Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Analyzátor VPN komunikace

Plzeň 2022 Aneta Koldovská

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Aneta KOLDOVSKÁ**
Osobní číslo:	**A18B0238P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Analyzátor VPN komunikace**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Proveďte srovnání vybraných VPN protokolů.
2. Vytvořte testovací scénáře VPN komunikace, zachyťte a analyzujte síťový provoz nástrojem pro analýzu síťových protokolů (např. Wireshark).
3. Implementujte vlastní síťový analyzátor a použijte ho pro detekci účastníků VPN komunikace.
4. Dle výsledků srovnání ruční analýzy zachyceného provozu a implementovaného analyzátoru vyhodnoťte jeho úspěšnost.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Martin Šimek, Ph.D.**
Centrum informatizace a výpočetní techniky

Datum zadání bakalářské práce: **4. října 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V textu práce jsou použity názvy produktů, software a firem, které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

V Plzni dne 2. května 2022

Aneta Koldovská

Poděkování

Děkuji vedoucímu této bakalářské práce Ing. Martinovi Šimkovi, Ph.D za jeho rady, ochotu a čas který mi věnoval při konzultacích. Také bych ráda poděkovala Ing. Antonínovi Vrbovi za věcné připomínky a praktické rady při implementaci.

Abstract

This thesis focuses on the development of a program which analyzes VPN communication and detects its participants.

First, the general concept and function of VPN is explained. Then, the principles of operation of selected VPN protocols are examined. Subsequently, test scenarios of this communication are designed and the existing detection options are described and compared. The resulting analyzer is then tested using the same scenarios and its success rate is evaluated. The resulting application will be used to analyze captured network traffic from the dormitory network of the University of West Bohemia.

Abstrakt

Tato bakalářská práce se zabývá vývojem programu, který bude sloužit pro analýzu VPN komunikace a detekci jejich účastníků.

Nejdříve je objasněn obecný koncept a fungování VPN. Poté jsou prozkoumány principy fungování vybraných VPN protokolů. Následně jsou navrženy testovací scénáře této komunikace a popsány a srovnány již existující možnosti detekce. Na stejných scénářích je poté výsledný analyzátor otestován a je zhodnocena míra jeho úspěšnosti. Výsledná aplikace bude využívána k analýze zachyceného síťového provozu z prostředí kolejní sítě Západočeské univerzity.

Obsah

1 Úvod

Od doby, co vznikla celosvětová síť Internet a lidé si jejím prostřednictvím začali sdílet informace, byl kladen důraz především na bezpečnost přenášených informací. Bylo nutné a chtěné docílit toho, aby protokoly, které byly (a většina z nich stále ještě je) pro internetovou komunikaci využívány, byly schopny data zabezpečit tak, aby nedošlo k jejich zneužití či aby se data nedostala k nesprávným lidem. S rostoucí popularitou tohoto média rostly i nároky nejenom na ochranu dat u koncových uživatelů, kteří data zpracovávají, ale i na jejich bezpečnost během samotného síťového přenosu. Právě tehdy se začaly objevovat první virtuální privátní sítě neboli *VPN*, které dokázaly vytvořit bezpečné spojení mezi uživateli, kteří spolu chtěli komunikovat a zároveň nechtěli riskovat ztrátu nebo kompromitaci sdílených informací.

Samotný koncept VPN je i dnes na vzestupu. Někteří uživatelé mohou využít těchto sítí například pro to, aby se připojili na webové stránky, které jsou za normálních okolností blokovány a přístup k nim je omezen či úplně znepřístupněn. VPN tedy může sloužit také k obcházení takových restrikcí, což je hlavní důvod pro detekování a případné omezování provozu virtuálních privátních sítí.

Tato práce se ve své teoretické části zabývá fungováním VPN protokolů a principem jejich komunikace. Nejdříve stručně popisuje základní fungování těchto sítí a jejich využití v praxi. Dále práce pojednává o komunikačních topologiích a režimech komunikace. Nechybí ani popis tunelovacích mechanismů z pohledu operačních systémů. Poté se práce zaměřuje již na konkrétní VPN protokoly. Podrobněji popisuje průběh navazování komunikace vybraných protokolů VPN a možnosti jejich detekce, včetně již existujících řešení.

V praktické části je cílem naprogramovat analyzátor IPv4 síťového provozu, který by umožňoval tyto vybrané VPN protokoly detekovat z předem zachycené síťové komunikace a zároveň by dokázal rozpoznat jejich účastníky. Součástí praktické části práce je také vytvoření typických komunikačních scénářů se zachycením datového provozu a jeho pozdější ruční analýza.

Na závěr dojde k porovnání výsledků této analýzy a vzniklého analyzátoru, ze kterého se stanoví míra úspěšnosti detekce VPN komunikace.

2 Virtuální privátní sítě

V této kapitole bude stručně popsán koncept virtuálních privátních sítí, princip jejich fungování a jejich využití v praxi. Zároveň zde proběhne základní rozdělení VPN. Dále budou vybrány tři nejpoužívanější protokoly, které budou popsány podrobněji.

2.1 Koncept VPN

Koncept virtuálních privátních sítí spočívá v zajištění určité míry soukromí přenášených informací (ať už se jedná o IP adresy či o data) mezi komunikujícími stranami. Cílem tedy je zajistit dostatečnou formu privátnosti dat a odstínění probíhající konkrétní komunikace od mezilehlých nebo okolních uzlů sítě pomocí privátního spojení.

Dříve se takovéto privátní spojení využívalo zejména v pracovním prostředí, kde firmy potřebovaly bezpečný způsob, kterým by si mohly mezi sebou (například mezi jednotlivými pobočkami) vzdáleně předávat informace či důležité firemní soubory. Zaměstnanci zároveň začali využívat těchto protokolů k připojování se například do svého pracovního zařízení, když se fyzicky nacházeli mimo firemní prostory a zároveň potřebovali přístup k datům [18].

Po nějaké době se začaly VPN šířit i mezi běžné uživatele. Nebyla to tedy již výsada pouze firem, ale i ostatních uživatelů, kteří chtěli svá data zabezpečit [37].

K tomuto významně dopomohlo několik bezpečnostních incidentů, které se uskutečnily na přelomu století, známé také jako „Operace Aurora“. Při těchto bezpečnostních incidentech byla odcizena citlivá data několika velkých společností jako například Google, Adobe, Yahoo či Symantec [9]. Lidé se tak i kvůli těmto hrozbám více strachovali o svá online data a chtěli je více chránit před podobnými útoky a zneužitím. Začali tedy hledat způsoby, jak tohoto docílit. Výsledkem byla zvýšená poptávka po takové technologii, která by tyto požadavky uživatelů splňovala, což vedlo k rozvoji právě VPN služeb.

2.2 Princip fungování VPN sítí

Při standardní internetové komunikaci má každé zařízení přiřazenou svoji IP adresu, kterou obvykle dostává od DHCP[1] serveru nebo mu je přidělena administrátorem staticky. S pomocí adres si mezi sebou dokáží jednotlivá zařízení posílat data, která putují počítačovými sítěmi ve formě paketů. To, jakou IP adresu přístroj zrovna dostane, závisí mimo jiné také na místě, odkud se uživatel připojuje. IP adresa přidělená při připojení v domácí síti bude jiná než IP adresa, kterou stejné zařízení dostane v kavárně či ve škole.

Při použití VPN může být veškerá komunikace posílána přes vzdálený server, tudíž poskytovatel internetového připojení (dále jen ISP[2]) nemůže vidět aktivitu uživatelů, z důvodu jejího šifrování a tunelování. Stejně tak například webové stránky, které uživatel navštíví, nemohou odhalit skutečnou IP adresu používaného zařízení, protože je maskována adresou výše zmíněného vzdáleného VPN serveru. Pokud se tedy tento server nachází například v jiném státě, pak to z pohledu ostatních serverů, ISP či klientských zařízení vypadá, že se uživatel připojuje a komunikuje z jiné geolokace, než ve které se ve skutečnosti nachází [23].

Kromě uživatelských VPN služeb existují také podnikové (enterprise) VPN, které fungují na principu značkování provozu a využívají se ve WAN[3] sítích. Zde je však nutné jednotlivé datové toky od sebe odlišit, aby bylo zajištěno jejich doručení na správné místo. Odlišení konkrétního VPN toku je zaručeno označením, které může být realizováno na úrovni Ethernetových rámců (využíváno v technologii jako například VLAN[4]), IP paketů (využíváno u MPLS[5]) nebo zapouzdřením celé komunikace do jiného paketu s odpovídajícím označením VPN (využíváno v GRE[6]) [3] [49] [26].

Princip fungování těchto sítí znázorňuje obrázek 2.1.

[1]Dynamic Host Configuration Protocol – automatické přidělení síťových parametrů

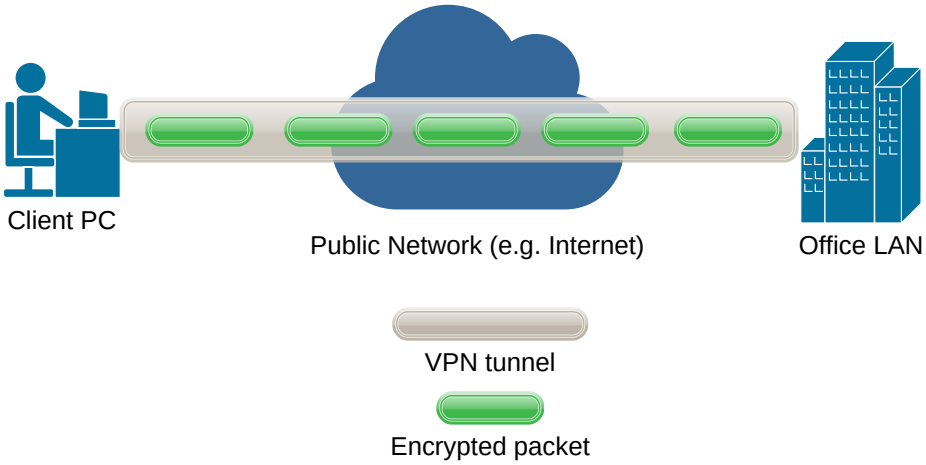
[2]Internet Service Provider – poskytovatel internetového připojení

[3]Wide Area Network – rozsáhlá počítačová síť, pokrývající často celé geografické území

[4]Virtual Local Area Network – virtuální lokální síť

[5]Multiprotocol Label Switching – jedna z metod směrování síťového provozu

[6]Generic Routing Encapsulation – protokol, určený k zapouzdření paketů jednoho protokolu do protokolu jiného



Obrázek 2.1: Princip fungování VPN sítí. Převzato z [46].

2.3 Využití VPN sítí

Účel a smysl v použití jakékoliv VPN služby spočívá především ve vytvoření zabezpečené privátní sítě (tunelu) přes veřejné síťové spojení. Dnes se tyto tunely využívají primárně k několika hlavním účelům: [37]

1. Šifrování informací během síťového přenosu z důvodu rostoucího počtu bezpečnostních útoků a sledování uživatelů při jejich online aktivitách.

2. Bezpečné propojení celých podsítí. VPN tunel se vytvoří mezi dvěma směrovači, čímž dojde k zajištění privátní komunikace mezi dvěma lokálními sítěmi.

3. Zvýšení ochrany dat v době, kdy je uživatel připojen k nijak nezabezpečené veřejné síti, například na svém mobilním zařízení či notebooku. Nejčastěji se jedná o bezdrátové sítě na veřejně dostupných místech jako jsou kavárny nebo restaurace.

4. Obcházení geo-blokace (omezení přístupu k Internetu na základě geografické polohy uživatele). Nejčastěji se tento princip využívá k omezení přístupu uživatelům ke konkrétním videím, webovým stránkám, či sociálním sítím, které se nacházejí například v jiném státě, ve kterém platí odlišná internetová politika.

2.4 Rozdělení VPN sítí

VPN sítě se mohou rozdělit do různých skupin pomocí rozdílných kritérií. Lze je dělit například dle:

- způsobu využití – mobilní, osobní, remote access [27];
- typu připojení – uživatelské, síťové [6];
- typu navazování tunelu – statické, dynamické [47];
- typu topologie – Site-to-Site, Hub and Spoke, Full Mesh [8].

To, kde se daná skupina využije, závisí například na typu prostředí, ve kterém bude VPN služba spuštěna, ale také může záviset na počtu uživatelů nebo velikosti firem/organizací, které pomocí ní budou komunikovat.

V následující části práce bude podrobněji popsána výše zmíněná skupina. Konkrétně se bude jednat o rozdělení dle typu topologie.

2.4.1 Site-to-Site

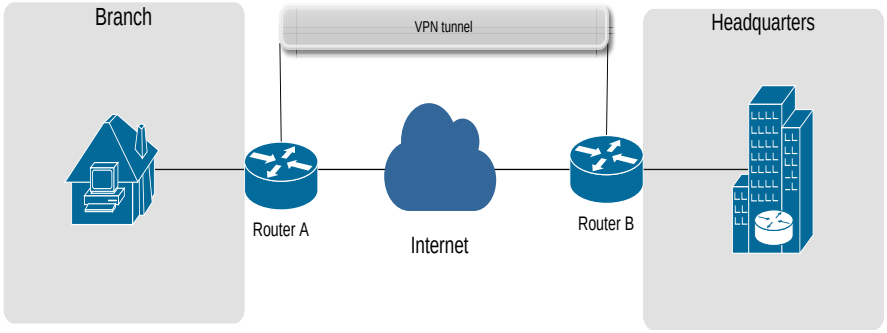
Tato topologie umožňuje propojení a přenos dat mezi dvěma sítěmi. Je určena spíše pro firemní prostředí a využívají ji zejména takové firmy, které mají oddělené pobočky na různých místech, mezi kterými potřebují komunikovat a sdílet informace. Propojuje tedy dvě a více LAN[7] sítí (samostatné pobočky), jak je znázorněno na obrázku 2.2.

Site-to-Site umožňuje vytvořit uzavřený tunel (tzv. vnitřní síť), přes který lze přistupovat k jednotlivým pobočkám dané organizace. Nejčastěji se v této topologii využívá protokolu *IPSec*, pomocí kterého se tunel navazuje a udržuje. Tento protokol a jeho fungování bude podrobněji rozepsáno v další části této práce. Někdy se této topologii také říká Intranet či Gateway-to-Gateway. Vyžaduje speciální hardware (většinou se jedná o směrovače s podporou VPN). Jednotliví uživatelé si tak v rámci lokální sítě (pobočky) nemusejí instalovat VPN klienty na všechna svá zařízení, ale spojení mají zajištěno právě prostřednictvím těchto směrovačů.

2.4.2 Hub and Spoke

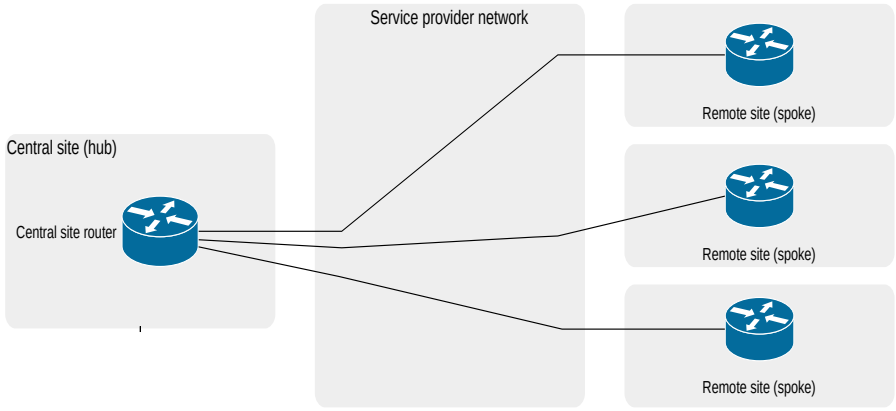
Této topologii se někdy také říká multi-site. Klíčovou roli zde hraje centrální prvek (Hub), ke kterému jsou připojeny ostatní prvky v hierarchii (Spokes).

[7]Local Area Network – lokální počítačová síť na omezeném území (nejčastěji škola, domácnost, kavárna, ...)



Obrázek 2.2: Site-to-Site topologie. Převzato z [2].

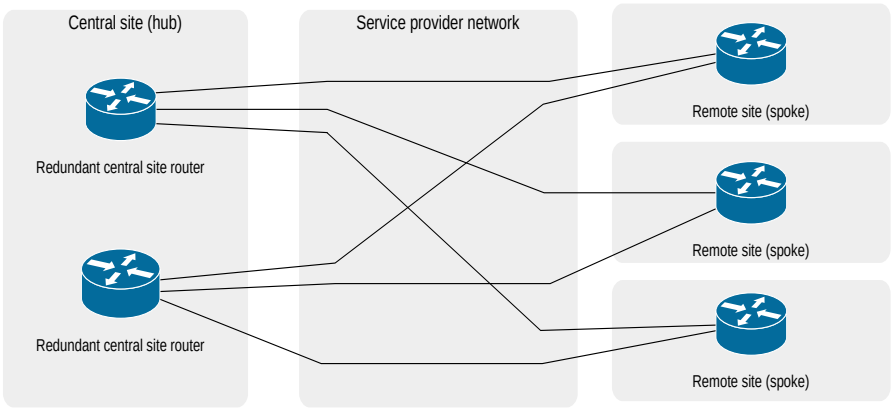
Jako hub je obvykle použit VPN router či VPN koncentrátor (funkčnost tohoto prvku je popsána na straně 11 této práce).



Obrázek 2.3: Hub and Spoke topologie. Převzato z [26].

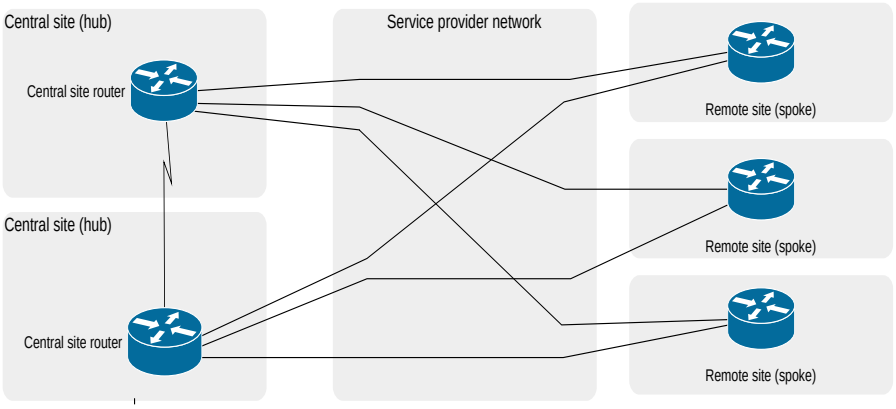
Hub and Spoke se také (stejně jako již popsaná Site-to-Site) nejčastěji využívá ve firemních prostředích. Na rozdíl od předchozí topologie se ale pomocí této připojuje více poboček k jedné centrále (hubu), přičemž všechny pobočky tak mohou komunikovat se všemi prostřednictvím zmíněného centrálního místa (viz obrázek 2.3).

Hub je tedy nekritičtější součást celé této topologie (pokud dojde k jeho poškození, celá topologie obvykle zůstane nefunkční). Z toho důvodu se často můžeme setkat s poupravenou formou Hub and Spoke, kde dojde k redundanci (navýšení počtu) VPN koncentrátorů či směrovačů. Zde je pak také rozdíl v umístění obou prvků. Mohou se nacházet v jedné centrále (jak je ukázáno na obrázku 2.4), či se může jednat o dvě různé centrální pobočky.



Obrázek 2.4: Hub and Spoke topologie s redundantními routery. Převzato z [26].

Tuto modifikaci znázorňuje obrázek 2.5. Změn může být i více. Vždy však záleží na konkrétním využití a cíli dané organizace.



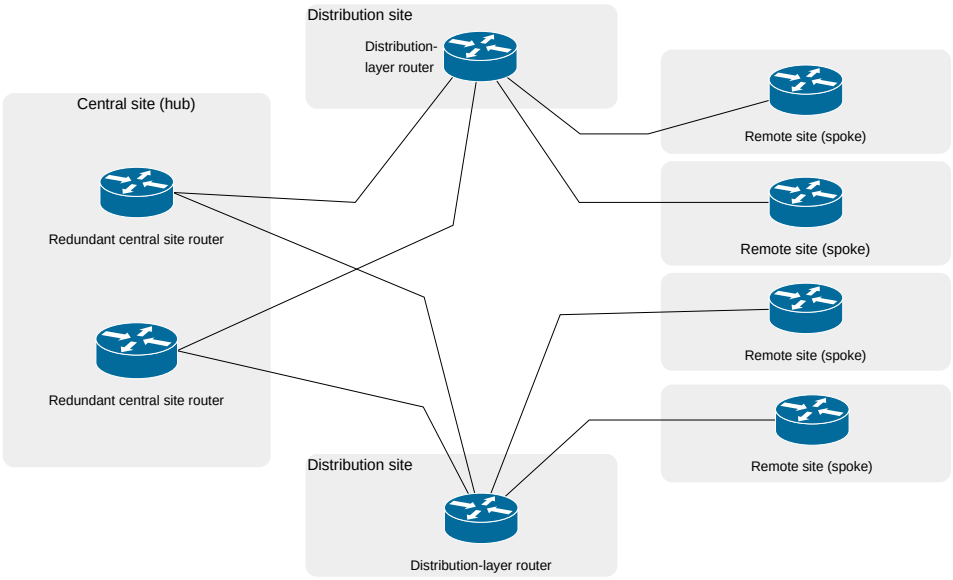
Obrázek 2.5: Hub and Spoke topologie se dvěma centrálními místy. Převzato z [26].

Pokud se firma rozrůstá (například vznikají nové pobočky), vzrůstají i požadavky na síťovou topologii: Nutnost připojit nově vybudované provozovny, modifikovat nastavení routovacích protokolů, které jsou v síti využívány nebo umožnit novým pobočkám komunikaci prostřednictvím zabezpečeného tunelu (VPN). Takto postupně dochází k různým úpravám dané struktury. Z jednoduché Hub and Spoke se tak stává tzv. víceúrovňová (multilevel) nebo také rekurzivní topologie, která má navíc ještě distribuční

vrstvu (další směrovače), která předává komunikaci mezi hubem a klienty (viz obrázek 2.6).

V případě, že obsah a tok dat mezi pobočkami je natolik velký, že představuje většinu síťového provozu (a hrozí tedy zahlcení koncentrátoru či směrovače) nebo je potřeba umožnit komunikaci i jednotlivým pobočkám přímo mezi sebou navzájem (tedy bez použití centrály) přistupuje se k výběru tzv. Partial nebo Full Mesh topologie [26]. Ty budou představeny v následující sekci 2.4.3.

Alternativní možností je využití dynamických VPN topologií (například Cisco DMVPN[8] nebo Fortinet ADVPN[9]). Tyto topologie umožňují navazovat a vytvářet VPN spojení až ve chvíli, kdy je to potřeba a dokážou zprostředkovat komunikaci mezi jednotlivými pobočkami, aniž by k tomu musel být využitý Hub.



Obrázek 2.6: Víceúrovňová Hub and Spoke topologie. Převzato z [26].

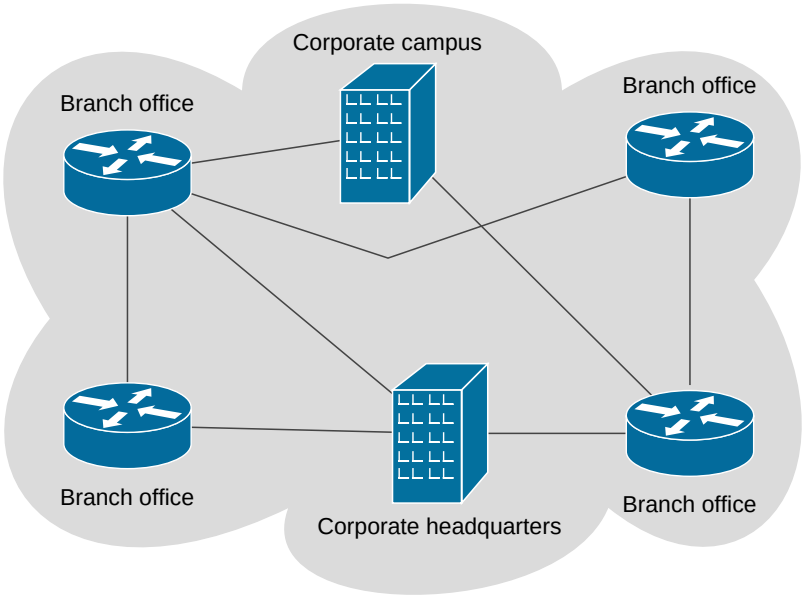
[8] https://www.cisco.com/c/en/us/products/collateral/security/dynamic-multipoint-vpn-dmvpn/data_sheet_c78-468520.html

[9] https://community.fortinet.com/t5/FortiGate/Technical-Tip-Fortinet-Auto-Discovery-VPN-ADVPN/ta-p/195698

2.4.3 Partial nebo Full Mesh

Oproti výše zmíněné topologii se částečná (partial) nebo Full Mesh topologie liší především v použití peer-to-peer architektury, tj. každá pobočka (nebo jejich většina) je propojena s ostatními a může s nimi komunikovat bez použití centrálního místa (VPN koncentrátoru nebo routeru). Jedná se tedy o decentralizovanou formu propojení. Toto řešení je více odolné oproti výpadku centrálního prvku v Hub and Spoke. Výpadek v této topologii tedy ovlivní pouze části dané sítě, které jsou na tomto prvku přímo závislé.

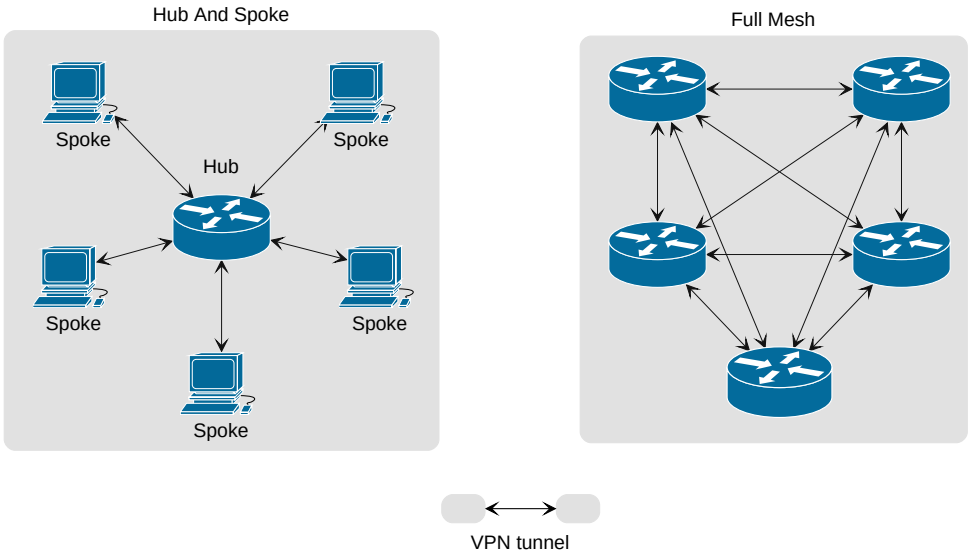
Dříve se využívala spíše Hub and Spoke topologie, protože většina síťových zařízení měla VPN podporu (firewally, routery), a tak bylo pro korporátní prostředí výhodnější využít spíše tyto prvky a sestavit tak centrální místo, přes které se posílala VPN komunikace dál. Postupem času se však zvýšila poptávka po možnosti pracovat vzdáleně, firmy začaly více využívat také cloudové řešení a virtuální stroje, které také musely mít přístup na VPN připojení, budovaly se nové pobočky, apod. Toto všechno vedlo k tomu, že se začalo více využívat právě tohoto typu topologie, tedy buď částečná Mesh (ne všechny oblasti jsou propojeny se všemi navzájem, ale mezi sebou může komunikovat pouze několik z nich (viz obrázek 2.7) nebo Full Mesh topologie, při které dojde ke spojení úplně všech).



Obrázek 2.7: Partial Mesh topologie. Převzato z [33].

Nejčastěji se tato topologie implementuje pomocí softwarového řešení (například *ZeroTier*[10], *ExpressRoute*[11] nebo *Nebula*[12]), jelikož u hardwarových komponent je častěji nutná rekonfigurace či jiný zásah administrátora [10]. Hlavní nevýhodou této topologie je především vyšší provozní údržba a nákladnější implementace, jelikož se musí zajistit (zakoupit) VPN služba pro každé zařízení v dané topologii. U použití úplné mesh sítě je důležité, aby v případě přidání nového uzlu došlo k jeho propagaci k již stávajícím uzlům v dané topologii [33].

Rozdíl mezi Hub and Spoke topologií a Full Mesh znázorňuje obrázek 2.8.



Obrázek 2.8: Hub and Spoke vs Full Mesh. Převzato z [1].

[10] https://www.zerotier.com/
[11] https://azure.microsoft.com/en-us/services/expressroute/
[12] https://github.com/slackhq/nebula

2.5 VPN koncentrátor

Jedná se o síťové zařízení, které poskytuje VPN spojení a přeposílá data mezi uzly (koncovými body topologie). Vytváří a udržuje tunely a umožňuje vícero tunelům využít stejnou síť. Umí tedy udržovat vícero spojení najednou a poskytovat zabezpečenou formu komunikace mezi rozdílnými koncovými stanicemi.

Pokud přijde na rozhraní (port) VPN koncentrátoru paket, který má být směrován ven do Internetu (jedná se tedy o odchozí paket), je jeho obsah zašifrován, přičemž způsob použitých šifrovacích mechanismů závisí na typu použité VPN služby. Nejčastěji se jedná o IPSec nebo SSL [39] [24]. Takto upravená data posílá toto zařízení dál do VPN tunelu, přičemž výběr správného tunelu se provádí na základě informací ve směrovací tabulce.

V opačném případě, kdy na port koncentrátoru přijde příchozí paket (tedy jde z vnějšku z tunelu do vnitřní sítě), probíhá opačný proces. Paket je zbaven přidané extra vrstvy, jeho obsah je dešifrován a poslán dál.

Kromě udržování komunikace se koncentrátor také stará o autentizaci (ověření) uživatelů, přiděluje IP adresy z daných VPN rozsahů (nejčastěji se jedná o privátní adresy) a uchovává informace o kryptografických klíčích.

Koncentrátory bývají součástí routerů a rozšiřují tak jejich funkčnost. Najdeme je ve výše popsaných topologiích. Zastávají tak například funkci centrálního prvku v topologii Hub and Spoke nebo v případě využití dvou VPN koncentrátorů ve dvou sítích lze utvořit Site-to-Site.

Některé výkonné VPN koncentrátory využívají ASIC[13] obvody pro IPSec a SSL offloading (šifrovaná komunikace se dešifruje s využitím zmíněného obvodu přímo na koncentrátoru a k cílovému serveru jde již v čitelné podobě, což vede ke snížení výpočetní zátěže CPU serveru a komunikace tedy probíhá rychleji) [32].

2.6 VPN režimy

Různé VPN topologie mohou fungovat v různých režimech. Vždy však záleží na použitém protokolu (jiné režimy má například OpenVPN a jiné IPSec (viz sekce 2.6).

V následující části práce budou představeny dva základní režimy VPN, a to *Tunnel* a *Transport*.

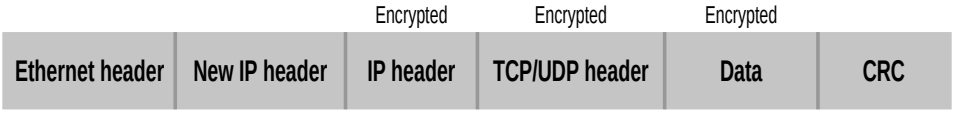
[13] Application-Specific Integrated Circuit

2.6.1 Tunnel

V tomto režimu je celý paket zapouzdřen do nového paketu a je k němu přidána nová IP hlavička (místo, kam se umisťuje například zdrojová a cílová IP adresa). Tato nová hlavička obsahuje mimo jiné také adresy VPN routerů (koncentrátorů), které umožňují skrýt původní směrovací údaje, což může být učinné zejména při nežádoucí analýze provozu, jelikož útočník je schopen identifikovat pouze přidané hlavičky a ty původní nezná. Zároveň ostatní routery, přes které je VPN provoz směrován ví, ke kterému cílovému bodu (výstupnímu bodu tunelu) má dané pakety doručit. Na konci tunelu se tyto přidané informace odstraní a router určí z původních IP adres příjemce, kterému paket doručí.

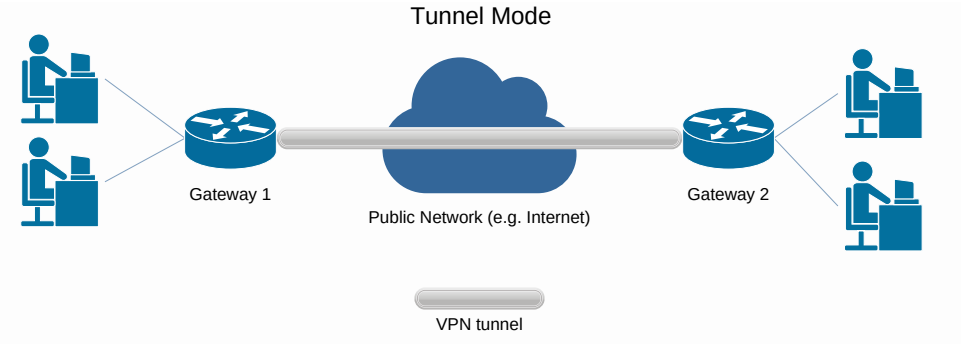
Tento režim je užitečný zejména pro zabezpečení síťového provozu mezi dvěma rozdílnými sítěmi. Využívá se zejména v Site-to-Site topologiích. Podporuje *NAT*[14]. Je jednoduchý na vytvoření a spolu s Transport režimem je součástí například IPSecu, který bude podrobněji popsán v další sekci. Tvorba a zapouzdření paketů se liší a závisí na použití konkrétního IPSec protokolu (viz sekce 2.6.1).

Tunnel režim může použít nejenom VPN brána, ale i klient (na rozdíl od transportního režimu, který bude popsán později). Mezi jeho hlavní nevýhody patří větší režie (z důvodu přidávání IP hlavičky) a menší MTU[15] hodnota, která udává maximální počet dat, které lze přenést v rámci jednoho paketu na konkrétní vrstvě [50] [28]. Formát paketu je na obrázku 2.9 a schématické znázornění tohoto režimu je na obrázku 2.10.



Obrázek 2.9: Formát paketu v režimu Tunnel. Převzato z [36].

[14]Network Address Translation – Překlad IP adres
[15]Maximum Transmission Unit



Obrázek 2.10: Režim Tunnel. Převzato z [50].

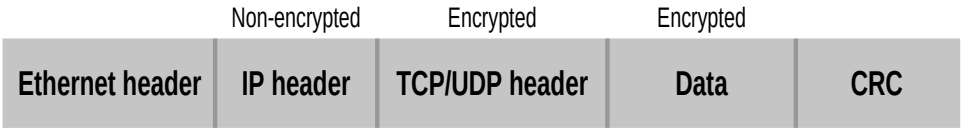
2.6.2 Transport

Transportní režim se od výše popsaného liší zejména tím, že nechává originální směrovací informace uvnitř hlavičky paketu (v nešifrované podobě). Nepřidává tedy žádné další informace navíc, což vede k menší režii. Zabezpečuje při přenosu pouze data, která jsou v paketu uložená. Šifrovaná data se posílají v tomto režimu napřímo mezi dvěma zařízeními, které spolu komunikují. Zabezpečuje tak například komunikaci typu klient-server, přičemž průběh přenosu opět závisí na typu použitého protokolu IPSecu [5]. Většinou se využívá ve chvíli, kdy komunikace neprobíhá mezi dvěma VPN routery, ale chtějí spolu komunikovat například dvě koncová zařízení (počítač, telefon, apod.), na kterých je nainstalován VPN program.

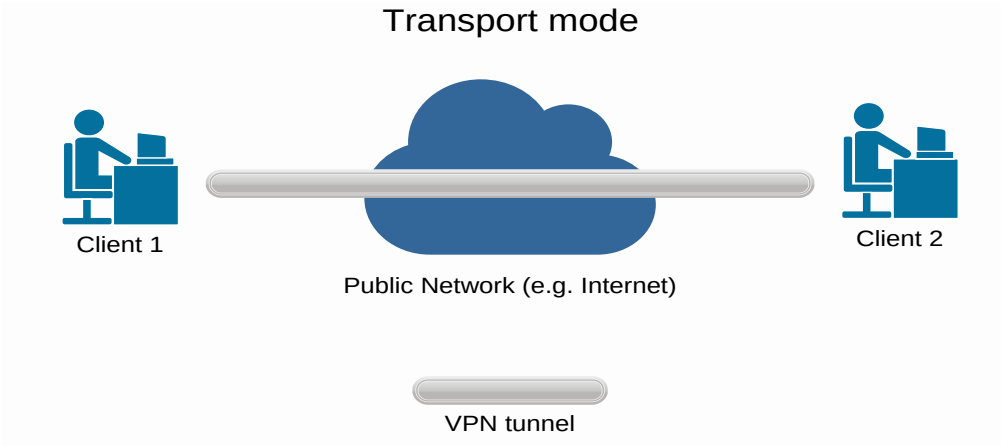
Tento režim má větší MTU, protože zde není nic navíc. Původní informace v hlavičce paketu se ponechávají v nešifrované podobě a nejsou zapouzdřeny v jiné. Je tedy považován za méně bezpečný než výše popsaný Tunnel mód.

Transport režim nepodporuje NAT a jeho hlavní nevýhodou je zejména horší kompatibilita s některými směrovači. Z tohoto důvodu se nemůže využívat v Site-to-Site topologiích [50].

Schématické znázornění tohoto režimu je na obrázku 2.12 a formát paketu v tomto režimu je vidět na obrázku 2.11.



Obrázek 2.11: Formát paketu v režimu Transport. Převzato z [36].



Obrázek 2.12: Režim Transport. Převzato z [50].

2.7 VPN z pohledu operačního systému

Ke správné funkčnosti a provozu VPN služeb na zařízeních je potřeba určitá pomoc jádra (kernelu) operačního systému. Konkrétně se jedná o možnost vytvořit a udržovat tzv. virtuální (softwarové) síťové rozhraní, které přijímá a odesílá pakety nikoliv přes fyzické síťové rozhraní počítače, ale přes tzv. „user space“ (uživatelský režim), což je část operačního systému, ve kterém běží uživatelské procesy.

Schopnost vytvořit toto virtuální síťové rozhraní mají dnes jádra všech operačních systémů (již to tedy není záležitost pouze Linuxu, ale zvládne to i Windows či MacOS).

Paket, který přijde na toto rozhraní projde procesem rozbalení (odstraní se IP hlavička a další subvrstvy) a obsah, který je uvnitř se zapíše do systémového souboru [43].

Uživatelský proces (třeba spuštěný konkrétní VPN klient (například NordVPN nebo TunnelBear) tedy může využívat tyto soubory pro zapisování

či čtení síťových dat. Předávání těchto dat mezi virtuálními a fyzickými rozhraními zajišťuje jádro OS [30].

Z důvodu kompatibility a zajištění správné funkčnosti síťové komunikace musejí být oba konce virtuálních sítí nastaveny na stejný typ rozhraní.

2.7.1 Tun

Jedná se o jedno ze dvou základních virtuálních rozhraní. Funguje na třetí vrstvě ISO/OSI[16]. Umožňuje manipulaci pouze s PDU[17] síťové vrstvy, tedy s pakety.

Typicky se využívá tehdy, když je potřeba vytvořit IP tunel (point-to-point spojení). Dále se hodí pro navázání a udržování VPN komunikace, jelikož použití těchto rozhraní umožňuje VPN klientům šifrovat a dešifrovat data předtím, než se pakety dostanou na fyzickou vrstvu [30].

Jak již bylo zmíněno, Tun dokáže fungovat jedině na síťové vrstvě, takže jej nelze využít při tvorbě síťového mostu (bridge), jelikož nedokáže manipulovat s PDU nižších vrstev. Neumožňuje posílání všesměrové komunikace (pakety, které jsou přijímány všemi aktivními zařízeními v dané síti) [15]. Tato vlastnost je však u VPN provozu většinou žádoucí, jelikož se broadcast či multicast provoz nedá poslat přes VPN tunel přímo. Tento typ komunikace tedy bývá před odesláním nejdříve zapouzdřen například pomocí GRE protokolu a následně je tunelován prostřednictvím VPN. Tuto vlastnost má například IPSec [52].

2.7.2 Tap

Druhé z virtuálních rozhraní je Tap. Od výše popsaného se liší zejména tím, že funguje na druhé vrstvě ISO/OSI modelu. Pracuje tedy s čistě Ethernetovými rámci. Není omezen jen na point-to-point spojení, může být součástí síťového mostu (bridge) a podporuje odesílání a přijímání broadcastu.

Nejčastěji se toto nastavení využívá v případě tvorby virtuálního přemostění síťových karet nebo při virtualizaci operačních systémů, které mezi sebou potřebují vzájemně komunikovat [51].

[16] https://www.iso.org/ics/35.100/x/

[17] Protocol Data Unit

2.8 Princip fungování vybraných VPN protokolů

V následujících částech práce budou představeny tři známé VPN protokoly [45]. Budou zde popsány principy jejich fungování, navazování a průběhy komunikace a tvorby tunelů, šifrování zpráv, a také struktura dat.

2.8.1 IPSec

IP Security či zkráceně IPSec (definován v RFC 6071 [19]), patrně nejznámější VPN protokol, jehož součástí je sada dalších protokolů, které zabezpečují síťovou komunikaci. Konkrétně se jedná o *ESP*[18], *AH*[19], *IKE*[20] a *ISAKMP*[21]. Komunikace probíhá prostřednictvím *UDP* či *TCP* protokolu a vytvořeného tunelu. Přenášená síťová data jsou zabezpečena pomocí šifrovacích a autentizačních mechanismů [13]. Zajišťuje integritu přenášených dat, jejich šifrování a autentizaci komunikujících zařízení. IPSec v dnešní době využívá k autentizaci i k šifrování obsahu zpráv sadu šifrovacích algoritmů Suite B [29].

Dále definuje způsob, jakým bude probíhající komunikace zabezpečena. Jinými slovy říká, který z protokolů použít či jakým způsobem bude provedena výměna bezpečnostních klíčů při navazování spojení. Komunikující strany se na tomto dohodnou při tvorbě tunelu. Konkrétně jsou tyto informace obsaženy v tzv. Security Associations (dále jen SA).

Pro porozumnění fungování tohoto protokolu je zapotřebí popsat a pochopit principy chování jeho jednotlivých částí.

Struktura paketů

Pro navazování spojení používá IPSec protokol ISAKMP. Pro přenos dat využívá dva protokoly, kterými dokáže přenášená data zabezpečit a zajistit jejich integritu. Prvním z těchto protokolů je AH, druhým z nich je ESP. Oba zároveň umožňují fungovat ve dvou režimech, které již byly představeny v sekci 2.6, a to Tunnel a Transport. U AH i ESP se liší způsob zapouzdření paketů.

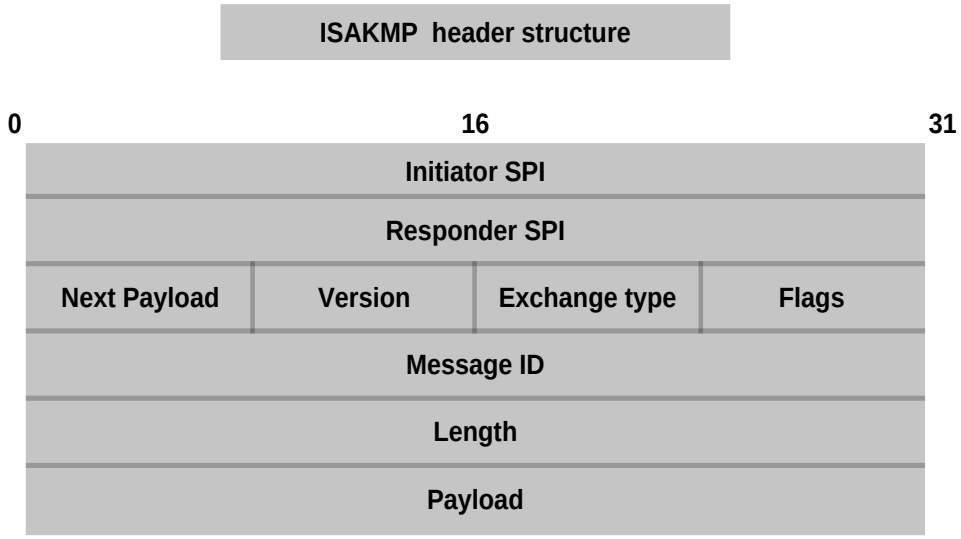
[18] Encapsulation Security Payload
[19] Authentication Header
[20] Internet Key Exchange
[21] Internet Security Association and Key Management Protocol

Internet Security Association And Key Management Protocol

Hlavička ISAKMP se skládá z následujících položek [35]:

- **Initiator SPI** – 32 bitů, identifikátor SA odesílatele;
- **Responder SPI** – 32 bitů, identifikátor SA příjemce;
- **Next Payload** – 8 bitů, typ posílaných dat v daném paketu;
- **Version** – 8 bitů, verze použitého protokolu;
- **Exchange type** – 8 bitů, typ zprávy (označení části navazání spojení, ve které se komunikace aktuálně nachází);
- **Flags** – 8 bitů, příznaky, obsahující informace o tom, kdo odesílal danou zprávu či zda se jedná o žádost o informace nebo o odpověď;
- **Message ID** – 32 bitů, identifikátor zprávy, ochrana proti replay útokům;
- **Length** – 32 bitů, délka ISAKMP hlavičky a dat;
- **Payload** – 32 bitů, datový obsah paketu.

Strukturu ISAKMP hlavičky lze vidět na obrázku 2.13.

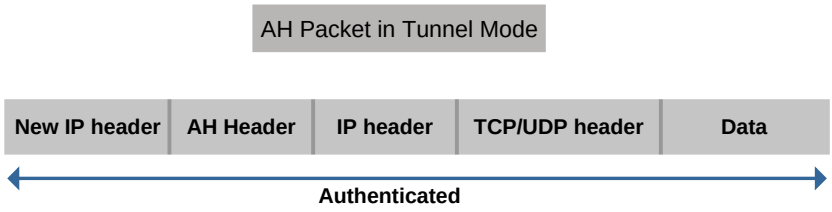


Obrázek 2.13: Struktura ISAKMP hlavičky. Převzato z [35].

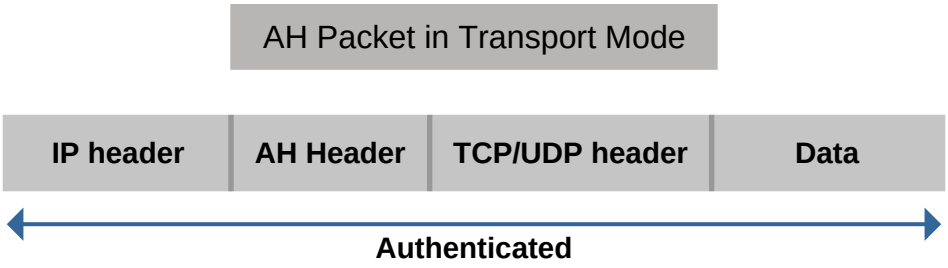
Authentication Header

V případě použití tohoto protokolu a režimu Tunnel dojde k přidání AH hlavičky a patičky k původnímu paketu, a zároveň nové IP hlavičky, ve které je umístěna například IP adresa cílové výchozí brány. Toto použití znázorňuje obrázek 2.14.

V případě režimu Transport jsou AH části vloženy za původní IP hlavičku, jak lze vidět na obrázku 2.15. Jako identifikátor AH protokolu se do IP hlavičky vloží IP protokol s číslem 51 [48].



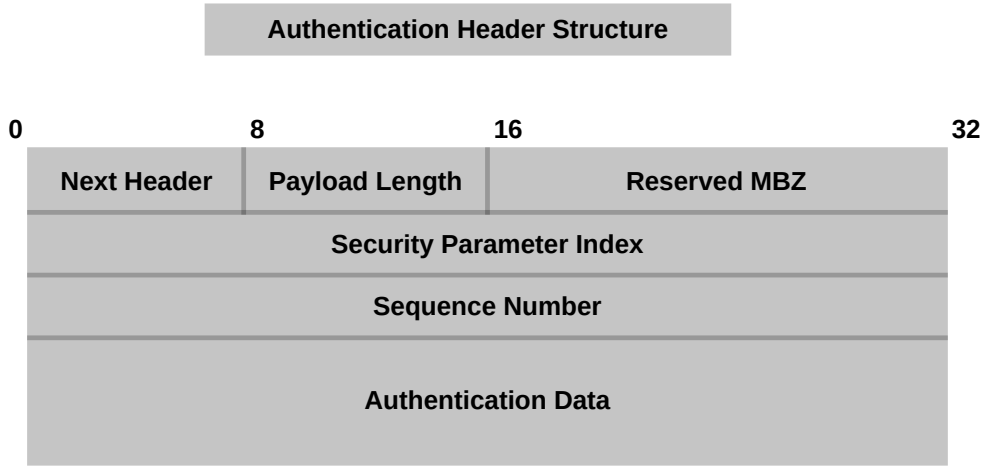
Obrázek 2.14: Formát paketu v režimu Tunnel s AH hlavičkou. Převzato z [7].



Obrázek 2.15: Formát paketu v režimu Transport s AH hlavičkou. Převzato z [7].

AH hlavička obsahuje následující položky a je znázorněná na obrázku 2.16 [20]:

- **Next Header** – 8 bitů, specifikuje typ transportního protokolu z vyšší vrstvy (v případě TCP je hodnota nastavena na číslo 6, v případě UDP je zde hodnota 17 [48]);
- **Payload Length** – 8 bitů, velikost AH hlavičky;
- **Security Parameter Index (SPI)** – 32 bitů, unikátní identifikátor SA, který je utvořen z cílové IP adresy a typu protokolu (zde AH);
- **Sequence Number** – 32 bitů, přenáší pouze odesílatel. Poskytuje tzv. „anti-replay“ ochranu paketu (jestliže příjemce obdrží paket se stejným Sequence Number, který již jednou přijal, pak tento paket zahodí);
- **Authentication Data** – proměnlivá velikost (závisí na použitém algoritmu, musí být zarovnána na 32 bitů či 64 bitů a bývá doplněna na tyto velikosti nulami). Slouží ke kontrole integrity paketu i samotné AH hlavičky;
- **Reserved MBZ** – 12 bitů, rezervované místo v hlavičce. Obsahuje nulové hodnoty.



Obrázek 2.16: Struktura AH hlavičky. Převzato z [7].

AH nešifruje obsah paketu, ale poskytuje autentizaci, integritu dat a ochranu proti „replay“ útokům, kdy útočník zopakuje během přenosu některé pakety a ty tak příjemce obdrží vícekrát [33].

Zajištění integrity

Integrita v přenášených datech je zajištěna pomocí autentizačního algoritmu, na kterém se komunikující strany dohodnou před zahájením přenosu (tato informace je součástí SA). Odesílatel spočítá pomocí tohoto algoritmu hodnotu, kterou vloží do pole `Authentication Data` AH hlavičky.

Příjemce po obdržení tohoto paketu spočítá pomocí stejného algoritmu své vlastní číslo, které následně porovná s hodnotou uvnitř výše zmíněné části hlavičky. Výsledná hodnota je spočítána z celého paketu, včetně AH hlavičky. Vynechávají se jen položky, které se mohou při přenosu měnit (jedná se například o `Time to Live` (TTL), která se dekrementuje při průchodu každým routerem v cestě sítí nebo `Checksum` či `Flags`). Hodnota bývá nejčastěji v podobě MAC[22] hashe nebo méně častěji v podobě digitálního podpisu [16].

Pokud se výsledné hodnoty shodují, znamená to, že nedošlo při přenosu k narušení integrity paketu a data tak mohou být zařízením zpracována. Pokud se čísla liší, zařízení paket zahodí.

Ověřování integrity dat příjemcem však může zkomplikovat použití překladu adres na směrovačích.

AH a Network Address Translation

Překlad adres komplikuje až znemožňuje použití AH protokolu v IPSecu. Problém nastává při ověřování integrity paketů.

Pokud je AH paket routován přes zařízení, na kterém je nastaven NAT, pak se do IP hlavičky vloží při tomto průchodu jiná adresa (proběhne překlad), což způsobí jiný výsledek při výpočtu MAC hodnoty na cílovém zařízení, jelikož se tato hodnota počítá v případě AH z IP hlavičky i vnitřku paketu.

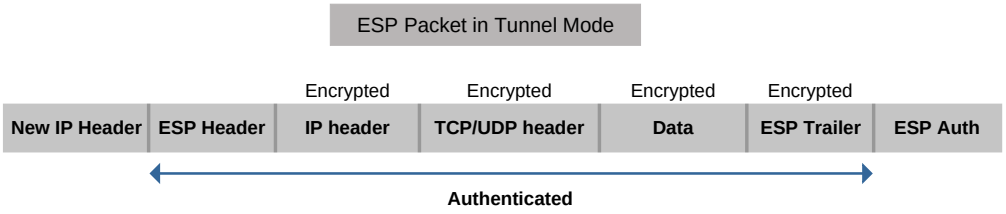
Možným řešením tohoto problému je, aby si překládající router spočítal vlastní MAC hash a vložil jej do AH hlavičky. To však není možné, jelikož použitý algoritmus k výpočtu této hodnoty zná pouze odesílatel a příjemce dané komunikace. Authentication Header v kombinaci s NATem (ať už v jakémkoliv režimu) tedy nelze použít [4]. Většinou se tyto potíže řeší využitím druhého protokolu (ESP) [21].

[22]Message Authentication Code

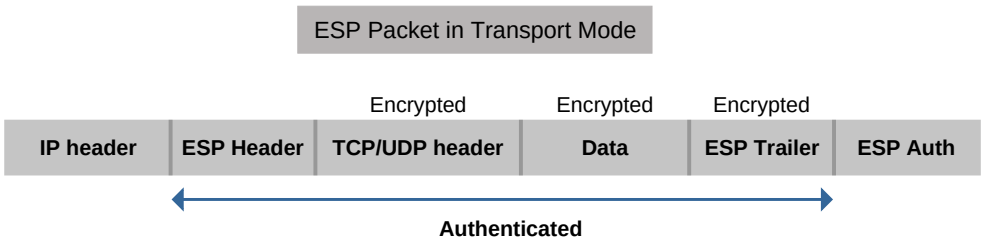
Encapsulation Security Payload

Tento protokol se nejčastěji využívá v kombinaci s režimem Tunnel, jelikož podporuje kromě autentizace a zajištění integrity také šifrování přenášených dat. V tomto módu se zabezpečuje celý vnitřek paketu, včetně původní IP hlavičky, jak je vidět na obrázku 2.17.

V transportním režimu se nezabezpečuje celý paket, ale pouze jeho obsah (vnitřek). Toto znázorňuje obrázek 2.18. Jako identifikátor ESP se do IP hlavičky vloží číslo IP protokolu 50 [48]. Na rozdíl od AH se tento protokol podílí i na zapouzdření paketu. Nejdříve obsah paketu zašifruje a poté jej vloží mezi ESP části (u AH vzniká pouze AH hlavička) [33].



Obrázek 2.17: Formát paketu v režimu Tunnel s ESP částmi. Převzato z [7].



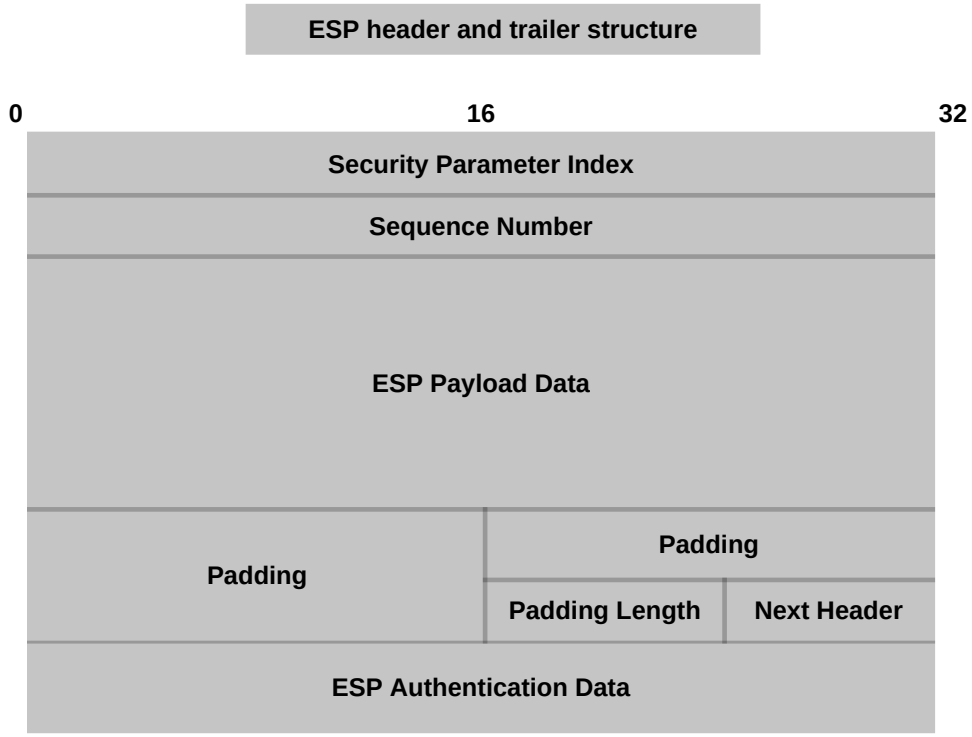
Obrázek 2.18: Formát paketu v režimu Transport s ESP částmi. Převzato z [7].

ESP má následující strukturu hlavičky (viz obrázek 2.19):

- **Security Parameter Index (SPI)** – 32 bitů, unikátní identifikátor SA, který je utvořen z cílové IP adresy a typu protokolu (zde ESP);
- **Sequence Number** – 32 bitů, přenáší pouze odesílatel, plní stejnou funkci (ochranu proti „anti-replay“ útoku) jako v případě AH.

Na rozdíl od AH paketu se položky `Authentication Data` (místo, kde je uložena MAC hodnota) a `Next Header` (identifikátor protokolu transportní vrstvy) nenacházejí v hlavičce, ale v ESP patičce. Zde jsou umístěny spolu

s informací o paddingu a jeho délce, což jsou informace, které využívá šifrovací algoritmus ke zjištění bytové hranice obsahu paketu, který má být tímto algoritmem zašifrován [21].



Obrázek 2.19: Struktura ESP hlavičky a patičky. Převzato z [7].

ESP a Network Address Translation

Jak již bylo zmíněno u AH protokolu, tak i ESP má s překladem adres problém, ale zde záleží na použitém režimu.

Problém nastává při přenosu v módu Transport. A to i přesto, že IP adresa paketu, která se nachází v IP hlavičce není v tomto případě zašifrována a tudíž není zahrnuta do výpočtu MAC hodnoty. Z důvodu nezabezpečené podoby IP hlavičky může na zařízeních, které podporují NAT dojít k překladu IP adresy v hlavičce paketu, což způsobí změnu v CRC součtu, který udržuje integritu TCP případně UDP části paketů. Tento součet již ale ve výpočtu MAC hodnoty obsažen je, tudíž jej změní. NAT zařízení nemá z důvodu šifrování obsahu paketu (s výjimkou hlavičky) přístup k CRC součtu, takže jej nemůže přepočítat na novou hodnotu, která by odpovídala modifikaci paketu. Toto povede opět k zahození paketu. U ESP je to z důvodu

změny CRC hodnoty u transportního režimu, u AH se jednalo o změnu MAC hodnoty.

Jestliže se však použije ESP v kombinaci s režimem Tunnel, pak tyto pakety přes NAT projdou. Provede se tzv. *NAT-Traversal*.

NAT-Traversal

Tento mechanismus zajistí průchod IPSec ESP paketů přes NAT zařízení. Probíhá ve dvou fázích:

1. Zjištění, zda oba peery podporují NAT-T;
2. Detekce NAT zařízení v cestě, kudy pakety budou posílány (NAT-Discovery (NAT-D)).

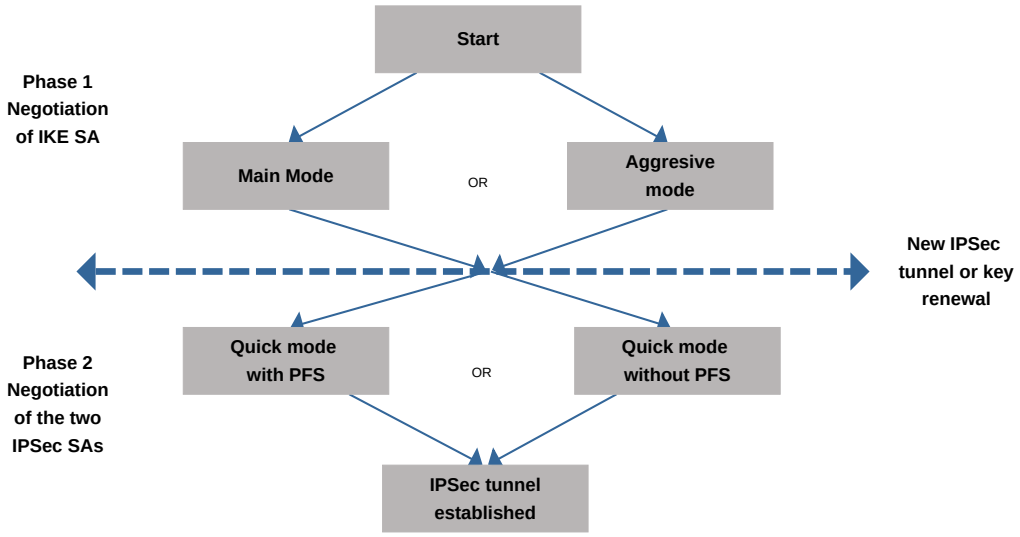
První fáze probíhá v prvních dvou zprávách v Main režimu IKE (viz strana 24). Pokud oba konce komunikace podporují NAT-Traversal, pak se přistoupí k druhému bodu, který nastává ve třetí a čtvrté zprávě Main režimu. Zařízení si pošlou dva hashe původní IP adresy a portu (jeden hash je pro zdrojové údaje, druhý pro cílové). Druhé zařízení, které tyto zprávy přijme, si spočítá vlastní hashe. Pokud se tyto čtyři hodnoty liší, znamená to, že při cestě sítí došlo k překladu adres, a tedy že mezi těmito dvěma body existuje někde NAT.

Pokud se tak stane, je další IPSec komunikace (včetně druhé fáze IKE a následně i uživatelské komunikace) zapouzdřena do UDP paketu s portem 4500. Pro odlišení datových a kontrolních zpráv se v tomto případě do IKE paketů přidává navíc tzv. *Non-ESP Marker*, což jsou čtyři nulové byty, nacházející se na stejné pozici jako hodnota SPI u ESP paketů. Takto upravené pakety již NAT zařízením projdou (překlad se povede správně, aniž by ovlivnil výslednou MAC hodnotu) [7].

Navazování spojení

Předtím než je možné šifrovat a autentizovat pakety posílané přes IPSec protokol, je zapotřebí sestavit tunel, kterým budou data posílána. K tomu, aby se tento tunel mohl vytvořit, si potřebují obě strany, které spolu chtějí komunikovat, vyměnit informace, dohodnout se na bezpečnostních principech, které v průběhu posílání dat využijí, a také potřebují získat bezpečnostní klíče, které si navzájem vymění. Právě výměna, uchování a předání těchto klíčů mezi komunikujícími stranami je u IPSecu velice kritická část a je nutné ji dobře zabezpečit.

Jak již bylo zmíněno, pro tento účel je využíván protokol s názvem ISAKMP, který podporuje několik různých technik, pomocí kterých lze klíče transportovat mezi komunikujícími stranami. Standardně komunikuje na portu UDP 500 [48]. Použitím tohoto protokolu se komunikující strany dohodnou na způsobu výměny klíčů a dalších informací (SA). Jestliže ISAKMP využívá techniky s názvem Oakley (popsán v RFC 2412 [42]) či SKEME[23] (což se v případě využití u IPSecu skutečně děje), pak se jedná o specifickou implementaci tohoto protokolu, které se říká IKE. Existuje ve dvou různých verzích, a to IKEv1 a IKEv2. V následujících odstavcích bude popsána IKEv1, jelikož se v běžném síťovém prostředí používá více. Ta se (stejně jako IKEv2) dělí na dvě fáze – IKE fáze 1 a IKE fáze 2 [33] [44]. Princip obou těchto fází je schématicky znázorněn na obrázku 2.20.



Obrázek 2.20: Schéma IKE procesu. Převzato z [33].

IKE fáze 1

V první fázi IKE se komunikující strany vzájemně dohodnou na konkrétním způsobu šifrování, autentizačních mechanismech, hashovacích algoritmech a na dalších parametrech, které budou při výměně IKE zpráv používat. Dále se provádí výměna klíčů Diffie-Hellman (DH). V této fázi vzniká ISAKMP tunel. Slouží pouze pro zajištění bezpečného navázání druhého tunelu (druhé fáze IKE) a posílají se přes něj především udržovací zprávy (například `KeepAlive`), aby nedošlo k přerušení již navázaného tunelu [33].

IKEv1 má dva režimy výměny zpráv (*Main* a *Agressive*).

[23] Secure Key Exchange Mechanism

V režimu Main proběhnou celkem tři výměny zpráv, z nichž každá se skládá z žádosti a z odpovědi. Celkem tedy proběhne šest zpráv.

1. V první výměně si komunikující strany mezi sebou pošlou parametry, kterými budou následně zabezpečovat další IKE zprávy. Jedná se konkrétně o:

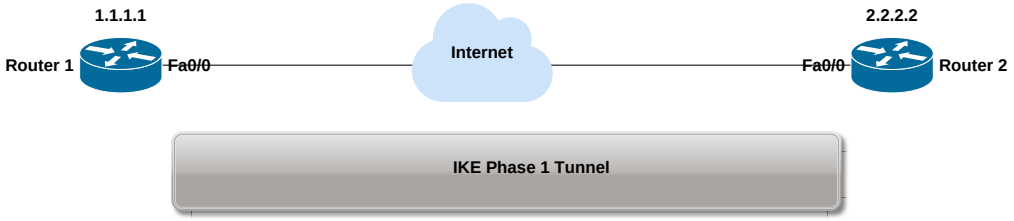
   - Volbu šifrovacího, hashovacího a autentizačního algoritmu, které budou sloužit pro výměnu klíčů.
   - Diffie-Hellman skupinu, která určuje jak velký (kolik bitů) bude mít klíč, který si vygenerují a pošlou v druhé výměně. Čím víc bitů klíč má, tím déle trvá jeho výpočet, ale je odolnější vůči prolomení.
   - Dobu, po kterou bude tunel z první fáze udržován.

   Odesílatel zašle možný výběr těchto parametrů příjemci. Pokud si příjemce jednu z nabízených možností vybere, pak lze tunel navázat.
   V opačném případě se proces počínající komunikace zastaví.

2. V druhé výměně probíhá samotné přeposlání klíčů pomocí
   Diffie-Hellman algoritmu. Odesílatel posílá hashe, které může přečíst pouze příjemce a naopak.

3. V poslední výměně dochází k autentizaci stran a ověření parametrů, které byly stanoveny v předchozích bodech komunikace. Tyto zprávy jsou zabezpečeny pomocí algoritmů, na kterých se komunikující strany dohodly v části 1.

V režimu Aggressive se posílá o polovinu méně zpráv. Tento mód je rychlejší, ale méně bezpečný, jelikož informace obsažené v paketech nejsou nijak šifrovány. Proběhne zde sloučení prvních dvou částí do jedné. Třetí zpráva se posílá jen jako potvrzení a je odesílána až po tvorbě IKE SA [33] [34] [44]. První fázi IKE znázorňuje obrázek 2.21.



Obrázek 2.21: První fáze IKE. Převzato z [38].

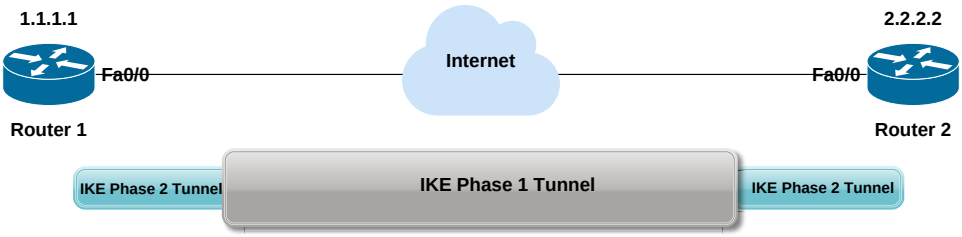
IKE fáze 2

V této části se využívají hodnoty z DH procesu z první fáze IKE. Buduje se další tunel v již existujícím tunelu. Ten slouží pro samotné zasílání uživatelských dat (IPSec paketů). V této fázi se využívá Quick mód, při kterém je využito IKE SA z předchozí fáze.

Zde si zařízení vyměňují následující informace:

- volba IPSec protokolu (AH nebo ESP);
- jaký režim při přenosu dat využijí (Transport nebo Tunnel);
- šifrovací a autentizační algoritmus;
- doba trvání tunelu. Pokud tento čas uplyne, SA se považují za nevalidní a tato fáze se opakuje znovu pro obnovení těchto informací.

Pro zvýšení bezpečnosti se někdy využívá ještě PFS[24], které vynucuje novou výměnu klíčů pomocí nového absolvování DH procesu i pro druhou fázi IKE, která je znázorněna na obrázku 2.22 [33] [34].

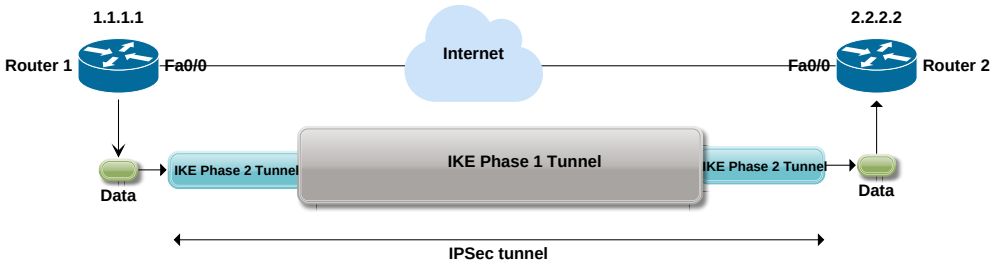


Obrázek 2.22: Druhá fáze IKE. Převzato z [38].

Po dokončení a výměně IPSec SA zpráv je tunel navázán a posílají se přes něj již uživatelská data, která jsou zabezpečena pomocí AH či ESP

[24]Perfect Forward Secrecy

protokolů a dalších mechanismů, na kterém se komunikující strany shodly v první a v druhé fázi tohoto procesu. Posílání uživatelských dat lze vidět na obrázku 2.23.



Obrázek 2.23: IPSec tunel. Převzato z [38].

2.8.2 OpenVPN

Tento VPN protokol (narozdíl od IPSecu) není definován v žádném RFC dokumentu, ale je veřejně dostupný, včetně svých zdrojových kódů. Jedná se o open source projekt, tudíž jeho kód bývá revidován a zkoumán rozdílnými lidmi, což mnohdy vede k objevení a odstranění možných bezpečnostních rizik. Vznikl v roce 2001, jeho autorem je James Yonan a spadá pod licenci GNU GPLv2 [12] [17].

Využívá virtuálních síťových adaptérů Tun nebo Tap (viz sekce 2.7), a je tedy kompatibilní se všemi operačními systémy, které tyto adaptéry podporují. Pomocí nich dokáže zprostředkovat komunikaci buď na síťové vrstvě (používá rozhraní operačního systému Tun) nebo na linkové vrstvě (používá rozhraní operačního systému Tap).

Ke komunikaci pomocí této VPN se využívá dvou kanálů – *kontrolního* a *datového*. Zabezpečení každého z nich se řeší odlišným způsobem.

Součástí OpenVPN je knihovna OpenSSL, která slouží k zajištění autentizace a šifrování dat. Komunikace může probíhat přes UDP i TCP protokoly. Při použití protokolu TCP může dojít k fragmentaci aplikačních dat do více TCP segmentů. Zároveň však také může dojít k slučování aplikačních dat do jednoho TCP segmentu. Vždy to závisí na velikosti těchto dat.

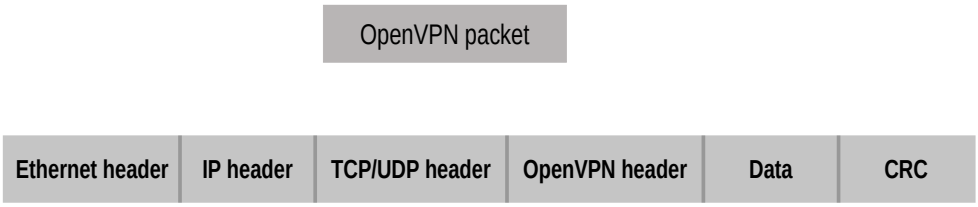
Standardní port OpenVPN je UDP 1194. Pokud využíváme starší klienty (od verze 2.0-beta16 a starší), můžeme se setkat s implicitním portem 5000 [12].

Režimy OpenVPN

Tato VPN služba obsahuje dva různé režimy, které využívají rozdílné způsoby autentizace. Prvním z nich je *point-to-point*. Při použití tohoto režimu se autentizace komunikujících stran realizuje pomocí statických sdílených klíčů, které si v rámci navazování spojení komunikující strany vymění. Druhým režimem je *multi-client*, který autentizaci provádí pomocí TLS spojení, a to buď prostřednictvím výměny certifikátů nebo pomocí jména a hesla, přičemž výběr konkrétní možnosti závisí vždy na aktuální konfiguraci OpenVPN služby [25].

Formát paketů

OpenVPN zprávy jsou zapouzdřeny do TCP nebo UDP paketů. Formát těchto dat je znázorněn na obrázku 2.24. Rozlišují se kontrolní a datové pakety, jejichž struktura bude popsána v následující části práce.



Obrázek 2.24: Formát OpenVPN paketu. Převzato z [25].

Typy zpráv

OpenVPN pakety se rozlišují `Opcode` hodnotou. Přehled těchto hodnot u kontrolních zpráv je zobrazen v tabulce 2.1. Přehled `Opcode` hodnot u datových zpráv je zobrazen v tabulce 2.2.

Název Opcode	Hodnota Opcode
P_CONTROL_HARD_RESET_CLIENT_V1	0x01
P_CONTROL_HARD_RESET_SERVER_V1	0x02
P_CONTROL_SOFT_RESET_V1	0x03
P_CONTROL_V1	0x04
P_ACK_V1	0x05
P_CONTROL_HARD_RESET_CLIENT_V2	0x07
P_CONTROL_HARD_RESET_SERVER_V2	0x08

Tabulka 2.1: Přehled Opcode v kontrolních paketech OpenVPN. Převzato z [25].

Název Opcode	Hodnota Opcode
P_DATA_V1	0x06
P_DATA_V2	0x09

Tabulka 2.2: Přehled Opcode v datových paketech OpenVPN. Převzato z [25].

Datové pakety se liší použitým TLS režimem (verze 1 nebo verze 2). Kontrolní zprávy `P_CONTROL_HARD_RESET_CLIENT` a `P_CONTROL_HARD_RESET_SERVER` se liší v použitém způsobu generování kryptografických klíčů. Verze 1 využívá funkce z OpenSSL knihovny (`RAND_bytes()`). Verze 2 využívá navíc pseudo-náhodnou TLS funkci (`TLS PRF`) [41].

Struktura kontrolních paketů

Kontrolní pakety se skládají z následujících položek [25]:

- **Packet Length** – 16 bitů, délka paketu. Je obsažena pouze při použití TCP protokolu a je vždy posílána nešifrovaně.
- **Packet Type** – 8 bitů, identifikátor typu OpenVPN zprávy. Je obsažen pouze v TLS režimu. Vyšších 5 bitů označuje `Opcode` zprávy, zbývající 3 dolní bity označují `key_id`, které identifikuje již navázané TLS spojení, přičemž může dojít k opětovnému obnovení tohoto spojení posláním nového `key_id`.
- **Local Session Id** – 64 bitů, identifikátor relace, ke které daný paket patří.

- **HMAC**[25] – proměnná velikost (závislá na použitém hashovacím algoritmu), hodnota, používající se k ověření integrity paketu.
- **Packet Id** – 32 nebo 64 bitů (v případe zahrnutí časové značky), „anti-replay“ ochrana (viz strana 19). Je obsaženo ve všech paketech, pokud je zapnutý TLS režim.
- **Message Id** – 32 bitů, identifikátor zprávy v paketu. Není obsažen v potvrzovacích (`ACK`) paketech.
- **Acknowledge Packet Id Array Size** – 8 bitů, velikost pole.
- **Acknowledge Packet Id Array** – 32 bitů × počet `ACK` paketů, pole identifikátorů `ACK` paketů. Využívá se pro potvrzení více paketů najednou.
- **Remote Session Id** – 64 bitů, identifikátor druhého konce komunikace. Paket jej obsahuje pouze tehdy, pokud zároveň obsahuje `ACK` pole.

Struktura datových paketů

Datové pakety obsahují [25]:

- **Packet Length** – 16 bitů, délka paketu. Je obsažena pouze při použití TCP protokolu a je vždy posílána nešifrovaně.
- **Packet Type** – 8 bitů, identifikátor typu OpenVPN zprávy. Je obsažen pouze v TLS režimu. Vyšších 5 bitů označuje `Opcode` zprávy, zbývající 3 dolní bity označují `key_id`, které identifikuje již navázané TLS spojení, přičemž může dojít k opětovnému obnovení tohoto spojení posláním nového `key_id`.
- **Peer Id** – 24 bitů, identifikuje druhý konec komunikace (druhé zařízení). Je obsaženo pouze v datových paketech verze 2.
- **Packet Id** – 32 nebo 64 bitů (v případě zahrnutí časové značky), „anti-replay“ ochrana (viz strana 19). Je obsaženo ve všech paketech, pokud je zapnutý TLS režim.
- **Compression Flag** – 8 bitů, informace o použití komprese.

[25] Hash-Based Message Authentication Code

- **HMAC** – proměnlivá velikost (závislá na použitém hashovacím algoritmu). V případě datových paketů může být velikost buď 32 nebo 64 bitů, hodnota, používající se k ověření integrity paketu.
- **IV** – 32 nebo 64 bitů, inicializační vektor. Používá se v kombinaci s `HMAC` a jeho velikost je závislá na použitém hashovacím algoritmu.
- **Payload** – proměnlivá velikost, data paketu.

V datových paketech jsou zašifrované výše zmíněné části, a to konkrétně: `Paket Id`, `Compression Flag`, `HMAC`, `IV` a `Payload`. V zachycené komunikaci je tedy vidět pouze délka paketu (v případě použití TCP protokolu), `Opcode` a identifikátor komunikující strany [25].

Zajištění integrity

K zajištění integrity používá tento protokol položku `HMAC` v kombinaci s hashovacím algoritmem *SHA-1* [12]. Jakýkoliv UDP paket, který v sobě nemá správnou hodnotu `HMAC`, je přijímacím zařízením zahozen.

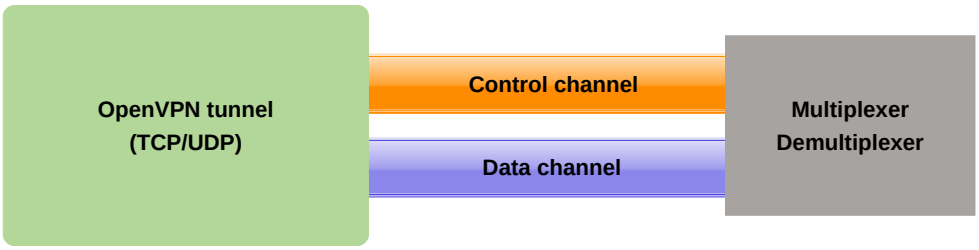
OpenSSL knihovna využívá dále blokové šifrovací algoritmy *AES*[26], *Blowfish* nebo *Triple DES (3DES)*. OpenVPN disponuje také *CBC*[27], což je operace u blokových šifer zajišťující závislost mezi bloky (částmi šifry). To případnému útočníkovi znemožní rozpoznání vzorců v šifrovaném textu, pokud se nějaké části původní zprávy opakují, a zároveň je to ochrana proti neoprávněné manipulaci s bloky [31].

Navazování spojení

Komunikace OpenVPN probíhá dvoufázově. Nejdříve je vytvořen kontrolní kanál, přes který proběhne autentizace komunikujících stran. Zároveň se naváže datový kanál, kterým se transportují data. Oba tyto kanály fungují nezávisle a jsou udržovány díky multiplexovacímu mechanismu, který je znázorněn na obrázku 2.25. Ten dle `Opcode` hodnoty v paketu (viz sekce 2.8.2) pozná do jakého kanálu má daný paket zařadit.

Vzhledem k různým možnostem konfigurace a jejich malým rozdílům bude následující text zaměřen na základní konfiguraci OpenVPN verze 2.0, která zahrnuje například druhý způsob generování klíčů při navázání spojení.

[26] Advanced Encryption Standard
[27] Cipher Block Chaining



Obrázek 2.25: Multiplexovací mechanismus OpenVPN. Převzato z [25].

Tvorba kontrolního kanálu

Navázání spojení začíná klient, který pošle zprávu k OpenVPN serveru. Celý proces zahajuje posláním jedné ze zpráv `P_CONTROL_HARD_RESET_CLIENT`, na kterou reaguje server odesláním paketu stejné verze `P_CONTROL_HARD_RESET_SERVER`. Tuto zprávu klient potvrdí `ACK` paketem. Následně se komunikující strany dohodnou na šifrovacích a autentizačních mechanismech. V případě multi-client režimu se nejdříve uskuteční TLS handshake a vymění se certifikáty nebo proběhne ověření jménem a heslem (volba vždy závisí na dané konfiguraci). Pokud je povolený point-to-point režim, pak musí administrátoři nahrát na komunikující zařízení statické klíče.

Utvoření kontrolního kanálu je dokončeno tehdy, pošle-li server jednu či více `P_CONTROL_V1` zpráv a klient je potvrdí. Poté může být přistoupeno k poslání dat datovým kanálem.

Po přenesení určitého počtu paketů, bytů či uplynutí předem určené doby (v sekundách) může server poslat kontrolním kanálem zprávu `P_CONTROL_SOFT_RESET_V1`, čímž si vynutí obnovu autentizačních položek (certifikátů, klíčů, ...). Klient mu v tuto chvíli odpovídá stejným paketem a server odesílá `ACK`. Poté proběhne nová výměna, která je následně opět potvrzena jednou nebo více kontrolními zprávami a `ACK` pakety [41] [25].

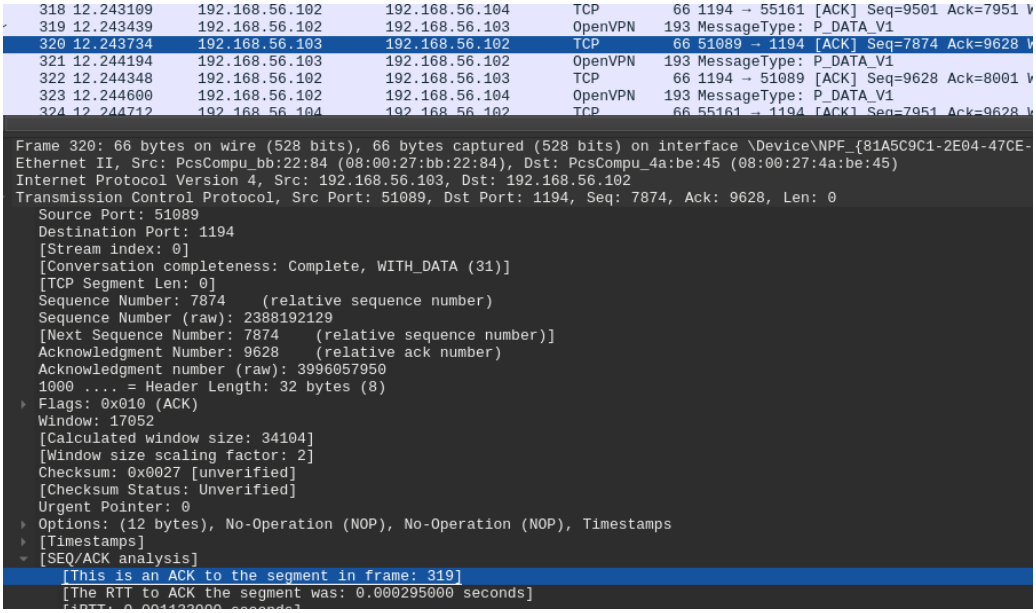
Zabezpečení kontrolního kanálu je zajištěno pomocí protokolu SSL/TLS [12].

Tvorba datového kanálu

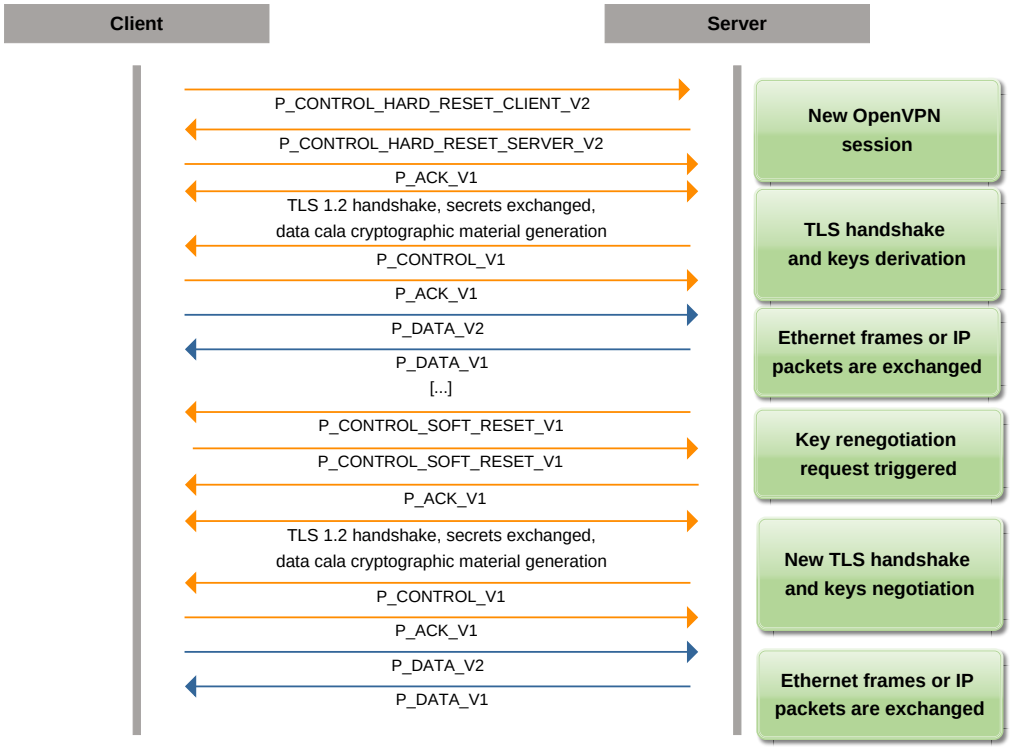
Datový kanál vzniká poté, co si komunikující strany vymění potřebné informace (certifikáty, klíče či jméno a heslo). Multiplexer dle `Opcode` v paketu pozná, že se jedná o data a pošle jej do datového kanálu.

V případě UDP protokolu data nejsou potvrzovaná. Při použití TCP jsou data potvrzovaná na transportní vrstvě posíláním `TCP ACK`, jak je vidět na obrázku 2.26.

Zabezpečení tohoto spojení probíhá pomocí předem dohodnutých šifrovacích a hashovacích algoritmů (například *BlowFish-CBC* či *SHA1*) [12]. Průběh celé komunikace je znázorněn na obrázku 2.27.



Obrázek 2.26: Potvrzování dat při použití TCP protokolu.



Obrázek 2.27: Průběh OpenVPN komunikace. Převzato z [25].

2.8.3 WireGuard

Jedná se o rychlou a snadno konfigurovatelnou VPN, která klade důraz na bezpečnost a je ze všech tří popisovaných VPN nejnovější. Stabilní verze byla vydána v roce 2020. Stejně jako v případě OpenVPN se jedná o open source software a jeho autorem je Jason Donenfeld.

Původně byl vyvinut jako náhrada OpenVPN a IPSecu pro linuxová jádra a postupem času začal podporovat i ostatní systémy. Na rozdíl od předchozí popisované VPN běží WireGuard pouze v režimu jádra operačního systému (kernel space). OpenVPN funguje i v uživatelském režimu OS (user space), jelikož využívá Tun a Tap (viz sekce 2.7).

Funguje na principu přiřazování veřejných klíčů ke komunikujícím IP adresám. Každý klient má svůj pár kryptografických klíčů, pomocí kterých probíhá šifrování a dešifrování paketů. WireGuard podporuje komunikaci přes TCP i UDP protokol a standardní port této služby je UDP 51820 [40] [11].

Byl také navržen jako „tichý“ protokol, takže pokud neprobíhá aktivní komunikace, pak je na lince ticho, tj. neprobíhá zde žádné pravidelné posílání konkrétních typů paketů, aby se spojení nepřerušilo. Výjimku tvoří

NAT, který naopak potřebuje komunikaci udržovat, jinak by po čase došlo k novému překladu adres i portů, což by způsobilo chyby při *CryptoKey-Routingu* (viz strana 37). WireGuard řeší NAT pomocí udržovacích zpráv (`KeepAlive`). Ty se posílají po určité době, kterou určuje administrátor při konfiguraci této VPN.

Formát paketů

Tato VPN používá ke komunikaci celkem čtyři druhy paketů, které je možné rozlišit typem uvedeným v hlavičce. `KeepAlive` pakety mají stejný typ jako datové. Typ a název paketu lze vidět v tabulce 2.3.

Název paketu	**Typ**
Handshake initiation	0x01
Handshake reply	0x02
Cookie Reply	0x03
Transport data a KeepAlive	0x04

Tabulka 2.3: Přehled typů paketů ve WireGuardu. Převzato z [53].

`Handshake` pakety se skládají z následujících položek [53] [14]:

- **Type** – 8 bitů, typ paketu;
- **Reserved** – 24 bitů, rezervované bity;
- **Sender Index** – 32 bitů, identifikátor odesílatele;
- **Reciever Index** – 32 bitů, identifikátor příjemce;
- **Responder epmhemeral public key** - 256 bitů, dočasný veřejný klíč. Komunikující zařízení si vytvoří nový klíč pokaždé, když dojde k novému handshake;
- **Encrypted Empty** – 128 bitů, zašifrovaná část (prázdná), která se přidává k handshake zprávám pro kontrolu integrity;
- **MAC1** – 128 bitů, hodnota, sloužící pro kontrolu integrity;
- **MAC2** – 128 bitů, hodnota, sloužící také pro kontrolu integrity. Je však obsažena pouze tehdy, pokud je linka přetížená (ochrana proti DoS útokům).

Pokud je linka přetížená, pak se na `Handshake init` zprávu odpoví zprávou `Cookie Reply`, která má menší velikost, a tudíž je pravděpodobnější, že se dostane k příjemci i přes vyšší provoz v síti.

`Cookie Reply` obsahuje [53] [14]:

- **Type** – 8 bitů, typ paketu;
- **Reserved** – 24 bitů, rezervované bity;
- **Reciever Index** – 32 bitů, identifikátor příjemce;
- **Nonce** – 192 bitů, náhodná číselná hodnota, přidávaná k zašifrované cookie hodnotě;
- **Encrypted Cookie** – 256 bitů, zašifrovaná cookie, bývá používána opakovaně.

Pakety, které mají v sobě typ `KeepAlive` a `Transport Data` obsahují následující položky [53] [14]:

- **Type** – 8 bitů, typ paketu;
- **Reserved** – 24 bitů, rezervované bity;
- **Reciever Index** – 32 bitů, identifikátor příjemce;
- **Counter** – 32 bitů, číslo přenášeného rámce mod 32. Po opětovné inicializaci spojení se vynuluje;
- **Encrypted Packet** – $\geq$ 128 bitů, zašifrovaná data. Pokud je jejich délka nulová, jedná se o `KeepAlive` zprávu.

Zajištění integrity

Integrita je u WireGuardu zajištěna symetrickým šifrovacím algoritmem *ChaCha20* v kombinaci s algoritmem *Poly1305* pro zajištění autentizace paketů.

Pro výměnu klíčů využívá *ECDH*[28] spolu s *Curve25519m* a také hashovací algoritmus *BLAKE2s*. Tyto postupy se využívají s *Noise*[29] frameworkem [11] [14].

[28] Elliptic-Curve Diffie-Hellman

[29] `https://noiseprotocol.org/`

Navazování spojení

Komunikaci začíná klient odesláním zprávou `Handshake init`. Server odpovídá `Handshake response`. Těmito zprávami se naváže spojení a vymění se dočasné veřejné klíče. Po dokončení handshake se začnou přenášet data. Pokud je v cestě od klienta k serveru NAT, pak se navázaný kanál udržuje pomocí zpráv `KeepAlive`. Jestliže uplyne určitá doba (specifikovaná parametrem *Rekey-Timeout* při konfiguraci), pak dojde k opětovnému handshake s novými klíči [14].

Jestliže dojde k přetížení linky a hrozí její zahlcení, pak se posílá paket `Cookie Reply` jako reakce na `Handshake init`.

WireGuard také používá v komunikaci CryptoKey Routing, kdy se veřejné klíče klientů spárují s konkrétní IP adresou, která s VPN serverem komunikuje. Každé síťové rozhraní (port) na serveru má svůj privátní klíč a seznam povolených IP adres a jejich veřejných klíčů, které slouží k autentizaci klientů.

Jestliže server přijme paket od klienta z lokální sítě, pak seznam s klíči a IP adresami funguje podobně jako směrovací tabulka – server dle cílové IP adresy v paketu zjistí, kam jej má odeslat a následně jej zašifruje veřejným klíčem, který má k této IP adrese přiřazený. Pokud daná adresa v seznamu není, pak se paket zahodí. Po přijmutí paketu z VPN tunelu server rozšifruje paket a zkontroluje, zda zdrojová IP adresa souhlasí s tou, ke které je přiřazen veřejný klíč. Pokud ano, paket pošle dál ke klientovi. Pokud ne, paket je zahozen [14] [22].

3 Již existující řešení detekce

V této kapitole budou popsány nástroje, které detekují síťový provoz. Některé z nich se (podobně jako analyzátor z této práce) zaměřují na rozpoznání konkrétních síťových služeb (jako jsou například VPN). Každý z níže popsaných pojal detekci jiným způsobem a dává k dispozici rozdílné informace o dané službě.

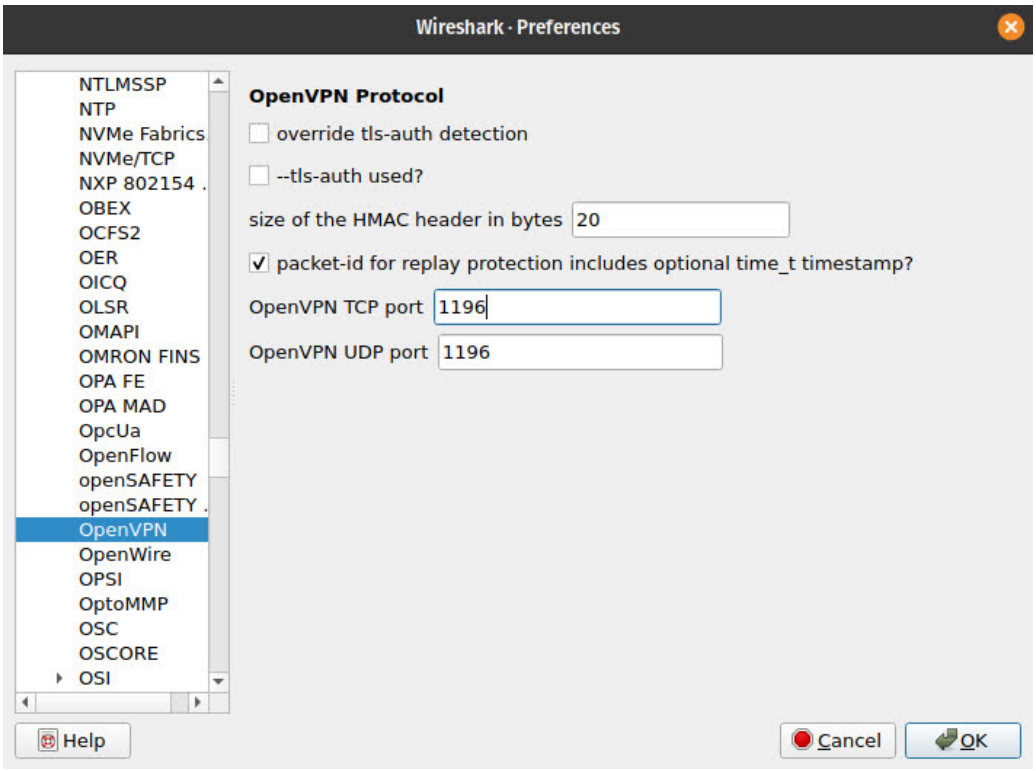
Postupně budou tedy popsány nástroje *Wireshark*, *EtherApe* a také *Síťový modul pro rozpoznání VPN provozu*.

3.1 WireShark

Jedná se o známý a populární síťový analyzátor vydaný pod licencí GNU GPL-2.0. Mezi administrátory je hodně rozšířený a bývá užitečný při řešení různých síťových problémů.

Většinou se využívá pro prvotní průzkum síťové komunikace. Dokáže nasbíraná síťová data ukládat do souborů pro další zpracování a zvládne extrahovat například různé zprávy do čitelnější podoby (například emailovou korespondenci). Umí detekovat mnoho protokolů a služeb, a to včetně jednotlivých vrstev ISO/OSI. Zvládne také převod mezi jednotlivými typy souborů (starší příponu `pcap` převede na novější `pcapng` a obráceně).

V této bakalářské práci byl WireShark využíván pro ruční analýzu síťového provozu z vytvořených scénářů. VPN pakety, které prošly sítí se základní konfigurací (tj. byla například využita standardní čísla portů) dokázal tento nástroj detekovat spolehlivě (viz sekce 4.6). Jakmile byla ale konfigurace sítě jiná (konkrétně stačilo, aby se změnila čísla portů například u OpenVPN), pak WireShark nedokázal tento provoz detekovat a muselo být upraveno jeho nastavení, aby dokázal tato data správně interpretovat. Ukázalo se tedy, že při detekci (nejenom) VPN provozu využívá pouze čísla portů a předpokládá tedy vždy základní (nezměněnou) konfiguraci dané služby. To lze vidět na obrázku 3.1. Z tohoto důvodu se jeho použití na analýzu VPN služeb ukázalo jako nevhodné.



Obrázek 3.1: Nastavení detekce služeb ve WireSharku.

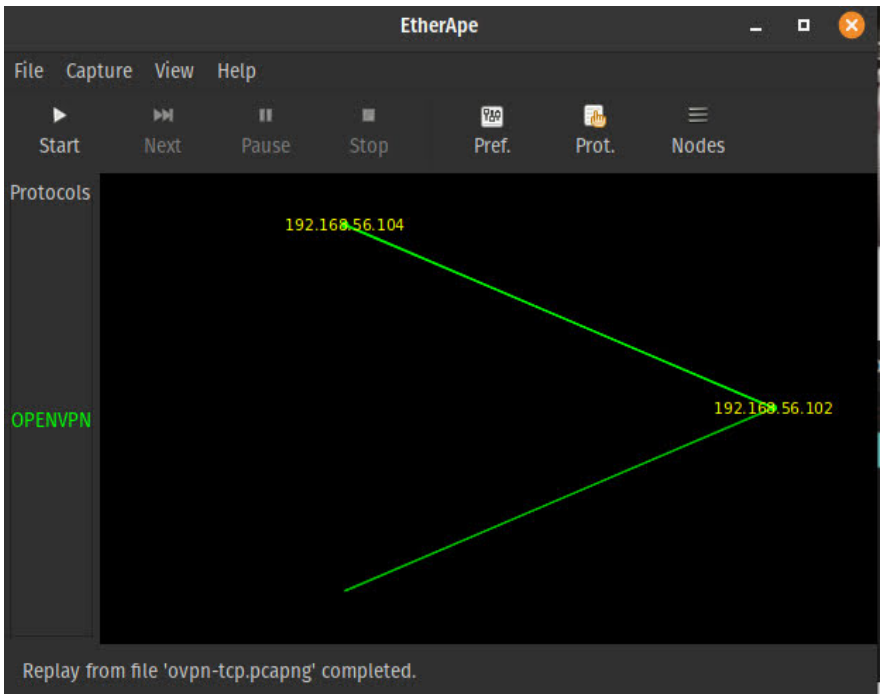
3.2 EtherApe

EtherApe[1] je program, který slouží k analýze, monitorování a především k vizualizaci síťového provozu v reálném čase. Zvládne však zpracovat i zachycený síťový provoz ze souboru. Jeho hlavní výhodou je grafické rozhraní, které umožňuje barevně rozlišit jednotlivé komunikace a protokoly. V průběhu zpracovávání dat se vykreslování mění tak, aby byl vidět aktuální stav (objem) dané komunikace. Dále zvládne překlad IP adres na doménové názvy.

Jedná se o open source nástroj pod licencí GNU GPL-2.0 a jeho zdrojový kód je tedy veřejně dostupný[2]. Při bližším zkoumání tohoto kódu bylo však zjištěno, že podobně jako WireShark i tento síťový program detekuje komunikaci pouze dle jejích základních charakteristik (základní čísla portů, protokolů, apod.). Potvrdilo se to i při následné analýze dvou různých síťových provozů. Konkrétně se jednalo o dva záchyty OpenVPN komunikace. Jeden obsahoval standardní konfiguraci a druhý měl změněné číslo portu.

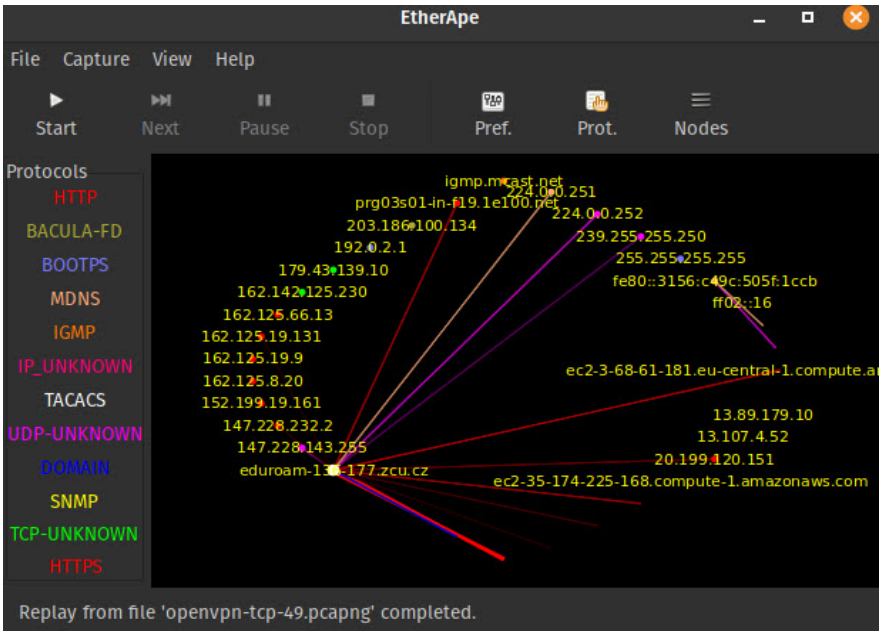
[1] `https://etherape.sourceforge.io/`
[2] `https://github.com/IFGHou/EtherApe`

První z nich EtherApe dokázal detekovat úspěšně (jak lze vidět na obrázku 3.2).



Obrázek 3.2: Úspěšnost detekce standardní komunikace OpenVPN při použití nástroje EtherApe.

Ve druhém souboru byla OpenVPN nastavena na TCP port 49, což tento software bohužel nerozpoznal (viz obrázek 3.3). Obdobně tomu bylo i u dalších souborů, ve kterém byl standardní UDP port nahrazen portem 1196. Ani ten bohužel EtherApe nepoznal jako OpenVPN. Ze stejného důvodu jako v případě WireSharku se tedy i tento software ukázal pro potřeby automatické detekce virtuálních privátních sítí spíše nevhodný.



Obrázek 3.3: Úspěšnost detekce nestandardní komunikace OpenVPN při použití nástroje EtherApe.

3.3 Softwarový modul pro rozpoznání VPN v síťovém provozu

Níže popsané řešení je součástí diplomové práce Bc. Martina Čtrnáctého z ČVUT v Praze s názvem „Softwarový modul pro rozpoznání VPN v síťovém provozu“ [54]. Bylo vyvinuto v roce 2021 a zaměřuje se především na detekci OpenVPN služby.

Jedná se o sadu detekčních aplikací, z nichž každá rozpoznává VPN jiným způsobem (například jde o detekci na základě IP toků či na základě charakteristických vlastností navazování spojení dané VPN).

Součástí textu práce je také popis konečného automatu pro detekci OpenVPN, které se stalo inspirací pro zpřesnění detekce analyzátoru implementovaného v této bakalářské práci.

Právě popisovaný softwarový modul je však omezen pouze na detekci OpenVPN služby a na VPN Cisco AnyConnect[3], která z OpenVPN vychází. Z toho důvodu se ukázal tento software pro potřeby detekce VPN služeb v kolejní síti ZČU jako nevyhovující, jelikož se v rámci síťové aktivity kolejní sítě předpokládá výskyt více rozdílných VPN služeb.

[3] `https://www.cisco.com/c/en/us/products/security/anyconnect-secure-mobility-client/index.html`

4 Vytvořené testovací scénáře

Součástí zadání této bakalářské práce bylo vytvořit testovací scénáře a zachytit VPN provoz, který by se následně analyzoval nástrojem pro sledování síťového provozu (Wireshark), a poté byl na stejných souborech otestován implementovaný analyzátor. Dle výsledků ruční analýzy a výstupu z analyzátoru byla zhodnocena míra úspěšnosti detekce.

Nejprve došlo k vytvoření a zachycení uměle simulovaného provozu ze tří scénářů, které umožnily jednodušší ruční kontrolu obsahu dat a následné odladění programu díky omezenému objemu zachyceného provozu. Poté byl zachycen reálný provoz z kolejní sítě Západočeské univerzity.

Jednotlivé (nasimulované) scénáře se tvořily v simulačním programu *GNS3*[1].

4.1 Použité technologie

V této části práce budou popsány jednotlivé technologie a programy, které byly využity při tvorbě testovacích scénářů.

4.1.1 GNS3

Jedná se o simulační program s grafickým uživatelským rozhraním, který umožňuje emulovat nejrůznější síťové prvky a vytvářet tak různorodé síťové topologie.

Spolu s technologií *Docker*[2], ve které byli spuštěni klienti, byl tento nástroj využit pro tvorbu celkem tří testovacích scénářů s různými síťovými topologiemi (konkrétně se jednalo o scénář s *IPSec VPN*, *OpenVPN* a *WireGuard VPN*).

4.1.2 Docker

Docker je open source platforma, která umožňuje vývojářům a administrátorům spustit aplikace v *kontejnerech* (což jsou vzájemně na sobě nezávislé a izolované jmenné prostory (namespace)). Jedná se o prostředí pro spuštění aplikací bez nutnosti zavádět jiné virtualizované jádro operačního systému.

[1] `https://www.gns3.comn`

[2] `https://www.docker.com/`

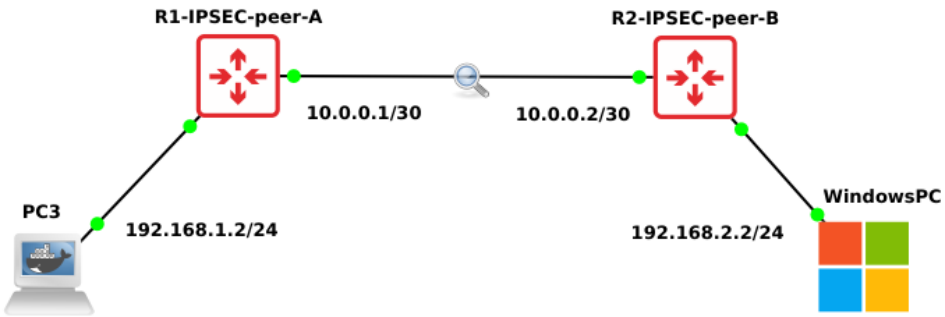
4.2 Scénář IPSec

Prvním vytvořeným scénářem v GNS3 byl IPSec. Využilo se dvou routerů od firmy MikroTik ve verzi 6.47.9, mezi kterými se vytvořil IPSec tunel. K nim se připojily dvě lokální sítě s klienty. Jeden klient byl zprovozněn za pomocí virtualizace v operačním systému Windows. Druhý byl utvořen s pomocí kontejnerizace (Dockeru) v linuxové distribuci Debian.

4.2.1 Adresace

V lokální síti, ve které je umísťen linuxový klient byla využita síť 192.168.1.0/24. Klient dostal adresu 192.168.1.2. Druhá LAN síť je 192.168.2.0/24 a klient s OS Windows má adresu 192.168.2.2. Mezi směrovači (a tedy IPSecovým tunelem) jsou adresy 10.0.0.1/30 a 10.0.0.2/30.

Adresace i konkrétní síťová topologie je znázorněna na obrázku 4.1.



Obrázek 4.1: Testovací topologie IPSec scénáře.

4.2.2 Komunikace v testovacím scénáři IPSec

V tomto scénáři byl bod záchytu síťového provozu umístěn na propojení dvou směrovačů, tedy přímo ve VPN tunelu. Mezi klienty proběhla komunikace pomocí programu *Ping*, která byla zachycena do souboru.

Jednalo se o první soubor, na kterém proběhla v rámci této práce ruční analýza provozu a zároveň to byl první soubor, na kterém byl vyzkoušen prototyp analyzátoru.

4.3 Scénář OpenVPN

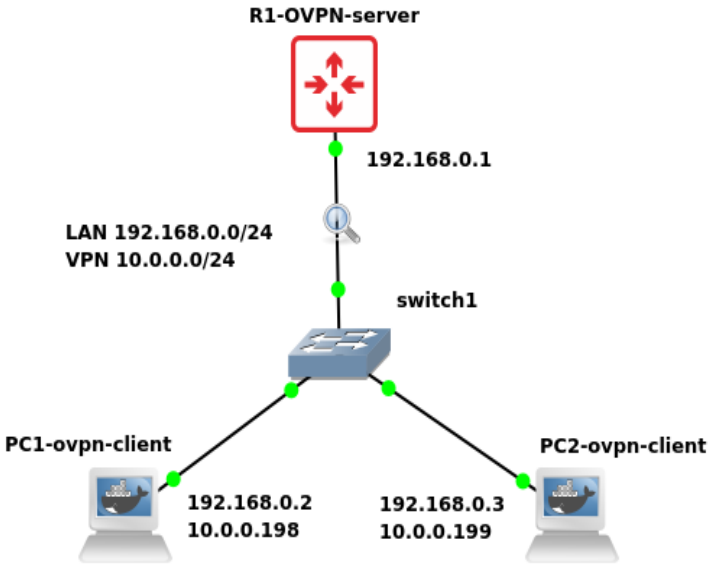
Druhý ze scénářů, který byl v simulačním prostředí GNS3 vytvořen, byl scénář s OpenVPN. Využilo se jednoho MikroTik směrovače, který byl nastaven jako OpenVPN server, jednoho přepínače (switche) a dvou počítačů. Ty mezi sebou komunikovaly s využitím TCP OpenVPN.

Klienti (počítače s linuxovou distribucí Debian) byli opět vytvořeni pomocí linuxových kontejnerů v Dockeru. Verze MikroTiku v této topologii musela být (z důvodu přidání podpory UDP OpenVPN komunikace) změněna na novější verzi 7.1.

4.3.1 Adresace

Zde je VPN tunel tvořen adresami ze sítě 192.168.0.0/24. OpenVPN server se nachází na adrese 192.168.0.1. PC1 byla přiřazena adresa 10.0.0.198 a PC2 10.0.0.199. Obě tato zařízení jsou v síti 10.0.0.0/24.

Konkrétní přehled adres a topologie se nachází na obrázku 4.2.



Obrázek 4.2: Testovací topologie OpenVPN scénáře.

4.3.2 Komunikace v testovacím scénáři OpenVPN

Místo pro zachycení síťového provozu bylo v tomto scénáři mezi směrovačem a přepínačem. Opět se mezi klientskými počítači testovalo spojení pomocí programu Ping. Celkem byly vytvořeny dva soubory se zachycenou komunikací. Jeden obsahuje OpenVPN komunikaci přes TCP protokol, druhý z nich obsahuje OpenVPN komunikaci přes UDP protokol.

4.4 Scénář WireGuard

Třetí síťovou topologii tvoří opět dva počítače s linuxovou distribucí Debian, jeden přepínač (switch) a celkem tři směrovače od MikroTiku, mezi kterými je jedna LAN síť.

Dva ze směrovačů byly nastaveni do režimu klient a ke každému je připojena další LAN síť s internetovými klienty (počítači). Poslední ze směrovačů má routovací politiku nastavenou jako server a umožňuje ostatním zařízením přistupovat k Internetu. Verze MikroTiku zde byla 7.1rc4, jelikož již obsahuje funkce WireGuard VPN v jádře.

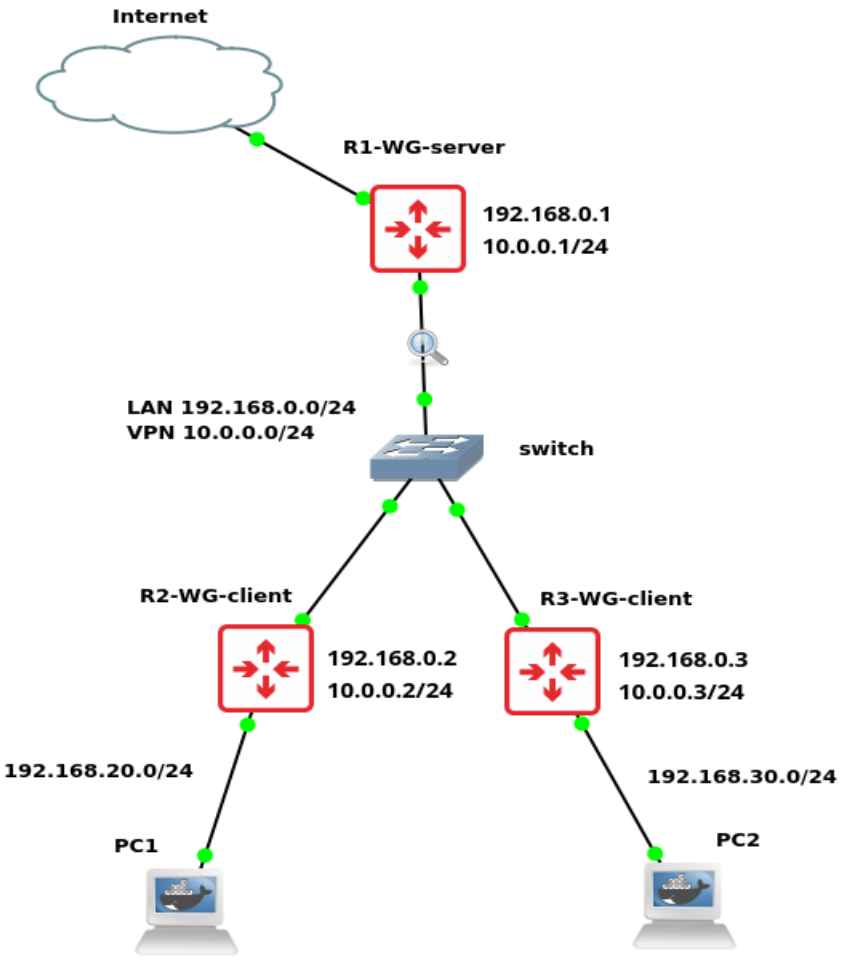
4.4.1 Adresace

Scénář obsahuje tři WireGuard peery – jeden server a dva klienty. Routery komunikují po síti 192.168.0.0/24, kde jsou klienti připojeni k serveru WG protokolem – vytvářejí virtuální síť (VPN) 10.0.0.0/24.

Klientské peery mají přímo připojené sítě 192.168.20.0/24 a 192.168.30.0/24. Používají výchozí bránu 10.0.0.1, což je server, přes který se směruje provoz mezi těmito sítěmi přes WG tunely. Tento server tedy obsahuje odpovídající statické cesty pro sítě 192.168.20.0/24 a 192.168.30.0/24. Úplné schéma sítě je vidět na obrázku 4.3.

4.4.2 Komunikace v testovacím scénáři WireGuard

Zde byl bod záchytu síťového provozu umístěn mezi WireGuard server a lokální síť 192.168.0.0/24. Stejně jako v předchozích dvou topologích se i zde testovala konektivita mezi koncovými zařízeními pomocí nástroje Ping.

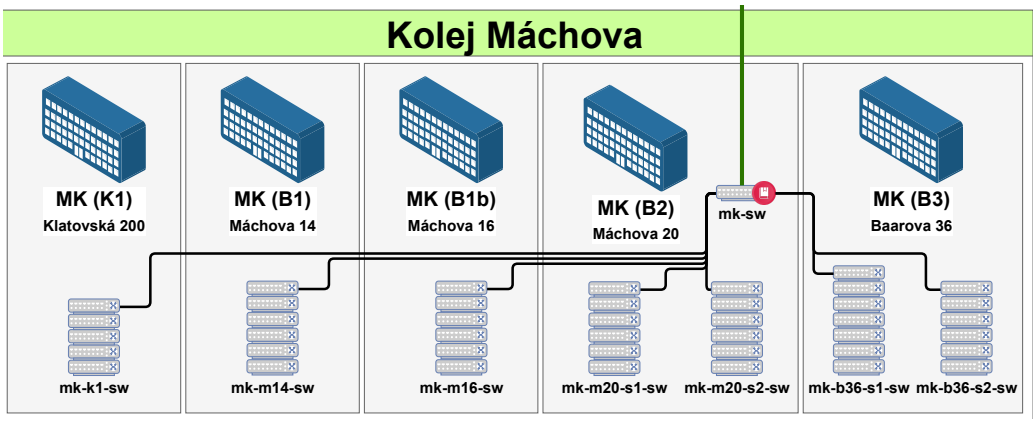


Obrázek 4.3: Testovací topologie WireGuard scénáře.

4.5 Kolejní síť

Po vytvoření tří scénářů v simulačním programu (které měly velikost řádově Kilobyty přenesených dat) bylo potřeba získat rozsáhlejší data o větším objemu, na kterých by se otestovala rychlost i přesnost implementovaného analyzátoru. K tomu posloužila kolejní síť ZČU, která je v určitých bodech vybavena konektivitou s kapacitou přenosu až 10 Gbit/s, což (pokud je na daném místě zvýšená síťová komunikace) vede až k miliónům paketů za sekundu, a tedy i k větším objemům (řádově Gigabyty) zachycených dat.

Schéma kolejní sítě, ve které probíhal záchyt komunikace, lze vidět na obrázku 4.4.



Obrázek 4.4: Schéma kolejní sítě Máchova ZČU.

4.5.1 Použitá zařízení

Provoz z této sítě byl pro účely testování analyzátoru zachycen na jednom z hlavních přepínačů v kolejní síti Máchova. Jednalo se o switch *Cisco Catalyst 9500*, který je na obrázku 4.4 označen červeným bodem.

4.5.2 Adresace

Vnitřní adresní rozsah je v této síti 10.0.0.0/16. Adresy z tohoto rozsahu se tedy objevily v zachycené komunikaci při ruční analýze a následně i při testování analyzátoru.

4.6 Analýza pomocí programu WireShark

Tento open source analyzátor byl v rámci bakalářské práce využit pro ruční analýzu provozu z připravených scénářů. Jako první se analyzoval provoz ze scénáře IPSec.

4.6.1 Analýza IPSec VPN

Po otevření souboru se zachycenou síťovou komunikací z IPSecového scénáře lze vidět několik zpráv, které s VPN nesouvisejí. V případě této analýzy se jedná například o DHCP či ARP[3] zprávy. Prvním krokem tak bylo vždy použití WireShark filtrů, kterými se docílilo přehledného zobrazení pouze konkrétního VPN provozu.

[3]Address Resolution Protocol – protokol pro získání fyzické adresy zařízení pomocí IP adresy

Nejprve bylo potřeba určit počet kontrolních zpráv. K tomu byl využit filtr „`isakmp`“. Ukázalo se, že se ve zkoumaném souboru nachází celkem 12 paketů, které si komunikující strany vyměnily při navazování spojení.

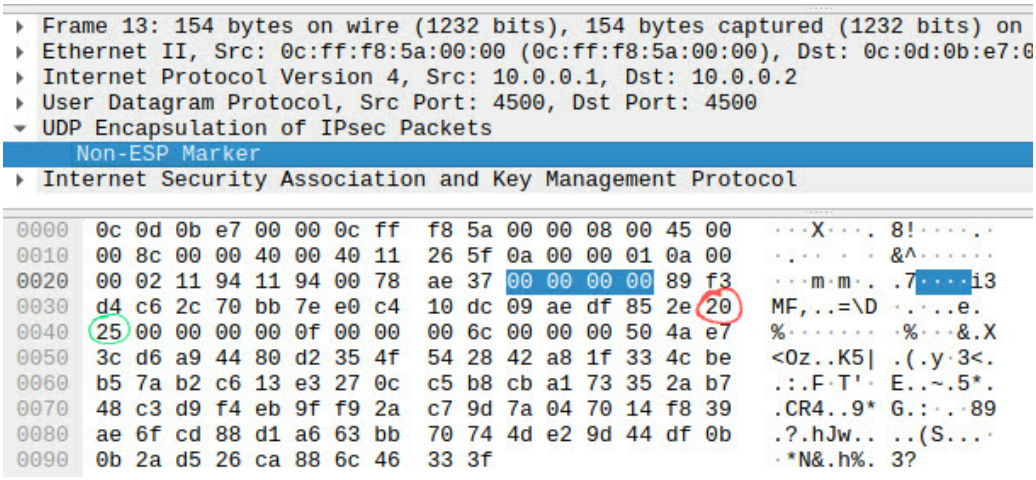
Následně se zjišťovalo množství datových paketů. Filtr byl tedy změněn na název dvou protokolů, které IPSec při posílání dat využívá. Konkrétně se jednalo o filtr „`ah or esp`“. Počet datových zpráv byl celkem 274.

Celkem se tedy přeneslo 286 paketů, což se ověřilo sloučením a následným aplikováním obou předchozích filtrů: „`isakmp or ah or esp`“.

V poslední fázi analýzy byla věnována pozornost jednotlivým částem IPSec hlavičky, ze které byly určeny unikátní identifikátory dané VPN. Jednalo se o verzi použitého IKE protokolu, použitý režim přenosu zpráv či (v případě kontrolních paketů) o Non-ESP marker.

IKEv1 má na 17. pozici v hlavičce hodnotu '`0x10`', zatímco IKEv2 má na 13. pozici hodnotu '`0x20`', jak je vidět na obrázku 4.5. Zároveň každý IKE paket obsahuje navíc identifikaci fáze navazování spojení. Jedná se o byte, nacházející se hned za specifikací IKE verze (například v obrázku 4.5 je byte '`0x25`', což označuje *Informational* část).

Shrnutí ruční analýzy je zobrazeno v tabulce 4.1.



Obrázek 4.5: Byty v IPSec paketu.

IP adresy zařízení	10.0.0.1	10.0.0.2
Počet paketů celkem	286	286
Počet bytů celkem	48 KB	48 KB
Příchozí pakety	153	133
Odchozí pakety	133	153
Kontrolní pakety	12	12
Datové pakety	274	274

Tabulka 4.1: Výsledek ruční analýzy v komunikaci ze scénáře IPSec.

4.6.2 Analýza OpenVPN

Zde se začalo analýzou souboru, ve kterém byl zachycen OpenVPN provoz probíhající přes UDP protokol. Prvním krokem bylo opět vyfiltrovat pouze VPN pakety. Nejdříve se zjišťoval počet kontrolních a následně i datových zpráv. Na to se v tomto případě hodil WireShark filtr, skládající se z jednotlivých `Opcode` hodnot kontrolních paketů. WireShark v OpenVPN hlavičce zobrazí hodnotu celého bytu. Použitý filtr tudíž porovnával celé toto číslo (byl tedy zkoumán konkrétní byte, ve kterém se nachází `Opcode` hodnota a zároveň `key_id`). Například část filtru „`openvpn.type == 0x28`“ zobrazí pouze `ACK` pakety. Poskládáním těchto dílčích filtrů tak dojde k získání celého zachyceného navazování spojení. Porovnávají se hodnoty '`0x40`', '`0x38`', '`0x28`', `0x20`', '`0x18`', '`0x10`' a '`0x08`'.

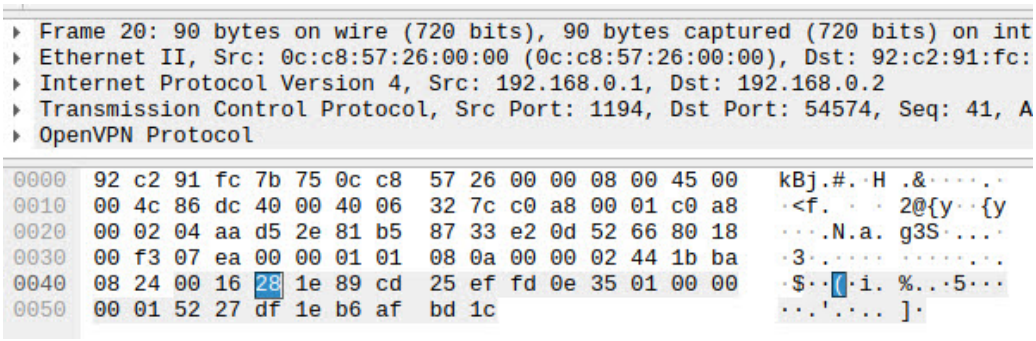
V další fázi analýzy se určovalo množství datových zpráv. Předchozí filtr byl tedy změněn tak, aby obsahoval pouze byty datových paketů,
a to: „`openvpn.type == 0x30 or openvpn.type == 0x48`“.

První soubor obsahoval 50 kontrolních a 1016 datových zpráv. Celkově tedy 1066 OpenVPN paketů. Tento počet byl ověřen použitím nejjednodušího filtru „`openvpn`“.

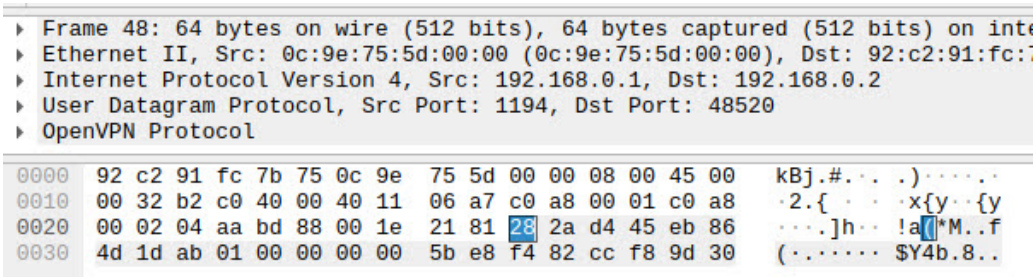
Při analýze druhého souboru s TCP provozem se použily stejné filtry. Počet kontrolních paketů byl 63 a datových bylo 876 (celkem tedy druhý soubor obsahoval 939 paketů).

V obou souborech byla zároveň objevena odlišnost v implementaci OpenVPN v případě využití zařízení MikroTik. Ukázalo se, že při použití UDP protokolu se po prvních třech zprávách (`P_CONTROL_HARD_RESET_CLIENT`, `P_CONTROL_HARD_RESET_SERVER` a `ACK` posílá navíc znovu paket `P_CONTROL_HARD_RESET_CLIENT`.

V případě použití TCP protokolu dojde k prohození dvou začátečních zpráv komunikace. První byla zachycena zpráva
`P_CONTROL_HARD_RESET_SERVER` a až po ní následovaly pakety



Obrázek 4.6: Byte kontrolní zprávy v TCP OpenVPN paketu.



Obrázek 4.7: Byte kontrolní zprávy v UDP OpenVPN paketu.

`P_CONTROL_HARD_RESET_CLIENT` a `ACK`, což neodpovídá specifikaci výše popisované VPN služby (viz sekce 2.8.2). Tomuto chování tedy musel být analyzátor přizpůsoben, což je více popsáno v sekci 5.3.

Následně byly u obou souborů, stejně jako v předchozí analýze IPSec provozu, zkoumány bytové charakteristiky OpenVPN hlavičky. Její struktura se však liší v závislosti na použitém transportním protokolu.

V případě TCP se na začátku hlavičky nachází dva byty navíc, označující její délku. Tento rozdíl lze vidět na obrázcích 4.6 a 4.7.

Shrnutí ruční analýzy provozu z druhého scénáře lze vidět v tabulkách 4.2 a 4.3.

IP adresy zařízení	192.168.0.1	192.168.0.2	192.168.0.3
Počet paketů celkem	1066	262	804
Počet bytů celkem	135 KB	34 KB	100 KB
Příchozí pakety	534	130	402
Odchozí pakety	532	132	402
Kontrolní pakety	50	25	25
Datové pakety	1016	237	779

Tabulka 4.2: Výsledek ruční analýzy v UDP komunikaci ze scénáře OpenVPN.

IP adresy zařízení	192.168.0.1	192.168.0.2	192.168.0.3
Počet paketů celkem	939	609	330
Příchozí pakety	466	308	165
Odchozí pakety	473	301	165
Kontrolní pakety	63	24	39
Datové pakety	876	585	291

Tabulka 4.3: Výsledek ruční analýzy v TCP komunikaci ze scénáře OpenVPN.

4.6.3 Analýza WireGuard VPN

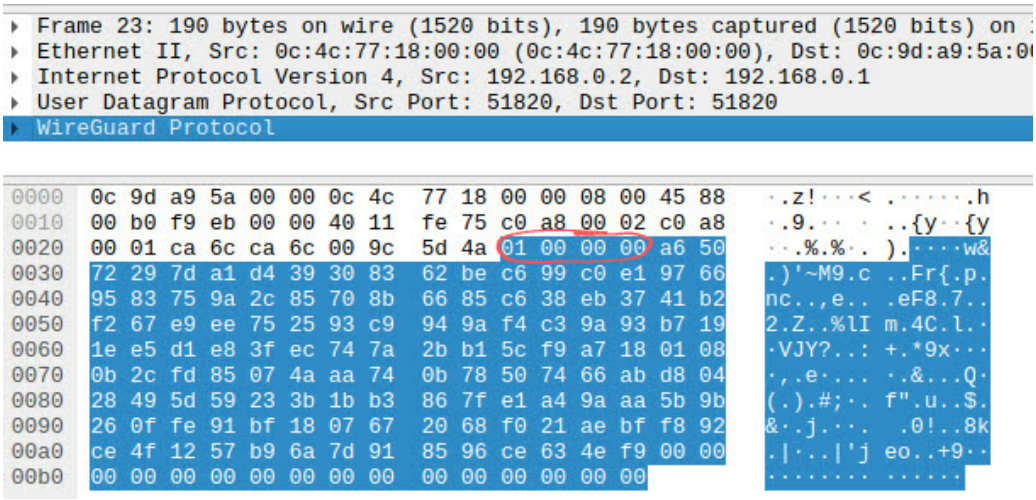
Pro analýzu dat z posledního scénáře, ve kterém byla využita WireGuard VPN, bylo opět nutné zobrazit ve WireSharku pouze ty pakety, které se dané služby týkají. Stejně jako v předchozích případech se nejprve vyfiltrovaly kontrolní zprávy. Pro tento účel byly využity filtry na jednotlivé typy WireGuard zpráv. Konkrétně se jednalo o filtr „`wg.type == 0x1 or wg.type == 0x2 or wg.type == 0x3`“. Jelikož však byla v tomto scénáři využita zařízení od MikroTiku, která při komunikaci využívala navíc zprávy ICMP, musel být filtr navíc upraven na „`(wg.type == 0x1 or wg.type == 0x2 or wg.type == 0x3) and not icmp`“. Komunikace obsahovala čtyři kontrolní pakety.

V dalším kroku se určoval počet datových paketů. Předchozí filtr byl tedy upraven na „`wg.type == 0x4 and not icmp`“. Datových zpráv se zachytilo 176. Celkově soubor obsahoval 180 WireGuard paketů, což se ověřilo použitím filtru „`wg and not icmp`“.

Takto vyfiltrovaný provoz byl dále zkoumán a z WireGuard hlavičky byly opět vypozorovány unikátní charakteristiky komunikace. Ty byly potvrzeny také v oficiálním dokumentu k dané VPN [14]. Jedná se celkem o 4 byty,

které lze vidět na obrázku 4.8. První byte označuje typ zprávy (v obrázku 4.8 jde o zprávu `Handshake initiation`) a zbylé tři jsou nulové hodnoty (`Reserved`).

Shrnutí analýzy WireGuardového provozu lze vidět v tabulce 4.4.



Obrázek 4.8: Byty ve WireGuard hlavičce paketu.

IP adresy zařízení	192.168.0.1	192.168.0.2
Počet paketů celkem	180	180
Počet bytů celkem	24 KB	24 KB
Příchozí pakety	100	80
Odchozí pakety	80	100
Kontrolní pakety	4	4
Datové pakety	176	176

Tabulka 4.4: Výsledek ruční analýzy v komunikaci ze scénáře WireGuard.

4.6.4 Analýza provozu z kolejní sítě

V kolejní síti ZČU byly pro testování analyzátoru vytvořeny celkem dva soubory. První z nich měl 7,1 GB, druhý měl 7,0 GB. Jednalo se o přibližně pět minut reálného provozu v dané síti. Již při takto velkých souborech začala být ruční analýza WireSharkem pomalejší než v předchozích případech. Postupovalo se však podobným způsobem.

Nejprve opět došlo k vyfiltrování čistě VPN provozu. V obou případech byly postupně vyzkoušeny všechny výše uvedené filtry (viz sekce 4.6).

V prvním souboru byla objevena IPSec VPN a WireGuard VPN. Konkrétně se jednalo o 41098 IPSec paketů, z toho 5 paketů bylo součástí navazování IPSec spojení. WireGuard komunikace obsahovala 32826 paketů. Součástí komunikace bylo celkem 9 IP adres. Některé z nich však v tomto případě nesplňovaly předpoklad záchytu komunikace implementovaným analyzátorem, který určuje, že v zachycené komunikaci musí být VPN provoz kompletní, tj. vždy po navázání VPN spojení se musí přenést alespoň jeden datový paket. Uživatel sítě tedy danou VPN musí skutečně využívat a posílat přes ní data. Tento předpoklad splňovaly pouze IP adresy 10.10.23.23, 178.255.170.247, 10.10.23.61 a 10.10.23.13. Zbylým adresám chyběla buď kontrolní či datová část komunikace. OpenVPN filtr neukázal žádný provoz.

Výsledek ruční analýzy z prvního souboru lze vidět v tabulce 4.5.

Ve druhém souboru byl po použití všech filtrů nalezen také pouze WireGuard a IPSec provoz. Konkrétně se jednalo o 4758 paketů, z nichž 50 paketů bylo součástí IPSec komunikace, která však také nesplnila podmínku analyzátoru (jednalo se pouze o kontrolní pakety). Korektní komunikace byla tudíž pouze WireGuard VPN, a to pouze jen některé IP adresy. Konkrétně se jednalo o adresy 10.10.23.23, 10.10.23.13, 10.10.23.61 a 178.255.170.247. Výsledek této analýzy je zapsán v tabulce 4.6.

Z důvodu více komunikujících zařízení byly z tabulek oproti testovacím případům vynechány řádky s příchozími a odchozími pakety, zejména kvůli zachování přehlednosti.

Název VPN	IPSec	OpenVPN	WireGuard
Nalezena	ANO	NE	ANO
IP adresy	10.10.43.48 185.216.35.28	0	3.73.99.148 10.10.13.56 10.10.23.13 10.10.23.23 10.10.23.61 10.10.45.17 176.31.233.32 155.133.226.75 178.255.170.247
Počet paketů celkem	41098	0	3286
Kontrolní pakety	5	0	47
Datové pakety	41093	0	3239

Tabulka 4.5: Nalezené VPN z prvního souboru z kolejní sítě.

Název VPN	IPSec	OpenVPN	WireGuard
Nalezena	ANO	NE	ANO
IP adresy	10.10.13.24 10.10.43.58 31.30.69.152	0	10.10.20.23 10.10.23.13 10.10.23.23 10.10.23.61 10.10.21.44 87.239.4.170 3.121.192.248 178.255.170.247
Počet paketů celkem	50	0	4708
Kontrolní pakety	50	0	48
Datové pakety	0	0	4660

Tabulka 4.6: Nalezené VPN z druhého souboru z kolejní sítě.

5 Návrh analyzátoru

V této části práce bude popsán návrh vlastního síťového analyzátoru, včetně jeho struktury, popisu jak probíhá filtrace, vlastní analýza a následné zpracování výsledků.

5.1 Výběr knihovny

V prvním návrhu programu bylo předzpracování a rozpoznání dat k filtraci a k analýze implementováno pomocí knihovny s názvem *Scapy*[1]. Postupem času se ale tato knihovna ukázala jako nevhodná a byla nahrazena jinou knihovnou, a to *dpkt*[2].

5.1.1 Scapy

Scapy je nativní knihovna pro Linux, Windows a většinu Unixových systémů, které podporují *libpcap*[3]. Podporuje Python 2 i Python 3. Dokáže různými způsoby manipulovat se síťovou komunikací (umožňuje vytvářet a odesílat vlastní pakety do sítě či přijímat a zpracovávat příchozí data, a to buď ze síťové karty nebo právě z `pcap` souborů). Poradí si jak s `pcap` formátem, tak s novější specifikací `pcapng` (viz sekce A.2).

Při implementaci analyzátoru byla tato knihovna využívána ve verzi 2.4.4 a zajišťovala především zpracování vstupních dat a jejich filtraci. Dokázala detekovat například jednotlivé vrstvy paketu, obsažené IP adresy a další užitečné informace. Bohužel se díky své robustnosti ukázala jako nevyhovující, a to zejména v požadavcích na rychlost běhu programu, které nesplňovala. Byla tak nahrazena jinou (rychlejší) knihovnou.

5.1.2 dpkt

Jedná se o knihovnu, která umožňuje základní rozpoznání TCP/IP protokolů síťové komunikace. Využívá se tedy pro filtr vstupních dat. Podobně jako výše popsaná Scapy umí vytvářet a zkoumat pakety.

Bohužel si dokáže poradit pouze s novější specifikací `pcap` souborů, a to `pcapng`. Z tohoto důvodu při použití této knihovny vzniká omezení podpory

[1] `https://scapy.net/`

[2] `https://github.com/kbandla/dpkt`

[3] `https://github.com/the-tcpdump-group/libpcap`

vlastního analyzátoru pouze na uvedený typ souborů. Na rozdíl od Scapy je však tato knihovna podstatně rychlejší, což byl hlavní důvod jejího použití. Srovnání časů běhů prototypu analyzátoru s těmito dvěma knihovnami lze vidět v tabulce 5.1. Porovnání rychlosti proběhlo na stroji s následujícími parametry:

- **CPU** – Intel Core i5-3350P 3,10 GHz;
- **RAM** – 16GB DDR3;
- **Disk** – Samsung SSD 850 EVO 250 GB.

Pokud by si uživatel chtěl spustit analýzu se Scapy (například kvůli podpoře `pcap` formátu), tak analyzátor poskytuje argument, který mu to umožní. Jedná se o parametr `--library`. Při jeho nepoužití se automaticky zvolí knihovna dpkt.

	Velikost dat	**Scapy** [s]	**dpkt** [s]
IPSec scénář	120,6 kB	00:00,18	00:00,05
OpenVPN TCP scénář	897,8 kB	00:00,96	00:00,25
OpenVPN UDP scénář	960,8 kB	00:00,80	00:00,17
WireGuard scénář	57,6 kB	00:00,12	00:00,03
Pcap z kolejní sítě 1	7,1 GB	44:18,57	09:23,15
Pcap z kolejní sítě 2	7,0 GB	42:01,63	08:44,36

Tabulka 5.1: Srovnání rychlostí knihoven

5.2 Struktura analyzátoru

V následujících sekcích se čtenář seznámí se všemi částmi výsledného programu.

Nejprve bude představen objektový návrh, dále bude popsána filtrace síťové komunikace a následně vlastní analýza a hledání VPN.

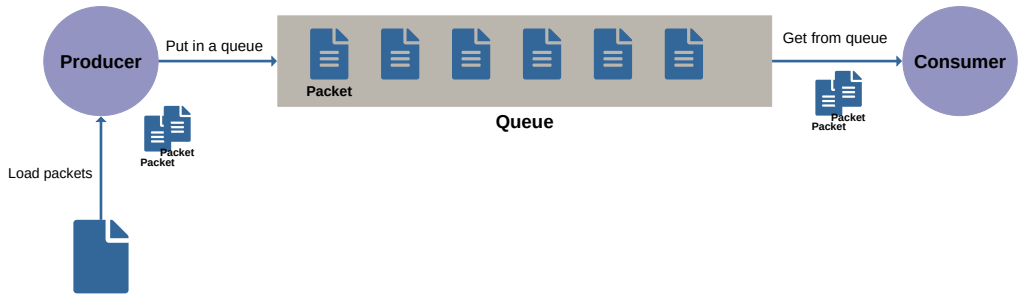
5.2.1 Objektový návrh

K zajištění efektivního načítání dat ze souboru a jejich následného zpracování byla využita jedna ze základních synchronizačních úloh. Konkrétně se jednalo o *producenta-konzumenta*.

Producent se stará o načítání dat z disku a jeho ukládání do klasické fronty, která je součástí Pythonu a zajišťuje většinu synchronizační části v této úloze.

Konzument vybírá postupně data z fronty a volá funkce pro filtrování nepotřebných paketů (viz sekce 5.2.2). Pokud projde paket filtrem, dojde k vytvoření struktury, která je nezávislá na použité knihovně (viz sekce 5.2.3) a zavolá se funkce pro rozpoznání VPN.

Schéma použití principu producenta-konzumenta v této bakalářské práci je znázorněno na obrázku 5.1.



Obrázek 5.1: Schéma producenta-konzumenta.

Výsledný analyzátor tak obsahuje několik objektů. Konkrétně se jedná o:

- **ScapySniffer** – Slouží k předzpracování síťové komunikace pomocí knihovny Scapy. Obsahuje tedy všechny funkce, které tuto knihovnu využívají. Jedná se o funkce pro načítání dat ze souboru, pro filtraci nepotřebných paketů a o tvorbu vlastní nezávislé struktury `Packet`.
- **DpktSniffer** – Slouží k předzpracování síťové komunikace pomocí knihovny dpkt. Zastává stejnou činnost jako `ScapySniffer`.
- **MainAnalyzer** – Hlavní analyzátor, který spouští analýzu VPN služeb pomocí volání funkcí jednotlivých VPN analyzátorů, jejichž instance jsou zde vytvořeny na základě filtrů od uživatele. Zároveň se stará o přidávání výsledků do slovníku, a také o odfiltrování případných falešně pozitivních detekcí, což opět zajišťuje pomocí volání funkcí z daných analyzátorů.

- **IPSecAnalyzer, WireGuardAnalyzer, OpenVPNAnalyzer, OpenVPNMikrotikAnalyzer** – Jednotlivé VPN analyzátory. Každý z nich obsahuje detekci „své“ VPN a odfiltrování falešných záchytů. OpenVPN analyzátory obsahují navíc konečné automaty (viz sekce 5.3.2).

5.2.2 Filtrace

Před zahájením vlastní analýzy síťové komunikace, bylo nejprve potřeba odfiltrovat pakety, které se VPN zcela jistě netýkají. Jednalo se konkrétně o následující typy zpráv a protokoly:

1. **IPv6** – Z důvodu rozsahu práce bylo stanoveno omezení detekce VPN pouze na IPv4 provoz.
2. **LLC**[4] **+ LLDPDU**[5] – Protokoly druhé (spojové) vrstvy ISO/OSI modelu.
3. **ARP**[6] – Protokol, sloužící ke zjištění MAC adresy zařízení, u kterého je známá pouze jeho IP adresa. Není součástí navazování ani udržování VPN spojení.
4. **Multicast + Broadcast** – Tyto dva typy zpráv jsou hojně využívány v lokálních sítích, ale ze stejného důvodu jako v případě ARP protokolu pro vlastní detekci VPN nejsou potřebné, jelikož pomocí nich neprobíhá navazování komunikace ani její udržování.
5. **ICMP**[7] – Protokol, který informuje o chybových stavech v síťové komunikaci.
6. **DNS**[8] – Překlad doménových názvů na IP adresy a opačně.
7. **CLDAP + LDAP**[9] – Umožňuje přístup a modifikaci dat uložených v síti typicky v Active Directory.
8. **SNMP**[10] – Sledování stavu sítě.

[4]Logical Link Control
[5]Link Layer Discovery Protocol
[6]Address Resolution Protocol
[7]Internet Control Message Protocol
[8]Domain Name System
[9](Connection-less) Lightweight Directory Access Protocol
[10]Simple Network Monitor Protocol

5.2.3 Struktura Packet

Z dat, která projdou filtrem, se nejprve sestaví vlastní struktura (vlastní `Packet`), která však obsahuje pouze informace, které jsou pro detekci potřeba, a to jsou:

- zdrojová a cílová IP adresa;
- číslo IP protokolu;
- typ transportního protokolu (TCP/UDP/AH/ESP);
- zdrojový a cílový port;
- obsah paketu (bez hlaviček);
- velikost obsahu paketu.

Součástí této struktury je také údaj (`mark`) o kterou rozpoznanou VPN službu se jedná a ke které části komunikace (zda ke kontrolní nebo k datové) se daný paket řadí (například `IPSEC_CONTROL`). Při tvorbě paketu je tato proměnná nastavena na výchozí hodnotu -1 a konkrétní hodnota se doplní až v průběhu analýzy.

5.3 Implementace detekce jednotlivých VPN

Jedná se o nejdůležitější část celého programu. Jsou zde postupně zkoumány pakety ve frontě a dle VPN služby, kterou chce uživatel rozpoznat, se provádí konkrétní část detekce. Vždy se vychází z předpokladu, že komunikace je v daném souboru zachycena celá, tj. přenosu dat předchází navázání spojení.

Program umí detekovat tři VPN: IPSec, OpenVPN a WireGuard a jednu modifikovanou implementaci OpenVPN (OpenVPN-Mikrotik). Rozlišuje, zda se jedná o navazování spojení nebo o probíhající komunikaci.

5.3.1 Detekce IPSec VPN

Rozpoznání této VPN zajišťuje objekt `IPSecAnalyzer`.

U IPSecu jsou kromě bytů analyzovány také porty a transportní protokoly, jelikož jako jediný ze tří vybraných a zkoumaných protokolů je definován v RFC dokumentu, a tudíž je komunikace standardizována.

V případě detekce IKE se porovnávají přímo byty na konkrétní pozici v hlavičkách, které odpovídají struktuře dané VPN služby (viz sekce 4.6.1).

K detekci datových paketů je využito knihoven (buď Scapy nebo dpkt), které dokáží AH či ESP protokol rozpoznat. Tato informace se vkládá do struktury `Packet` při její tvorbě.

5.3.2 Detekce OpenVPN

Analýzu této služby zajišťují objekty `OpenVPNAnalyzer`, případně `OpenVPNMikrotikAnalyzer`.

OpenVPN má několik typů kontrolních zpráv, které jsou mezi zařízeními vyměňovány na začátku komunikace (viz sekce 2.8.2). Při analýze je porovnávána celá hodnota v hlavičce (tj. je zkoumán celý byte, ve kterém se nachází `Opcode` hodnota a zároveň `key_id` (konkrétní hodnoty byly popsány v sekci 4.6.2)).

U OpenVPN probíhá detekce (kromě porovnávání jednotlivých bytů v paketech), také pomocí konečného automatu. K tomuto řešení se přistoupilo zejména kvůli zvýšenému výskytu falešně pozitivních výsledků z průběhu celé OpenVPN komunikace, což bylo dáno porovnáváním pouze jediného bytu v hlavičce tohoto protokolu. Byla zde tedy šance 1 : 256, že náhodný paket bude klasifikován jako OpenVPN.

Přechodový graf konečného automatu je znázorněn na obrázku 5.2.

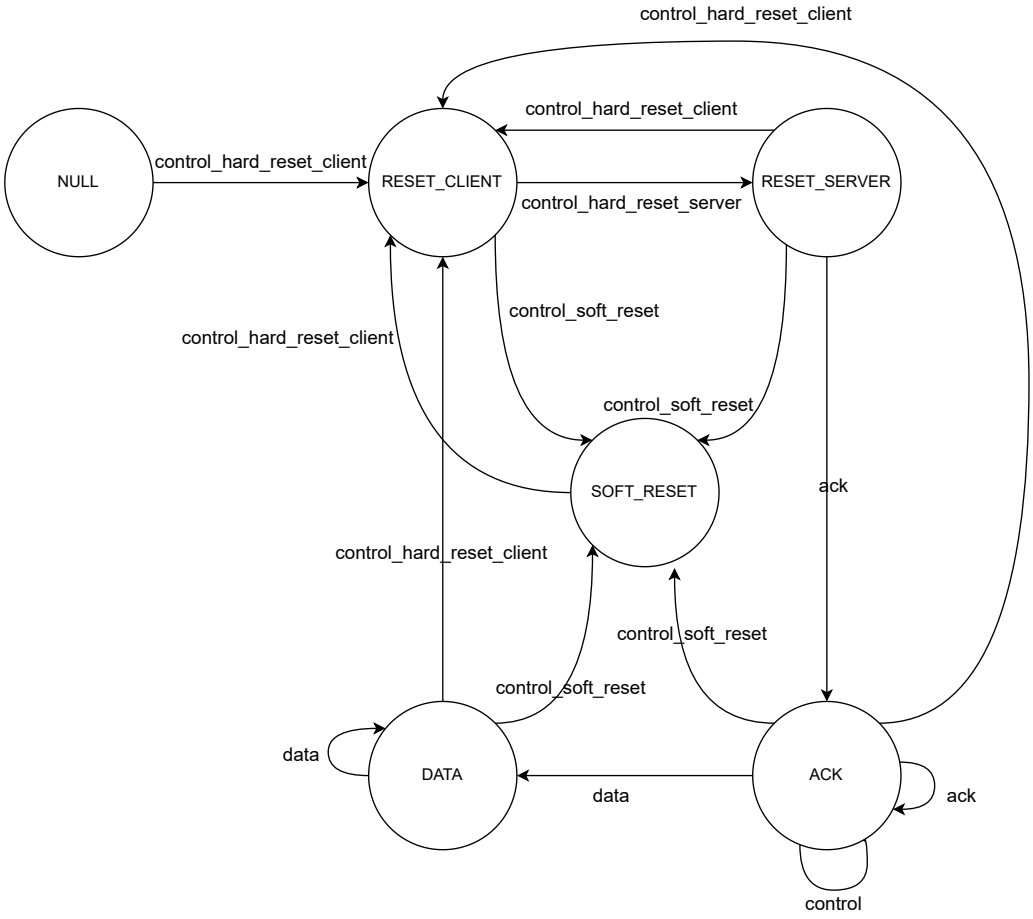
Z důvodu objevené nesrovnalosti u MikroTik zařízení při ruční analýze v komunikaci OpenVPN bylo nutné udělat i druhý automat, který byl modifikován tak, aby detekoval i MikroTik OpenVPN (původní automat z objektu `OpenVPNAnalyzer` byl oddědě n a bylo nutné vyměnit počáteční přechody a stavy komunikace). Vznikla tak tedy možnost detekovat další implementaci OpenVPN služby.

Zároveň bylo potřeba se vypořádat s fragmentací a skládáním více paketů do jednoho při použití OpenVPN protokolu, který komunikuje přes TCP. K tomu byl využit buffer, do kterého jsou ukládány rozdělené OpenVPN zprávy.

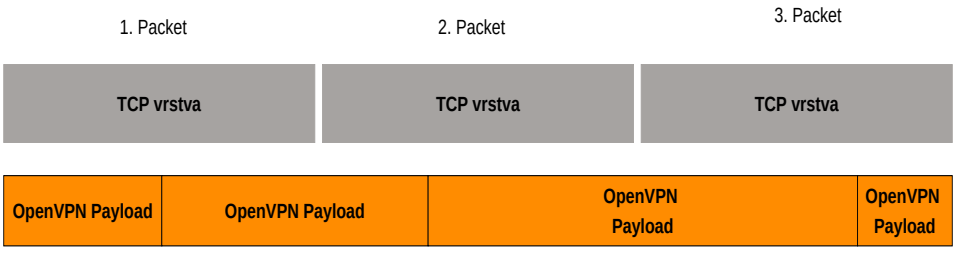
To, zda je OpenVPN obsah (payload) fragmentován, se poznává dle rozdílu délky dat v paketu (bez hlaviček) a velikosti TCP payloadu. Jestliže je tento rozdíl větší než nula, pak data v paketu nejsou úplná a předpokládá se, že zbytek chybějících dat obsahují následující OpenVPN pakety. Neúplná část se tedy umístí do bufferu a je spojena s dalšími OpenVPN daty. Toto se opakuje do doby, dokud se rozdíl pro danou komunikaci nerovná nule. Fragmentovaný OpenVPN payload je znázorněn na obrázku 5.3.

Analogický postup se uplatnil při počítání více OpenVPN payloadů obsažených v jednom paketu. Je zjištěna délka OpenVPN obsahu (OpenVPN při komunikaci s využitím protokolu TCP tuto informaci obsahuje hned na

začátku dat), toto množství dat je přeskočeno a zkoumá se, zda byty na aktuální pozici obsahují validní OpenVPN `Opcode`. Pokud tomu tak je, postup se opakuje a postupně dochází k započítávání těchto vnořených paketů.



Obrázek 5.2: Přechodový graf konečného automatu OpenVPN.



Obrázek 5.3: Schéma segmentace OpenVPN.

5.3.3 Detekce WireGuard VPN

Podobným způsobem (tedy zkoumáním jednotlivých bytů na konkrétních pozicích paketů) je realizovaná i analýza WireGuardu, o kterou se stará objekt `WireGuardAnalyzer`. Zkoumají se byty na začátku WireGuard hlavičky. První byte určuje typ zprávy (například paket `Handshake init` má na této pozici hodnotu '`0x1`', apod. (viz tabulka 2.3)). Za tímto bytem následují další tři nulové byty. Pokud tuto sekvenci paket ze zkoumaného souboru obsahuje, pak je pravděpodobné, že se jedná o WireGuard komunikaci.

5.4 Formát výstupu analýzy

Jestliže je v průběhu činnosti programu detekován VPN paket, provede se jeho přidání do asociativního pole (slovníku). V něm se přiřazuje počet paketů dané VPN služby ke komunikujícím IP adresám a zároveň se rozlišuje, zda se jedná o handshake (navazování spojení) nebo o data. Také se zde uvádí informace, jestli jde o příchozí či odchozí komunikaci do nebo z předem specifikované sítě.

Každá VPN komunikace je identifikována pomocí čtveřice informací:

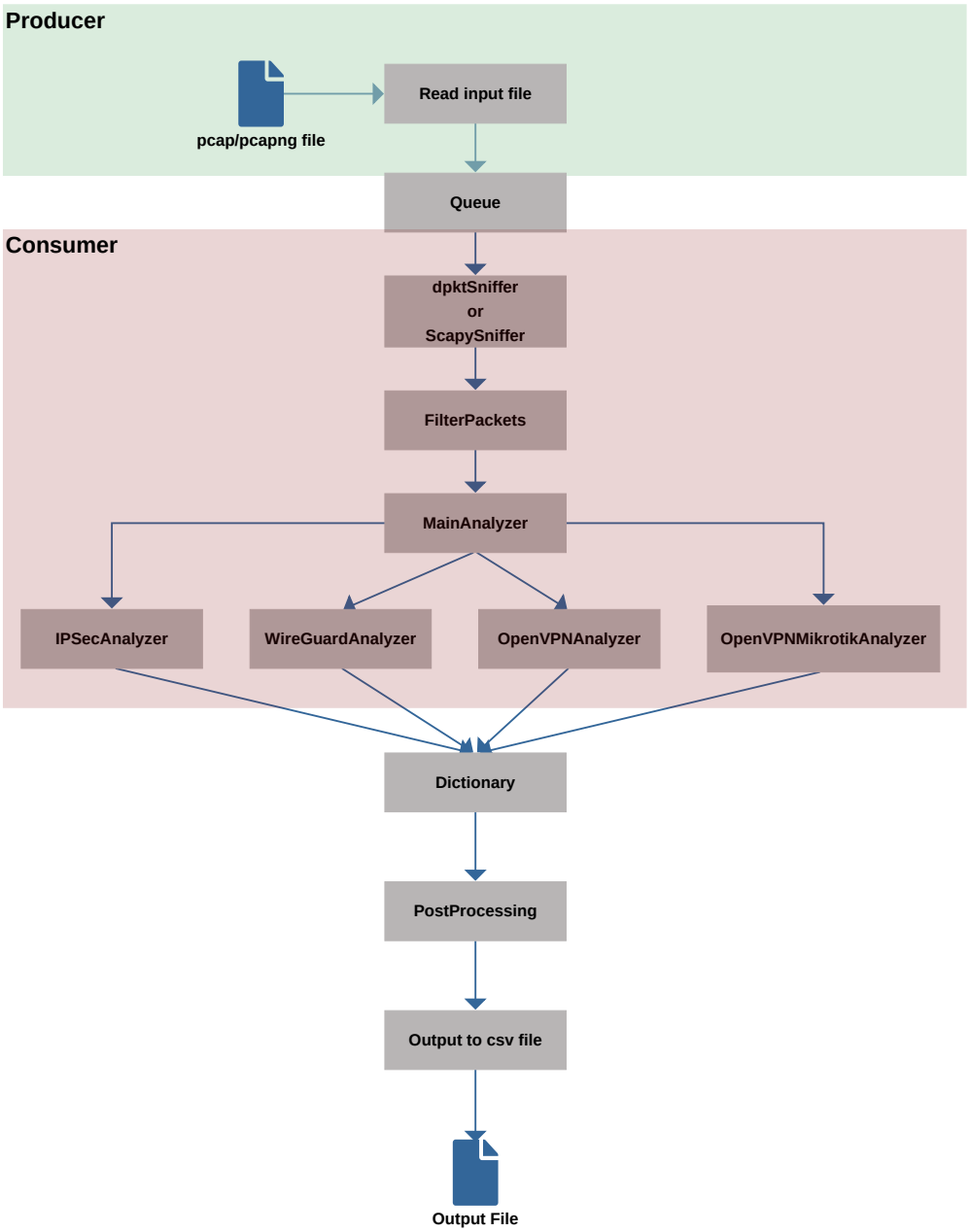
1. zdrojová IP adresa;
2. zdrojový port;
3. cílová IP adresa;
4. cílový port.

5.5 Eliminace falešně pozitivních výsledků

Po skončení analýzy se ještě kontroluje přítomnost kontrolních a datových záchytů v rámci každé komunikace. Pokud nějaká z těchto dvou složek chybí, považuje se daný záchyt za nevalidní a z výsledků je odstraněn.

Ze zbývajících výsledků se vytvoří soubor, přičemž jeho jméno nemusí být specifikováno uživatelem. Pokud uživatel explicitně nezadá jméno tohoto souboru, vytvoří program soubor s názvem *„Detection.csv“* a po zapsání výsledků se program ukončí.

Aktuální průběh a zpracování vstupních dat znázorňuje obrázek 5.4.



Obrázek 5.4: Průběh zpracování vstupních dat v analyzátoru.

6 Srovnání výsledků analýzy

Po implementaci analyzátoru přišlo na řadu testování. To se provádělo na stejných datech, na kterých byla udělána ruční analýza.

Nejprve se testovaly soubory z jednotlivých scénářů. Poté se přešlo k testování na záchytech reálného provozu, což mělo odhalit zejména výkonnostní a rychlostní nedostatky implementovaného řešení.

Jako první se analyzátor testoval na souboru ze scénáře IPSec.

6.1 Analýza provozu ze scénáře IPSec

Jak ukázala ruční analýza, v zachyceném provozu se nachází IPSec komunikace od zařízení s IP adresami 10.0.0.1 a 10.0.0.2. Obsahoval celkem 286 paketů patřících k této komunikaci.

6.1.1 Konfigurace analyzátoru

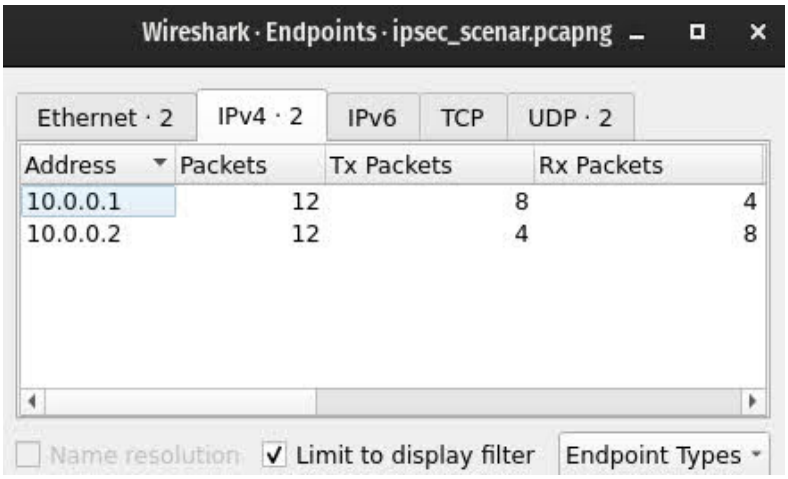
Nejprve je nutné analyzátor spustit se správnými parametry. Parametr `network` se omezil na síť 10.0.0.0/24, ve které se VPN provoz hledá. Použily se všechny filtry, aby se případně odhalily falešně pozitivní výsledky.

Seznam parametrů použitých na tento scénář byl následující:

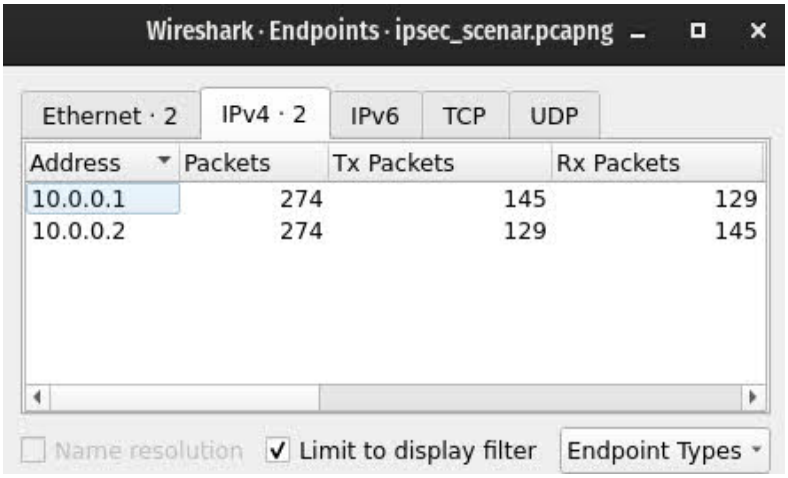
```
—pcap ipsec_scenar.pcapng
—outputFile ipsec_scenar.csv
—network 10.0.0.0/24
—filterNames IPSEC,WIREGUARD,OPENVPN,OPENVPN_MIKROTIK
```

6.1.2 Výsledky analýzy

Při zobrazení statistik z WireSharku (viz obrázek 6.1 a 6.2) bylo zjištěno, že scénář obsahuje celkem 12 kontrolních a 274 datových IPSec paketů.



Obrázek 6.1: Analýza IPSec scénáře z WireSharku – kontrolní část.



Obrázek 6.2: Analýza IPSec scénáře z WireSharku – datová část.

Výstup z analýzy tvoří vždy *CSV*[1] soubor, který má následující strukturu:

- **VPN** – název detekované VPN + rozlišení kontrolní a datové části komunikace;
- **ADDRESS_A** – první IP adresa z páru komunikujících zařízení;
- **ADDRESS_B** – druhá IP adresa z páru komunikujících zařízení;
- **PACKETS_A** – počet paketů odeslaných zařízením A;
- **PACKETS_B** – počet paketů odeslaných zařízením B.

[1]Comma Separated Value

VPN	ADDRESS_A	ADDRESS_B	PACKETS_A	PACKETS_B
IPSEC_CONTROL	10.0.0.1	10.0.0.2	8	4
IPSEC_DATA	10.0.0.1	10.0.0.2	145	129

Tabulka 6.1: Výsledek analýzy IPSec scénáře – výstup z analyzátoru.

Tabulka 6.1 ukazuje výsledky první automatické analýzy síťového provozu, kterou provedl implementovaný analyzátor.

Po dokončení této analýzy se počet paketů detekovaných programem WireShark shodoval s výstupem programu a nebyly zde zaznamenány žádné falešně pozitivní výsledky.

6.2 Analýza UDP provozu ze scénáře OpenVPN

Dle ruční analýzy se v souboru OpenVPN (komunikující přes UDP protokol) nachází tři komunikující zařízení s IP adresami 192.168.0.1, 192.168.0.2 a 192.168.0.3. Celkem se přeneslo 1066 paketů.

6.2.1 Konfigurace analyzátoru

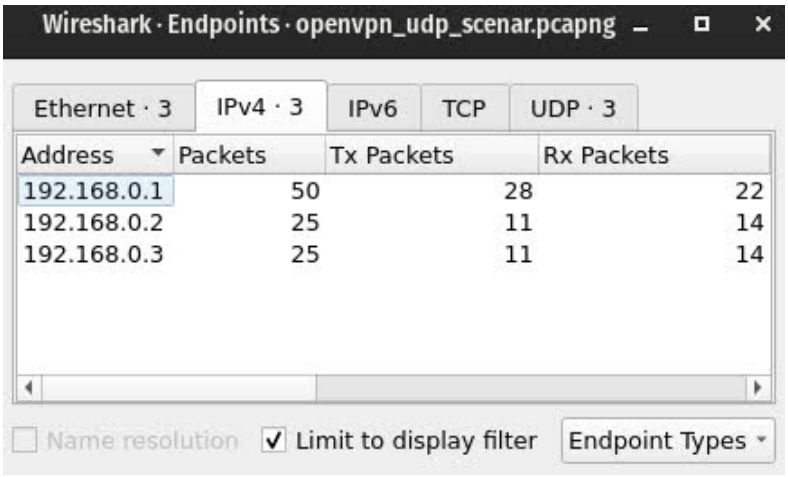
V tomto scénáři došlo k úpravě jména vstupního a výstupního souboru. Také bylo třeba nastavit parametr zkoumané sítě na 192.168.0.0/24.

Seznam parametrů tedy byl:

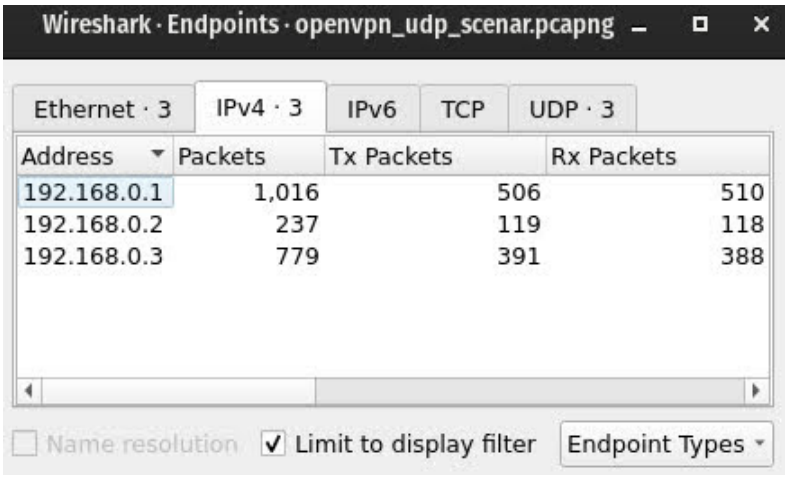
```
—pcap openvpn_udp_scenar.pcapng
—outputFile openvpn_udp_scenar.csv
—network 192.168.0.0/24
—filterNames IPSEC,WIREGUARD,OPENVPN,OPENVPN_MIKROTIK
```

6.2.2 Výsledky analýzy

Dle statistik z WireSharku by měl analyzátor detekovat stejný počet paketů, jaký se nachází na obrázcích 6.3 a 6.4.



Obrázek 6.3: Analýza OpenVPN UDP scénáře z WireSharku – kontrolní část.



Obrázek 6.4: Analýza OpenVPN UDP scénáře z WireSharku – datová část.

Výstup z analýzy OpenVPN UDP, kterou prováděl implementovaný analyzátor je vidět v tabulce 6.2.

VPN	ADDRESS_A	ADDRESS_B	PACKETS_A	PACKETS_B
OPENVPN_MK_CONTROL	192.168.0.1:1194	192.168.0.2:48520	14	11
OPENVPN_MK_CONTROL	192.168.0.1:1194	192.168.0.3:59474	14	11
OPENVPN_MK_DATA	192.168.0.1:1194	192.168.0.2:48520	118	119
OPENVPN_MK_DATA	192.168.0.1:1194	192.168.0.3:59474	388	391

Tabulka 6.2: Výsledek analýzy UDP OpenVPN scénáře – výstup z analyzátoru.

Ve sloupečku s názvem *VPN* vidíme, že rozpoznaná komunikace patřila

MikroTikové specifikaci OpenVPN, což se shoduje s testovacím scénářem, ve kterém skutečně byla použita zařízení MikroTik.

Jestliže spočítáme kontrolní pakety pro každý komunikující pár, dostaneme výsledné číslo 50 paketů ($14 + 14 + 11 + 11 = 50$), které se shoduje se statistikami z WireSharku. To samé platí o datových paketech ($118 + 119 + 388 + 391 = 1016$).

Počty paketů, které odeslala jednotlivá zařízení se také shodují s ruční analýzou i se statistikami (viz tabulka s výsledky 6.2).

Ani v tomto scénáři se žádné falešně pozitivní výsledky nenacházely.

Podobně dopadl i druhý testovací OpenVPN soubor, ve kterém byla komunikace přes TCP protokol.

6.3 Analýza TCP provozu ze scénáře OpenVPN

Ve druhém souboru s OpenVPN provozem se v průběhu ruční analýzy nalezlo celkem 939 OpenVPN paketů.

6.3.1 Konfigurace analyzátoru

Nastavení analyzátoru pro OpenVPN TCP soubor se lišilo pouze změnou v názvu vstupního a výstupního souboru.

```
—pcap openvpn_tcp_scenar.pcapng
—outputFile openvpn_tcp_scenar.csv
—network 192.168.0.0/24
—filterNames IPSEC,WIREGUARD,OPENVPN,OPENVPN_MIKROTIK
```

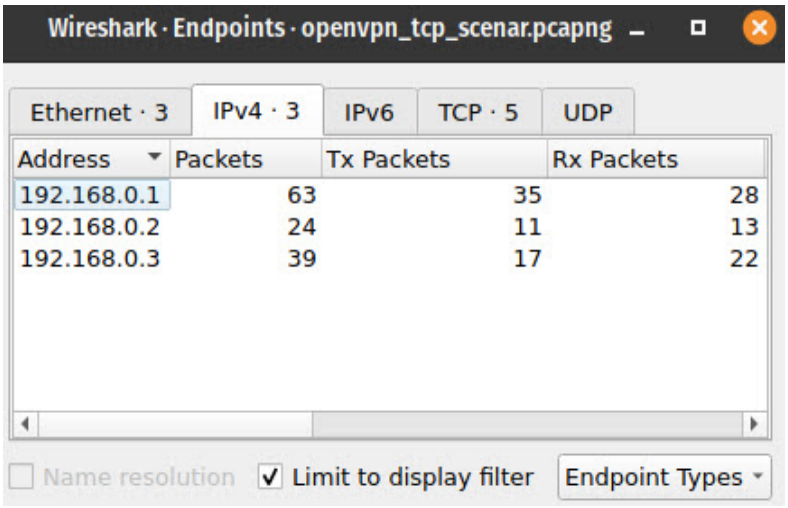
6.3.2 Výsledky analýzy

Dle WireSharku by měl analyzátor detekovat následující množství paketů (viz obrázky 6.5 a 6.6).

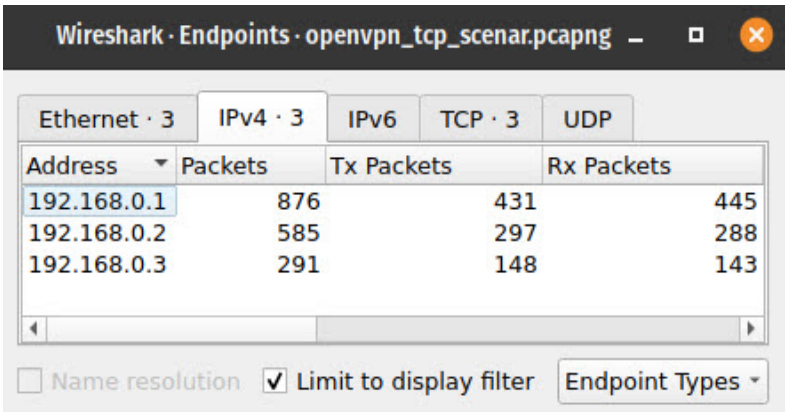
Výsledky automatické analýzy za pomocí implementovaného analyzátoru jsou uvedeny v tabulce 6.3.

Množství datových paketů ve výstupním souboru se shoduje s množstvím, které bylo zjištěno pomocí statistik WireSharku.

Množství kontrolních paketů se však liší. Analyzátor identifikoval celkem 46 těchto zpráv, zatímco v průběhu ruční analýzy se jich našlo 63. Důvodem



Obrázek 6.5: Analýza OpenVPN TCP scénáře z WireSharku – kontrolní část.



Obrázek 6.6: Analýza OpenVPN TCP scénáře z WireSharku – datová část.

těchto rozdílných počtů je TCP fragmentace. WireShark každý fragment počítá jako jeden samostatný paket. Analyzátor započítává všechny fragmenty, patřící k jednomu OpenVPN obsahu jako jeden paket.

VPN	ADDRESS_A	ADDRESS_B	PACKETS_A	PACKETS_B
OPENVPN_MK_CONTROL	192.168.0.1:1194	192.168.0.2:54574	12	11
OPENVPN_MK_CONTROL	192.168.0.1:1194	192.168.0.3:52700	12	11
OPENVPN_MK_DATA	192.168.0.1:1194	192.168.0.2:54574	288	297
OPENVPN_MK_DATA	192.168.0.1:1194	192.168.0.3:52700	143	148

Tabulka 6.3: Výsledek analýzy TCP OpenVPN scénáře – výstup z analyzátoru.

6.4 Analýza provozu ze scénáře WireGuard

Ruční analýza ukázala v zachyceném souboru celkem 180 WireGuard paketů. Komunikující zařízení měla IP adresy 192.168.0.1 a 192.168.0.2.

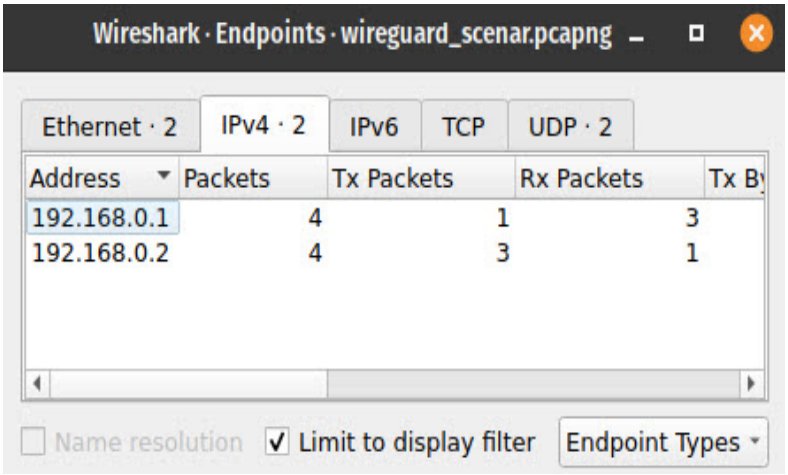
6.4.1 Konfigurace analyzátoru

Před spuštěním automatické analýzy se program nastavil s těmito parametry:

```
—pcap wireguard_scenar.pcapng
—outputFile wireguard_scenar.csv
—network 192.168.0.0/24
—filterNames IPSEC,WIREGUARD,OPENVPN,OPENVPN_MIKROTIK
```

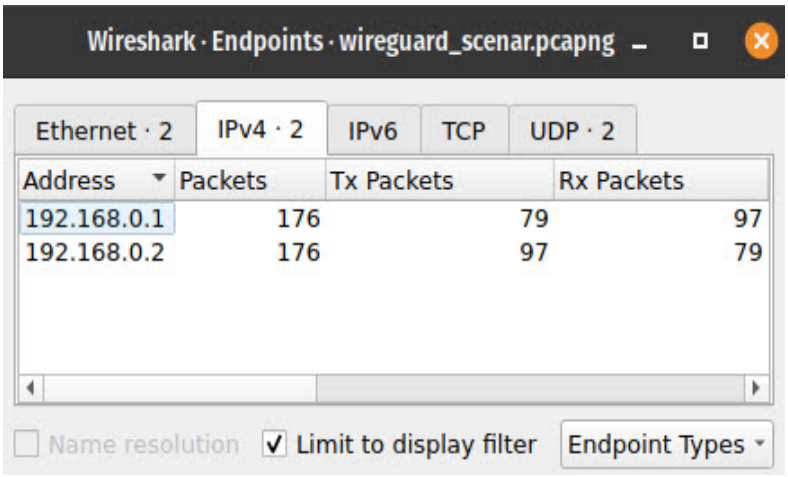
6.4.2 Výsledky analýzy

Statistiky z WireSharku ukazují, že by se ve výstupním souboru analyzátoru mělo nacházet následující množství paketů (viz obrázky 6.7 a 6.8).



Obrázek 6.7: Analýza WireGuard scénáře z WireSharku – kontrolní část.

Po porovnání výstupu z analyzátoru (který lze vidět v tabulce 6.4) s ruční analýzou se potvrdila přesnost automatické detekce VPN služeb prostřednictvím implementovaného programu i u třetího testovacího scénáře.



Obrázek 6.8: Analýza WireGuard scénáře z WireSharku – datová část.

VPN	ADDRESS_A	ADDRESS_B	PACKETS_A	PACKETS_B
WIREGUARD_DATA	192.168.0.1:51820	192.168.0.2:51820	79	97
WIREGUARD_CONTROL	192.168.0.1:51820	192.168.0.2:51820	1	3

Tabulka 6.4: Výsledek analýzy WireGuard scénáře – výstup z analyzátoru.

6.5 Analýza provozu z kolejní sítě

Po otestování funkčnosti analyzátoru na testovacím provozu se přešlo k testování výkonosti a přesnosti automatické analýzy na reálných datech z kolejní sítě Západočeské univerzity. V obou zkoumaných souborech byly během ruční analýzy nalezeny vždy dvě VPN – IPSec a WireGuard.

6.5.1 Konfigurace analyzátoru pro první soubor

Při zkoumání reálného síťového provozu se parametry programu upravily následujícím způsobem:

```
--pcap provoz_z_koleje1.pcapng
--outputFile provoz_z_koleje1.csv
--network 0.0.0.0/0
--filterNames IPSEC,WIREGUARD,OPENVPN,OPENVPN_MIKROTIK
```

Zkoumaná síť byla nastavena na nulové hodnoty (což znamená, že projdou všechny komunikace obsažené v souboru), aby se zachytily případné jiné VPN detekce, které by ruční analýza programem WireShark nedokázala odhalit (například kvůli změně portů).

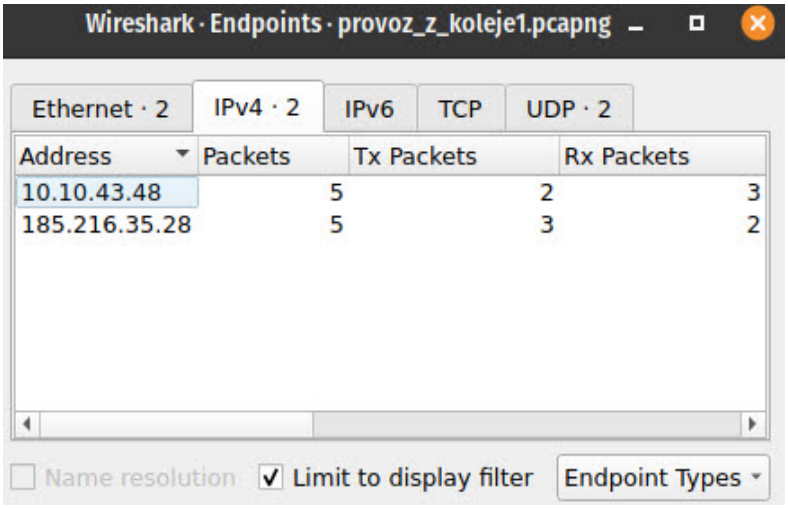
6.5.2 Výsledky analýzy prvního souboru z koleje

V prvním souboru, který měl velikost 7,0 GB, se při ruční analýze objevily VPN služby IPSec a WireGuard. Ve výsledcích z analyzátoru by měla však být jen část WireGuard provozu, jelikož některé WireGuardové komunikace nesplňovaly v tomto případě předpoklad analyzátoru na rozpoznání VPN přenosu (datovému provozu musí vždy předcházet navazování spojení). V případě IPSecu to v tomto případě platilo u všech zjištěných IP adres.

Množství kontrolních paketů u IPSecové komunikace je vidět na obrázku 6.9, datové pakety jsou na obrázku 6.10.

Statistiky z WireGuardové komunikace lze vidět na obrázcích 6.11 a 6.12.

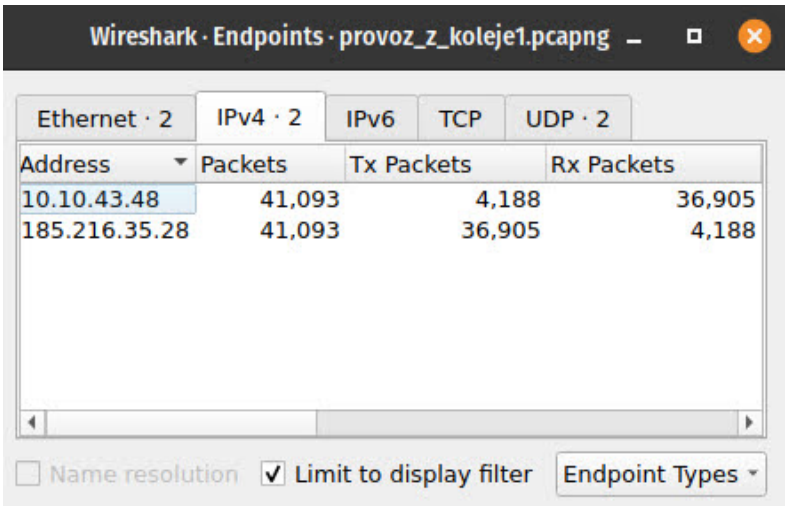
Obsah výstupního souboru z průběhu automatické analýzy zobrazuje tabulka 6.5.



Obrázek 6.9: Analýza prvního provozu z koleje ZČU ve WireSharku — IPSec, kontrolní část.

Pokud srovnáme počty z výstupního souboru z analyzátoru, zjistíme, že množství zachycených a rozpoznaných paketů IPSec i WireGuard souhlasí se statistikami ve WireSharku.

V průběhu automatické analýzy však byla navíc identifikovaná ještě OpenVPN, kterou WireShark nerozpoznal. Důvodem byla změna portu u této služby (analyzátor klasifikoval v rámci OpenVPN komunikace dvě zařízení s IP adresami 10.10.10.28 a 92.223.65.152, využívající při komunikaci porty 63293 (což je dynamický port, pravděpodobně používaný klientem) a 443 (což je port, který obvykle značí HTTPS komunikaci, avšak v tomto případě byl využit pravděpodobně také k „zamaskování“ OpenVPN provozu).



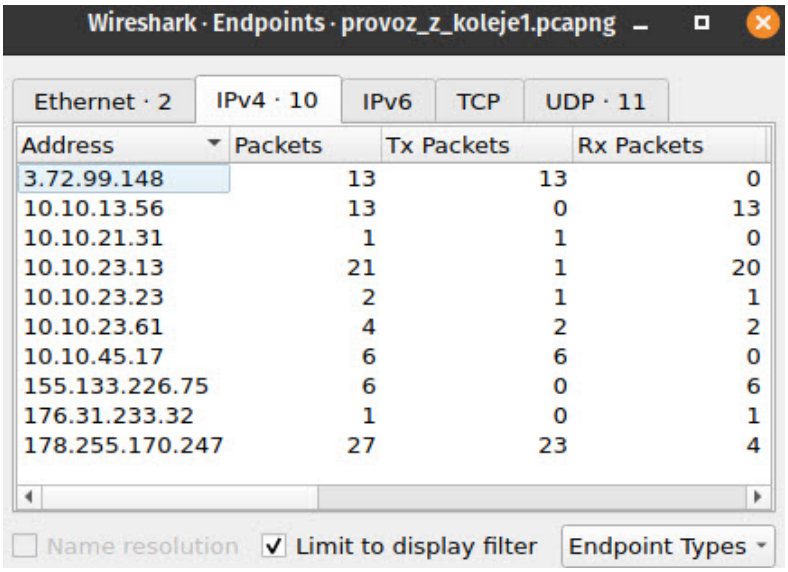
Obrázek 6.10: Analýza prvního provozu z koleje ZČU ve WireSharku — IPSec, datová část.

VPN	ADDRESS_A	ADDRESS_B	PACKETS_A	PACKETS_B
WIREGUARD_DATA	10.10.23.23:58035	178.255.170.247:50069	99	57
WIREGUARD_DATA	10.10.23.61:51866	178.255.170.247:50069	110	85
WIREGUARD_DATA	10.10.23.13:41003	178.255.170.247:50002	914	1974
OPENVPN_CONTROL	10.10.20.28:63293	92.223.65.152:443	11	9
WIREGUARD_CONTROL	10.10.23.61:51866	178.255.170.247:50069	2	2
WIREGUARD_CONTROL	10.10.23.13:41003	178.255.170.247:50002	1	20
WIREGUARD_CONTROL	10.10.23.23:58035	178.255.170.247:50069	1	1
IPSEC_DATA	10.10.43.48	185.216.35.28	4188	36905
OPENVPN_DATA	10.10.20.28:63293	92.223.65.152:443	37	3
IPSEC_CONTROL	10.10.43.48	185.216.35.28	2	3

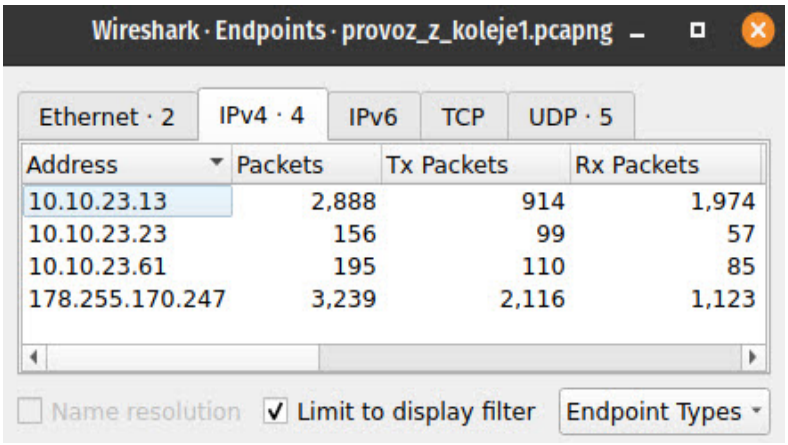
Tabulka 6.5: Výsledek analýzy prvního provozu z koleje ZČU – výstup z analyzátoru.

Aby tato hypotéza mohla být ověřena, muselo se upravit nastavení WireSharku. Nejprve se změnil port OpenVPN UDP z 1994 na 443 a vyfiltroval se OpenVPN provoz. Některé pakety však WireShark nedokázal identifikovat (označoval je jako *Malformed Packet*) a nedokázal z těchto paketů sestavit validní OpenVPN komunikaci. Z tohoto důvodu se nastavení WireSharku upravilo z portu UDP 443 na TCP port 443, což lze vidět na obrázku 6.13. Vzhledem k používanosti tohoto portu bylo nutné ještě přesně specifikovat komunikující pár, který identifikoval analyzátor, aby se předešlo falešně pozitivním výsledkům. Filtr ve WireSharku se tedy skládal z „`openvpn and (ip.addr == 92.223.65.152 or ip.addr == 92.223.65.152)`“.

Tento filtr dopadl lépe a opravdu ukázal „skrytou“ OpenVPN komunikaci. Ta byla ve WireSharku fragmentovaná a zároveň obsahovala i více



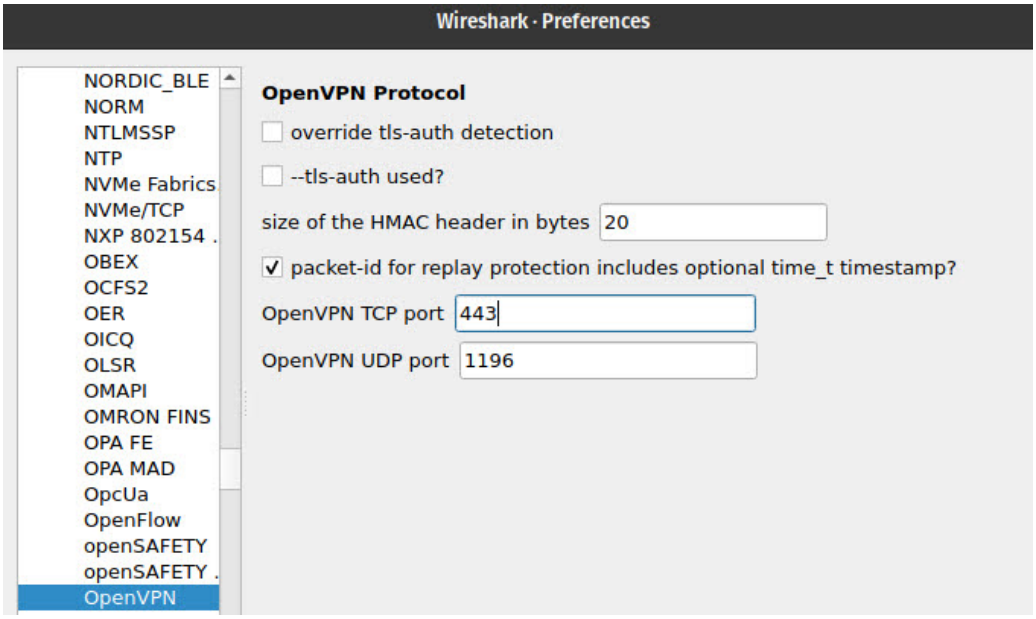
Obrázek 6.11: Analýza prvního provozu z koleje ZČU ve WireSharku — WireGuard, kontrolní část.



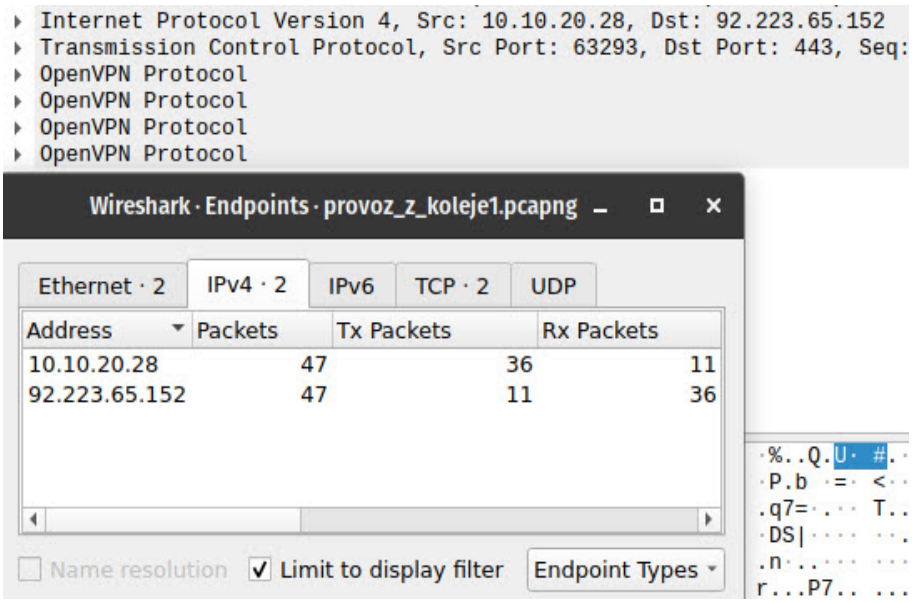
Obrázek 6.12: Analýza prvního provozu z koleje ZČU ve WireSharku — WireGuard, datová část.

poskládaných OpenVPN obsahů do samostatných paketů, takže statistiky z WireSharku (viz obrázek 6.14) se tentokrát více rozchází s výsledkem analyzátoru.

Implementovaný analyzátor však dokázal rozpoznat VPN, která ve WireSharku detekována nebyla.



Obrázek 6.13: Změna TCP portu ve WireSharku u služby OpenVPN.



Obrázek 6.14: OpenVPN na portu 443 v prvním souboru z kolejní sítě ZČU.

6.5.3 Konfigurace analyzátoru pro druhý soubor

U druhého souboru, který měl velikost 7,1 GB, se změnily pouze názvy vstupního a výstupního souboru.

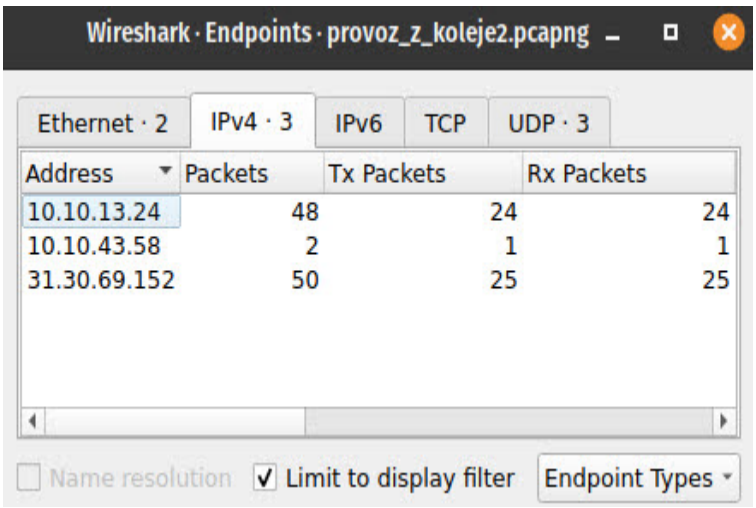
```
--pcap provoz_z_koleje2.pcapng
--outputFile provoz_z_koleje2.csv
--network 0.0.0.0/0
--filterNames IPSEC,WIREGUARD,OPENVPN,OPENVPN_MIKROTIK
```

6.5.4 Výsledky analýzy druhého souboru z koleje ZČU

V druhém souboru byly při ruční analýze identifikovány stejné VPN, jako v předchozím případě – IPSec a WireGuard.

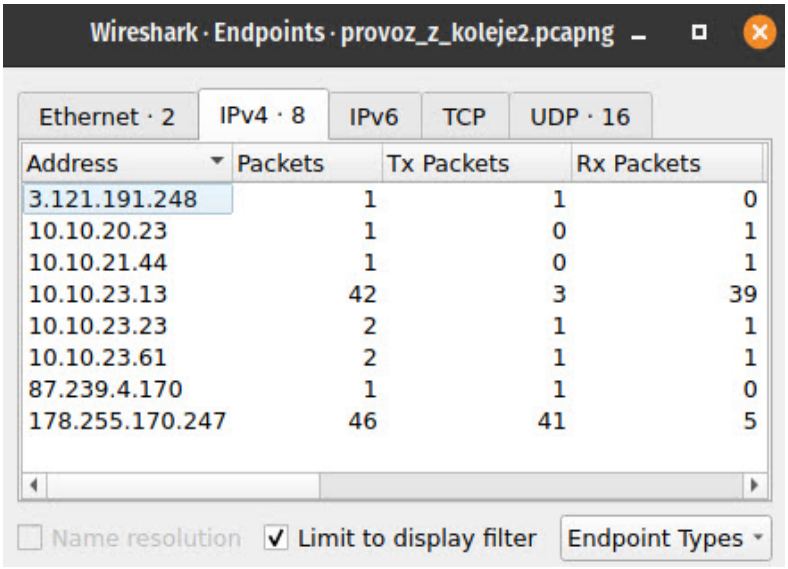
Zde by analyzátor měl detekovat jenom WireGuard provoz, jelikož všechny zjištěné komunikace prostřednictvím IPSec VPN patřily pouze navazování spojení (a tedy opět nesplňovaly předpoklad implementovaného analyzátoru). Žádná datová část IPSec komunikace nebyla v tomto souboru detekována.

Statistiky z kontrolní části jsou zobrazeny na obrázku 6.15.

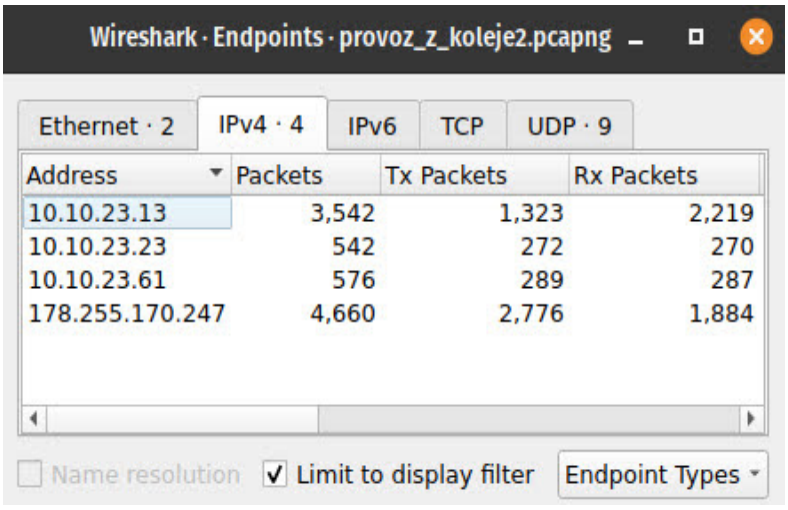


Obrázek 6.15: Analýza druhého provozu z koleje ZČU ve WireSharku — IPSec, kontrolní část.

WireGuard statistiky ukazují obrázky 6.16 a 6.17. Lze z nich vidět, že validní komunikaci, kterou by měl analyzátor identifikovat, zprostředkovala pouze čtyři zařízení, která mají záznam jak v kontrolní části komunikace, tak i v datovém přenosu. Jedná se o IP adresy z obrázku 6.17.



Obrázek 6.16: Analýza druhého provozu z koleje ZČU ve WireSharku — WireGuard, kontrolní část.



Obrázek 6.17: Analýza druhého provozu z koleje ZČU ve WireSharku — WireGuard, datová část.

VPN	ADDRESS_A	ADDRESS_B	PACKETS_A	PACKETS_B
WIREGUARD_DATA	10.10.23.61:53701	178.255.170.247:50069	289	287
WIREGUARD_DATA	10.10.23.23:56930	178.255.170.247:50069	272	270
WIREGUARD_DATA	10.10.23.13:55517	178.255.170.247:51820	9	5
WIREGUARD_DATA	10.10.23.13:51573	178.255.170.247:50001	433	426
WIREGUARD_DATA	10.10.23.13:35238	178.255.170.247:50002	881	1788
WIREGUARD_CONTROL	10.10.23.13:55517	178.255.170.247:51820	1	1
WIREGUARD_CONTROL	10.10.23.13:51573	178.255.170.247:50001	1	13
WIREGUARD_CONTROL	10.10.23.23:56930	178.255.170.247:50069	1	1
WIREGUARD_CONTROL	10.10.23.13:35238	178.255.170.247:50002	1	5
WIREGUARD_CONTROL	10.10.23.61:53701	178.255.170.247:50069	1	1

Tabulka 6.6: Výsledek analýzy druhého provozu z koleje ZČU – výstup z analyzátoru.

Výsledek automatické analýzy znázorňuje tabulka 6.6. Z této tabulky lze vidět, že zařízení s IP adresou 10.10.23.13 navazovalo WireGuard spojení vícekrát (pokaždé s jiným dynamickým portem) a pokaždé přes vytvořené spojení přenášelo data.

Tyto informace se ve statistikách WireSharku ani v ruční analýze nenacházely. Implementované řešení tedy tento program překonává.

7 Možnosti rozšíření

Po dokončení implementace výsledného analyzátoru se ukázalo několik možných cest, které by vedly k rozšíření funkčnosti programu.

Jednou z možností je umožnit automatickou analýzu nejenom ze souboru, ale přímo ze síťového rozhraní daného zařízení, na kterém by tento analyzátor byl nasazen. Vzhledem k počátečnímu předpokladu (analyzovat zachycený síťový provoz z prostředí kolejní sítě Západočeské univerzity) se nabízí umístění tohoto programu na server, na jehož rozhraní je zrcadlen provoz v dané síti.

K realizaci tohoto rozšíření se však musí zlepšit výkonnost a efektivita analyzátoru, jelikož byl již z počátku navržen na zpracování pouze zachycené komunikace ze vstupního souboru, nikoliv ze síťového rozhraní.

Další možností je rozšířit nabídku detekcí o další VPN služby. Vzhledem k modulárnosti aktuálně existujícího programu by toto rozšíření mělo být implementačně jednodušší, jelikož bude stačit postupovat podobným způsobem, který byl popsán v této práci (zjistit unikátní charakteristiky dané VPN, které by se pak začlenily do samostatného modulu pro analýzu. Ten následně stačí v kódu zahrnout do průběhu analýzy).

Také se nabízí možné propojení aktuálního řešení s databází a webovým rozhraním, ve kterém by se přehledně zobrazily statistiky detekce.

Výstupem analyzátoru by tedy již nebyl pouze *CSV* soubor, ale zjištěné výsledky by se ukládaly do databáze. Ta by byla propojena s webovou stránkou, která by obsah databáze přehledně vizualizovala administrátorům sítě.

8 Závěr

V prvních kapitolách této práce byl nejprve popsán obecný koncept VPN sítí a byly přiblíženy základní principy jejich fungování a využití v praxi. Poté následovalo rozdělení a popis jednotlivých VPN topologií, vysvětlení termínu VPN koncentrátor a byly představeny dva režimy, ve kterých jsou tyto virtuální privátní sítě schopné fungovat. Také byla popsána funkčnost VPN služeb z pohledu operačních systémů. Následně se podrobněji popsalo fungování tří konkrétních VPN protokolů, včetně navazování spojení a struktury jednotlivých paketů.

V závěru teoretické části se tato práce zaměřila na již existující řešení detekce VPN služeb.

Na základě poznatků a nabytých znalostí z teoretické (poznávací) části práce se podařilo v praktické části vyvinout prototyp analyzátoru, který dokáže zpracovat soubor se zachycenou síťovou komunikací a provést její základní filtraci a analýzu.

Součástí praktické části bylo také navrhnout a vytvořit celkem tři testovací scénáře, ve kterých se následně zachytila síťová data, která byla podrobena ruční analýze programem WireShark. Výsledky z této ruční analýzy sloužily pro porovnání přesnosti výsledného programu. Také se podařilo zachytit provoz z kolejní sítě Západočeské univerzity. Vznikly tak navíc dva testovací soubory s reálnou síťovou komunikací, na kterých byl analyzátor testován především na rychlost, přesnost a efektivitu analýzy.

Výsledný program byl v průběhu implementace nejprve testován na zmíněných síťových tocích z testovacích scénářů. Při tomto testování byly odhaleny nedostatky tehdejšího řešení (například zvýšený výskyt falešně pozitivních výsledků u OpenVPN komunikace), které se v další části implementace povedlo úspěšně eliminovat.

Na závěr proběhlo otestování na reálném provozu z kolejní sítě, ve kterém analyzátor dokázal rozpoznat i takové komunikace, které v průběhu ruční analýzy nebyly odhaleny.

Seznam zkratek

- **ACK** – Acknowledgment;
- **ADVPN** – Auto Discovery Virtual Private Network;
- **AES** – Advanced Encryption Standard;
- **AH** – Authentication Header;
- **ARP** – Address Resolution Protocol;
- **ASIC** – Application-Specific Integrated Circuit;
- **CBC** – Cipher Block Chaining;
- **CLDAP** – Connection-less Lightweight Directory Access Protocol;
- **CPU** – Central Processing Unit;
- **CRC** – Cyclic Redundancy Check;
- **ČVUT** – České vysoké učení technické;
- **DES** – Data Encryption Standard;
- **DHCP** – Dynamic Host Configuration Protocol;
- **DH** – Diffie-Hellman;
- **DMVPN** – Dynamic Multipoint Virtual Private Network;
- **DNS** – Domain Name System;
- **ECDH** – Elliptic-Curve Diffie-Hellman;
- **ESP** – Encapsulation Security Payload;
- **GNS3** – Graphical Network Simulator-3;
- **GNU GPL** – GNU General Public License;
- **GRE** – Generic Routing Encapsulation;
- **HMAC** – Hash-Based Message Authentication Code;
- **ICMP** – Internet Control Message Protocol;

- **IKE** – Internet Key Exchange;
- **IPSec** – Internet Protocol Security;
- **IP** – Internet Protocol;
- **ISAKMP** – Internet Security Association and Key Management Protocol;
- **ISO/OSI** – International organization of Standardization/Open Systems Interconnection;
- **IV** – Initialization Vector;
- **LAN** – Local Area Network;
- **LDAP** – Lightweight Directory Access Protocol;
- **LLC** – Logical Link Control;
- **LLDPDU** – Link Layer Discovery Protocol;
- **MAC** – Message Authentication Code;
- **MPLS** – Multiprotocol Label Switching;
- **MTU** – Maximum Transmission Unit;
- **NAT-D** – Network Address Translation-Discovery;
- **NAT-T** – Network Address Translation-Traversal;
- **NAT** – Network Address Translation;
- **OS** – Operating System;
- **PDU** – Protocol Data Unit;
- **PFS** – Perfect Forward Secrecy;
- **RAM** – Random Access Memory;
- **RFC** – Request For Comments;
- **SA** – Security Associations;
- **SHA** – Secure Hash Algorithms;
- **SKEME** – Secure Key Exchange Mechanism;

- **SNMP** – Simple Network Monitor Protocol;
- **SPI** – Security Parameter Index;
- **SSD** – Solid State Drive;
- **SSL** – Secure Sockets Layer;
- **TCP** – Transmission Control Protocol;
- **TLS** – Transport Layer Security;
- **TTL** – Time To Live;
- **UDP** – User Datagram Protocol;
- **VLAN** – Virtual Local Area Network;
- **VPN** – Virtual Private Network;
- **WAN** – Wide Area Network;
- **WG** – WireGuard;
- **ZČU** – Západočeská univerzita.

Seznam tabulek

Seznam obrázků

Literatura

[1] How to set up VPN tunnels between registered vigor routers by SD-Wan on ACS 3?, May 2021. Dostupné z: https://www.draytek.com/support/knowledge-base/7499.

[2] How do I configure site-to-site IPSec VPN?, Aug 2021. Dostupné z: https://support.huawei.com/enterprise/it/doc/EDOC1000079719/95a8cbf3/how-do-i-configure-site-to-site-ipsec-vpn#fig_dc_ar_faq_ipsec_001001.

[3] Abdullah, Z. N. – Ahmad, I. – Hussain, I. Segment routing in software defined networks: A survey. *IEEE Communications Surveys & Tutorials*. 2018, 21, 1, s. 464–486.

[4] Aboba, B. – Dixon, W. IPsec-Network Address Translation (NAT) Compatibility Requirements", RFC 3715. Technical report, RFC Editor, 2004.

[5] Azad, T. B. Chapter 8 - Security Guidance for ICA and Network Connections. In Azad, T. B. (Ed.) *Securing Citrix Presentation Server in the Enterprise*. Burlington: Syngress, 2008. s. 557–618. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9781597492812000081. ISBN 978-1-59749-281-2.

[6] Banks, E. *4 Common VPN Types (And When to Use Them)* [online]. auvik.com, 2015. [cit. 2021/11/11]. Dostupné z: https://www.auvik.com/franklyit/blog/vpn-types/.

[7] Bollapragada, V. – Khalid, M. – Wainner, S. *IPSec VPN Design*. Cisco Press, 2005.

[8] Brown, J. *Which VPN Topology is Also Known as a Hub-and-Spoke Configuration?* [online]. vpnprofy.com, 2021. [cit. 2021/11/11]. Dostupné z: https://vpnprofy.com/which-vpn-topology-is-also-known-as-a-hub-and-spoke-configuration/.

[9] Cimpanu, C. *A decade of hacking: The most notable cyber-security events of the 2010s* [online]. zdnet.com, 2019. [cit. 2021/23/10]. Dostupné z: https://www.zdnet.com/article/a-decade-of-hacking-the-most-notable-cyber-security-events-of-the-2010s/.

[10] Constantin, L. *Mesh VPNs explained: Another step toward zero-trust networking* [online]. www.csoonline.com, 2020. [cit. 2021/11/12].

Dostupné z: https://www.csoonline.com/article/3575088/mesh-vpns-explained-another-step-toward-zero-trust-networking.html.

[11] Constantin, L. *What is WireGuard? Secure, simple VPN now part of Linux* [online]. www.csoonline.com, 2020. [cit. 2022/03/04]. Dostupné z: https://www.csoonline.com/article/3434788/what-is-wireguard-secure-simple-vpn-still-in-development.html.

[12] Crist, E. F. – Keijser, J. J. *Mastering OpenVPN*. Packt Publishing Ltd, 2015.

[13] Dhall, H. et al. Implementation of IPSec protocol. In *2012 Second International Conference on Advanced Computing & Communication Technologies*, s. 176–181. IEEE, 2012.

[14] Donenfeld, J. A. Wireguard: next generation kernel network tunnel. In *NDSS*, s. 1–12, 2017.

[15] Doozer. *Understanding TUN TAP Interfaces* [online]. naturalborncoder.com, 2014. [cit. 2021/12/02]. Dostupné z: https://www.naturalborncoder.com/virtualization/2014/10/17/understanding-tun-tap-interfaces/.

[16] Elkeelany, O. et al. Performance analysis of IPSec protocol: encryption and authentication. In *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, 2, s. 1164–1168 vol.2, 2002. doi: 10.1109/ICC.2002.997033.

[17] Feilner, M. *OpenVPN: Building and integrating virtual private networks*. Packt Publishing Ltd, 2006.

[18] Ferguson, P. – Huston, G. What is a VPN?, 1998.

[19] Frankel, S. – Krishnan, S. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, RFC Editor, February 2011. Dostupné z: http://www.rfc-editor.org/rfc/rfc6071.txt. http://www.rfc-editor.org/rfc/rfc6071.txt.

[20] Goralski, W. Chapter 33 - IP Security. In Goralski, W. (Ed.) *The Illustrated Network (Second Edition)*. Boston: Morgan Kaufmann, second edition edition, 2017. s. 813–830. doi: https://doi.org/10.1016/B978-0-12-811027-0.00033-3. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9780128110270000333. ISBN 978-0-12-811027-0.

[21] Grasdal, M. et al. Chapter 10 - MCSE 70-293: Planning, Implementing, and Maintaining Internet Protocol Security. In Grasdal, M. et al. (Ed.) *MCSE (Exam 70-293) Study Guide*. Rockland: Syngress, 2003. s. 709–779. doi: https://doi.org/10.1016/B978-193183693-7/50014-2. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9781931836937500142. ISBN 978-1-931836-93-7.

[22] Haga, S. et al. 5G network slice isolation with WireGuard and open source MANO: a VPNaaS Proof-of-Concept. In *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, s. 181–187. IEEE, 2020.

[23] Heinzman, A. *Why do some websites block VPNs?* [online]. www.howtogeek.com, 2019. [cit. 2021/24/10]. Dostupné z: https://www.howtogeek.com/403771/why-do-some-websites-block-vpns/.

[24] J. Peter Bruzzese, T. D. *VPN Concentrators: A Must for Small Business* [online]. redmondmag.com, 2008. [cit. 2021/11/14]. Dostupné z: https://redmondmag.com/articles/2008/05/01/vpn-concentrators-a-must-for-small-business.aspx.

[25] Jean-Baptiste Bédrune, G. C. J. B. OpenVPN 2.4.0 Security Assessment. Technical report, Quarkslab SAS, 2017.

[26] Jim Guichard, I. P. *MPLS and VPN Architectures, CCIP™ Edition*. Cisco Press, 2002. ISBN 1-58705-081-1.

[27] Jones, J. *The 4 Main Types of VPN* [online]. www.top10vpn.com, 2021. [cit. 2021/11/11]. Dostupné z: https://www.top10vpn.com/what-is-a-vpn/vpn-types/.

[28] Kent, S. – Seo, K. Security Architecture for the Internet Protocol. RFC 4301, RFC Editor, December 2005. Dostupné z: http://www.rfc-editor.org/rfc/rfc4301.txt. http://www.rfc-editor.org/rfc/rfc2408.txt.

[29] Law, L. – Solinas, J. Suite B Cryptographic Suites for IPsec. RFC 6379, RFC Editor, October 2011. Dostupné z: http://www.rfc-editor.org/rfc/rfc6379.txt. http://www.rfc-editor.org/rfc/rfc6379.txt.

[30] Li, H. *TUN/TAP Interface* [online]. hechao.li, 2018. [cit. 2021/12/02]. Dostupné z: https://hechao.li/2018/05/21/Tun-Tap-Interface/.

[31] Likhar, P. – Yadav, R. S. – others. Securing IEEE 802.11 g WLAN using OpenVPN and its impact analysis. *arXiv preprint arXiv:1201.0428*. 2012.

[32] Lin, Y.-N. et al. VPN gateways over network processors: implementation and evaluation. In *11th IEEE Real Time and Embedded Technology and Applications Symposium*, s. 480–486, 2005. doi: 10.1109/RTAS.2005.58.

[33] Liu, D. et al. *Firewall policies and VPN configurations*. Elsevier, 2006.

[34] Mason, A. IPSec Overview Part Four: Internet Key Exchange (IKE). *Ciscopress. com*. 2002.

[35] Maughan, D. – Schneider, M. – Schertler, M. Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408, RFC Editor, November 1998. Dostupné z: `http://www.rfc-editor.org/rfc/rfc2408.txt`. `http://www.rfc-editor.org/rfc/rfc2408.txt`.

[36] Milanovic, S. – Mastorakis, N. E. Architecting the Next Generation End-to-End e-Business Trust Infrastructure. *WSEAS Transactions on Communications*. 2002, , 1, s. 1–8.

[37] Mocan, T. *VPN history* [online]. cactusvpn.com, 2018. [cit. 2021/24/10]. Dostupné z: `https://www.cactusvpn.com/beginners-guide-to-vpn/vpn-history/`.

[38] Molenaar, R. *IPsec (Internet Protocol Security)* [online]. NetworkLessons.com, 2016. [cit. 2021/12/09]. Dostupné z: `https://networklessons.com/cisco/ccie-routing-switching/ipsec-internet-protocol-security/`.

[39] Moon, L. *What is a VPN Concentrator?* [online]. www.netgeekpro.com, 2021. [cit. 2021/11/14]. Dostupné z: `https://www.netgeekpro.com/vpn-concentrator/`.

[40] Niedermeier, T. *WireGuard Basics* [online]. www.thomas-krenn.com, 2019. [cit. 2022/03/05]. Dostupné z: `https://www.thomas-krenn.com/en/wiki/WireGuard_Basics`.

[41] Novickis, T. – Poll, E. – Altan, K. *Protocol state fuzzing of an OpenVPN*. PhD thesis, PhD thesis. MS thesis, Fac. Sci. Master Kerckhoffs Comput. Secur., Radboud Univ, 2016.

[42] Orman, H. RFC2412: The OAKLEY Key Determination Protocol, 1998.

[43] Pavlík, R. *VPN s OpenVPN* [online]. roman-pavlik.cz, 2013. [cit. 2021/12/02]. Dostupné z: `https://roman-pavlik.cz/vpn-s-openvpn-uvod`.

[44] Perlman, R. – Kaufman, C. Key exchange in IPSec: Analysis of IKE. *IEEE Internet Computing*. 2000, 4, 6, s. 50–56.

[45] Phillips, G. *The 6 Major VPN Protocols Explained* [online]. makeuseof.com, 2020. [cit. 2021/12/02]. Dostupné z: `https://www.makeuseof.com/tag/major-vpn-protocols-explained/`.

[46] Pollards, E. *Proxy vs. VPN* [online]. vpnclientapp.com, 2019. [cit. 2021/12/09]. Dostupné z: `https://vpnclientapp.com/blog/proxy-vs-vpn/`.

[47] Rahul Awati, T. P. *dynamic multipoint VPN (DMVPN)* [online]. www.techtarget.com, 2021. [cit. 2022/12/01]. Dostupné z: `https://www.techtarget.com/searchnetworking/definition/dynamic-multipoint-VPN-DMVPN`.

[48] Reynolds, J. – Postel, J. Assigned numbers. Technical report, STD 2, RFC 1700, October, 1994.

[49] Rosu, S. M. et al. The Virtual Enterprise Network based on IPSec VPN Solutions and Management. *International Journal of Advanced Computer Science and Applications*. 2012, 3, 11.

[50] *IPsec Tunnel Mode vs. Transport Mode* [online]. twingate.com, 2021. [cit. 2021/11/27]. Dostupné z: `https://www.twingate.com/blog/ipsec-tunnel-mode/`.

[51] Walsh, R. *What is TUN/TAP and Why do VPNs Use Them?* [online]. proprivacy.com, 2021. [cit. 2021/12/02]. Dostupné z: `https://proprivacy.com/guides/tun-tap`.

[52] Wang, C. – Chen, J.-y. Implementation of GRE over IPsec VPN enterprise network based on cisco packet tracer. In *2nd International Conference on Soft Computing in Information Communication Technology*, s. 142–146. Atlantis Press, 2014.

[53] Wu, P. Analysis of the WireGuard protocol. *Master's Thesis, Analysis of the WireGuard protocol, Eindhoven University of Technology*. 2019.

[54] Čtrnáctý, B. M. Softwarový modul pro rozpoznání VPN v síťovém provozu. Diplomová práce, České vysoké učení technické, Praha, 2019.

A Uživatelská příručka

V této příloze budou popsány parametry výsledné aplikace a vstupní data.

A.1 Parametry programu

Analyzátor lze spustit s pěti parametry, kde kromě názvu výstupního souboru a volby knihovny jsou zbylé tři argumenty povinné:

- **pcap** – název vstupního souboru (viz sekce A.2);
- **outputFile** – název výstupního souboru (nepovinný parametr);
- **library** – název použité knihovny (nepovinný parametr, pokud není uveden, automaticky se použije knihovna dpkt);
- **network** – IPv4 síť, v jejímž rozsahu se budou hledat adresy komunikující pomocí VPN;
- **filterNames** – názvy VPN filtrů.

A.2 Vstupní data

Jak již bylo zmíněno v úvodu, vstupní data analyzátoru tvoří vždy část zachycené síťové komunikace. Program si načte do paměti soubor, který je předán jako parametr při spuštění, a následně jej zpracovává. Při puštění analyzátoru bez parametru `--library` musí být soubor ve formátu `pcapng`. `Pcap` formát lze snadno převést na `pcapng`, například pomocí programu WireShark, který umožňuje zobrazovat a filtrovat zachycené pakety (viz sekce 3.1). Zároveň umožňuje otevřít oba typy souborů. `Pcap` tedy stačí otevřít ve WireSharku a pak jej uložit jako `pcapng`.

Pokud uživatel chce analýzu pustit s knihovnou Scapy (tedy uvedl daný parametr při spuštění), pak mu tato knihovna umožní zpracovat i `pcap` formát, avšak celkový běh programu bude trvat déle.

B Adresářová struktura elektronické přílohy

Struktura elektronické přílohy práce je následující:

- **Adresář „Text_prace“** – obsahuje všechny zdrojové soubory: *main.tex, literatura.bib, thesiskiv.cls, csplainatkiv.bst*, výsledný PDF soubor, a také podadresář „images“, který obsahuje obrázky použité v textu práce.
- **Adresář „Aplikace_a_knihovny“** – obsahuje zdrojové kódy a soubor *requirements.txt*, který obsahuje potřebné knihovny pro spuštění aplikace.
- **Adresář „Vstupni_data“** – obsahuje čtyři testovací soubory se zachycenou síťovou komunikací z testovacích scénářů. Vzhledem k objemu dat z reálného provozu jsou zbývající dva testovací soubory z kolejní sítě dostupné u vedoucího této práce.
- **Adresář „Vysledky“** – obsahuje výstupní *CSV* soubory z aplikace.
- **Soubor Readme.txt** – soubor s popisem adresářové struktury.