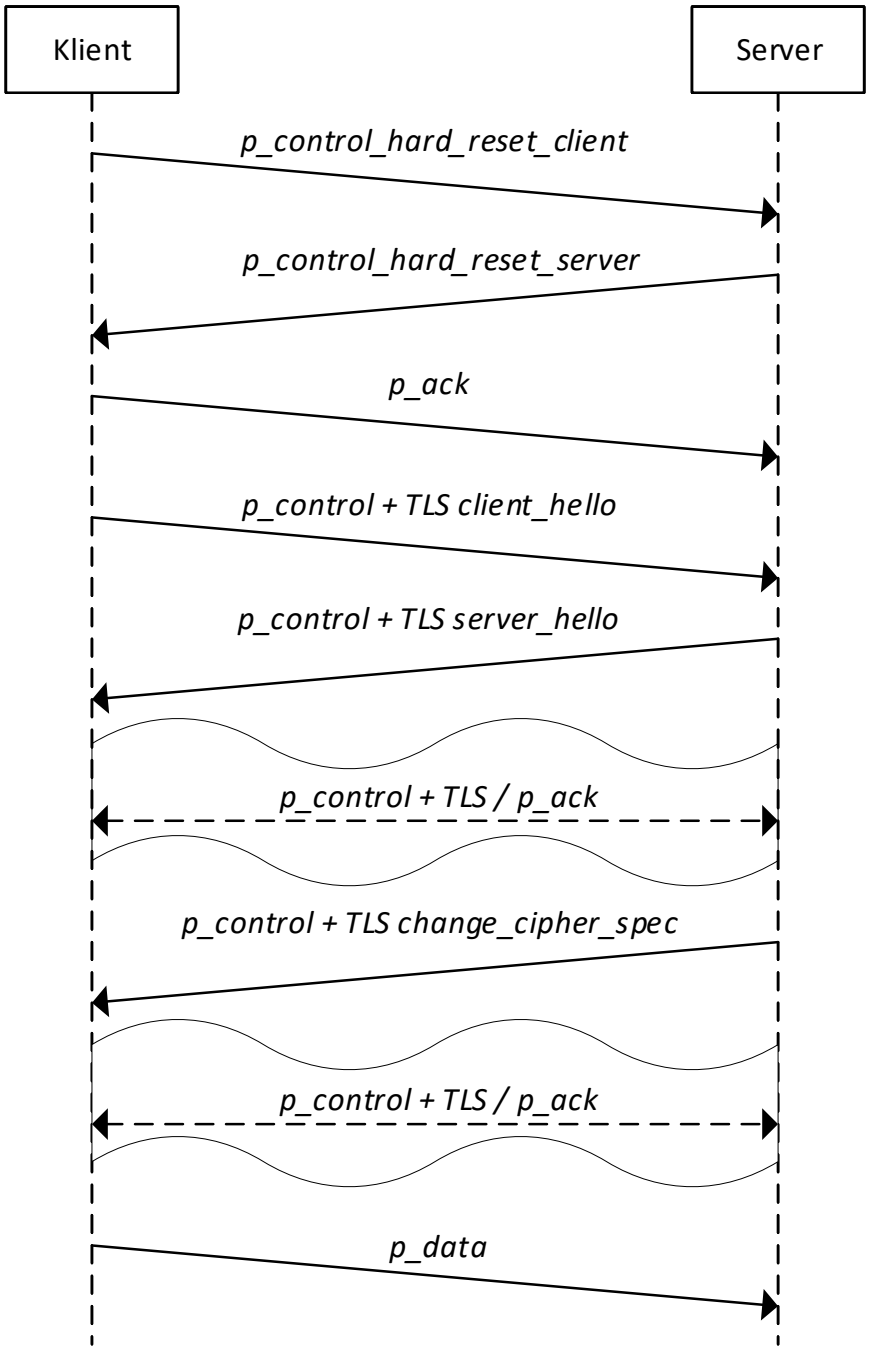
Protocol *application_data*

Poslední z typů druhé části TLS je typ *application_data*, který zajišťuje zapouzdření přenosu šifrovaných dat.

1.5.3 Navazování spojení

Komunikaci při navazování spojení VPN ukazuje Obrázek 1.2. Komunikace probíhá následovně:

1. Klient pošle zprávu *p_control_hard_reset_client*.



Obrázek 1.2: Navazování spojení OpenVPN

2. Server pošle zprávu *p_control_hard_reset_server*.
3. Klient potvrdí přijetí zprávou *p_ack*.
4. Klient pošle zprávu *p_control* s TLS zprávou *client_hello*.
5. Server odpoví zprávou *p_control* s TLS zprávou *server_hello*.
6. Zprávami *p_control* s TLS proběhne výměna certifikátů, klíčů atd. dle způsobu autentizace, zároveň může probíhat potvrzování zprávami typu *p_ack*.
7. Server pošle zprávu *p_control* s TLS zprávou *change_cipher_spec*, čímž zahájí šifrování.
8. Proběhne výměna informací zašifrovaným kanálem zprávami *p_control* s *TLS application_data*, zároveň může probíhat potvrzování zprávami *p_ack*.
9. Proběhne navázání komunikace a začne odesílání zpráv typu *p_data* obsahujících samotný šifrovaný provoz.

Standardně už dále probíhá komunikace pouze zprávami typu *p_data*, pouze v případě překlíčování po určitém čase nebo objemu přenesených dat dochází k výměně klíčů a k zasílání zpráv typu *p_control*.

1.6 Existující řešení

Knihovna nDPI

Jedná se o knihovnu pro hloubkovou analýzu paketů (DPI – Deep Packet Inspection) [13]. Modul pro detekci VPN se snaží detekovat OpenVPN tak, že hledá dva pakety OpenVPN spojení, kdy si uloží *session_id* prvního z nich a v druhém kontroluje, jestli odpověď obsahuje potvrzení daného *session_id*. Vzhledem k tomu, že se *session_id* posílá pouze při navazování spojení, dokáže tento způsob rozpoznat OpenVPN pouze při navazování spojení, nikoliv ve chvíli, kdy už je spojení navázáno.

Nástroj libprotoident

Jde o nástroj sloužící k identifikaci provozu na síti na základě analýzy paketů [14]. Jedním z mnoha modulů tohoto nástroje je i modul pro identifikaci VPN. Provádí identifikaci na základě známého portu, který OpenVPN používá na zachycení paketů *reset_client* nebo *reset_server*. Tento způsob identifikace funguje pouze při navazování spojení OpenVPN. Již běžící spojení identifikovat nedokáže.

Detekce pomocí frekvenční analýzy

Jedná se o detekování OpenVPN na základě analýzy četností typů zpráv v komunikaci. Vzhledem k tomu, že typ zprávy lze z paketu OpenVPN jednoduše získat, je možné analyzovat počet různých typů zpráv v komunikaci a výsledek porovnat s obvyklým množstvím typů jednotlivých zpráv v běžné OpenVPN komunikaci. Pokud bude charakteristika typů zachycených zpráv podobná charakteristice zpráv běžné OpenVPN komunikace, je možné s určitou pravděpodobností identifikovat provoz jako OpenVPN.

Kapitola 2

Analýza a návrh

2.1 Testovací data

Pro účely práce bylo potřeba získat dostatek testovacích dat. Vzhledem k tomu, že bylo zkoušeno několik různých způsobů detekce VPN, bylo zapotřebí i odlišných datových sad.

2.1.1 Data pro paketovou analýzu

Pro analýzu jednotlivých paketů bylo vytvořeno několik VPN serverů s různou konfigurací. Pro OpenVPN to byly čtyři konfigurace, které se lišily použitím protokolu TCP či UDP a použitím či nepoužitím HMAC autentizace paketů. Pro komerční Cisco AnyConnect se podařilo získat pouze jeden server, s nímž byla komunikace zachycována.

Protože paketová analýza cílí zejména na detekci inicializace spojení typickou pro daný druh VPN, byly tyto datové sady vytvářeny tak, aby obsahovaly pakety zasílané při navazování VPN komunikace. Nebylo potřeba, aby tyto datové sady obsahovaly větší množství komunikace ani aby trvaly delší dobu.

2.1.2 Data pro analýzu IP toků

Dat pro analýzu IP toků bylo třeba získat větší množství, protože u tohoto způsobu detekce VPN je rozhodující sledování charakteristik těchto dat typických jak pro provoz s VPN, tak pro provoz bez VPN, což bez dostatečně velkého množství dat není možné. Na tato data také byly kladeny velké nároky, neboť bylo potřeba, aby splnily několik kritérií.

Základním požadavkem bylo rozlišení, zda se jedná o VPN komunikaci, nebo komunikaci bez VPN. To bylo samo o sobě velmi náročné, neboť sama práce se detekcí VPN zabývala a nástroje na určení její přítomnosti nejsou. Data, o kterých stoprocentně víme, že VPN komunikaci obsahují či neobsahují,

bylo možné získat pouze tak, že byly v rámci práce vygenerovány a zachyceny vlastní datové sady.

Pro tento účel byly v rámci práce vygenerovány dva virtuální počítače. Jeden s operačním systémem GNU/Linux Ubuntu 20.04 LTS, druhý s operačním systémem Windows 10. Na těchto počítačích bylo vygenerováno velké množství testovacích datových sad. Byly získávány jak datové sady bez použití VPN, tak byly využity i různé VPN programy, které nabízely možnost zdarma vyčerpat určité množství dat přes VPN pro získání dostatečně rozmanité škály zachycené VPN komunikace. Pro VPN spojení byly použity tyto VPN služby: ProtonVPN, Windscribe, TunnelBear, HotspotShield, Hide.me a Speedify. Většina těchto programů nabízela různé režimy spojení, například TCP, UDP, IKEv2 atd. Všechny režimy, které se podařilo zprovoznit, byly vždy zachyceny. Při spuštěném zachytávání provozu pak byla náhodně generována komunikace jako procházení webu, streamování hudby či videa, stahování souborů, odesílání pošty atd.

Dalším požadavkem bylo rozlišení dle IP adresy až na konkrétní počítač. To znamená například nemožnost sbírat data za překladem adres, který sdružuje více uživatelů. Taková data by mohla obsahovat namíchanou komunikaci z několika počítačů, a to by již nebylo relevantní pro sledovaní charakteristik.

Zapotřebí bylo také znát rozsah IP adres, ze kterých komunikace probíhala. Komunikace těchto sledovaných IP adres musela být kompletně zachycena. V případě zachycení pouze části komunikace by nebyly charakteristiky dostatečně vypovídající. Znalost rozsahu IP adres, ze kterých komunikace probíhala, je důležitá ze dvou důvodů. Za prvé z hlediska toho, že víme, že pro tyto IP adresy máme komunikaci kompletní a má cenu se tedy zabývat její charakteristikou, a za druhé také pro snížení výpočetní náročnosti, protože mohou být počítány charakteristiky pouze těch IP adres, pro které víme, že to má smysl.

Pro analýzu charakteristik z IP toků byly datové sady připravovány tak, aby obsahovaly buď pouze VPN komunikaci, nebo pouze komunikaci bez VPN. Z datových sad obsahujících VPN byla tedy odstraňována komunikace před navázáním VPN spojení a po ukončení VPN spojení. Bylo také kontrolováno, zda nedošlo v průběhu zachytávání dat k výpadku VPN spojení.

V rámci práce byla vytvořena datová sada ve formě IP toků ze sond na páteřní síti CESNET2. Tato data se však nepodařilo dostatečně klasifikovat s ohledem na to, zda VPN komunikaci obsahují, či nikoliv. I tak se ale dala využít k ověření chování detekce nad reálným provozem a podařilo se díky nim detekci v mnoha ohledech vylepšit, zejména pak pomohla odhalit nedostatky v metodách hodnocení. Dále byla tato data využívána k vyzkoušení metod strojového učení.

2.2 Detekce OpenVPN analýzou paketů

Detekce OpenVPN analýzou paketů je možná díky znalosti protokolu OpenVPN. Lze zachytávat jednotlivé zprávy, které se posílají mezi každou dvojicí IP adres, a pokud se v komunikaci objeví paket typický pro inicializaci OpenVPN, začne se komunikace mezi touto dvojicí IP adres monitorovat a pomocí stavového automatu ověřovat posloupnost zpráv vůči standardní posloupnosti paketů při navazování OpenVPN.

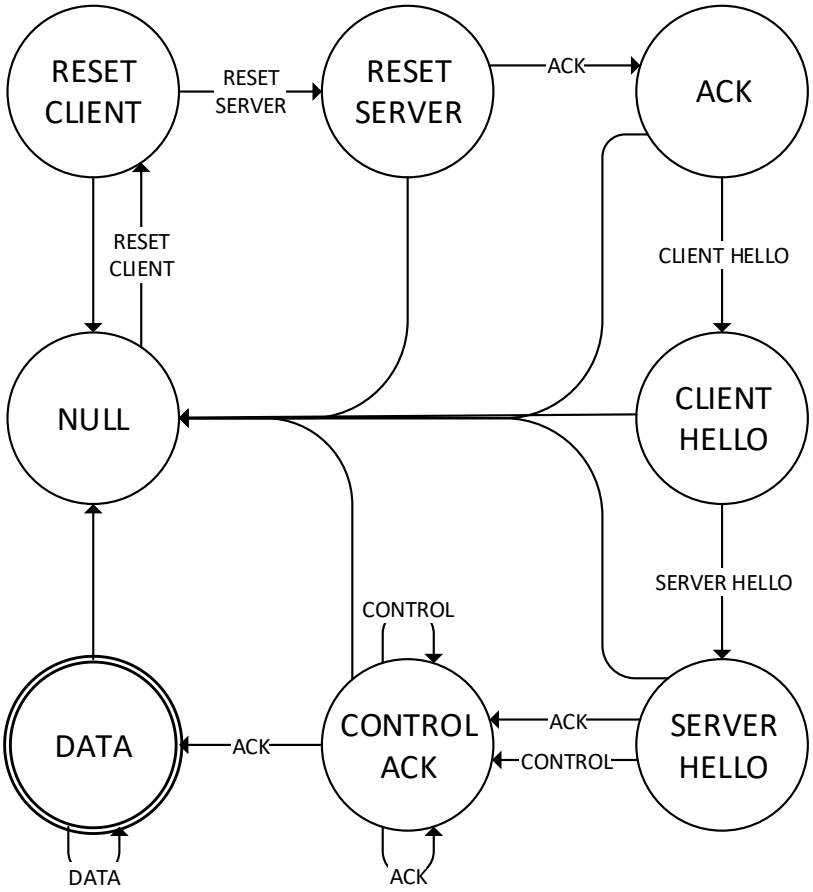
Při zachycení paketu nejprve dojde k jeho rozparsování na úrovni první až čtvrté vrstvy modelu ISO/OSI a k získání dat, která jsou na pozici, kde by byl v případě OpenVPN opcode. Pokud získaný opcode obsahuje hodnotu, kterých může OpenVPN opcode nabývat, postupuje paket k dalšímu zpracování. Pokud paket neobsahuje validní opcode, jeho zpracování tímto končí.

Pro další zpracování se využívá struktura, která uchovává aktuální stavy jednotlivých spojení. Touto strukturou je mapa, která u každého spojení identifikovaného zdrojovou IP adresou, zdrojovým portem, cílovou IP adresou a cílovým portem udržuje stav, ve kterém se spojení nachází. Výjimkou je stav NULL, který se pozná tak, že ve struktuře uložen není.

Pokud má tedy paket validní opcode, postupuje k dalšímu zpracování. Tím je spuštění algoritmu dle příslušného opcode. Algoritmus se pro různé opcode liší, ale většinou proběhne ve struktuře udržující stavy spojení vyhledání spojení, a to buď jedním, nebo oběma směry. Vzhledem k tomu, že inicializace probíhá vždy zprávou *reset_client* ve směru klient → server, je známa orientace komunikace v paměti stavů. Vyhledání pouze v jednom směru se děje pro zprávy s typem opcode, které mají jasně definovaný směr, například *server_hello*. Vyhledávání v paměti stavů pro oba směry se děje pro zprávy s typem opcode, které jsou zasílány obousměrně, například *data*.

Na základě vyhledaného aktuálního stavu spojení se určí, zda je požadovaný přechod dle opcode možný, či nikoliv. Pokud přechod odpovídá grafu přechodů, dojde k úspěšné změně stavu a uložení této informace do paměti stavů. Pokud požadovaný přechod neodpovídá grafu přechodů, stav se nezmění. V takovém případě je inkrementován čítač pokusů o neplatný počet přechodů a pokud je počet neplatných pokusů o přechod vyšší než stanovená mez, dojde k přesunu spojení do stavu NULL, tedy k jeho smazání z paměti stavů. Při každém úspěšném přechodu stavu je čítač pokusů o neplatný přechod vynulován. Čítač pokusů o neplatný přechod je používán, protože může občas dojít například k opakování paketu z důvodu jeho ztráty na síti nebo může dorazit například fragmentovaný paket a je potřeba, aby taková skutečnost detekci ihned nevyřadila.

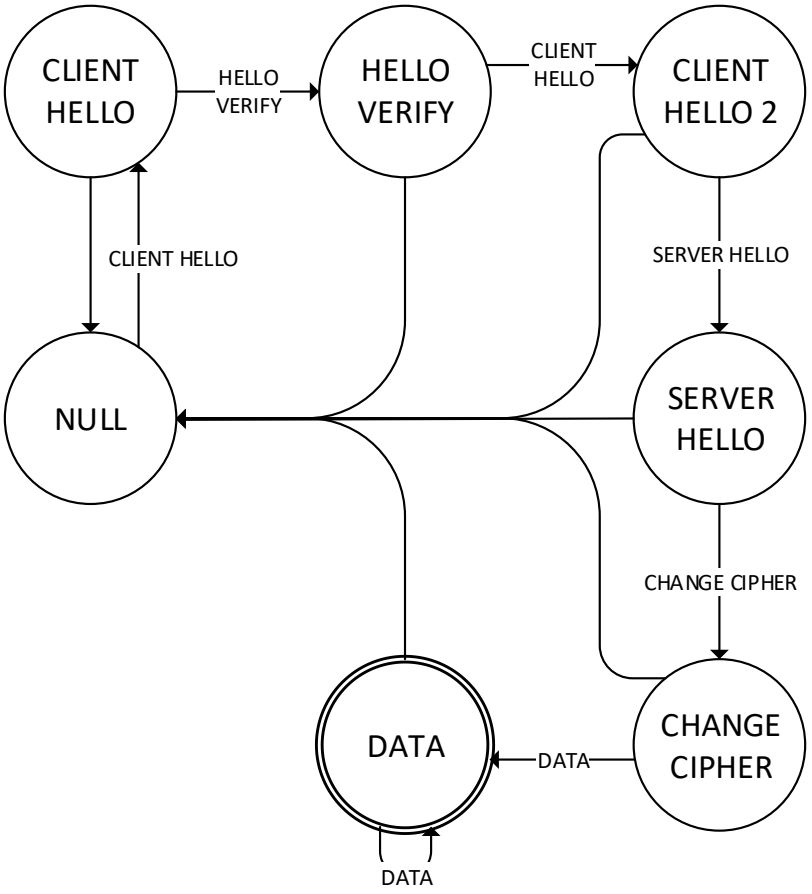
Pokud pro dané spojení není uložen žádný stav, nic se neděje. Výjimkou je situace, kdy se jedná o paket s opcode *reset_client*. V případě detekce paketu s opcode *reset_client* dojde vždy k nastavení stavu v paměti spojení do stavu *reset_client*, bez ohledu na předchozí stav. Stejně se chová OpenVPN, která při přijetí této zprávy vše zapomene, a nastavování spojení začíná od začátku.



Obrázek 2.1: Schéma detekce OpenVPN

Přechody mezi jednotlivými stavy ilustruje Obrázek 2.1. Samotná detekce se považuje za dokončenou, jakmile spojení úspěšně projde grafem do stavu *data* a proběhne odeslání určitého množství zpráv typu *data*. To zajišťuje spolehlivou a zároveň rychlou detekci.

Detekce tohoto typu byla zvolena jako pokus o zjištění, jestli je vůbec detekce nějakého druhu VPN možná. Pracuje na té nejnižší úrovni, na úrovni paketů, takže se jedná o výpočetně i datově nejnáročnější způsob. Také je zaměřena pouze na jednu konkrétní implementaci VPN a pro jiné implementace VPN bude neúčinná. Na druhou stranu se jedná o účinné řešení s rychlou detekcí, která je nutná pro další reakci.



Obrázek 2.2: Schéma detekce AnyConnect

2.3 Detekce AnyConnect analýzou paketů

Detekce VPN Cisco AnyConnect analýzou paketů pracuje na podobném principu jako detekce OpenVPN, ale protože AnyConnect používá protokol DTLS, používají se jiné stavy a přechody mezi nimi. Princip však zůstává stejný. V případě zachycení paketu, který odpovídá inicializaci komunikace VPN, dojde k uložení stavu spojení do mapy a v případě zachycení dalšího paketu tohoto spojení se vyhodnocují přechody mezi jednotlivými stavy. Buď dojde k přesunu do dalšího stavu, nebo k vymazání z paměti stavů, pokud přechod neodpovídá očekávanému chování. Detekce končí přechodem do přijímacího stavu. Stavy VPN AnyConnect znázorňuje Obrázek 2.2.

Jedná se o rychlé řešení vyžadující přístup k obsahu jednotlivých paketů.

Jeho zásadní nevýhodou je však založení detekce na navázání spojení protokolem DTLS. Tento protokol pracuje přímo nad protokolem UDP a detektor nijak nepozná, zda se jedná o spojení DTLS navazované pro účely VPN spojení nebo za jakýmkoliv jiným účelem. V praxi by tak detektor způsoboval velké množství falešně pozitivních detekcí.

Pro jeho korektní funkčnost by bylo potřeba v komunikaci vyhledat typický znak pro AnyConnect VPN. Jedním takovým znakem je, že AnyConnect klient vždy před navázáním komunikace pomocí UDP a DTLS komunikuje s VPN serverem pomocí TCP a ověřuje mimo jiné například možnost aktualizace VPN klienta. Bylo by tedy možné sledovat, zda v krátkém časovém intervalu před navázáním DTLS spojení neproběhlo TCP spojení mající charakteristiku AnyConnect. V rámci práce nebyla tato kontrola implementována, protože se práce dále snažila věnovat detekci VPN na základě charakteristik, které nejsou závislé na konkrétní VPN implementaci, místo zlepšování detekce jedné konkrétní implementace.

2.4 Detekce s využitím IP toků z Cisco Joy

Detekce za pomoci datových toků generovaných z Cisco Joy se snaží odhalit VPN provoz pomocí agregace charakteristik z IP toků. Práce vycházela z předpokladu, že použití VPN bude mít určitý dopad na charakteristiku provozu, který z daného počítače, tedy IP adresy, probíhá. Vzhledem k tomu, že pouhý jeden IP tok neobsahuje dostatek informací na to, aby bylo možné rozhodnout, zda dané spojení je nebo není VPN, je nutné data nějakým způsobem dále agregovat, jak co se týče dat, tak co se týče času. Samotná agregace potom znamená, že je například počítán průměrný čas trvání komunikace jednoho IP toku, kolikrát se komunikovalo s určitou IP adresou, kolik dat bylo přeneseno atd.

Vzhledem k předpokladu, že při použití VPN se bude lišit charakteristika VPN provozu, je vhodným způsobem, jak data agregovat, počítání charakteristik získaných z IP toků pro každou sledovanou IP adresu. Sledovaná IP adresa je taková IP adresa, která je z rozsahu, u něhož chceme kontrolovat, zda nepoužívá VPN, a u které předpokládáme, že zachytáváme většinu její komunikace do internetu. Rozsah sledovaných IP adres je třeba detektoru předat jako vstup, neboť bez něj není možné poznat, od kterých IP adres zpracovávaných IP toků máme nebo nemáme kompletní komunikaci. Typicky se jedná například o lokální rozsah IP adres.

Agregaci by bylo možné provádět například i na základě dvojice spolu komunikujících IP adres, ale to by nedostatečně vystihovalo charakteristiky celkové komunikace sledované IP adresy jako celku. Zároveň by ale taková detekce měla větší šanci odhalit případnou VPN, kterou není směrován veškerý provoz, ale pouze jeho část. Tato metoda by však agregovala menší množství dat a detekce by proto mohla být nepřesnější. Protože se práce snažila ově-

řit, jestli je detekce pomocí charakteristik vůbec možná, tuto méně přesnější metodu netestovala.

Další prováděná agregace dat je agregace v čase. Pro účely detekce je potřeba nasbírat dostatečné množství dat, ale zároveň je zapotřebí detekci provést co nejdříve od započetí komunikace. Na základě testování úspěšnosti detekce v závislosti na časové agregaci, jak ukazuje Tabulka 2.1, byl zvolen časový interval pět minut. Tento interval je dostatečný pro získání kvalitních charakteristik a zároveň se jedná o ještě akceptovatelnou detekční dobu. Časová agregace také umožňuje vyřadit ty IP adresy, které v daném časovém úseku komunikovaly méně, než je pro VPN spojení typické. U VPN spojení se předpokládá, že bude spojení neustále udržováno a kontrolováno, takže budou zasílány pakety, na které bude druhá strana odpovídat, i kdyby zrovna tunelem žádná komunikace neprocházela. Je tedy možné vyřadit IP adresy, u nichž je již z nízkého množství agregované komunikace a jejího rozložení v čase evidentní, že neprobíhala po celou dobu. To znamená, že její charakteristiky nejsou počítány z dostatečného množství dat, a tudíž nemusí být dostatečně přesné. Zároveň to však znamená, že se o VPN komunikaci s největší pravděpodobností nejedná, a je možné komunikaci z rozhodování vyřadit, neboť by její nepřesné charakteristiky vzniklé z malého množství dat mohly vést k falešné pozitivitě.

Tabulka 2.1: Úspěšnost detekce v závislosti na časové agregaci

	VPN			bez VPN		
čas (s)	vzorků	chyb	úspěšnost	vzorků	chyb	úspěšnost
60	509	68	87 %	222	0	100 %
150	206	9	96 %	88	0	100 %
300	99	2	98 %	42	0	100 %
450	62	1	98 %	27	0	100 %
600	45	1	98 %	20	0	100 %

Samotné IP toky generuje na základě surových zachycených paketů Cisco Joy, který umí mimo základních informací exportovat i rozšířené informace. Z těchto rozšířených dat byly pro účely detekce VPN exportovány a používány následující:

DNS obsahuje informace o DNS dotazech,

TLS exportuje informace o navazování nových TLS spojeních,

Entropie obsahuje informace o entropii dat,

Walsh-Hadamardova transformace pomáhá odhalit periodicitu v datech,

Data exportuje několik prvních bajtů prvního paketu v IP toku,

Biflow spojuje běžně jednosměrné IP toky do obousměrných IP toků.

Obousměrné IP toky neboli biflow jsou IP toky, ve kterých je obsažena komunikace oběma směry, pokud nějaká existovala a exportéru IP toků se ji podařilo spojit. Použití obousměrných IP toků bylo otestováno a bylo zjištěno, že jejich použitím a počítáním statistik pouze z takových IP toků, které obsahují obousměrná data, dochází k pročištění a zvýraznění charakteristik VPN provozu. Při vytváření a přípravě testovacích dat bylo zjištěno, že zejména ihned po zapnutí VPN ještě nějakou dobu dobíhá komunikace, na kterou již často nejsou odesílány reakce.

V Tabulce 2.2 je možné vidět rozdíly mezi výsledky získanými s využitím jednosměrných ($\rightarrow$) IP toků a obousměrných ($\leftrightarrow$) IP toků. Zatímco u provozu bez VPN nejsou vidět žádné výrazné rozdíly kromě logického zdvojnásobení průměrného počtu paketů v IP tocích, což je důsledek obousměrných IP toků, u VPN provozu je možné vidět výrazný nárůst průměrného času IP toku a snížení počtu unikátních IP adres i portů v komunikaci.

Tabulka 2.2: Porovnání charakteristik jednosměrných a obousměrných IP toků

charakteristika	$\rightarrow$ IP toky bez VPN	$\leftrightarrow$ IP toky bez VPN	$\rightarrow$ IP toky s VPN	$\leftrightarrow$ IP toky s VPN
průměrně paketů v IP tocích	9,0	18,6	38,9	121,9
unikátních portů	4676,8	4531,0	134,7	14,4
unikátních IP adres	573,7	562,7	26,1	5,3
poměr DNS	0,46	0,48	0,00	0,00
průměrný čas trvání (s)	2,5	2,6	15,4	25,4
poměr TCP SYN příznaků	0,37	0,39	0,04	0,04
poměr inic. TLS spojení	0,36	0,38	0,00	0,00
poměr nejčastější IP	0,09	0,09	0,88	0,90

Program tedy zpracovává vstup, na kterém dostává obousměrné IP toky. Pro každý obousměrný IP tok zkontroluje, jestli je zdrojová nebo cílová IP adresa ze sledovaného rozsahu, a pokud ano, provede pro tuto sledovanou IP adresu uložení dat potřebných pro výpočet charakteristik. Počítaných charakteristik je mnoho a podle toho se liší data potřebná pro jejich výpočet. Někdy může jít pouze o inkrementaci čítače, který zaznamenává počet výskytu určité události, jindy je potřeba uložení celé hodnoty z obousměrného IP toku do pole.

Jakmile se na vstupu programu objeví obousměrný IP tok, který již dle agregace podle času patří do dalšího časového okna, provede program výpočet charakteristik pro aktuální časové okno. Vyhodnotí, která komunikace byla VPN a která nebyla VPN, proběhne smazání paměti dat pro výpočet charakteristik a proces začíná znovu.

Pro zjištění, jaké charakteristiky by mohly být pro VPN provoz typické, byla vydefinována datová sada obsahující třicet dva charakteristik:

1. počet IP toků,
2. rozptyl počtu paketů v IP tocích,
3. průměrný počet paketů v IP tocích,
4. minimální počet paketů v IP tocích,
5. maximální počet paketů v IP tocích,
6. počet unikátních portů v komunikaci,
7. počet unikátních protokolů v komunikaci,
8. počet unikátních IP adres v komunikaci,
9. počet DNS dotazů na jeden IP tok,
10. rozptyl času trvání jednoho IP toku,
11. průměrný čas trvání jednoho IP toku,
12. minimální čas trvání jednoho IP toku,
13. maximální čas trvání jednoho IP toku,
14. počet SYN příznaků TCP na jeden IP tok,
15. rozptyl entropie dat,
16. průměr entropie dat,
17. minimum entropie dat,
18. maximum entropie dat,
19. počet inicializací TLS spojení na jeden IP tok,
20. rozptyl velikosti paketů,
21. průměr velikosti paketů,
22. minimální velikost paketu,
23. maximální velikost paketu,
24. přítomnost komunikace na blacklistu,
25. počet OpenVPN data hlaviček na jeden IP tok,

26. procento komunikace s nejčastěji komunikovanou IP adresou,
27. procento komunikace s druhou nejčastěji komunikovanou IP adresou,
28. procento komunikace s třetí nejčastěji komunikovanou IP adresou,
29. průměr první složky Walsh-Hadamardovy transformace,
30. průměr druhé složky Walsh-Hadamardovy transformace,
31. průměr třetí složky Walsh-Hadamardovy transformace,
32. průměr čtvrté složky Walsh-Hadamardovy transformace.

Tyto charakteristiky byly následně vypočítány z testovacích dat jak bez VPN, tak s VPN. Výsledky byly zpracovány a byly hledány takové charakteristiky, ve kterých se průměrná hodnota charakteristiky komunikace bez VPN a s VPN výrazně liší. Ideální jsou takové charakteristiky, které se liší natolik, že rozsah mezi minimální a maximální hodnotou charakteristiky v souboru bez VPN je disjunktní s rozsahem mezi minimální a maximální hodnotou dané charakteristiky v souboru s VPN. Tedy například charakteristika průměrný čas trvání jednoho IP toku se ukázala jako vhodná, neboť průměrný čas trvání jednoho IP toku bez VPN se pohybuje mezi 0,7 až 2,7 sekundami, zatímco při použití VPN trvá jeden IP tok průměrně 14,3 až 31,3 sekundy. Protože se rozsahy mezi minimální a maximální hodnotou nepřekrývají, je možné podle tohoto parametru odlišit VPN provoz od provozu bez VPN. Hodnoty napočítaných charakteristik, které jsou z hlediska detekce VPN zajímavé a které byly následně využity v různých metodách hodnocení k detekci VPN provozu, ukazuje Tabulka 2.3. Jedná se o následující charakteristiky:

- průměrný počet paketů v IP tocích,
- počet unikátních portů v komunikaci,
- počet unikátních IP adres v komunikaci,
- počet DNS dotazů na jeden IP tok,
- průměrný čas trvání jednoho IP toku,
- počet SYN příznaků TCP na jeden IP tok,
- počet inicializací TLS spojení na jeden IP tok,
- procento komunikace s nejčastěji komunikovanou IP adresou.

Jako nevhodné charakteristiky se projevily blacklist a hledání názvu VPN v reverzním DNS dotazu. Blacklist nebyl dostatečně kvalitní a označoval jako závadnou většinu agregované komunikace i bez použití VPN. Reverzní DNS dotazy naopak nedetekovaly v testovacích datech žádné pozitivní výsledky a jejich provádění velmi zpomalovalo samotnou detekci.

Tabulka 2.3: Napočítané charakteristiky VPN vs. běžný provoz Cisco Joy

	bez VPN			s VPN		
charakteristika	min	ø	max	min	ø	max
paketů v tocích	7,4	18,7	43,1	61,2	118,6	168,7
unik. portů	57,0	645,2	1727,0	3,0	6,8	28,0
unik. IP adres	21,0	119,0	323,0	2,0	3,6	10,0
poměr DNS	0,12	0,49	0,80	0,00	0,00	0,00
čas trvání (s)	0,7	2,7	5,7	14,3	25,2	31,3
TCP SYN příz.	0,10	0,35	0,84	0,00	0,02	0,18
inic. TLS spojení	0,09	0,34	0,78	0,00	0,00	0,06
nejčastější IP	0,05	0,16	0,44	0,58	0,91	0,95

2.5 Detekce s využitím IP toků z NEMEA modulu

Detekce s využitím NEMEA modulu funguje na podobném principu jako detekce s využitím Cisco Joy. U NEMEA modulu se však využil exportér IP toků z NEMEA frameworku nazývaný *flow_meter*. Z tohoto exportéru byly používány pouze základní IP toky, nebyla tedy využívána rozšíření na export dat o DNS, TLS, Entropie atd. Detekce ze základních IP toků je lepší z hlediska praktického použití detektoru, neboť z důvodu nižších nároků na vstupní data o provozu je možné takový detektor používat s menšími nároky na výkon a kapacitu paměti, a to i na místech, kde není možné rozšířené IP toky získávat. Zároveň ale rozšířené IP toky poskytují další charakteristiky, podle kterých je možné provoz klasifikovat, takže jejich nepřítomnost znamená horší kvalitu detekce.

Stejně jako v případě Cisco Joy, i zde byly využívány obousměrné IP toky, které se získávaly pomocí spojování jednosměrných IP toků NEMEA modulem *preprocessing module*. Ze souboru charakteristik pro detekci byly vyřazeny ty charakteristiky, které není možné ze základních IP toků zjistit. Pro detekci pomocí NEMEA modulu je tedy využíván tento soubor charakteristik:

- průměrný počet paketů v IP tocích,
- počet unikátních portů v komunikaci,
- počet unikátních IP adres v komunikaci,
- průměrný čas trvání jednoho IP toku,
- procento komunikace s nejčastěji komunikovanou IP adresou.

Vzhledem k tomu, že soubor obsahuje pouze pět charakteristik, bude velmi výrazně dotčena kvalita detekce takového modulu, neboť bude jednoduché takový modul přimět k falešným detekcím. Napočítané hodnoty pro tyto charakteristiky jsou v Tabulce 2.4. Hodnoty některých charakteristik se výrazně liší

od hodnot stejných charakteristik napočítaných pomocí Cico Joy. To je dáno použitím různých exportérů IP toků, kdy každý může používat jiné algoritmy pro generování IP toků a spojování jednosměrných IP toků do obousměrných IP toků.

Tabulka 2.4: Napočítané charakteristiky VPN vs. běžný provoz NEMEA

	bez VPN			s VPN		
charakteristika	min	ø	max	min	ø	max
paketů v tocích	100,6	40,7	12,5	117837,1	50780,4	14288,2
unik. portů	7508,0	4562,3	1296,0	15,0	5,2	1,0
unik. IP adres	776,0	575,2	231,0	3,0	1,9	1,0
čas trvání (s)	6,1	4,3	2,8	291,6	177,2	50,8
nejčastější IP	0.55	0.27	0.09	1.00	1.00	0.97

2.6 Metody hodnocení

Pro účely rozhodování o tom, zda provoz je nebo není VPN, vzniklo několik různých způsobů hodnocení napočítaných charakteristik.

2.6.1 Bodový součet

Každý sledovaný parametr se převede na určitý počet bodů, kdy vyšší počet bodů znamená větší podobnost s VPN provozem. Hodnotí se potom celkový součet nasbíraných bodů za jednotlivé charakteristiky. Výhodou této metody je rychlá a jednoduchá změna citlivosti rozhodování, možnost používání charakteristik, které jsou pozitivní pouze u některých druhů VPN, a možnost přiřadit různým charakteristikám různou důležitost, tedy větší maximální bodové skóre. Nevýhodou je hodnocení charakteristik pouze jako celku a i když v souboru byla jedna charakteristika, která jasně naznačovala, že se o VPN provoz jednat nemůže, stále je možné, aby v ostatních charakteristikách nasbírala komunikace dostatek bodů pro to, aby byla klasifikována jako VPN.

Tento způsob hodnocení dobře fungoval na nasbíraných testovacích datech. Ve chvíli, kdy byla algoritmu předložena data z běžného provozu, se však rychle ukázal nedostatek v hodnocení dat jako celku bez vyžadování minimální výše jednotlivých charakteristik.

2.6.2 Nastavení rozsahu charakteristik

Tato metoda používala striktně nastavený rozsah hodnot pro každou charakteristiku, kterou musela komunikace splnit, aby byla za VPN označena. Oproti předchozímu způsobu má výhodu ve vyřazení provozu, který v některé charakteristice nesplnil minimální požadovanou hodnotu. Nevýhodou je pak obtížné

nastavení citlivosti rozhodování, pouze binární výstup, který nereflektuje, jak moc si je algoritmus jistý, a striktní vyřazení komunikace, která se, byť v jediné charakteristice, odliší od předem definovaných hodnot.

Také není možné používat charakteristiky, které se projevují pouze u některých typů VPN, neboť vyžaduje od každé charakteristiky určitou minimální hodnotu a pokud ji splní, již dále hodnocení neovlivní. Tedy charakteristika, která by byla u poloviny VPN nulová a u druhé poloviny VPN vysoká, by buď zařídila falešně negativní výsledky poloviny VPN spojení, anebo by se hodnocení vůbec neúčastnila.

2.6.3 Minimum a průměr

Poslední metoda hodnocení se snažila vzít si to dobré z předchozích metod. Každá charakteristika byla obodována nula až sto body podle toho, jak moc se podobá charakteristikám VPN komunikace. Na bodování souboru charakteristik jsou stanoveny požadavky na minimum a průměr. Kontrola minima zajišťuje, že každá z charakteristik naplňuje alespoň minimální podobnost s VPN provozem, a kontrola průměru zajišťuje, že se charakteristiky podobají VPN komunikaci jako celek.

Například je stanoveno minimum více než padesát bodů a průměr více než osmdesát bodů, což zajišťuje, ža každá z charakteristik má minimálně padesát bodů a v průměru mají více než osmdesát bodů. Tím je zajištěno jednoduché nastavování citlivosti detekce, hodnocení přesnosti detekce pomocí průměru hodnocení a nezamítnutí komunikace kvůli pouze jediné horší charakteristice. Nevýhodou je nemožnost nastavení důležitosti jednotlivých charakteristik a nemožnost používat charakteristiky, které se projevují pouze u některých typů VPN, zatímco u ostatních jsou nulové či velmi nízké.

Pro nejlepší výsledky by bylo vhodné použít algoritmy strojového učení, které by dokázaly odvodit hodnotící kritéria z dat. Použití strojového učení bylo v případě tohoto problému taktéž vyzkoušeno, ale potíží byl nedostatek vhodně rozlišených trénovacích dat.

2.7 Detekce s využitím strojového učení

Poslední vyzkoušenou metodou bylo strojové učení. Předpokladem bylo, že strojové učení dokáže vytvořit sofistikovanou hodnotící metodu, která bude dávat lepší výsledky než manuálně vytvořené hodnotící algoritmy. Pro to je ale potřeba dostatečné množství trénovacích dat. Pro tento účel byla využita data ze sond na páteřní síti CESNET2. Tato data nebyla sice vhodně klasifikovaná, ale bylo jich k dispozici velké množství a bylo možné provést vyhodnocení výsledné naučené hodnotící metody na řádně klasifikovaných vlastních nasbíraných datech. Učení algoritmu na vlastních nasbíraných datech nebylo vhodné, neboť tato data zvládal algoritmus rozdělit podle jednoho či dvou parametrů.

Pro tento účel tedy vznikly dvě nové datové sady IP toků ze sítě CESNET2. První, která se snažila zachytit VPN provoz, a druhá, která se snažila zachytit běžný provoz. První datová sada byla vytvářena tak, že se několik minut sledoval provoz na všech sondách v síti CESNET2 a zaznamenávaly se IP adresy, které komunikovaly na portu 1194, běžně využívaném OpenVPN. Ihned poté se spustilo zachytávání IP toků na všech sondách v síti CESNET2 a zachytávala se komunikace IP adres, které byly v předchozím časovém intervalu zaznamenány jako komunikující na portu 1194. Rozsah IP adres ze sítě CESNET2, pro které byla prováděna detekce, byl následující:

- 146.102.0.0/16
- 147.228.0.0/16
- 147.230.0.0/15
- 147.251.0.0/16
- 147.32.0.0/15
- 158.194.0.0/16
- 158.196.0.0/16
- 160.216.0.0/15
- 185.8.160.0/22
- 193.84.116.0/23
- 193.84.160.0/20
- 193.84.192.0/19
- 193.84.192.0/20
- 193.84.208.0/20
- 193.84.32.0/20
- 193.84.53.0/24
- 193.84.55.0/24
- 193.84.56.0/21
- 193.84.80.0/22
- 195.113.0.0/16

Druhá datová sada, která se snažila zachytit běžnou komunikaci, probíhala tak, že se zachytávaly IP toky pouze na jedné sondě v síti CESNET2. Vzhledem k tomu, že běžného provozu je dostatek, nebylo nutné, aby probíhal záchyt na všech sondách současně. Rozsah IP adres, na kterých byla prováděna detekce, odpovídá rozsahu IP adres náležících v síti CESNET2 k těmto sondám. Konkrétně s jedná o rozsahy:

- 147.32.112.0/23
- 147.32.30.0/23

Datové sady byly ovlivněny tím, že se jednalo o data, kde již mohla být část komunikace agregována pomocí překladu adres z více počítačů. Jejich klasifikace byla provedena pouze na základě výskytu portu 1194 v komunikaci, což nemusí nutně znamenat VPN, a část komunikace opouštějící síť CESNET2 nemusela být vůbec sondami zachycena. Také se v datových sadách pravděpodobně vyskytoval takový VPN provoz, kde byla VPN tunelem směrována pouze část komunikace, což sice znesnadňuje detekci pomocí charakteristik, ale na druhou stranu je to část VPN provozu, kterou by bylo také vhodné umět detekovat.

Datové sady byly vygenerovány pomocí Cisco Joy včetně stejných rozšířených informací využívaných při detekci pomocí charakteristik s využitím Cisco Joy IP toků. Proto mohlo být vstupem algoritmu strojového učení všech třicet dva charakteristik napočítaných z těchto IP toků. Tyto charakteristiky byly společně s jejich klasifikací předávány knihovně pro strojové učení scikit-learn, která poté vygenerovala pro dané klasifikované vstupy binární rozhodovací strom. Tento binární rozhodovací strom byl aplikován na vlastní správně klasifikovaná data a byla pozorována míra úspěšnosti detekce.

Datové sady ze sítě CESNET2 byly zpracovávány dvěma různými způsoby. První způsob nebral ohled na to, zda datová sada měla nebo neměla obsahovat VPN, a prováděl klasifikaci pro strojové učení pouze na základě přítomnosti nebo nepřítomnosti portu 1194 v komunikaci. To vedlo k získání výrazně většího množství dat, která byla předána ke zpracování strojovému učení, neboť každá zachycená komunikace ze sledovaného rozsahu byla předána buď jako VPN, nebo jako běžný provoz bez VPN. Zároveň se však zvyšovala možnost chyby v klasifikaci.

Druhý způsob poté zohledňoval původní určení datové sady a strojovému učení předal pouze takovou komunikaci, která buď byla z VPN datové sady a obsahovala provoz na portu 1194 nebo byla z datové sady běžného provozu bez VPN a neobsahovala provoz na portu 1194. Komunikace, která byla z VPN datové sady a neobsahovala komunikaci na portu 1194 nebo byla z datové sady běžného provozu bez VPN a obsahovala komunikaci na portu 1194, nebyla do strojového učení zařazena. Tím došlo k omezení množství dat předaných strojovému učení, ale snížila se možnost chyby v klasifikaci.

Kapitola 3

Realizace

Vzhledem k tomu, že práce zkoušela různé způsoby detekce VPN, vzniklo několik verzí programů, které byly použity pro experimentální vyhodnocení a porovnání.

3.1 Detekce OpenVPN analýzou paketů

Detektor OpenVPN je aplikace napsaná v jazyce C++ s využitím knihovny libpcap. Program je tvořen hlavní funkcí, která zajišťuje zpracovávání parametrů, nastavování zdroje dat a spouštění detekční smyčky. Jako zdroj dat může být použit buď soubor s uloženou dříve zachycenou komunikací, nebo může být prováděno zachytávání přímo na číslem specifikovaném rozhraní. Detekční smyčka potom zajistí pro každý paket spouštění detekční funkce, která paket zpracuje. Dojde také k vytvoření datové struktury mapa, která se používá k ukládání stavu spojení.

Detekční funkce nejprve zajistí zpracování jednotlivých hlaviček. To znamená načtení Ethernet hlavičky, IP hlavičky, TCP nebo UDP hlavičky a nakonec OpenVPN hlavičky. K tomu využívá ukazatele na připravené struktury jednotlivých protokolů. U protokolů, jejichž délka může být různá, jako například TCP, se zjišťuje také délka dané hlavičky, aby bylo možné pokračovat zpracováním protokolů vyšších vrstev. Pokud je v průběhu zpracovávání zjištěno, že hlavička vyšší vrstvy není jednou z podporovaných, například paket s protokolem ICMP, zpracování se ukončí a začne zpracovávání dalšího paketu. Na konci této fáze tedy známe číslo, které je v paketu na pozici, na které by v případě OpenVPN měl být opcode.

Následuje kontrola, zdali je daný opcode skutečně z rozsahu jedna až deset, kterého může opcode v případě OpenVPN nabývat. Pokud paket nemá opcode z tohoto rozsahu, nejedná se o OpenVPN, není tedy nutné se jeho zpracováním dále zabývat a je možné pokračovat zpracováváním dalšího paketu. Pokud opcode je z rozsahu používaného OpenVPN, dojde k přípravě datové struktury typu dvojice, kdy první člen je struktura obsahující zdrojovou IP adresu a

zdrojový port a druhý člen je struktura obsahující cílovou IP adresu a cílový port.

Tato dvojice, u které záleží na orientaci komunikace, se používá jako klíč v mapě, která byla deklarována při startu programu a která slouží jako paměť pro ukládání spojení. Jako hodnota příslušnému klíči se v mapě používá struktura obsahující informace o aktuálním stavu spojení, počtu kontrolních paketů, počtu datových paketů a počtu posledních neplatných pokusů o změnu stavu.

Ukázka deklarace struktury IP adresa a port, vytvoření dvojice zdroj cíl, deklarace struktury stavu, vytvoření mapy pro ukládání stavů spojení:

```
struct s_ipv4_port{
unsigned char ip[4];
unsigned short int port;
};

pair<s_ipv4_port, s_ipv4_port> from_to;

struct s_status{
unsigned char status;
unsigned int control;
unsigned int data;
unsigned int invalid;
};

map< pair<s_ipv4_port, s_ipv4_port> , s_status > connections;
```

Jakmile je připravena dvojice definující dané spojení, probíhá větvení programu podle opcode. Vzhledem k tomu, že opcode definuje přechod, který je potřeba z aktuálního stavu udělat, provádí se následující kroky:

1. prohození směru v klíči komunikace (pokud je potřeba),
2. vyhledání struktury stavu v mapě,
3. posouzení možnosti přechodu na základě aktuálního stavu.

   V případě, že provedení přechodu je možné:

   a) vynulování čítače neplatných pokusů o přechod,
   b) inkrementace čítače datových či kontrolních paketů,
   c) kontrola přeneseného množství datových paketů a případná notifikace o detekci VPN.

   V případě, že provedení přechodu není možné:

a) inkrementace čítače neplatných přechodů,

b) kontrola povoleného množství pokusů o neplatné přechody a vymazání spojení z mapy v případě jejího překročení.

Tím je proces zpracování paketu ukončen a detekční smyčka předává ke zpracování další paket.

Při realizaci jsem se setkal s problémem, že při záchytu dat přímo z rozhraní program nepracoval správně, zatímco při použití souboru s již zachycenými daty fungoval v pořádku. Příčinou bylo přidávání dvou nulových bajtů do hlavičky Ethernetu hned za zdrojovou MAC adresu, které při zachytávání dat přímo z rozhraní předávala knihovna libpcap.

3.2 Detekce AnyConnect analýzou paketů

Jedná se, stejně jako u detekce OpenVPN analýzou paketů, o program v jazyce C++ a detekce probíhá stejným způsobem. Rozdílem je, že místo opcode protokolu OpenVPN se pro přechody používá *content_type* protokolu DTLS.

3.3 Detekce s využitím IP toků z Cisco Joy

Detekce s využitím IP toků je prováděna pomocí programu v jazyce Python. Program zpracovává vstupní IP toky ve formátu JSON a provádí výpočet charakteristik. Ty následně využívá ke klasifikaci provozu. Program se skládá ze tří tříd, které zajišťují potřebné funkcionality. Pro účely práce byl program napsán tak, aby jej bylo možné jednoduše upravovat pro vyzkoušení různých způsobů detekce. Program vznikl jak ve variantě pro zpracovávání jednosměrných IP toků, tak ve variantě pro zpracovávání obousměrných IP toků.

3.3.1 Třída flow

Třída flow zajišťuje parsování IP toků z JSON formátu a načítání potřebných dat, aby bylo možné s nimi pracovat. Konstruktor třídy dostane jako vstup řádek v JSON formátu, který rozparsuje, a poté načítá do proměnných třídy informace o IP toku, které jsou potřebné pro výpočet charakteristik. Konkrétně načítá z IP toku tyto informace:

sa zdrojová IP adresa,

da cílová IP adresa,

pr protokol,

sp zdrojový port,

dp cílový port,

bytes_out počet odeslaných bajtů,

num_pkts_out počet odeslaných paketů,

bytes_in počet přijatých bajtů,

num_pkts_in počet přijatých paketů,

time_start čas začátku komunikace,

time_end čas konce komunikace,

time čas trvání komunikace,

dns použití DNS,

flags TCP příznaky,

entropy entropie,

tls navazování TLS spojení,

b velikosti jednotlivých paketů,

wht Walsh-Wadamardova transformace,

payloadOut prvních 32 bajtů prvního odeslaného paketu,

payloadIn prvních 32 bajtů prvního přijatého paketu.

Třída obsahuje také funkci `isValid()`, která má za úkol ověřit, jestli je IP tok platný. Kontroluje se, zda není zdrojová nebo cílová IP adresa multicast a zda je IP tok obousměrný, tedy jestli v rámci IP toku proběhla výměna paketů oběma směry.

3.3.2 Třída features

Třída features je využívána k agregaci dat z více IP toků. Pro každou sledovanou IP adresu, která se komunikace zúčastní, je vytvořena jedna instance této třídy, do které se provádí agregace informací o komunikaci této IP adresy.

Konstruktor této třídy slouží pouze k inicializaci proměnných. K samotné agregaci je používána funkce `update()`. Vstupem této funkce je instance třídy flow, jejíž obsah má být agregován. Konkrétně jsou agregována tato data:

flows počet IP toků,

packets počet přenesených paketů,

ports množina použitých portů,

protocols množina použitých protokolů,

addresses mapa IP adres a počet přenesených paketů,

dns počet provedených DNS dotazů,

time čas trvání komunikace,

flagSYN počet SYN TCP hlaviček,

entropy součet entropie,

tls počet IP toků s navazováním TLS spojení,

b pole velikostí přenesených paketů,

wht složky Walsh-Hadamardovy transformace,

openVPNData počet IP toků s OpenVPN hlavičkou,

openVPNPort počet IP toků na OpenVPN portu.

Třída dále obsahuje následující funkce, které slouží pro výpočet charakteristik z agregovaných dat:

`variance()` rozptyl,

`min()` minimum,

`max()` maximum,

`maxN()` n nejvyšších hodnot,

`average()` průměr,

`sum()` součet,

`bestIP()` IP adresa, se kterou se komunikovalo nejčastěji,

`bestIPcount()` počet paketů s nejčastější komunikovanou IP adresou,

`reverseDNS()` kontroluje reverzní DNS záznam na přítomnost řetězce „vpn“,

`onBlacklist()` kontroluje přítomnost IP adresy na blacklistu.

Funkce `finalize()` je zavolána na každou instanci třídy features po skončení jednoho detekčního intervalu. Tato funkce má za úkol spočítat samotné charakteristiky a provést rozhodnutí, zda se jedná nebo nejedná o VPN komunikaci.

3.3.3 Třída detector

Třída detector zajišťuje obsluhu detekčního intervalu, načítání dat a ukládání výstupů. Obsahuje následující funkce:

`konstruktor()` slouží k inicializaci pole pro ukládání charakteristik, jejich klasifikace a samotného klasifikátoru,

`loadBlacklist()` zajišťuje načtení blacklistů z on-line zdroje,

`loadfile()` načítá soubor s IP toky,

`loadNetwork()` zpracovává zadaný rozsah sítí, pro které má být prováděna detekce,

`checkNetwork()` kontroluje, jestli IP adresa patří do rozsahu sítí, pro který má být prováděna detekce,

`printFeatures()` zajišťuje výpis napočítaných charakteristik do souboru,

`procesTimeWindow()` provádí operace související s ukončením jednoho detekčního intervalu,

`analyzeSample()` zajišťuje zpracovávání vstupního souboru s daty,

`learn()` spouští strojové učení na základě zpracovaných dat.

3.4 Detekce s využitím IP toků z NEMEA modulu

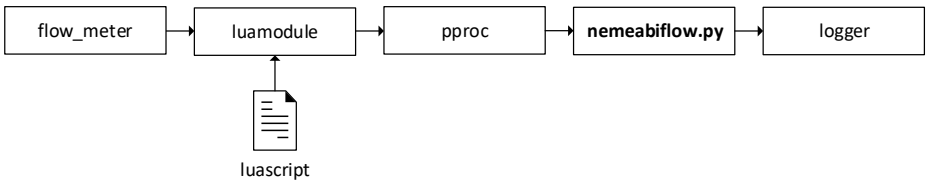
NEMEA modul pracuje na podobném principu jako detekce s využitím IP toků z Cisco Joy. Jedná se také o program v Pythonu, který ale využívá pro vstup a výstup dat standardní datové rozhraní NEMEA modulů. Díky tomu není potřeba třída flow, neboť není nutné parsovat data z JSON formátu, ale je možné k nim přistupovat přímo z objektu NEMEA zprávy.

Modul má jedno vstupní rozhraní, přes které získává IP toky, a jedno výstupní rozhraní, na které předává informace o detekovaných VPN. Při použití jednosměrných IP toků je modul připojen k rozhraní modulu *flow_meter*, odkud získává IP toky, jak ukazuje Obrázek 3.1.

Obrázek 3.1: Komunikace mezi moduly při použití jednosměrných IP toků

V případě použití obousměrných IP toků je potřeba mezi *flow_meter* a detekční modul zařadit procesovací modul *pproc*, který zajistí spojení jednosměrných IP toků do obousměrných IP toků. Protože modul *pproc* vyžaduje

jiný formát dat, než jaká jsou výstupem modulu *flow_meter*, je použit ještě konverzní modul *luamodule*, který zajišťuje přejmenovávání určitých polí datových zpráv podle zadaného skriptu. Schéma toku zpráv ukazuje Obrázek 3.2.



Obrázek 3.2: Komunikace mezi moduly při použití obousměrných IP toků

3.5 Detekce s využitím strojového učení

Pro detekci s využitím strojového učení bylo využíváno stejného programu jako pro detekci s využitím IP toků z Cisco Joy. K učení se tedy používaly datové soubory s IP toky ve formátu JSON. K samotné implementaci strojového učení byla využita knihovna sci-kit learn, konkrétně klasifikace rozhodovacím stromem. Pro učení jsou předány knihovně sci-kit learn dvě pole, jedno obsahuje záznamy s jednotlivými množinami charakteristik a druhé obsahuje klasifikaci daných charakteristik. Pro detekci poté stačí předat knihovně pouze první pole s charakteristikami a ona vrátí pole s klasifikací příslušných charakteristik.

Rozhodovací strom byl využit kvůli možnosti snadného porozumění jeho vygenerované klasifikační logice. Osvědčil se také k výběru parametrů pro předchozí způsoby detekce. Přestože vlastní testovací data rozdělil většinou na základě jedné charakteristiky, bylo možné díky němu snadno odhalit charakteristiky, které jsou pro detekci důležité. To probíhalo tak, že se charakteristika, podle které strojové učení komunikaci rozdělilo, ze souboru odstranila, a pokus se opakoval. Tím bylo možné jednoduše z množiny všech charakteristik vybrat ty nejdůležitější pro detekci.

Kapitola 4

Testování a výsledky

4.1 Detekce OpenVPN analýzou paketů

Testování tohoto detektoru probíhalo pomocí šesti serverů s různou konfigurací. Byly využity dva již existující servery a čtyři servery, které byly pro účely práce nakonfigurovány. K testování byla využita jak již zachycená data, tak zachytávání komunikace z běžícího rozhraní. Konkrétně byly testovány tyto konfigurace:

- UDP, autentizace certifikátem, tls-auth, Windows,
- TCP, autentizace certifikátem, tls-auth, Windows,
- UDP, autentizace certifikátem, bez tls-auth, Windows,
- TCP, autentizace certifikátem, bez tls-auth, Windows,
- TCP, autentizace jménem a heslem, tls=auth, Linux,
- TCP, autentizace certifikátem, bez tls-auth, RouterOS,

Pro všechny tyto konfigurace fungoval detektor bez problémů a ať už byl aktivní na rozhraní v době, kdy probíhalo navazování komunikace, nebo byl v tu dobu pořízen zpracovávaný záchyt provozu, vždy detekoval sestavení VPN tunelu. Detektor také umí zjistit, která strana je klient a která server, neboť to lze z typu zasílaných zpráv jednoduše odvodit. Při úspěšné detekci tedy program vypíše následující výpis:

```
OPENVPN DETECTED -
CLIENT: 192.168.43.91:54113 SERVER: 90.178.247.107:10103
```

4.2 Detekce AnyConnect analýzou paketů

Program pro detekci AnyConnect byl testován pouze oproti jedinému serveru, který byl realizován na hardwaru Cisco ASA 5506-X. Vzhledem k tomu, že se jedná o komerční řešení, nebylo k dispozici více serverů. Detekce navázání VPN fungovala bezchybně, ale vzhledem k realizaci na základě detekce navázání spojení protokolem DTLS bude tato metoda náchylná na falešně pozitivní detekci pro aplikace komunikující tímto protokolem. I tento protokol rozlišuje stranu klienta a serveru, což umožňuje při detekci jejich rozlišení. Při detekci VPN poskytne program následující výstup:

```
ANYCONNECT DETECTED -
CLIENT: 192.168.43.91:60120 SERVER: 195.122.193.66:8080
```

4.3 Detekce s využitím IP toků z Cisco Joy

Testování programu na detekci s využitím IP toků z Cisco Joy bylo prováděno zachycenými testovacími daty. Konkrétně se jednalo o tyto zachycené VPN komunikace:

novpnL1.json Bez VPN; Linux; 15 min; 0,05 GB

novpnL2.json Bez VPN; Linux; 45 min; 0,2 GB

novpnL3.json Bez VPN; Linux; 61 min; 0,1 GB

novpnL4.json Bez VPN; Linux; 50 min; 0,2 GB

novpnW1.json Bez VPN; Windows; 18 min; 0,1 GB

novpnW2.json Bez VPN; Windows; 45 min; 0,4 GB

protonvpnL1TCP.json ProtonVPN; Linux; TCP; 57 min; 1,2 GB

protonvpnL1UDP.json ProtonVPN; Linux; UDP; 14 min; 0,1 GB

protonvpnW1TCP.json ProtonVPN; Windows; TCP; 15 min; 0,2 GB

protonvpnW2UDP.json ProtonVPN; Windows; UDP; 20 min; 0,2 GB

protonvpnW3UDP.json ProtonVPN; Windows; UDP; 61 min; 1,2 GB

protonvpnW4TCP.json ProtonVPN; Windows; TCP; 58 min; 1 GB

protonvpnW5UDP.json ProtonVPN; Windows; UDP; 59 min; 1,1 GB

tunnelbearW1TCP.json TunnelBear; Windows; TCP; 14 min; 0,2 GB

tunnelbearW1UDP.json TunnelBear; Windows; UDP; 17 min; 0,1 GB

tunnelbearW2TCP.json TunnelBear; Windows; TCP; 30 min; 0,5 GB

windscribeL1UDP.json Windscribe; Linux; UDP; 14 min; 0,1 GB

windscribeW1IKEv2.json Windscribe; Windows; IKEv2; 15 min; 0,3 GB

windscribeW1TCP.json Windscribe; Windows; TCP; 14 min; 0,2 GB

windscribeW1UDP.json Windscribe; Windows; UDP; 18 min; 0,05 GB

windscribeW2IKEv2.json Windscribe; Windows; IKEv2; 55 min; 1,2 GB

windscribeW2TCP.json Windscribe; Windows; TCP; 38 min; 0,7 GB

windscribeW2UDP.json Windscribe; Windows; UDP; 64 min; 0,8 GB

Celkem se jednalo o téměř čtrnáct hodin zachycené komunikace s objemem přes 10 GB. Tyto komunikace byly klasifikovány programem a byla pozorována jeho úspěšnost. Protože detekce je prováděna v menších časových oknech a součástí každého okna může být více hodnocených IP adres, je výsledný počet ohodnocených vzorků komunikací vyšší než počet souborů se zachycenou komunikací.

Pro klasifikaci byla použita metoda minimum a průměr, bylo vyžadováno minimálně padesát bodů ze sta z každé charakteristiky a průměr hodnocení musel být vyšší než osmdesát bodů ze sta. Výsledky lze vidět v Tabulce 4.1. Došlo pouze ke čtyřem falešně negativním detekcím a žádná spojení nebyla klasifikována jako falešně pozitivní.

Účinnost detekce může být ovlivněna tím, že hodnotící metody byly na těchto datech odladěny. Pro potlačení tohoto vlivu byla využita metoda křížové validace. Testovací data byla rozdělena do pěti skupin, kdy jedna skupina vždy sloužila jako testovací a zbytek skupin sloužil k nastavení charakteristik hodnotícího algoritmu. Tento postup se opakoval pro všechny skupiny, aby každá skupina byla jednou testovací.

Tabulka 4.1: Testování Cisco Joy

	VPN			bez VPN		
výsledky	vzorků	chyb	úspěšnost	vzorků	chyb	úspěšnost
skupina 1	20	3	85%	9	0	100 %
skupina 2	20	0	100%	9	0	100 %
skupina 3	19	1	95%	8	0	100 %
skupina 4	20	0	100%	8	0	100 %
skupina 5	20	0	100%	8	0	100 %
celkem	99	4	96%	42	0	100 %

V testovacích datech se nachází výhradně takové VPN, které splňují všechny požadavky na ideální detekovatelnost. Tedy veškerá komunikace je směrována prostřednictvím tunelu, byla zachycena veškerá komunikace IP adresy do

internetu, známe rozsah sledovaných IP adres a každá IP adresa patří pouze jedinému počítači, není tedy spojována komunikace více počítačů za překladem adres.

Prováděno bylo také testování detektoru nad neklasifikovanými daty z reálného provozu. Tyto výsledky nelze vzhledem k chybějící klasifikaci dat objektivně prezentovat, ale probíhala manuální kontrola dat a reakcí detektoru na takováto data, na základě čehož bylo implementováno několik úprav.

4.4 Detekce s využitím IP toků z NEMEA modulu

NEMEA modul pro detekci z IP toků byl testován stejnými daty jako detektor používající IP toky z Cisco Joy a zároveň byl testován zachytáváním komunikace z běžícího rozhraní. Byl použit stejný hodnotící systém a výsledky je možné vidět v Tabulce 4.2.

Tabulka 4.2: Testování NEMEA

	VPN			bez VPN		
výsledky	vzorků	chyb	úspěšnost	vzorků	chyb	úspěšnost
skupina 1	14	4	71 %	9	0	100 %
skupina 2	15	5	67 %	9	0	100 %
skupina 3	14	2	86 %	9	0	100 %
skupina 4	14	3	79 %	9	0	100 %
skupina 5	15	2	87 %	9	0	100 %
celkem	72	16	78 %	45	0	100 %

Rozdílné výsledky mezi Cisco Joy a NEMEA jsou způsobené rozdílným exportérem IP toků a menším množstvím použitých charakteristik u NEMEA modulu.

4.5 Detekce s využitím strojového učení

Detekce s využitím strojového učení byla trénována na datech získaných ze sítě CESNET2, kterých bylo k dispozici velké množství a byla klasifikována na základě výskytu OpenVPN portu v komunikaci. Následně byla testována schopnost klasifikace natrénovaného algoritmu na vlastních klasifikovaných datech. Bylo vyzkoušeno i trénování strojového učení přímo na správně klasifikovaných datech, ale algoritmus se jim vzhledem k jejich menšímu množství snadno přizpůsobil. Ačkoliv na nich dosahoval stoprocentní úspěšnosti, na jiných datech by byla detekce nepoužitelná.

Vyzkoušeny byly dva způsoby detekce. První, který využíval ke strojovému učení všechna data, a druhý, který respektoval původní určení vzniku datové sady. Výsledky prvního z nich zobrazuje Tabulka 4.3. Je možné vidět, že navzdory tomu, jak nedostatečně byla data klasifikována, se jedná o poměrně

vysokou úspěšnost detekce, což dokazuje, že detekce VPN pomocí charakteristik je vhodným nástrojem.

Tabulka 4.3: Testování strojového učení 1

	vzorků	chyb	úspěšnost
bez VPN	42	14	67 %
VPN	99	12	88 %
celkem	141	26	82 %

Výsledky druhé metody jsou zobrazeny v Tabulce 4.4. Tato metoda kladla větší důraz na rozdělení datových sad, a proto se jí podařilo dosáhnout nulového množství falešně pozitivních detekcí. Úspěšnost detekce je sice nižší, ale to je vzhledem k menšímu počtu VPN vzorků pochopitelné.

Tabulka 4.4: Testování strojového učení 2

	vzorků	chyb	úspěšnost
bez VPN	42	0	100 %
VPN	99	48	52 %
celkem	141	48	67 %

4.6 Porovnání jednotlivých metod

Detekce na úrovni paketů nabízí velmi krátkou detekční dobu a detekci s vysokou spolehlivostí. Její detekční schopnost je zcela nezávislá na způsobu směrování komunikace skrz tunel. Nevýhodou je potřeba zpracování jednotlivých paketů, a tím velké nároky na výkon a prostor. Detekce je závislá na konkrétním protokolu, a proto nedokáže detekovat jiný než jí podporovaný protokol, což lze řešit implementací podpory pro ostatní protokoly. Protože detekuje navazování spojení, nedokáže odhalit již běžící VPN, což ale nebude v praxi představovat problém, protože lze předpokládat, že detektor poběží neustále.

Metody detekce na základě charakteristik mají velkou výhodu v nezávislosti na protokolu a nižších nárocích na prostor a výkon. Vygenerované IP toky lze navíc použít například i k dalšímu monitorování provozu na síti, jejich generování tak není pouze jednoúčelné. Nevýhodou tohoto způsobu je delší detekční doba způsobená potřebou nasbírat dostatečné množství charakteristik potřebných pro detekci. Detekce je také citlivá na to, kolik provozu je směrováno VPN tunelem, a obecně má menší spolehlivost než detekce na úrovni paketů. Tento nedostatek by se dal minimálně částečně vyřešit nasbíráním ještě většího množství vhodně klasifikovaných testovacích dat, na základě kterých by bylo možné detekci zlepšit.

Závěr

Diplomová práce se zabývala analýzou možností detekce VPN v síťovém provozu. Tomuto tématu se práce věnovala z toho důvodu, že VPN je možné mimo legitimních účelů zneužít i k obcházení kontroly provozu či exfiltraci dat. Takové chování je zejména v podnikových sítích považováno za nežádoucí a je potřeba umět takové VPN spojení detekovat.

Práce nejprve sbírala vzorová data VPN spojení, která následně analyzovala za účelem navržení vhodných detekčních mechanismů. Protože VPN je možné detekovat na několika úrovních, práce navrhla a analyzovala několik možných způsobů detekce. Jednotlivé metody byly popsány a byly analyzovány jejich požadavky na nasazení.

Na základě analýzy byly navrženy dvě základní detekční metody. První z nich je detekce na základě identifikace navazování spojení daného VPN protokolu analýzou jednotlivých paketů. Tato metoda je velmi spolehlivá a nabízí krátkou detekční dobu, vyžaduje však zpracovávání jednotlivých paketů a jejich obsahu.

Druhou použitou detekční metodou je detekce na základě charakteristik komunikace agregovaných z IP toků a jejich porovnání s obvyklou charakteristikou VPN komunikace. Tato metoda je zcela nezávislá na použitém VPN protokolu a vzhledem k používání agregovaných IP toků je méně náročná na výkon a prostor.

V práci bylo implementováno celkem pět různých detekčních programů. První dva využívají detekci na základě identifikace navazování spojení daného VPN protokolu analýzou jednotlivých paketů a mezi sebou se liší podporovanými VPN protokoly. Další tři programy využívají detekci na základě charakteristik VPN provozu s využitím IP toků a liší se mezi sebou použitými zdroji IP toků a využitými rozhodovacími metodami.

Náročnou součástí práce bylo sbírání a klasifikace síťového provozu s ohledem na VPN komunikaci. Zejména pro účely detekce na základě charakteristik bylo potřeba získat velké množství dat pro výpočet obvyklých charakteristik VPN provozu. Pro účely testování byly také využívány IP toky ze sond v síti CESNET2.

V závěru práce byly všechny programy otestovány se zaměřením na přesnost rozpoznávání a četnost falešně pozitivních výsledků. Programy byly testovány jak uloženými záznamy provozu, tak zachytáváním reálného síťového provozu. Byla také provedena diskuze o porovnání jednotlivých metod, což je pro tuto práci zásadní, neboť jejím cílem bylo zejména vyzkoušení a porovnání hned několika přístupů k detekci VPN.

Značným problémem bylo v práci zajištění správně klasifikovaných dat pro testování úspěšnosti detekce. Detekci na úrovni paketů je možné nadále rozšiřovat o další protokoly. Detektory pracující s charakteristikami provozu je možné do budoucna rozvíjet o identifikaci konkrétního protokolu VPN, například na základě známých používaných portů, nebo zlepšovat detekci získáváním většího množství testovacích dat.

Literatura

[1] Patsakis, C.; Casino, F.; Katos, V.: Encrypted and covert DNS queries for botnets: Challenges and countermeasures. *Computers & Security*, ročník 88, 2020: str. 101614.

[2] Haddon, D. A. E.; Alkhateeb, H.: Investigating Data Exfiltration in DNS Over HTTPS Queries. In *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*, 2019.

[3] Hofstede, R.; Čeleda, P.; Trammell, B.; aj.: Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials*, ročník 16, č. 4, 2014: s. 2037–2064.

[4] Perricone, P.; McGrew, D.; Anderson, B.: GitHub - cisco/joy: A package for capturing and analyzing network flow data and intraflow data, for network research, forensics, and security monitoring. 2018, [cit. 2020-03-14]. Dostupné z: `https://github.com/cisco/joy`

[5] Cejka, T.; Bartos, V.; Svepes, M.; aj.: NEMEA: A Framework for Network Traffic Analysis. In *12th International Conference on Network and Service Management (CNSM 2016)*, 2016.

[6] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; aj.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, ročník 12, 2011: s. 2825–2830.

[7] Boutaba, R.; Salahuddin, M. A.; Limam, N.; aj.: A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, ročník 9, č. 1, Jun 2018, doi:10.1186/s13174-018-0087-2. Dostupné z: `http://dx.doi.org/10.1186/s13174-018-0087-2`

[8] Ovsienko, D.; Abdulkadir, A.; Richardson, M.; aj.: the-tcpdump-group/libpcap: the LIBpcap interface to various kernel packet capture

mechanism. 2020, [cit. 2020-04-06]. Dostupné z: https://github.com/the-tcpdump-group/libpcap

[9] Perricone, P.; McGrew, D.; Anderson, B.: joy/wht.c at master · cisco/joy · GitHub. 2018, [cit. 2020-03-11]. Dostupné z: https://github.com/cisco/joy/blob/master/src/wht.c

[10] Liaudat, N.: GitHub - ejrv/VPNs: List of datacenter & VPN IP addresses. 2020, [cit. 2020-03-22]. Dostupné z: https://github.com/ejrv/VPNs

[11] Karger, S.; Sommerseth, D.; Schwabe, A.; aj.: openvpn/ssl.h at master · OpenVPN/openvpn · GitHub. 2019, [cit. 2020-02-09]. Dostupné z: https://github.com/OpenVPN/openvpn/blob/master/src/openvpn/ssl.h

[12] Dierks, T.; Rescorla, E.: *The Transport Layer Security (TLS) Protocol Version 1.2 [online], organization = The Internet Engineering Task Force.* srpen 2008, [cit. 2020-02-09]. Dostupné z: https://tools.ietf.org/rfc/rfc5246.txt

[13] Deri, L.; Campus, M.; Faranda, E.; aj.: nDPI/openvpn.c at dev · ntop/nDPI · GitHub. 2020, [cit. 2020-03-22]. Dostupné z: https://github.com/ntop/nDPI/blob/dev/src/lib/protocols/openvpn.c

[14] Alcock, S.: libprotoident/lpi_openvpn.cc at master · wanduow/libprotoident · GitHub. 2019, [cit. 2020-03-22]. Dostupné z: https://github.com/wanduow/libprotoident/blob/master/lib/udp/lpi_openvpn.cc

Příloha A

Seznam použitých zkratek

VPN Virtual Private Network

HMAC Keyed-hash Message Authentication Code

TLS Transport Layer Security

TCP Transmission Control Protocol

UDP User Datagram Protocol

VoIP Voice over Internet Protocol

ISP Internet Service Provider

FTP File Transport Protocol

IP Internet Protocol

JSON JavaScript Object Notation

DPI Deep Packet Inspection

NEMEA Network Measurements Analysis

DTLS Datagram Transport Layer Security

CESNET Czech Education and Scientific Network

Příloha B

Obsah přiloženého CD

- `readme.txt` ... stručný popis obsahu CD, návod na spuštění implementací
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `data` ..... vybraná testovací data
  - `impl` ..... zdrojové kódy implementace
  - `pictures` ..... použité obrázky a jejich zdrojový kód
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF