VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

BEZPEČNOSTNÍ TESTOVÁNÍ PROTOKOLŮ RODINY IPV6 A SOUVISEJÍCÍCH ZRANITELNOSTÍ

SECURITY TESTING OF IPV6 FAMILY PROTOCOLS AND RELATED VULNERABILITIES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Matěj Vopálka

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Jan Jeřábek, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Matěj Vopálka ***ID:*** 220834

Ročník: 2 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Bezpečnostní testování protokolů rodiny IPv6 a souvisejících zranitelností

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte z literatury problematiku fungování protokolů IPv6, ICMPv6, NAT64, DNS64 a souvisejících protokolů. Zaměřte se i na nástroje na testování fungování těchto protokolů a případné odlišnosti jednotlivých platforem. Nastudujte aktuálně známé bezpečnostní problémy těchto protokolů a problematiku nástrojů na testování slabin a zranitelností. V rámci diplomové práce vytvořte seznam testovacích případů pro bezpečnostní testování vybraných služeb a protokolů. V praktické části se soustřeďte zejména na popis testu, způsoby jak ho provést, možné dopady zranitelnosti na daný systém a jejich závažnost, návaznost na případný seznam zranitelností, které lze tímto testem odhalit. Součástí práce bude vytvoření nástroje na automatické testování vybraných případů v programovacím jazyce Python a také redesign či optimalizace některých již existujících nástrojů. Vytvořené dílo musí být kompatibilní s nástrojem penterep. Kompletní specifikace musí být předem konzultována a schválena vedoucím práce. Výsledný nástroj důkladně otestujte a zdokumentujte jeho vlastnosti, stejně tak jako zdrojové kódy. Vytvořte ukázky možného použití. V rámci semestrálního projektu zpracujte teoretickou část, vytvořte seznam testovacích případů se základním popisem a připravte testovací prostředí.

DOPORUČENÁ LITERATURA:

[1] Kurose, J. F., Ross, K. W., Computer networking: a top-down approach. 8th global ed. Essex: Pearson, 2022, 852 s. ISBN 978-1-292-15359-9.

[2] JEŘÁBEK, J. Pokročilé komunikační techniky. Skriptum FEKT Vysoké učení technické v Brně, 2023. s. 1-179.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 21.5.2024

Vedoucí práce: doc. Ing. Jan Jeřábek, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato práce se zabývá problematikou Internetového Protokolu verze 6 (IPv6), zejména pak v oblasti jeho bezpečného nasazení. Jsou zde zmíněny nedostatky protokolu IPv4 a důvody vzniku protokolu IPv6. Dále je zde probrána problematika adresace, struktury rámce IPv6 a jsou uvedeny základní typy rozšiřujících záhlaví IPv6. Práce se také zabývá souvisejícími protokoly s IPv6 jako NDP, SLAAC a DHCPv6. Obsahuje i úvod do penetračního testování, popis základních typů hackerů a obecný popis útoků v oblasti informační bezpečnosti. Praktická část se věnuje vývoji aplikace pro automatické testování zranitelností IPv6 sítí Penvuhu6. Nástroj je vyvinut v programovacím jazyce Python za pomocí knihovny Scapy. Penvuhu6 byl testován v emulovaném síťovém prostředí programem GNS3. Pro nástroj byly vyvinuty tři testové scénáře zaměřující se na testování průchodu repetetivních a špatně seřazených záhlaví, překrývajících se fragmentů a Router advertisement a DHCPv6 advertisement zpráv. Penvuhu6 bylo testováno na emulovaném zařízení RouterOS se základní a restriktivní konfigurací.

KLÍČOVÁ SLOVA

IPv6, IP, informační bezpečnost, penetrační testování, python, kali linux, NDP, ICMPv6, DHCPv6, rozšiřující záhlaví, Mikrotik, RouterOS, RouterBOARD, Scapy

ABSTRACT

This thesis discusses the Internet Protocol version 6 (IPv6), especially the secure deployment of the protocol. The thesis deals with the shortcomings of IPv4 protocol and reason of development of IPv6 protocol. It covers topics like IPv6 addressing, structure of frames, the initial types of IPv6 extension headers. Additionally, the thesis explores related protocols to IPv6, such as NDP, SLAAC, adn DHCPv6. The thesis provides an introduction to penetration testing, describes the basic types of hackers and gives a general overview of information security attacks. The practical part is devoted to the development of an application for automatic vulnerability testing of IPv6 networks Penvuhu6. The tool is developed in Python programming language using Scapy library. Penvuhu6 has been tested in an emulated network environment with the GNS3 program. Three test scenarios were developed for the tool focusing on testing the passage of repetitive and misaligned headers, overlapping fragments, and Router advertisement and DHCPv6 advertisement messages. Penvuhu6 was tested on an emulated RouterOS device with basic and restrictive configurations.

KEYWORDS

IPv6, IP, information security, penetration testing, python, kali linux, NDP, ICMPv6, DHCPv6, extension headers, overlapping fragments, Mikrotik, RouterOS, RouterBOARD, Scapy

VOPÁLKA, Matěj. *Bezpečnostní testování protokolů rodiny IPv6 a souvisejících zranitelností*. Diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: doc. Ing. Jan Jeřábek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Matěj Vopálka
VUT ID autora:	220834
Typ práce:	Diplomová práce
Akademický rok:	2023/24
Téma závěrečné práce:	Bezpečnostní testování protokolů rodiny IPv6 a souvisejících zranitelností

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Janu Jeřábkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Internetový protokol verze 4 (IPv4) provází informační sítě již od roku 1980 [1]. V době vzniku IPv4 vypadaly informační sítě podstatně jinak, než je tomu v dnešní době. Připojení k datové veřejné síti nebylo v této době běžné a disponovaly jím pouze instituce jako vysoké školy, nebo vojenské síly. Hlavní důraz byl při vývoji kladen na funkčnost a vznikající protokoly tak nebyly dostatečně zabezpečené. Nepředpokládalo se totiž tak masivní nasazení jako je tomu v dnešní době. Tomu nebyl přizpůsoben, ani dnes již vyčerpaný adresní prostor, kterého byl v době návrhu dostatek, ani bezpečností funkce tohoto protokolu.

Právě z těchto důvodů byl roku 1995 představen nový síťový Internet Protocol version 6 [2]. Výhody toho protokolu jsou zejména ve zjednodušení záhlaví vynecháním méně využívaných funkcí IPv4, navýšení adresního prostoru ze třiceti dvou bitového na sto dvaceti osmi bitový a vyhrazení místa pro případné rozšíření za záhlaví IPv6 formou rozšiřujících záhlaví. I přes zmíněné výhody a vyčerpání adresního rozsahu IPv4 je snaha ze strany jeho provozovatelů protokol zachovávat z důvodu kompatibility, avšak postupně by měl být nahrazen protokolem IPv6. Počet sítí s nasazeným protokolem IPv6 stále roste, což vede k objevování nových bezpečnostních problémů tohoto protokolu. Oproti IPv6 je protokol IPv4 za dobu jeho existence dobře prozkoumán a je známo velké množství jeho bezpečnostních slabin, avšak některé tyto slabiny jsou znovu přítomné i v případě protokolu IPv6. Kromě samotného IPv6 přináší jeho implementace také nutnost provozu komplementárních protokolů jako je ICMPv6, SLAAC, DHCPv6, NDP, MLD, NAT64, nebo DNS64. Provoz těchto protokolů samozřejmě představuje další potenciální místo pro bezpečností chyby.

Tato práce si klade za cíl adresaci a popis dnes známých bezpečnostních slabin IPv6 a jednoduché ověření jejich přítomnosti v síti. Součástí práce bude vyvinutí nástroje pro snadné ověření zranitelností v síti IPv6. Vzhledem k charakteru oboru bezpečnosti je na nástroj kladen požadavek jeho rozšířitelnosti pro možnost vytvoření nových testů pro lepší detekci aktuálních zranitelností, či testování nově objevených zranitelností. Dále nástroj musí uživateli detekovanou zranitelnost představit, nasměrovat uživatele k identifikaci problému a nabídnout možné řešení mitigace detekované bezpečnostní zranitelnosti. Nástroj by měl pracovat nejlépe automaticky a to tak, aby bylo možné test spustit bez další interakce do dokončení testu.

Součástí vytvořeného nástroje budou testové scénáře adresující aktuálně známé bezpečnostní slabiny IPv6 spolu s jejich popisem, uvedení nevyhovujícího případu a možností mitigace.

1 IPv6

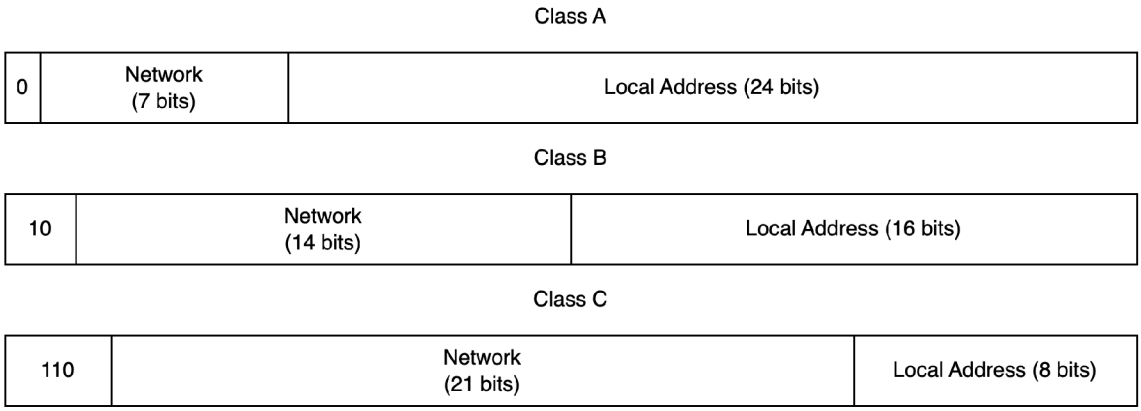
Internetový Protokol verze 6 (IPv6) je nejnovější verzí internetového protokolu umožňující identifikaci a komunikaci zařízení uvnitř sítě i mezi sítěmi. Protokol operuje na 3. vrstvě referenčního modelu ISO/OSI, neboli 2. vrstvě modelu TCP/IP a je nástupcem Internetového Protokolu verze 4 (IPv4) [3].

1.1 Důvody ke vzniku IPv6

Původní protokol IPv4 byl navržen s ohledem na dobu jeho vzniku a nepředpokládalo se tak ohromujícího masivního rozmachu osobních počítačů a informačních zařízení obecně. Z tehdejší perspektivy se adresní prostor, tvořený kombinací 32 bitů, jevil jako nevyčerpatelný. Tato představa však byla mylná a začátkem devadesátých let dvacátého století bylo zřejmé, že adresní prostor nestačí.

Adresní prostor IPv4 byl původně dělen do tří tříd. Třídy měli různé délky adresy sítě a adresy hostů a lišily se prefixem v prvním oktetu adresy [4] viz obr. 1.1. Jednalo se o následující třídy [5]:

- Třída A - měla prefix 0 a dělila adresu na prvních 8 bitů (bez prefixu 7 bitů) adresy sítě a zbylých 24 bitů pro adresu hostů. Ve třídě A tedy bylo možné vytvořit až 128 sítí (0 - 127).
- Třída B - začínala prefixem 10. Následovalo 14 bitů pro adresu sítě a 16 bitů pro hosty. Třída B tak umožňovala vytvořit 16 384 sítí.
- Třída C - měla prefix 110, 21 bitů adresy sítě a 8 bitů pro hosty. Umožňovala tak vytvořit 2 097 152 sítí.
- Třída D - zbylý prefix 111 byl původně rezervován a nemohl být použit. Občas se tomuto prefixu říkalo třída D [6].



Obr. 1.1: Třídní adresování IPv4 sítí

Takto dělený adresní prostor však nebyl šetrný k adresnímu prostoru a velikosti směrovacích tabulek [7]. Tento problém vedl k zavedení beztřídního adresování, které je šetrnější k adresním rozsahům. Beztřídní adresování zavádí síťovou masku. Příklad beztřídního adresování je uveden na obr. 1.2. Síťová maska je 32 bitová, ve které řada jedniček ukazuje na místo, po které bity v síťové adrese udávají adresu sítě a následná řada nul udává část adresy hosta [8, 9].

Beztřídní adresace	IPv4 adresa			
Adresa	10	168	10	1
	00001010	10101000	00001010	00000001
Síťová maska	255	255	255	128
	11111111	11111111	11111111	10000000
Adresa sítě	10	168	10	0
	00001010	10101000	00001010	00000000
Všesměrová adresa sítě	10	168	10	127
	00001010	10101000	00001010	01111111

Obr. 1.2: Příklad beztřídní adresace v IPv4 sítích

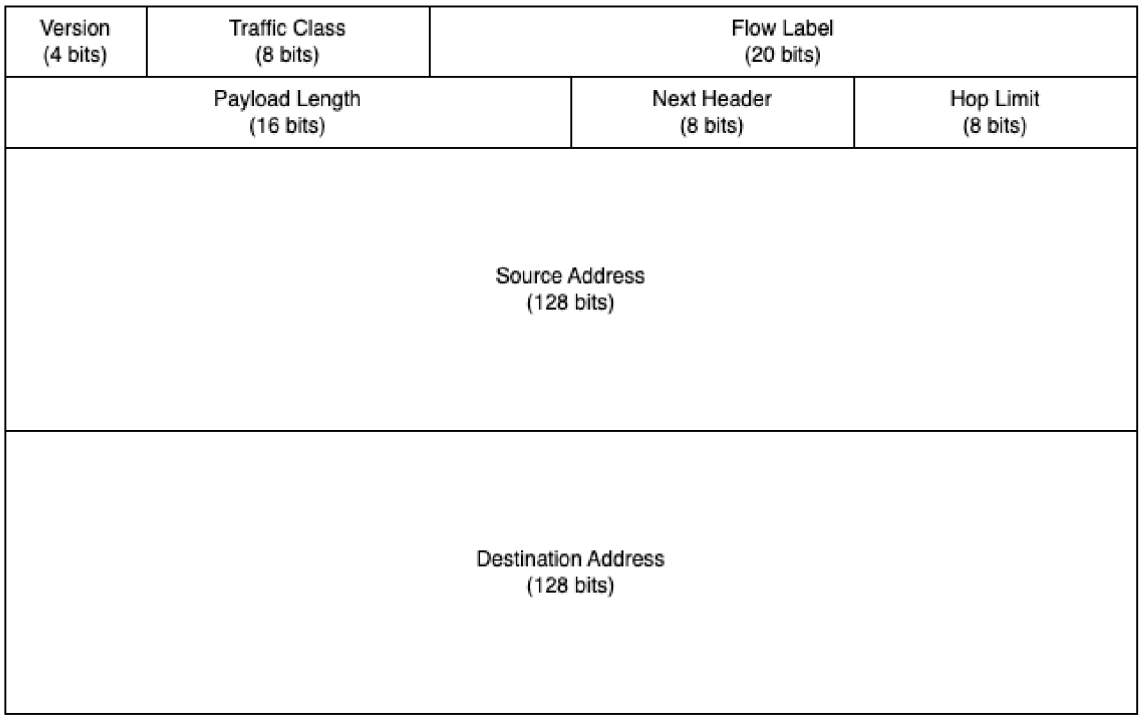
Navzdory zavedení beztřídního adresování se volný adresní prostor díky stále narůstajícímu počtu zařízení dále zmenšoval. Bylo tedy potřeba najít jiné řešení. Tím se stala kombinace privátních adres [10, 11] a Network Address Translator (NAT) [12, 13]. Privátní adresy přinesly možnost organizacím vytvářet si vlastní síť bez nutnosti registrace adresního rozsahu. Tyto rozsahy však nemohou být směrovány do mezipodnikových sítí. Mechanismus překladu adres (NAT) umožňuje měnit zdrojovou a cílovou adresu na jinou. Kombinace těchto dvou mechanismů umožnila další úsporu adres. Principiálně může mít organizace jednu, nebo malý počet veřejných adres a následně může překládat adresy, které potřebují komunikovat s veřejnou sítí. Mezi mechanismy překladu patří statický NAT, dynamický NAT, nebo překlad za pomocí portů. Toto řešení má však nevýhodu, protože spolu stanice nekomunikují přímo, tak jak původně bylo v protokolu IPv4 zamýšleno. Dále kvůli změně adresy a udržování tabulky mapující spojení narůstají nároky na síťové prvky a zvyšuje se odezva v síti. Tyto nevýhody řeší protokol IPv6, který obsahuje 128 bitů dlouhé adresy a poskytuje tedy z dnešního hlediska dostatečný adresní prostor. Kromě problému s adresním prostorem IPv6 řeší i následující nedostatky protokolu IPv4 [2]:

- Zjednodušení formátu hlavičky - oproti protokolu IPv4 se IPv6 snaží o co nejjednodušší hlavičku a ostatní funkce jsou přidávány pouze v případě potřeby.
- Vylepšená podpora rozšíření a dodatečných možností - díky zjednodušené hlavičce se v IPv6 zpracovávají převážně jen nezbytná data. Pakety, které potřebují další možnosti, jsou obdařeny dalšími záhlavími. Není tak potřeba například zasílat informace o fragmentaci při nefragmentovaném paketu a například právě fragmentace nezatěžuje směrovač, ale o rozdělení paketu se stará vysílací stanice.
- Značkování toku dat - dle hodnoty v poli flow label lze se souvisejícími toky dat fungovat efektivněji [14, 15].
- Bezpečnost - v IPv6 protokolu existují rozšíření pro autentizaci, integritu a volitelně důvěrnost přenášených zpráv.

1.2 Záhlaví IPv6

Jednou z nejdůležitějších část protokolu IPv6 je právě jeho záhlaví. Záhlaví udává význam jednotlivým přenášeným bitům a tvoří jej hlavička protokolu a přenášená data. Struktura hlavičky je vyobrazena na obrázku 1.3. Přenášená data jsou předány z vyšší vrstvy a před tyto data je vložena hlavička. Tomuto mechanismu se říká zapouzdření a při příchodu paketu k příjemci je využit opačný proces rozbalování datových jednotek. Hlavička se skládá z následujících polí:

- Verze protokolu (version) - 4 bitové pole udává, o který protokol se jedná. Pro protokol IPv6 nabývá pole hodnoty 6 v desítkové soustavě.
- Třída provozu (traffic class) - 8 bitové udává, jak by se s daným paketem mělo zacházet. Využití je například pro tzv. "realtime aplikace", které potřebují ke svému provozu co nejmenší odezvu. Samotné chování směrovačů však záleží na jejich nastavení.
- Značka toku (flow label) - hodnota označuje posloupnost dat odeslanou od zdroje k cíli, kterou chce odesílatel označit jako tok. S tímto tokem se pak může v síti zacházet jako s celkem [15]. Skutečné chování síťových prvků je však opět ovlivněné jejich konfigurací.
- Délka dat (payload length) - označuje délku užitečných dat přenášených v datové části paketu.
- Další záhlaví (next header) - pole umožňuje specifikovat další informace o paketu, které nejsou součástí základní hlavičky. Může se jednat například o fragmentaci paketu, informace o preferovaném směrování, nebo pole označuje protokol další vrstvy.
- Limit skoků (hop limit) - pole funguje obdobně jako pole TTL v IPv4 a určuje kolika uzly může paket projít, dokud nebude zahozen. Mechanismus slouží



Obr. 1.3: Záhlaví IPv6

proti nekonečnému směrování paketů.

- Zdrojová adresa (source address) - slouží jako identifikátor stanice, která paket poslala. Příjemce se díky poli dozví, na jakou adresu má odpovědět. Pole se také využívá v případě, kdy paket nelze doručit. Směrovač tak při zahození může zdroj informovat o této skutečnosti.
- Adresa cíle (destintion address) - pole slouží pro určení příjemce zprávy. Na základě tohoto pole se pak paket směruje sítí tak, aby dorazil ke správnému příjemci, nebo příjemcům

1.3 Adresace a adresní prostor

Adresace se v IPv6 opírá o podobné principy jako je tomu u adresování v IPv4. Například zde fungují stejné principy podsíťování. IPv6 na rozdíl od IPv4 předpokládá využití více adres na jednom rozhraní. Druhy komunikace u IPv6 se dělí na individuální, skupinové a výběrové. Zvláštním typem je pak nespecifikovaná adresa.

- individuální (unicast) adresa - slouží ke komunikaci mezi konkrétními rozhraní na stanicích.
- skupinová (multicast) adresa - slouží pro komunikaci mezi skupinou zařízení. Pakety zaslané na multicastovou adresu by měli obdržet všichni hosté přihlášení do dané skupiny. Do multicastu se řadí i všesměrové vysílání (broadcast),

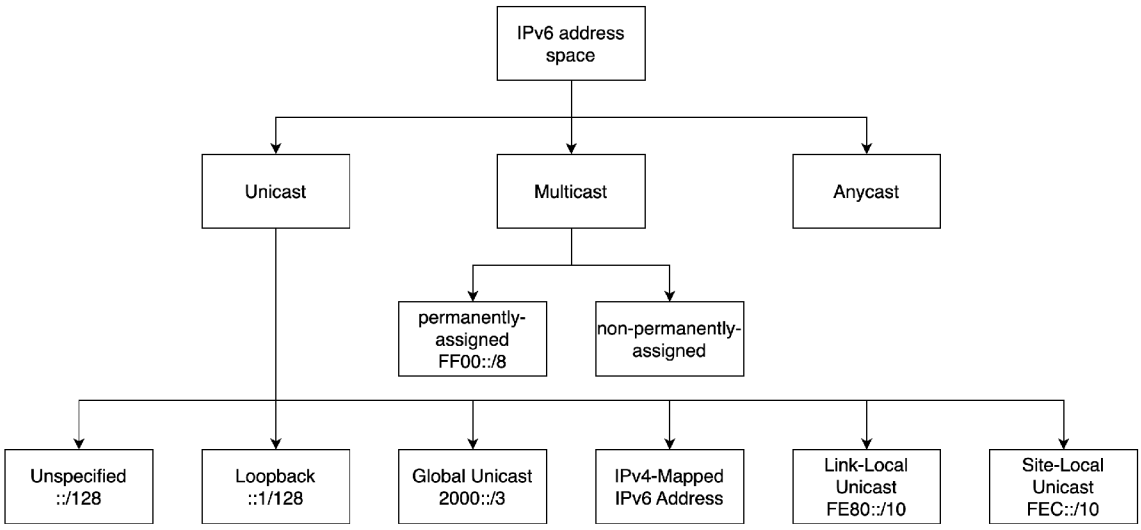
které je realizováno multicastovou adresou všech uzlů v síti.
- výběrová (anycast) adresa - funguje podobně jako skupinová, ale na rozdíl od ní obdrží paket pouze jeden příjemce ze skupiny.

1.3.1 Formát IPv6 adres

IPv6 adresy jsou oproti IPv4 adresám čtyřikrát delší. Není proto vhodné je zapisovat v desítkové či binární soustavě. Pro zápis IPv6 se využívá hexadecimální tvar. Každých 16 bitů (4 čísla hexadecimální soustavy) jsou přitom oddělené dvojtečkou [16]. Adresa se tedy skládá z 8 bloků o 16-ti bitech. V případě, kdy adresa obsahuje na začátku bloku odděleného dvojtečkami řadu nul, je možné tyto nuly vynechat a zápis tak zkrátit, avšak v případě, kdy je blok plný nul, musí zůstat minimálně jedna. Výjimkou, kdy v bloku nemusí být žádné číslo, je případ kdy se v IP adrese nachází řada bloků s nuly. Nejdelší řada nul lze zkrátit vynecháním těchto nul a celá řada se zastoupí dvěma dvojtečkami [16]. Toto zkrácení lze využít pouze jednou, protože by pak nebylo jednoznačné, kolik bloků dvě dvojtečky reprezentují a nebylo by tak možné zjistit význam adresy.

1.3.2 Skupiny adres IPv6

Jednotlivé typy adres, popsány na začátku sekce, jsou dále děleny do menších skupin dle jejich účelu viz obr. 1.4. Toto dělení probíhá na základě jejich prefixu [16].

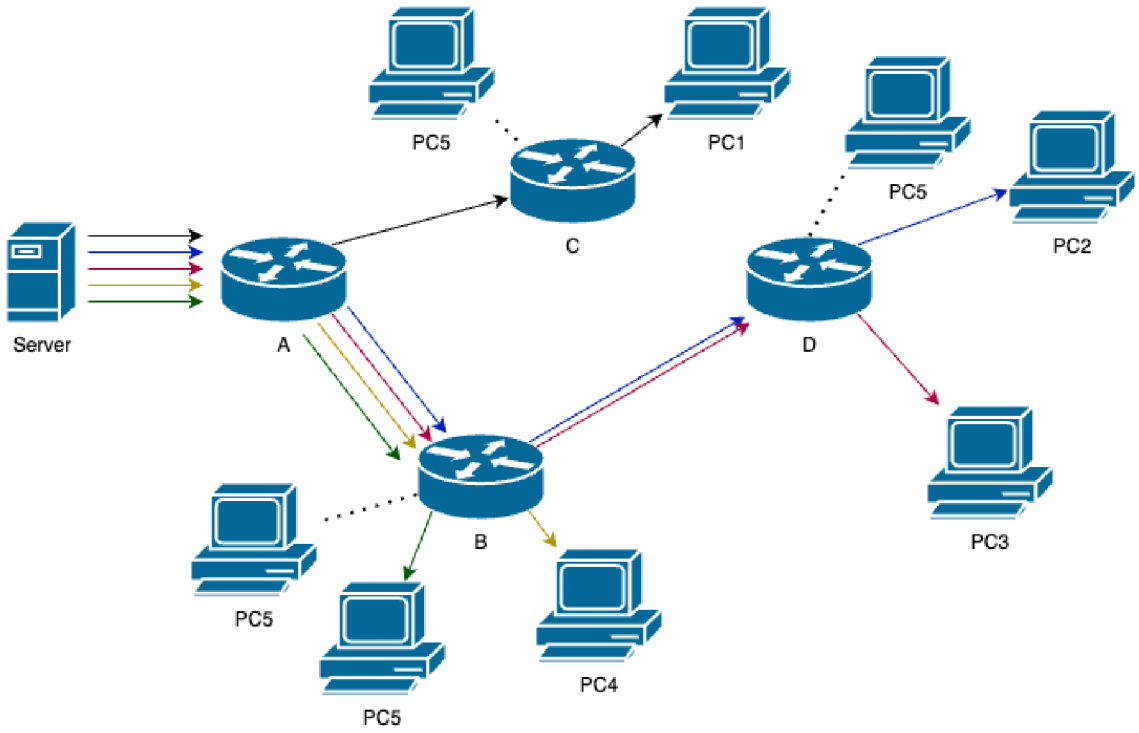


Obr. 1.4: Struktura IPv6 adresování

Unicastové adresy

Unicastové adresy se dále dělí na skupiny unspecified, loopback, IPv4 kompatibilní, link-local, site-local, global unicast.

- Nespecifikovaná (unspecified) adresa - vyjadřuje nepřítomnost IPv6 adresy na stanici. Tato adresa nesmí být přiřazena žádnému uzlu, nacházet se v poli adresy cíle, nebo v hlavičkách směrování, ani být směrována IPv6 routery. Příklad jejího použití může být v poli zdrojová adresa v paketech vygenerovaných právě se inicializujícím hostem, který ještě nemá vlastní IPv6 adresu. Nespecifikovaná adresa obsahuje na všech pozicích nuly a lze ji tedy zapsat jako "::/128" [16].
- IPv4 kompatibilní - vzhledem k tomu, že je IPv6 adresa oproti IPv4 čtyřikrát delší, je možné IPv4 adresu zapsat do IPv6 pro zpětnou kompatibilitu [16]. Existují mechanismy, které dokáží přeložit IPv6 adresu na IPv4 a zpětně IPv6 adresu na IPv4. Díky těmto mechanismům je pak umožněna komunikace mezi sítěmi založenými pouze na IPv6 a pouze na IPv4. Je však potřeba, aby tyto sítě byly propojeny těmito mechanismy fyzicky, nebo logicky pomocí síťových tunelů.
- Lokální linkové (Link-local) adresy - tyto adresy jsou obdobou rozsahu APIPA u IPv4. Jejich využití, jak název napovídá, je pouze v konkrétní síti a nesmí být směrovači směrovány do jiných sítí [16]. Adresy začínají prefixem fe80::/64, kde adresa hosta v posledních 64 bitech je vygenerována EUI-64, či jiným mechanismem. Díky automatickému vygenerování adresy je dostupná vždy a využívá se následně například pro komunikaci s DHCPv6 serverem, nebo mezi stanicemi používajících pouze přepínač.
- Lokální místní (Site-local) adresy - slouží k adresaci v rámci určité lokality. Tento typ lze však nahradit globálně unikátními adresy, protože z hlediska optimalizace routovacích tabulek místo odpovídá adrese sítě, a tak pozdější normy [16] tento typ vyřadily.
- Globální (global) adresa - adresy z této skupiny musí být globálně unikátní. Skupina je obdoba původně zamýšleného adresního prostoru IPv4 tak, aby spolu mohly komunikovat jakékoliv adresy na přímo. Adresní rozsahy jsou přidělovány organizací IANA mezi jednotlivé Regional Internet Registries (RIR). Mezi tyto registry aktuálně patří Réseaux IP Européens Network Coordination Centre (RIPE NCC), Asia-Pacific Network Coordination Centre (APNIC), American Registry for Internet Numbers (ARIN), Latin American and Caribbean Internet Addresses Registry (LACNIC) a African Network Coordination Centre (AFRINIC). Tyto RIR dále zprostředkovávají adresní rozsahy pro National Internet Registries (NIR), kteří působí již na úrovni států. NIC



Obr. 1.5: Příklad vysílání stejného obsahu pomocí unicast

následně zprostředkovávají rozsahy adres pro Internet Service Provider (ISP), kteří tyto adresy již přidělují koncovým uživatelům. Díky tomuto mechanismu je umožněno adresy rozdělovat systematicky a optimalizovat směrovací tabulky IPv6 [17].

Adresu hosta lze získat v IPv6 různými způsoby. Oproti klasické manuální konfiguraci IPv6 přináší bezstavovou konfiguraci Stateless Address Autoconfiguration (SLAAC). Routery v síti zasílají pravidelně, nebo na vyžádání oznámení směrovače Router Advertisement (RA) zprávy. Obsahem zpráv je prefix sítě a způsob konfigurace adresy. Ta může být vygenerována EUI-64, nebo podle *Dynamic Host Configuration Protocol version 6* (DHCPv6). Mechanismus EUI-64 adresu stanice za pomocí MAC adresy zařízení. DHCPv6 pak funguje obdobně jako u IPv4. Dělí se však na bezstavový (stateless) a stavový (statefull). V případě bezstavového DHCPv6 neřeší přiřazování adres, ale obstarává pouze dodatečné informace jako například DNS. V případě statefull musí server držet seznam přiřazených adres.

Každá unicastová adresa musí bez ohledu na způsob přiřazení před přiřazením (kromě definovaných výjimek [18]) použít mechanismus Duplicate Address Detection (DAD). Zařízení, které se adresu snaží vygenerovat následně zkouší, zda se již na síti daná stanice nenachází pomocí ICMPv6 Sending Neighbor Solicitation Messages. V případě, kdy žádná stanice neodpoví pomocí Neighbor Advertisement message, adresa je volná a zařízení ji přiřadí k rozhraní. V opačném případě zařízení gene-

ruje novou adresu a pokus opakuje [19]. Mechanismus umožňuje útok, kdy útočník nedovolí oběti přiřazení adresy. V praxi se tedy proces opakuje pouze několikrát.

Multicastové adresy

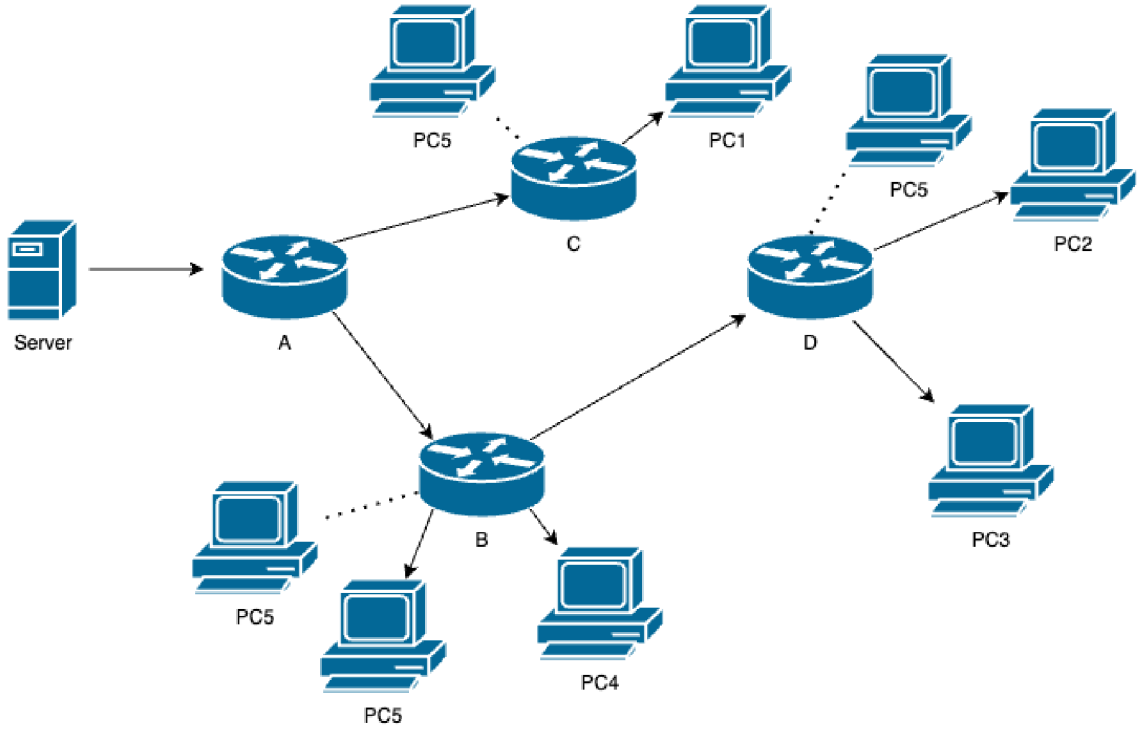
Jak již bylo zmíněno, pakety s cílovou mutlicastovou adresou jsou určeny pro všechny členy dané multicastové skupiny. Tyto adresy lze jednoduše odlišit díky prefixu FF. Unicastové vysílání je neefektivní v případě kdy se vysílá stejná zpráva různým stanicím viz Obr 1.5. Oproti unicastovému vysílání nabízí multicastové vyslání pouze jedné zprávy, která se v případě potřeby duplikuje viz obr. 1.6.

Po tomto prefixu následují 4 bity pro značky (flagy). První bit je nastaven na nulu a následují bity R, P a T. Bit T označuje zda je multicastová adresa takzvaně "dobře známá" (well-know) tím, že ji organizace Internet Assigned Numbers Authority (IANA) zařadila jako permanentně přiřazenou. V takovém případě je bit nastaven na hodnotu nula, v opačném na hodnotu jedna. V případě hodnoty jedna může být adresa přechodná, nebo dynamicky přiřazená [16]. Bit P označuje, zda je IP adresa přidělena na základě prefixu. V případě že ano, pak je bit nastaven na hodnotu jedna, v opačném případě na hodnotu nula. Pokud je bit P nastaven na hodnotu 1, musí být nastaven i bit T na hodnotu 1 [20]. Bit R označuje, zda v sobě adresa obsahuje adresu Rendezvous Point (RP). Pokud je bit nastaven na hodnotu 1, multicastová adresa obsahuje adresu RP, pokud je nastaven na 0, tak neobsahuje. Pokud je nastaven bit R na hodnotu jedna, pak musí být nastaveny i bity P a T na hodnotu 1. První bit je nastaven na hodnotu 0 a zatím není využit [21].

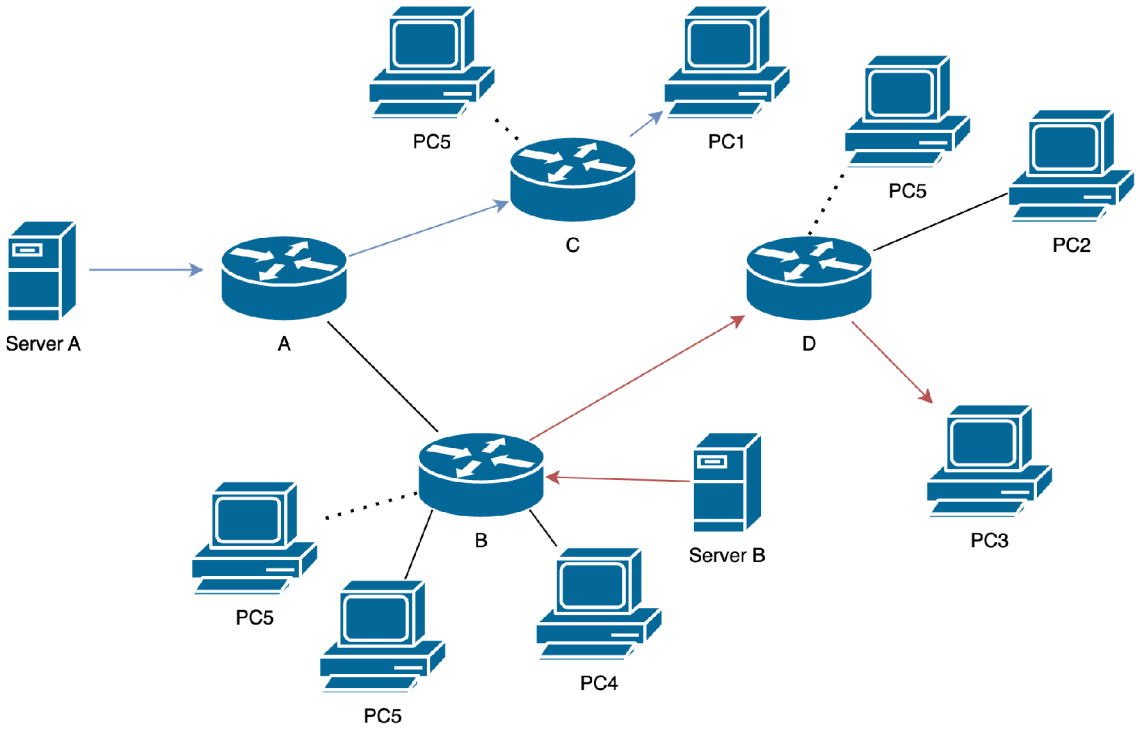
Další 4 bity označují rozsah (scope). Pole se využívá pro omezení rozsahu multicastových adres. Tyto rozsahy jsou rezervované, lokální, globální, nebo nepřiřazené. Mezi lokální pak patří například interface, link, realm, admin, site, nebo organization [16]. Po poli scope následuje již pole adresy skupiny. Toto pole identifikuje konkrétní multicastovou skupinu.

Anycastové adresy

Anycastové adresy využívají stejného rozsahu jako unicastové a nejsou tak rozeznatelné. Adresy by měly svým rozpoložením respektovat rozdělení prefixů, protože v případě, kdy se prefix neagreguje do routovací tabulky, může anycastová adresa být jako samostatný záznam ve všech směrovačích v síti internet. Tyto adresy samozřejmě existovat mohou, avšak jejich využitím by se mělo v rámci optimalizace routovacích tabulek šetřit. Cesta k cíli je vypočítána pomocí směrovacích protokolů a využije se tedy takový cíl, který je danou metrikou protokolu vyhodnocen jako nejbližší [16]. Princip anycastového vysílání je vyobrazen na obr. 1.7.



Obr. 1.6: Příklad vysílání obsahu pomocí multicast



Obr. 1.7: Příklad vysílání stejného obsahu pomocí anycast

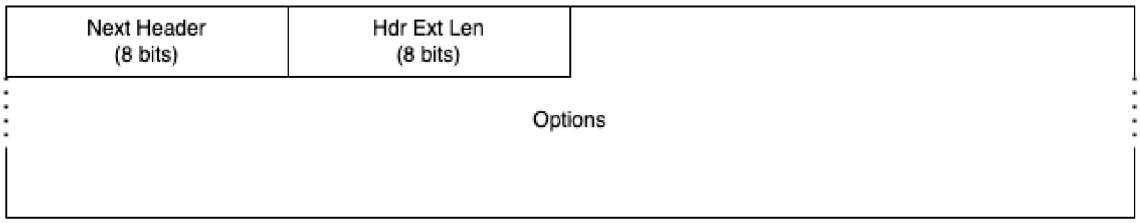
1.4 Rozšiřující záhlaví

Jak již bylo zmíněno, IPv6 přináší pevně definovanou délku záhlaví. Toto pevně dané záhlaví v sobě obsahuje nejdůležitější informace pro směrování a vynechává další rozšíření, které nabízí IPv4. Pro další funkcionalitu namísto IPv4 přináší IPv6 rozšiřující záhlaví. V základním záhlaví IPv6 je definováno pole next header, které udává právě přítomnost a typ záhlaví, které následuje za hlavičkou IPv6. Formáty rozšiřujících záhlaví se liší dle potřeby konkrétních rozšíření. Obsahují však opět pole next header, díky kterému může jeden IPv6 paket obsahovat více rozšiřujících záhlaví. Tyto záhlaví se poté skládají za sebou. O aktuální seznam rozšiřujících záhlaví se stará organizace IANA [22]. V samotném standartu IPv6 jsou definovány záhlaví Hop-by-Hop Options, Fragment, Destination Options, Routing v RFC 8200 [23] a rozšiřující záhlaví Authentication specifikované v RFC 4302 [24] a Encapsulating Security Payload specifikované v RFC 4303 [25]. Pokud je využito více rozšiřujících záhlaví, je doporučené pořadí následující. Jako první po záhlaví IPv6 by mělo následovat záhlaví Hop-by-Hop, následované záhlavími Destination Options, Routing, Fragment, Authentication, Encapsulating Security Payload, Destination Options header následované záhlavími protokolů vyšších vrstev [23]. Polem Next Header jsou definovány i další protokoly jako je ICMPv6 (hodnota 58), OSPFv3 (hodnota 89), nebo protokoly vyšších vrstev jako TCP (hodnota 6), nebo UDP (hodnota 17).

1.4.1 Hop-by-Hop Options Header

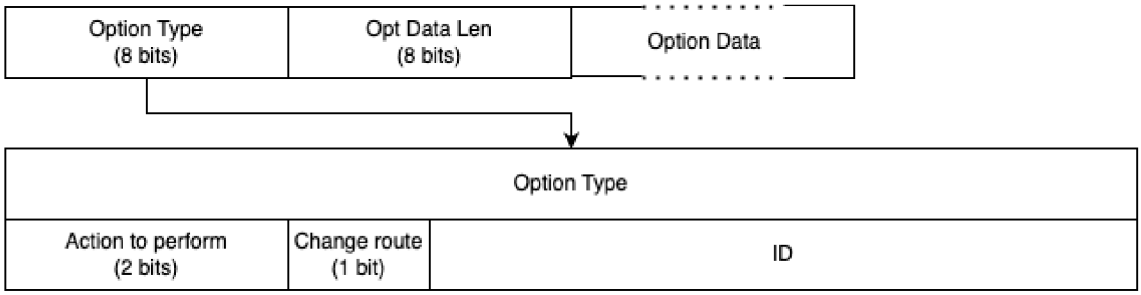
Záhlaví Hop-by-Hop, identifikované polem Next Header s hodnotou 0, je využito pro akce, které se mají vykonat při zpracovávání paketu. Záhlaví Hop-by-hop je znázorněno na obr. 1.8. Hlavička obsahuje 8 bitové pole option type, kde první 2 bity udávají, jak s paketem zacházet, pokud tento typ option není na zařízení implementován. V případě, kdy první 2 bity obsahují nuly, je obsah option hlavičky ignorován a pokračuje se ve zpracování paketu. V ostatních případech je paket zahozen. Třetí bit obsahuje informaci o tom, jestli data v option mohou změnit cestu k cílové stanici. Zbylých 5 bitů je přiřazené číslo dané option vydané organizací IANA. Po poli option type následuje délka option také dlouhé 8 bitů [23]. Pole délka udává délku těla option počtem oktetů. Tělo option už je definováno samostatně a liší se.

Například se může jednat o option Minimum Path MTU Hop-by-Hop. Tato option má zjišťovat nejmenší velikost path MTU, kterou lze poslat danou cestou. Principiálně funguje mechanismus následovně. Zdroj zašle paket s next header 0 (IPv6 Hop-by-Hop Option), kde option type je nastaven na 00110000. První dva bity označují uzlům, aby paket v případě, kdy nepodporují danou option, nezahazovali. Další bit udává, že data, které obsahuje option, mohou změnit cestu paketu. Posledních



Obr. 1.8: Hop-by-hop záhlaví

pět bitů (10000) označuje číslo option přiřazené organizací IANA. Option typ je graficky znázorněn na obr. 1.9. Tato option je 4 bajty dlouhá a obsahuje 3 další pole. Jedná se o pole Min-PMTU dlouhé 16 bitů, Rtn-PMTU dlouhé 15 bitů a bit R. Pole Min-PMTU udává nejmenší MTU, která byla po cestě detekována. Rtn-PMTU slouží pro zjištění Min-PMTU cíle. Bit R slouží pro cíl jako signalizace, zda má vyplnit pole Rtn-PMTU svým Min-PMTU [26]. Tuto hlavičku je vhodné použít například v případě, kdy směrovač pošle zdroji zprávy ICMP zprávu Packet Too Big [27] a zdroj tak musí zjistit, jak velkou zprávu může poslat, aby paket prošel úspěšně celou cestou.

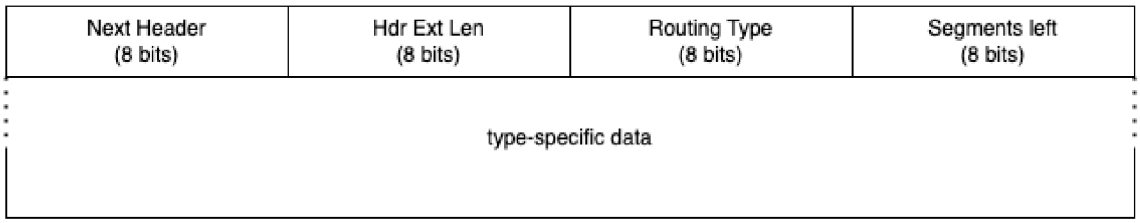


Obr. 1.9: IPv6 extension header option type

1.4.2 Routing Header

Funkcionalita záhlaví Routing Header je podobná jako funkce Loose Source and Record Route option v protokolu IPv4 [23, 4]. Záhlaví Routing Header obsahuje seznam jednoho, případně více mezilehlých uzlů, které by měly být při cestě paketu sítí navštíveny viz obr. 1.10. Rozšiřující záhlaví Routing Header je identifikováno hodnotou 43 v poli next header. Záhlaví Rourting Header je tvořeno 8 bitovým polem next header popsaným výše. Dále 8 bitovým polem Hdr Ext Len určující délku záhlaví, následovaný polem 8 bitovým polem Routing type určující typ Routing Header, příkladem může být type 2 Routing Header definovaný v RFC 6275 [28]. Dále následuje 8 bitové pole Segments Left, které udává počet segmentů, které je

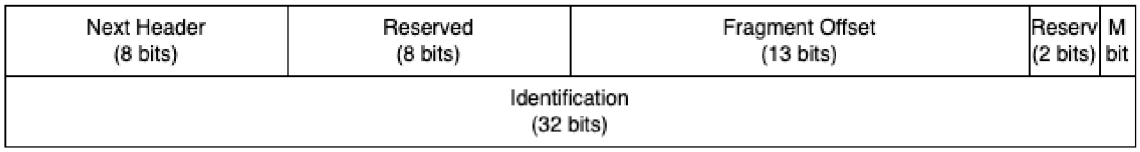
ještě potřeba navštívit. Poslední pole se nazývá type-specific data, obsahující data dle zvoleného Routing type.



Obr. 1.10: Routing záhlaví

1.4.3 Fragment Header

Rozšiřující záhlaví Fragment Header, identifikované polem Next Header s hodnotou 44, plní podobnou funkci jako v IPv4 fragmentace, která je implementovaná rovnou v záhlaví IPv4 viz obr. 1.11. U IPv6 funguje fragmentace odlišně. Fragmentovat v síti IPv4 může jakékoliv zařízení po cestě a činí tak v případě, kdy je paket příliš velký pro transport přes daný linkový protokol. Tento mechanismus však může způsobit několikanásobnou fragmentaci. Je totiž možné, že je paket fragmentován na jednom uzlu na velikost například 1400 bytů a na dalším je potřeba velikost 1300 bytů. Vzniknou tedy 3 fragmenty, i když by bylo potřeba pouze dvou. U protokolu IPv6 mezilehlé uzly nesmí zprávu fragmentovat. Místo fragmentace se paket zahodí a pomocí ICMPv6 odešle uzel zdroji zprávu Packet Too Big [23]. Pomocí této zprávy zdroj zjistí, že paket neprošel sítí, protože je příliš dlouhý. Zdroj tedy musí zjistit minimální MTU trasy, například pomocí výše zmíněné Hop-by-Hop option Minimum Path MTU záhlaví. Následně přímo zdroj fragmentuje zprávu dle minimální MTU cesty.



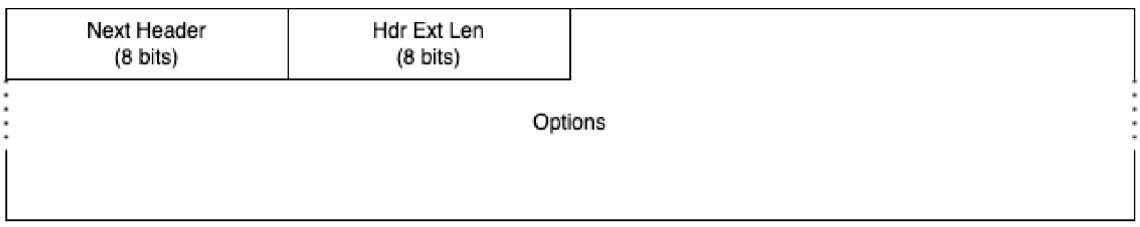
Obr. 1.11: Fragment záhlaví

Fragment Header jako první obsahuje 8 bitů dlouhé pole Next Header, jehož využití je zmíněno výše. Následuje 8 rezervovaných bitů nastavených na hodnotu 0. Dalších 13 bitů udává Fragment Offset, který slouží stejně jako v IPv4 ke zpětnému složení paketu. Pole udává kolik bytů zprávy předchází tomuto paketu. Další 2 bity jsou rezervované a nastaveny na hodnotu 0. Další bit informuje, zda je paket posledním fragmentem, nebo následují další fragmenty. Hodnota 1 informuje o dalších

fragmentech, hodnota 0 značí, že je daný fragment poslední. Posledním polem je pole Identification, tvořeným 32 bity, které identifikuje danou skupinu fragmentů. Tímto polem cílová stanice pozná, které fragmenty patří k sobě [23]. Všechny fragmenty patřící k jedné zprávě musí obsahovat stejnou hodnotu pole identifikace. Vzhledem k tomu, že pole identifikace identifikuje skupinu fragmentů, musí toto pole být unikátní pro skupinu fragmentů po předpokládanou dobu jejich existence v síti. Generace pole identifikace se věnuje například RFC 7739 [29]. Při fragmentaci není nutné redundantně přenášet všechny hlavičky paketu, ale pouze ty které jsou důležité pro každý fragment zvlášť. Hlavičky které následují po hlavičce Fragment Header, jako jsou Destination Options header, Encapsulating Security Payload header, nebo hlavičky vyšších protokolů jako UDP, TCP, RSTP jsou přenášeny pouze v prvním fragmentu. Hlavičky, které jsou v pořadí před Fragment Header, je potřeba duplikovat a jsou uloženy před hlavičkou Fragment Header pro každý fragment. Při zpětném sestavení pak sestavený paket obsahuje všechny extension headers prvního fragmentu vyjma hlavičky Fragment Header. Dále je nutné při vyjmutí hlavičky změnit pole Next Header předcházející hlavičky na hodnotu, která se původně nacházela ve Fragment Header poli Next Header a upravit Payload lenght na hodnotu sestaveného paketu.

1.4.4 Destination Options Header

Záhlaví Destination Options Header je určené pouze pro příjemce zprávy. Next Header pole pro identifikaci tohoto záhlaví je hodnota 60. Záhlaví informuje cílovou stanici dodatečných informací a jelikož je určené pouze pro příjemce zprávy a není zpracováváno mezilehlými uzly, je toto pole při fragmentaci umístěno pouze do prvního fragmentu. Při zpětném sestavení totiž cíl obdrží celý paket s jedním záhlavím Destination Options. Struktura záhlaví (viz obr. 1.12) obsahuje jako první opět 8 bitové pole Next Header. Následuje 8 bitové pole Hdr Ext Len, které udává délku záhlaví. Posledním polem je Options proměnné délky, které obsahuje data konkrétní Option [23].



Obr. 1.12: Destination záhlaví

Pro Destination Options Header například existuje option Home Address Op-

tion. Tato option je využívána při mobilitě IPv6. Struktura option se skládá z 8 bitového pole Option Type, které je v případě Home Address Option nastaveno na hodnotu 201. Následuje 8 bitové pole Option Length, udávající délku option. V délce se u option ignorují pole Option Type a Option Length, takže v případě Home Address Option je v tomto poli hodnota 16, udávající 16 bytů následujícího pole Home Address. Pole Home Address má tedy délku 16 bytů a udává adresu domovského routeru, který se využívá při mobilitě IPv6 [28].

1.4.5 No Next Header

Rozšiřující záhlaví No Next Header identifikované hodnotou 59 v poli Next Header předchozího záhlaví signalizuje nepřítomnost dalších dat [23]. Pokud je dle délky IPv6 paketu zjevné, že následují další data, musí být tato data v případě předání paketu dalšímu uzlu nezměněny.

1.5 ICMPv6

Nedílnou součástí protokolu IPv6 je protokol *Internet Control Message Protocol version 6* (ICMPv6), bez kterého je IPv6 nefunkční a musí být implementován ve všech prvcích s podporou IPv6. Jedná se o servisní protokol, který nepřenáší žádná uživatelská data. Protokol ICMPv6 se stejně jako protokol *Internet Group Management Protocol* (ICMP) využívá k ohlašování chybových stavů, testování dostupnosti, zjišťování a přesměrování cesty paketů, nebo k výměně provozních informací. Oproti původnímu ICMP je ICMPv6 rozšířen o funkcionalitu zjišťování sousedů, překlad logických a fyzických adres, který v IPv4 poskytuje protokol *Address Resolution Protocol* (ARP), podporu multicastových skupin, které v IPv4 zajišťuje protokol *Internet Group Management Protocol* (IGMP), bezstavovou automatickou konfiguraci adres *Stateless Address Autoconfiguration* (SLAAC), nebo podporu mobility [3].

1.5.1 Formát ICMPv6

Záhlaví protokolu ICMPv6 je vloženo do datové části protokolu IPv6 s polem Next Header nastaveným na hodnotu 58 [27]. Formát záhlaví je tvořen 8 bitovým polem Type, které udává typ přenášené zprávy. První bit signalizuje, zda je zpráva typu error, nebo informační. V případě, kdy je první bit nastaven na hodnotu 0, jedná se o chybovou zprávu. V opačném případě se jedná o zprávu informační. Následuje 8 bitové pole Code, které blíže specifikuje typ zprávy. Dalších 16 bitů označuje Checksum určené pro detekci poškození dat ICMPv6 zprávy a části hlavičky IPv6 viz obr. 1.13. Poslední pole obsahuje tělo zprávy ICMP [27].

1.5.2 NDP

Jak již bylo zmíněno v IPv6 neexistuje protokol ARP. Místo ARPu je využit protokol *Neighbor Discovery protocol* (NDP), který kombinuje funkcionalitu ARP, ICMP Router Discovery a ICMP Redirect do jednoho protokolu [30]. ICMPv6 těchto mechanismů dosahuje pomocí zpráv Router Solicitation, Router Advertisement, Neighbor Solicitation, Neighbor Advertisement a Redirect.

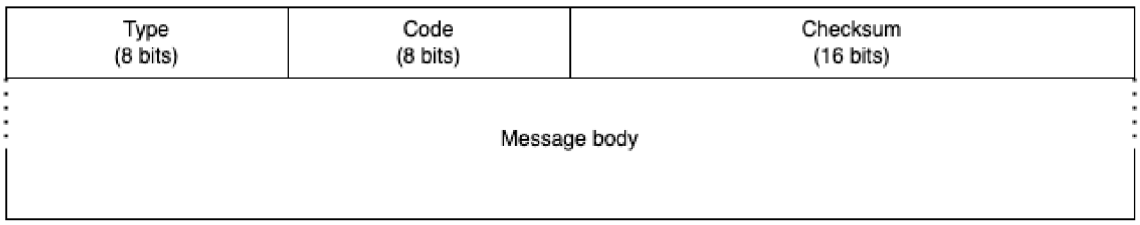
Router Solicitation

Router Solicitation je zpráva, kterou zasílá host, aby dotázal router o zprávu Router Advertisement [30]. Zdrojová adresa je adresa přidělená na odchozím rozhraní, případně nespecifikovaná adresa, pokud host nemá nastavenou adresu. Cílová adresa je v případě Router Solicitation typicky multicastová adresa všech směrovačů (ff02::2). V záhlaví ICMPv6 je nastaven Type na hodnotu 133, Code na hodnotu 0. Následuje 32 rezervovaných bitů a pole Options, které obsahuje například linkovou adresu.

Router Advertisement

Zpráva Router Advertisement, neboli oznámení směrovače oznamuje pravidelně zařízením v síti informace o síti, v které se nacházejí [30]. Ve zprávě se nacházejí důležité informace jako prefix sítě, jak nastavit adresu, MTU v síti, nebo nastavení časovačů pro objevování sousedních uzlů či platnost routeru.

Zdrojová adresa je adresa přidělená odchozímu rozhraní. Cílová adresa je multicastová adresa všech uzlů v síti (ff02::1) případně adresa uzlu, který zaslal zprávu Router Solicitation. Type ICMPv6 je nastaven na hodnotu 134 a Code na hodnotu 0. Po základním záhlaví ICMPv6 následuje 8 bitové pole Cur Hop Limit udávající hodnotu, která by měla být nastavena v IPv6 paketech v poli Hop Limit. Další bit značí flag M signalizující použití DHCPv6 a bit O, který informuje o dodatečných informací u DHCPv6. Příkladem může být adresa DNS serveru. Po bitech M a O následuje 16 bitové pole Router Lifetime určující dobu, po kterou je daný směrovač brán jako výchozí směrovač [30]. V případě hodnoty 0 není tento směrovač brán jako výchozí směrovač. Dalším polem je 32 bitové Reachable Time, které udává dobu, po



Obr. 1.13: Záhlaví ICMPv6

kterou je sousední zařízení po potvrzení dostupnosti považováno za stále dostupné. Následuje opět 32 bitové pole Retrans Timer, udávající dobu mezi Neighbor Solicitation zprávy pro sousední zařízení. Posledním polem jsou Options, které obsahují zdrojovou fyzickou adresu, hodnotu MTU v síti, nebo informace o prefixu sítě.

Neighbor Solicitation

Neighbor Solicitation slouží k zjištění fyzické adresy, dostupnosti zařízení, nebo jako mechanismus proti duplicitní konfiguraci adres. Zpráva je zasílána jako mutlicast, pokud zdroj zjišťuje fyzickou adresu, nebo unicast pokud zdroj ověřuje dostupnost zařízení [30]. Jako zdrojová adresa je adresa přiřazená na odchozí rozhraní, případně nespecifikovaná adresa. Cílová je adresa uzlu, nebo adresa typu solicited-node multicast. ICMPv6 type je nastaven na hodnotu 135 a Code na hodnotu 0. Po poli checksum následuje 32 rezervovaných bitů a 128 bitů reprezentující tázanou adresu. Pole Options se využívá pro zaslání zdrojové fyzické adresy.

Neighbor Advertisement

Neighbor Advertisement je odpovědí na zprávu Neighbor Solicitation, případně může být poslána bez vyzvání, pokud je potřeba příjemce informovat dříve. Zdrojová adresa je adresa odchozího rozhraní a cílová adresa může být adresa tázajícího uzlu, nebo v případě nespecifikované adresy tázajícího uzlu adresa všech uzlů v síti (ff02::1) [30]. ICMPv6 Type obsahuje hodnotu 136 a Code hodnotu 0. Po poli Checksum následují signalizační bit R udávající, zda je uzel směrovač, bit S říkající, že zpráva je odeslána jako odpověď na zprávu Neighbor Solicitation pro cílovou stanici a bit O, který příjemci říká, aby přepsal zprávou jeho aktuálně uložené informace. Po 3 signalizujících bitech je 29 bitů rezervovaných. Opět se zde nachází pole Target Address, které obsahuje tázanou adresu, nebo v případě, kdy Neighbor Advertisement nebylo vyžádané adresu, pro kterou se změnila fyzická adresa. Pole Options je využito pro fyzickou adresu k Target Address.

Redirect

Zpráva Redirect je využívaní pro informování hosta o lepší cestě k cíli v případě, kdy host využívá špatnou bránu a pakety tak musí být přeposílány dvakrát v rámci stejné sítě. Zpráva může být využita, i pokud zdroj neví, že cíl se nachází ve stejné síti. Zdroj je v tomto případě uzel, který zprávu poslal a cíl obsahuje zdrojovou adresu paketu, který zprávu Redirect vyvolal [30]. ICMPv6 Option obsahuje hodnotu 137 a Code hodnotu 0. Po poli Checksum následuje 32 bitů rezervy a 128 bitové Target pole Address, které obsahuje adresu lepšího uzlu a Destination Address obsahující cílovou

adresu v paketu, který vyvolal Redirect. Pole Options může obsahovat fyzickou adresu Target Address, nebo Redirect Header.

1.5.3 SLAAC a DHCPv6

IPv6 umožňuje automatickou bezstavovou konfiguraci adres SLAAC. Tato funkcionalita je výhodná z hlediska nepotřebnosti *Dynamic Host Configuration Protocol* (DHCP) serveru, který si musí uchovávat v paměti adresy, které již rozdal. Stanice si na základě prefixu, který obdrží ve zprávě Router Advertisement vygenerují adresy v daném prefixu [18]. Existuje více způsobů generace, je možné využít fyzickou adresu, nebo si adresu zvolit podle jiného mechanismu. Je však důležité ověřit, zda se daná adresa v síti nenachází. Toho je docíleno pomocí zprávy Neighbor Solicitation. V případě, kdy na tuto adresu zařízení odpoví, adresa není volná a je třeba vygenerovat novou. Pokud však zařízení odpověď neobdrží, může adresu používat.

DHCPv6 může pracovat ve dvou režimech. Může být využito bezstavově a v takovém případě slouží pouze pro předání dodatečných informací. Může se jednat například o adresy DNS serverů, mail serverů, nebo doménové jméno klienta. Druhým režimem je stavové DHCPv6, které funguje podobně jako klasické DHCP.

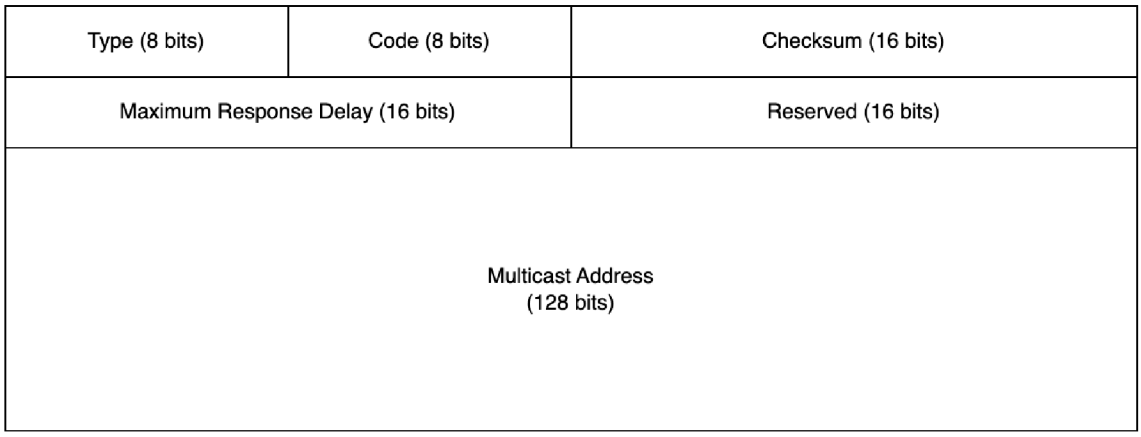
1.5.4 Multicast Listener Discovery

Protokol *Multicast Listener Discovery* (MLD) je obdobou protokolu IGMP v IPv6 sítích. Obdobně jako protokol IGMP zajišťuje v lokálních IPv6 sítích podporu multicastového přenosu, avšak oproti IGMP je MLD součástí protokolu ICMPv6 [31]. Protokol zjišťuje, které stanice mají zájem o multicastový přenos a jaký multicastový přenos chtějí přijímat. Router se pravidelně dotazuje, zda má jakákoliv stanice zájem o příjem daného multicastového přenosu. Pokud stanice má zájem, odpoví na tuto zprávu a router tak získá informaci, zda má daný multicastový přenos dále vyžadovat od multicastových routovacích protokolů. Všechny MLDv1 a MLDv2 zprávy musí obsahovat link-local IPv6 adresu, Hop limit nastaven na hodnotu 1 a obsahovat Hop-by-Hop Options header s Router Alert option [31, 32]. Hodnota Value v záhlaví této option je pro MLD nastavena na hodnotu 0 [33].

MLDv1

První verze protokolu MLD je odvozena od protokolu IGMPv2. Avšak oproti IGMPv2 zprávě [34] se pole MLDv1 zprávy mírně liší [31]. Formát MLD zprávy je vždy stejný a liší se pouze obsahem polí.

Zpráva MLDv1 je znázorněna na obrázku 1.14. Prvních osm bitů reprezentuje pole Type, které může nabývat tří různých hodnot dle typu zprávy. První typ Multicast Listener Query je reprezentován hodnotou 130 (decimální soustavy). Tento typ se dělí na dva další podtypy a to General Query, která slouží pro zjištění, které adresy jsou na lince požadovány a Multicast-Address-Specific Query, sloužící pro zjištění, zda je daná multicastová skupina vyžadována. Dále může pole Type nabývat hodnoty 131 pro zprávu Multicast Listener Report sloužící pro informování, že je daná skupina vyžadována, nebo 132 pro zprávu Multicast Listener Done, kterou stanice informuje směrovač o jejím ukončení odebírání dané multicastové skupiny. Následujících 8 bitů reprezentuje pole Code. Toto pole je odesílatelem nastaveno na hodnotu 0 a příjemci ho pro MLDv1 ignorují [31]. Po poli Code je 16 bitů využito pro pole Checksum sloužící pro kontrolu nepoškozenosti zprávy. Následujících 16 bitů je využito pro pole Maximum Response Delay, které udává maximální dobu, po kterou mohou příjemci multicastového vysílání vyslat odpověď. Po vypršení může směrovač přestat odebírat multicastovou skupinu v multicastových směrovacích protokolech. Pole je využito pouze pro typ zprávy Multicast Listener Query, v ostatních typech je nastaveno pole na hodnotu 0. Následujících 16 bitů je rezervováno pro budoucí použití. Poslední pole o velikosti 128 bitů reprezentuje Multicast Address. V případě General Query je nastaveno pole na hodnotu 0. V případě Multicast-Address-Specific Query je pole nastaveno na tázanou multicastovou adresu skupiny. V případě zpráv Multicast Listener Report a Multicast Listener Done je v poli nastavena adresa, kterou odesílatel poslouchá, nebo přestává poslouchat.



Obr. 1.14: Záhlaví MLDv1 zprávy

MLDv2

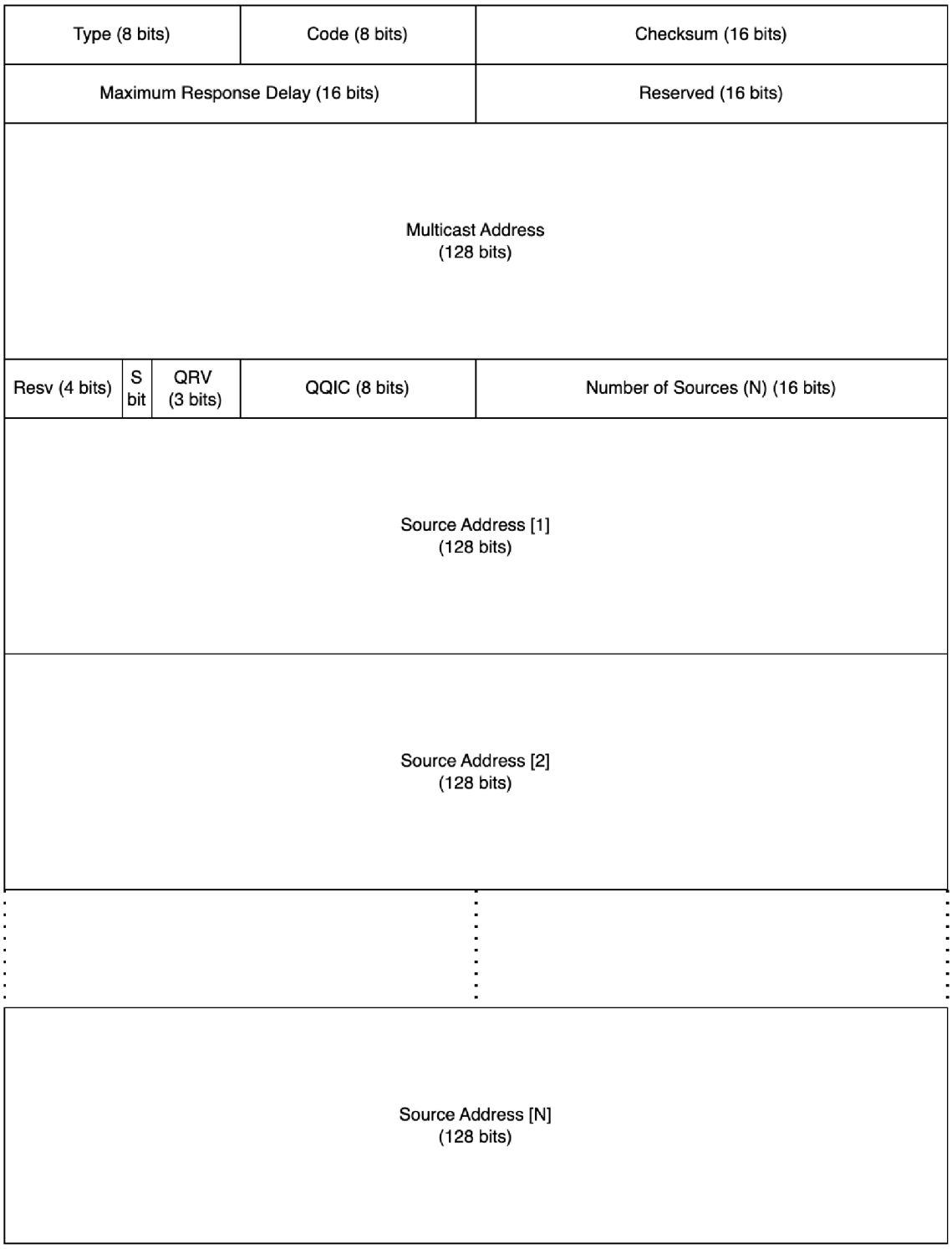
Oproti MLDv1, které je založené na protokolu IGMPv2 je MLDv2 založeno na protokolu IGMPv3. Oproti původnímu protokolu MLDv1 přidává MLDv2 možnost

filtrování zdrojů a to dvěma způsoby. První možností je přijímání multicastu pouze od vybraných zdrojů a druhou možností je přijímat od všech zdrojů s výjimkou vyčleněných. MLDv2 zároveň zachovává zpětnou kompatibilitu s předchozí verzí MLDv1. Oproti verzi MLDv1 definuje MLDv2 nový typ zprávy Version 2 Multicast Listener Report s hodnotou Type nastavenou na 143 decimálně [32].

Pole zprávy MLDv2 Multicast Listener Query Message jsou v prvních 192 bitech totožná se zprávou MLDv1 Multicast Listener Query. Po původní MLDv1 zprávě následují 4 rezervované bity pro budoucí použití. Následující bit reprezentuje flag S. V případě, kdy je flag nastaven na hodnotu jedna, všechny routery přijímající multicastové směrování musí potlačit běžné aktualizace časovače [32]. Další 3 bity jsou využity pro pole QRV (Querier's Robustness Variable). Pole je využito pro případy, kde je linka ztrátová a může nabývat hodnoty 0-7. Následujících 8 bitů reprezentuje pole QQIC (Querier's Query Interval Code). Toto pole reprezentuje dobu mezi zprávami Multicast Listener Query dotazujícího se routeru. Dalších 16 bitů označuje pole Number of Sources (N), které udává počet zdrojů obsažených v MLDv2 zprávě. Počet zdrojů je omezen MTU (Maximum transmission unit) linky. Po poli Number of Sources (N) následují pole Source Address [i], kde i reprezentuje index zdrojové adresy dle jejich počtu. Každé pole Source Address [i] je reprezentováno 128 bity viz obr. 1.15.

Zprávy Version 2 Multicast Listener Report Message jsou odesílány stanicemi k oznámení aktuálního stavu a změn přijímaných multicastových vysílání. Struktura této zprávy je znázorněna na obrázku 1.16. Po 8 bitovém poli typ zprávy, které je nastaveno na hodnotu 143 následuje 8 rezervovaných bitů pro budoucí použití a 16 bitů využitých pro pole Checksum. Po poli Checksum následuje 16 bitů opět vyhrazených pro budoucí využití. Po těchto rezervovaných bitech následuje pole Nr of Mcast Address Records (M) udávající počet polí Multicast Address Record ve zprávě. Multicast Address Record jsou datové bloky obsahující informace o přijímané multicastové adrese z pohledu příjemce [32]. Tyto záznamy mohou nabývat různých délek bitů převážně dle počtu definovaných zdrojových adres, avšak v novějších verzích protokolu může zprávu prodloužit i pole Auxiliary Data, které v RFC 3810 [32] nemá využití.

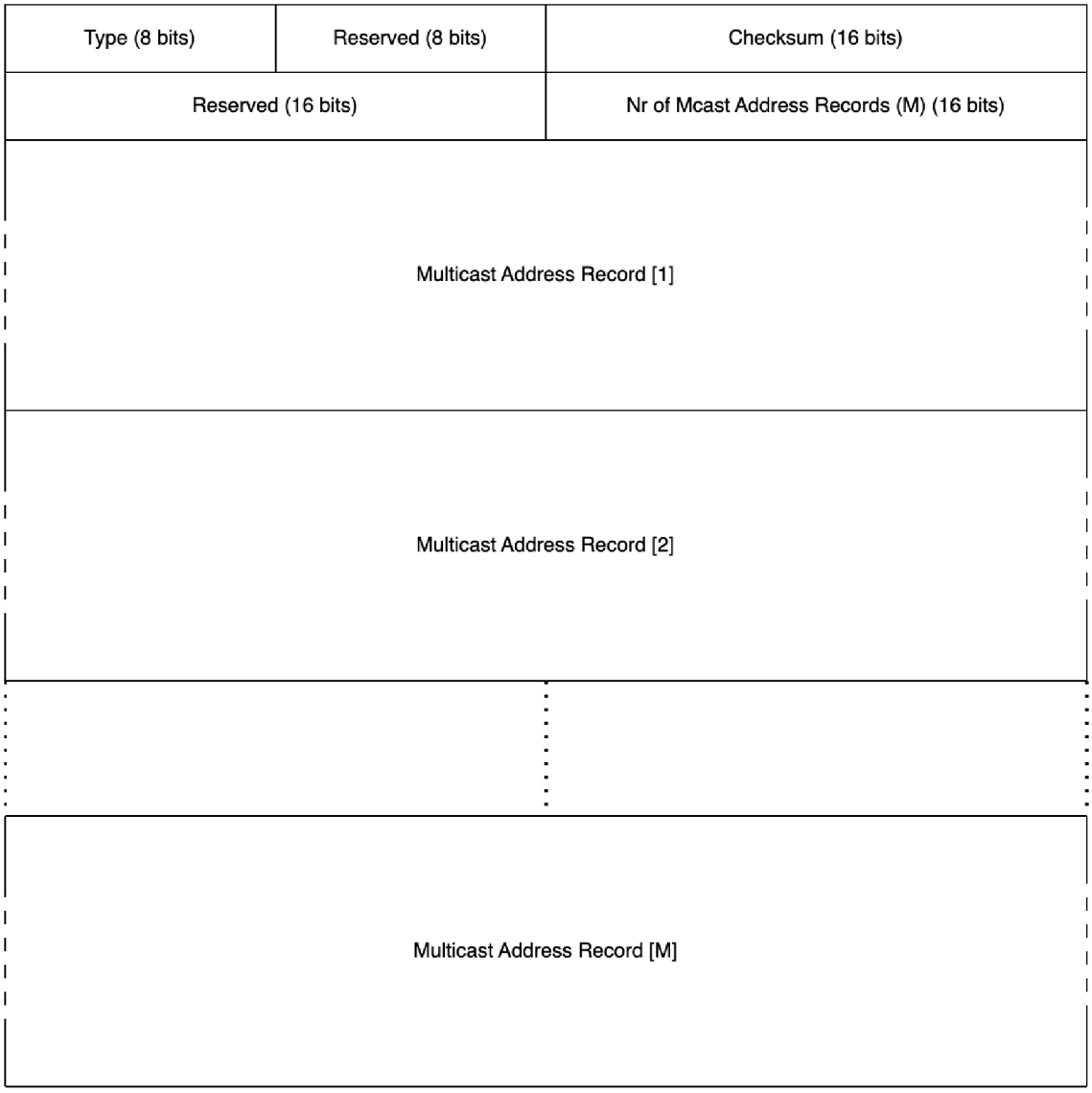
Pole Multicast Address Record je znázorněno na obrázku 1.17. Tento záznam obsahuje 8 bitové pole Record Type. RFC 3810 [32] specifikuje 6 různých typů pole Record Type. Hodnota 1 reprezentuje mód MODE_IS_INCLUDE, který udává, od kterých adres chce stanice multicastové vysílání přijímat. V případě prvního typu musí stanice uvést minimálně jeden zdroj. Tím se liší od druhého typu reprezentovaným hodnotou 2 udávající MODE_IS_EXCLUDE. Tento mód naopak přijímá multicastové vysílání od všech zdrojů, kromě zdrojů uvedených v Multicast Address Record. Následující hodnoty 3 a 4 reprezentují přechod na mód INCLUDE resp.



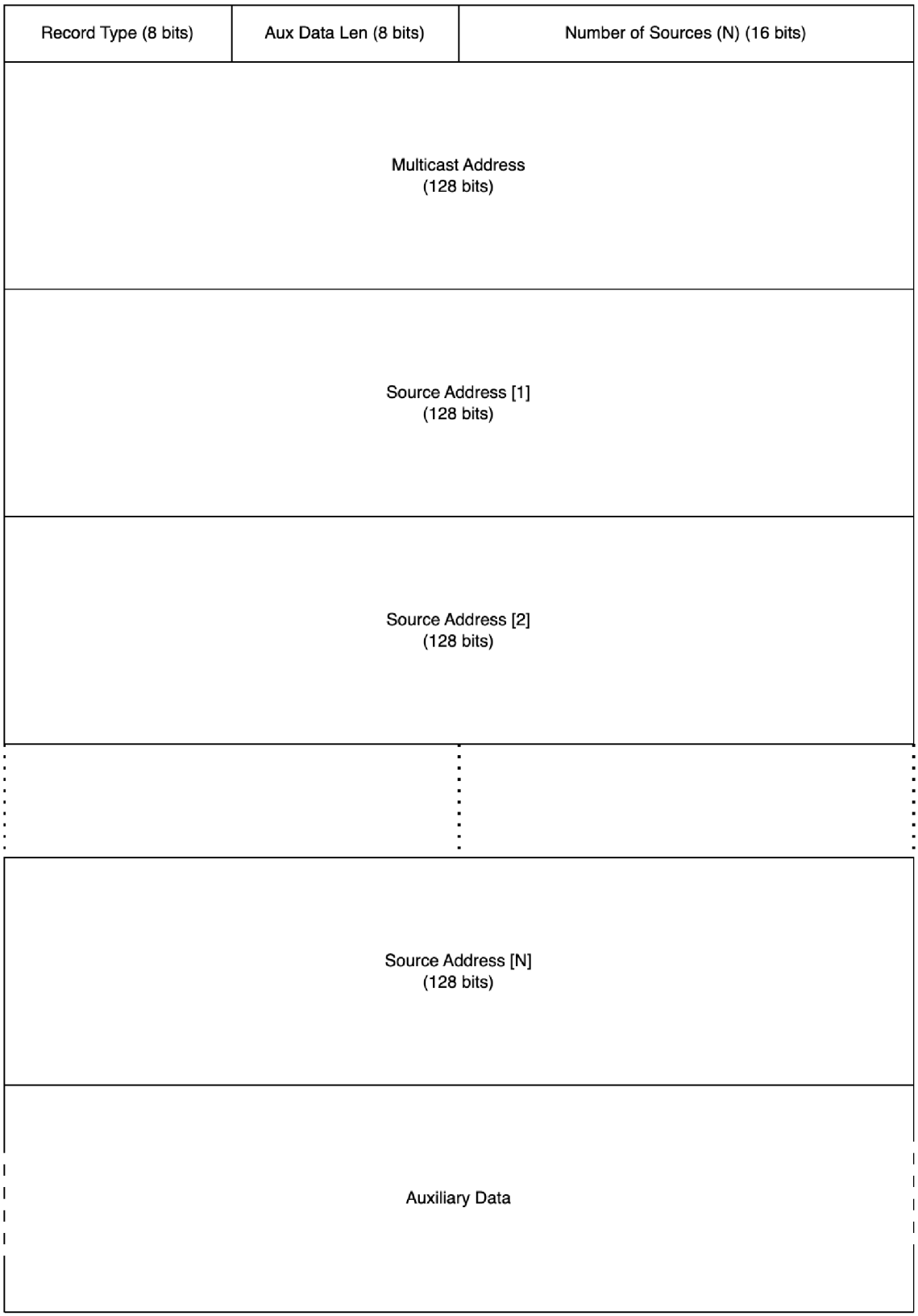
Obr. 1.15: Záhlaví MLDv2 Multicast Listener Query Message

EXCLUDE [32]. Poslední hodnoty 5 a 6 v poli type reprezentují přidání zdrojů resp odebrání zdrojů multicastového vysílání. Tyto pole respektují aktuální mód INCLUDE, nebo EXCLUDE. Po poli Record Type následuje 8 bitové pole Aux Data Len. Toto pole udává délku Auxiliary Data v jednotkách 32 bitových slov, avšak pro

MLDv2 je pole Auxiliary Data nevyužité a délka je tak nastavena na hodnotu 0. Dalším polem je 16 bitové pole Number of Sources (N), které udává počet Source Address v Multicast Address Record. V případě, že je využit mod EXCLUDE, může být toto pole nastaveno na hodnotu 0. Po počtu zdrojů multicastu následuje 128 bitů reprezentující adresu dané multicastové skupiny. Po této adrese následují opět 128 bitové adresy zdrojů. Posledním polem Multicast Address Record je pole Auxiliary Data, které jak již bylo zmíněno zůstává pro MLDv2 nevyužité.



Obr. 1.16: MLDv2 Multicast Listener Report Message



Obr. 1.17: MLDv2 Multicast Address Record

1.6 Přechod na IPv6

Vzhledem k tomu, že většina sítí vznikla na základě IPv4 a protokol IPv6 není zpětně kompatibilní s protokolem IPv4, bylo vyvinuto několik mechanismů, které usnadňují přechod na protokol IPv6. Postupný přechod z IPv4 sítí na IPv6 sítě lze rozdělit do 5 různých fází přechodu, pro které existují různé mechanismy přechodu [3].

První fází je provoz sítě postavené pouze na protokolu IPv4, bez jakékoliv podpory IPv6. Taková síť umožňuje provoz jednodušších a tedy levnějších zařízení, které podporují pouze protokol IPv4. Nevýhodou je nemožnost připojení se k sítím využívající protokol IPv6.

Druhá fáze představuje stav, kdy je stále využíván převážně protokol IPv4, ale protokol IPv6 je již také využíván. Pro zpřístupnění IPv4 sítím sítě IPv6 je možné využít tunelování protokolu IPv6 přes sítě IPv4. Výhodou tohoto řešení je zpřístupnění IPv6 sítí, bez nutnosti podpory IPv6 ve všech zařízeních. Nevýhody pak představuje stálá nutnost provozu protokolu IPv4 a potenciálně problematické fungování tunelování [3].

Třetí fází je souběh obou protokolů současně. V tomto případě musí všechna zařízení podporovat jak protokol IPv4, tak IPv6, což představuje nákup složitějších zařízeních a tedy vyšší pořizovací cenu. Oba protokoly pracují nezávisle na druhém. Další nevýhodou tohoto přístupu je stálá potřeba IPv4 adres. Výhodou je plná funkcionalita obou protokolů.

Čtvrtou fází je upozaďováni protokolu IPv4. Hlavním protokolem je IPv6 a IPv4 sítě jsou zprostředkovávány pomocí mechanismů přechodu. Mezi tyto mechanismy lze znovu zařadit tunelování, tentokrát však protokolu IPv4 přes síť IPv6. Toto řešení však opět nese jisté komplikace pro fungování IPv4 protokolu [3]. Výhodou je, že IPv6 není ve svém fungování omezeno. Dalším způsobem je překlad IPv4 adres. Oproti IPv4 má IPv6 daleko větší adresní prostor. Z adresního prostoru IPv6 lze tedy vyčlenit část, kterou reprezentují část, nebo celý adresní prostor, a to dokonce i několikrát. Tato technika se nazývá NAT-PT (Network Address Translator – Protocol Translator) a zajímavý je především protokol NAT64 spolu s DNS64 probraných v kapitole 1.6.1 a 464XLAT probraný v kapitole 1.6.2.

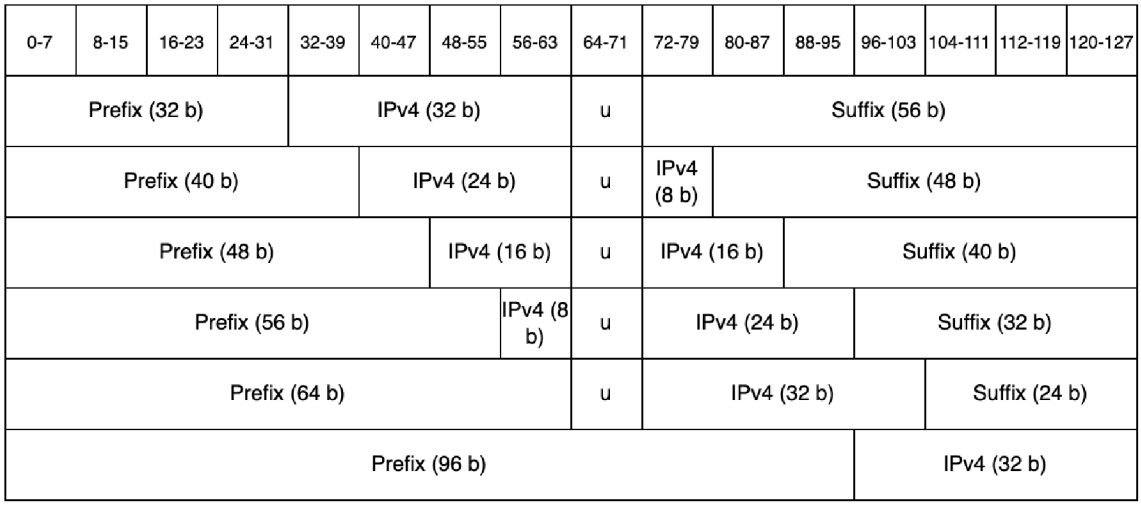
Poslední fází je úplné upuštění od protokolu IPv4. Protokol IPv4 se v této fázi vůbec nevyužívá, nebo je dokonce blokovaný. Výhodou je zjednodušení síťových prvků i koncových stanic. Nevýhodou je pak kompletní nedostupnosti protokolu IPv4. Otázkou je kdy a případně, zda tento stav vůbec nastane [3].

1.6.1 NAT64 a DNS64

Protokol NAT64 spolu s protokolem DNS64 představují řešení, které umožňuje v IPv6 only sítích komunikaci se stanicemi podporující pouze protokol IPv4. Tento protokol řeší jak případ, kdy je podporován pouze jeden z protokolů na straně stanice, tak na straně síťových prvků. Pro fungování tohoto mechanismu jsou potřebné oba protokoly.

NAT64

NAT64 je v podstatě překlad IPv6 na IPv4 adresy. V části IPv6 je vymezen rozsah, ve kterém 32 bitů IPv6 adresy reprezentuje adresu IPv4. Díky 128 bitovému adresnímu prostoru IPv6 je možné obsáhnout 32 bitový adresní prostor IPv4 dokonce několikrát. IPv4 adresní prostor je pak pomocí IPv6 adresy reprezentován pomocí prefixu, IPv4 adresy a suffixu. Bity 64 až 71 IPv6 adresy jsou rezervovány pro kompatibilitu s host identifier format [35]. Existují dva druhy prefixů a to well-know prefix (64:ff9b::/96), nebo specifický prefix sítě. Povolené délky specifický prefixů sítě jsou 32, 40, 48, 56, 64, nebo 96. Všechny povolené varianty prefixů jsou znázorněny na obr. 1.18. Adresy v těchto prefixes generuje DNS64 server, který v případě neexistence záznamu IPv6 generuje IPv6 adresu pro NAT64.



Obr. 1.18: Reprezentace IPv4 adresy v IPv6 adrese

Při DNS dotazu na zařízení, které nepodporuje IPv6 vrátí DNS64 adresu IPv6 s vnořenou adresou IPv4 daného zařízení. Díky prefixu je paket následně směrován k NAT64. Nastavení směrování prefixu k NAT64 je zásadním předpoklad pro správnou funkcionalitu. NAT64 při příchodu paketu určeného k překladu do IPv4 sítě vybere aktuálně nevyužitý port a spolu s adresou IPv4 adresou NAT64 vytvoří mapovací záznam skládající se z IPv6 adresy, zdrojového portu, IPv4 adresy NAT64

a vybraného volného portu. Následně je paket přeložen dle RFC 7915 [36]. IPv6 záhlaví je nahrazeno záhlavím IPv4 se zdrojovou IPv4 adresou NAT64 a cílovou adresou obsaženou v původní IPv6 cílové adrese. V případě použití TCP, nebo UDP je nahrazen zdrojový port vybraným volným portem. V případě ICMP je využito například pole identifier. IPv4 paket je následně směrován do IPv4 sítě. Při přijmutí odpovědi NAT64 vybere odpovídající mapovací záznam a paket zpětně přeloží pro IPv6 síť a odešle k odpovídající IPv6 stanici.

Výše popsaný mechanismus se věnuje stavovému NAT64. Stavový NAT64 slouží převážně k překladu IPv6 adres na adresy IPv4 a jejich zpětnému překladu. Předpokládá se však, že komunikaci inicializuje IPv6 stanice [37]. Hlavním důvodem je rozdíl ve velikosti adresních prostorů. V případě, kdy je třeba zpřístupnit IPv4 stanici zařízení v IPv6 síti, je možné využít bezstavového NAT, nebo port forwardingu. V případě bezstavového NAT je vytvořen statický záznam IPv6 adresy a odpovídající IPv4 adrese případně lze využít port forwardingu a zpřístupnit tak službu na stanici IPv6 do sítě IPv4.

DNS64

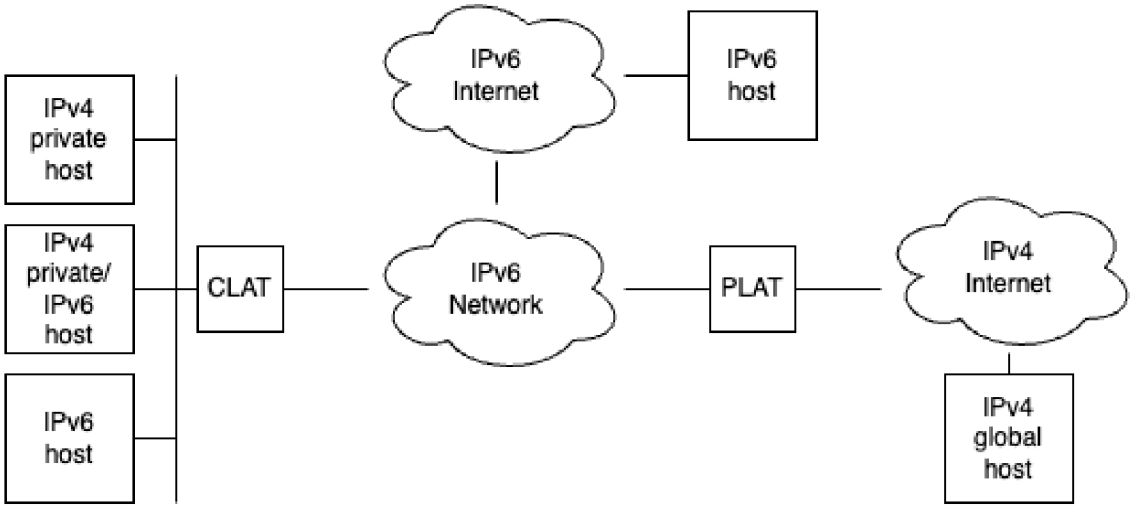
DNS64 spolu s NAT64 umožňují koncové stanici v IPv6 síti komunikovat se stanici v IPv4 síti bez nutnosti další konfigurace koncových stanic. DNS64 funguje jako standardní DNS do okamžiku, kdy není k tázané URL adrese záznam AAAA, který obsahuje odpovídající IPv6 adresu. V takovém případě vybere DNS64 záznam A obsahující IPv4 adresu a dle předem nakonfigurovaného prefixu tuto adresu přeloží na IPv6 adresu. Následný AAAA záznam vrátí jako odpověď [38]. Při obdržení AAAA odpovědí stanice komunikuje s daným zařízení za pomocí NAT64.

Použití DNS64 spolu s NAT64 však přináší různé problémy, zejména pak z hlediska bezpečnosti problémy s rozšířením DNSSEC. DNSSEC je rozšíření protokolu DNS, které si klade za cíl zabránit modifikaci DNS odpovědi po cestě k rekurzivnímu serveru, případně ke koncové stanici. Toto zabezpečení přímo odporuje principu modifikace odpovědi pomocí DNS64. Při použití DNS64 a DNSSEC tedy mohou nastat různé případy, kdy některé neovlivní funkcionalitu a některé skončí zamítnutím DNS64 odpovědi z důvodu její modifikace. Tyto případy jsou popsány v RFC 6147 [38].

1.6.2 464XLAT

464XLAT je dalším mechanismem přechodu mezi IPv6 a IPv4 sítí. Tento mechanismus se dále skládá z PLAT (provider-side translator (XLAT)) a CLAT (customer-side translator (XLAT)). Jak název vypovídá PLAT je zařízení na straně poskytovatele internetu a překládá adresy dle [37]. CLAT je zařízení na straně zákazníka

a využívá různé prefixy IPv6 [39]. 464XLAT umožňuje IPv4 i IPv6 zařízení komunikovat s IPv4 zařízeními pomocí IPv6 infrastruktury. Nejlépe tento mechanismus vysvětluje obrázek 1.19



Obr. 1.19: Příklad zapojení 464XLAT

1.7 Mobilita

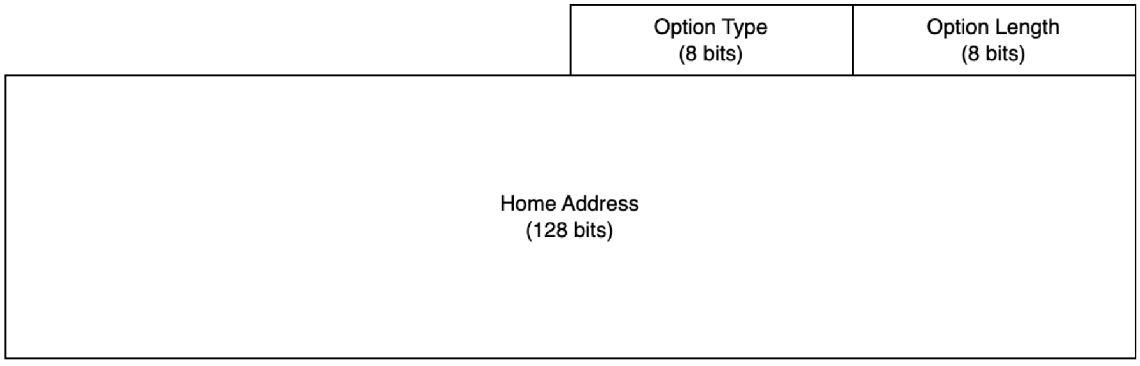
Při návrhu IPv6 bylo myšleno i na zařízení, která se mohou během komunikace pohybovat a to i v rámci různých sítích. Při klasické komunikaci se zařízení při směrování paketu řídí jeho směrovací tabulkou, která je nejčastěji vyplněna dynamickými směrovacími protokoly. Směrovače však paket pro danou adresu směřují do cílové sítě a netuší o možném pohybu mobilního zařízení do cizích sítí. Právě pro tyto případy existuje protokol Mobile IPv6 [28].Protokol definuje nové termíny používané v IPv6 mobilitě. Mezi nejdůležitější patří následující:

- Mobile node - stanice schopna měnit místo svého připojení v síti.
- Home address - permanentní unicastově routovatelná adresa mobilní stanice.
- Home link - síť jejíž prefix obsahuje Home address mobilního zařízení.
- Foreign link - jakákoliv síť mimo domovskou síť.
- Care-of address - unicastově routovatelná adresa v cizí síti, která je přiřazena mobilnímu zařízení při jeho připojení do cizí sítě. Mobilní zařízení může mít přiřazeno více Care-of adres. Care-of adresa registrovaná s home agent je nazvaná "primary".
- Correspondent node - stanice komunikující s Mobile node.
- Home agent - router připojený do domovské sítě mobilní stanice. V případě, kdy je mobilní stanice mimo svou domovskou síť, přeposílá home agent pakety

pro mobilní stanici na její primary care-of adresu.

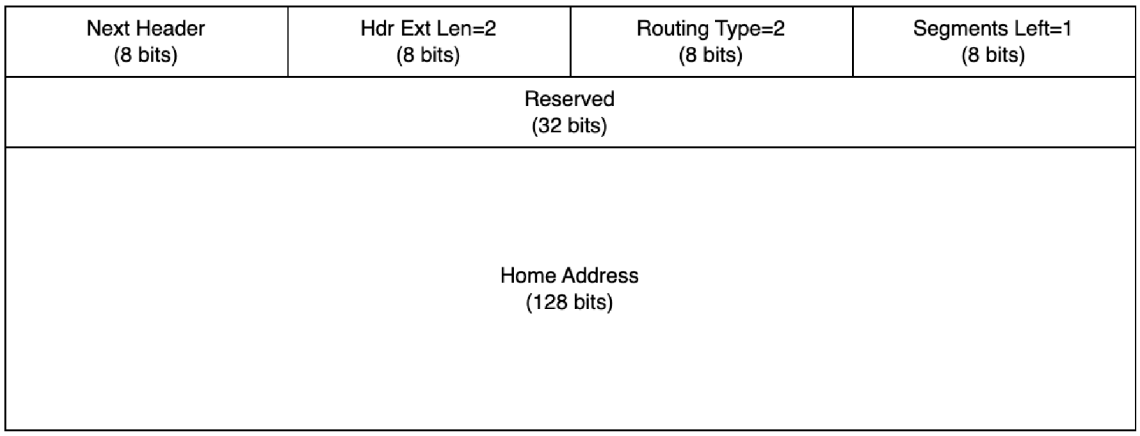
Pro podporu tohoto mechanismu existují dva způsoby komunikace. První způsob potřebuje ke své funkci podporu Mobile IPv6 na straně mobilní stanice a home agenta. V případě, kdy mobilní stanice opustí svou domovskou síť, musí se zaregistrovat u svého home agenta. Ten vytvoří vazbu mezi domovskou adresou a care-of adresou [28]. Následně je v případě, kdy chce correspondent node komunikovat s mobile node, pošle correspondent paket s destination adres odpovídající home address mobile node. Tento paket přijme home agent, který paket zapouzdří a odešle na care-of adresu mobile node. Pro odpověď se využije stejného principu. Mobile node prvně zapouzdří odpověď a odešle home agentovi. Ten paket následně odešle ke correspondent node [28]. Při využití tohoto způsobu nemusí correspondent node podporovat Multicast IPv6, což je velkou výhodou. Naopak nevýhodou tohoto způsobu je potenciální delší trasa paketu, který musí vždy přejít i přes home agenta.

Druhý způsob vyžaduje podporu correspondent node, avšak umožňuje optimalizování cesty toku paketů. Tento způsob vyžaduje registraci vazby mezi home address a care-of adress u correspondent node. Ta při odeslání paketu použije jako cíl care-of adresu mobile node a přidá k paketu Routing Header s option Type 2 Routing Header, odpovídající hodnotě 2 [28] znázorněný na obrázku 1.21. Tato option obsahuje home adress mobile node, se kterou chce correspondent node komunikovat. Při přijetí je nahrazena zdrojová adresa home address za care-of address a přidán Destination options header obsahující option "Home Address" obsahující home address mobile node [28] znázorněný na obrázku 1.20. Díky tomuto mechanismu nejsou ovlivněny protokoly vyšších vrstev. Mezi výhody tohoto mechanismu patří optimalizování cesty paketu a zachování komunikace v případě, kdy není dostupný home agent.



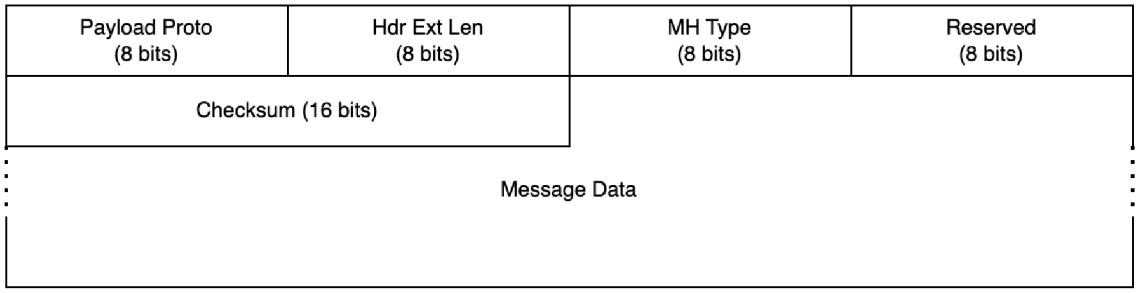
Obr. 1.20: Home address option

Mobility IPv6 dále definuje Mobility header obr. 1.22, který slouží pro ustanovení a management vazeb home address a care-of address. Dále Mobility IPv6 definuje řadu bezpečnostních mechanismů pro zabezpečení obou způsobů komunikace. Mezi



Obr. 1.21: Type 2 Routing Header

ty nejdůležitější patří využití IPsec extension headers, nebo využití Binding Authorization Data option [28]. Přičemž se Binding Authorization Data option vkládá právě do nově definovaného Mobility header.



Obr. 1.22: Mobility header

1.8 Bezpečnost IPv6

Od doby vzniku IPv6 bylo objeveno spoustu bezpečnostních problémů. Některé tyto problémy IPv6 sdílí s IPv4, jiné jsou pro IPv6 nové. Z hlediska bezpečnosti je potřebné tyto problémy hledat a nalézat jejich řešení.

1.8.1 Fragmentace

Problémem fragmentace je horší filtrování za pomocí firewallů. Paket totiž není při průchodem firewallu celý a tak firewall nemusí poznat data vyšších vrstev [40]. To je také důvodem proč jsou často fragmentované pakety v síti blokované. Fragmentace sice umožňuje přenést pakety s velkými objemy dat pomocí rozdělení, ale nastává otázka, zda je opravdu potřeba tato data fragmentovat. Ve skutečnosti se totiž může

namísto snížení nadbytečných dat díky menšímu počtu hlaviček objem přenášených dat zvýšit, kvůli nadbytečné hlavičce fragmentace. Aplikace by tedy, pokud nutně nepotřebují přenášet data pomocí velkých paketů, měla data dělit rovnou na menší úseky tak, aby fragmentace nebyla vůbec nutná.

Dalším problémem fragmentace jsou překrývající se fragmenty (overlapped fragments) [41]. Tato technika využívá nekorektní manipulace s fragmenty některých systémů. Příkladem útoku může být vyslání prvního fragmentu, který je úspěšně propuštěn firewallem, ale na koncové stanici je část toho fragmentu přepsána hodnotou, která by firewallem neprošla. Následně si může koncová stanice sestavit škodlivý paket, který měl být původně firewallem blokován. Útoků však může být více. V průběhu útoku se může například zaslat několikrát fragment s polem Offset nastaveným na hodnotu 0. Díky rozšiřujícím záhlavím může být detekce překrývajících se fragmentů ztížena. Dle RFC 5722 [42] musí být při detekci překrývajících se fragmentů zahozeny všechny fragmenty spadající do daného paketu.

Ačkoliv minimální MTU pro IPv6 je stanoveno na hodnotu 1280 bajtů, RFC2460 [43] definuje způsob chování při komunikace mezi IPv6 a IPv4 sítí při MTU menší než 1280 bajtů. V takovém případě totiž lze obdržet ICMPv6 zprávu Packet Too Big, která informuje o MTU menší než je 1280 bajtů. Vzhledem k tomu, že IPv4 a IPv6 řeší fragmentaci odlišným způsobem, RFC2460 přichází s řešením, kdy zdroj přidá záhlaví fragmentace a IPv4 následně využívá například hodnotu identifikace pro fragmentaci v IPv4. Tento mechanismus umožňuje vytvoření takzvaného atomického fragmentu (atomic fragment), který obsahuje hodnotu offset 0 signalizující první fragment paketu, ale zároveň hodnotu bitu M nastavenou na 0 signalizující poslední fragment. Některé implementace tak mohou mít problém se skládáním fragmentů. U atomických fragmentů je možné překrytí s jinými fragmenty při zpětném sestavení a následné zahození dle RFC 5722 [42]. V případě, kdy potenciální útočník odesílá fragmenty se stejnými poli Source Address, Destination Address a Identification je schopný způsobit útok typu *Denial of service* (DoS). Na tento problém se zaměřuje RFC6946 [41], které mění způsob zacházení s atomickými fragmenty. Pokud je doručen atomický fragment na stanici, musí s ním být zacházeno individuálně a nesmí tedy ovlivňovat nesouvisející fragmenty.

Každá stanice dle RFC 8200 [23] musí být schopna přijmout fragmentovaný paket o velikosti nejméně 1280 bytů. Je však možné, aby stanice přijala i delší zprávy. V RFC 8200 se však píše, že by se měl odesílatel přesvědčit, zda cílová stanice je schopna paket o větší délce sestavit. RFC 8200 [23] doporučuje MTU 1500 bytů. Otázkou tak je, jak bude konkrétní implementace reagovat na fragmentované pakety delší než 1500 bytů a zda lze takto dlouhé fragmenty použít k potenciálnímu útoku.

1.8.2 Velké množství rozšiřujících hlaviček

Protokol IPv6 sice šetří místo oproti protokolu IPv4, který obsahuje pole, která jsou často nevyužita, avšak právě způsob jakým IPv6 šetří místo může být zneužit. Lze totiž vkládat rozšiřující hlavičky neustále za sebou a následný firewall, či stanice musí projít všechny hlavičky až nakonec [40]. To může firewall podstatně zpomalit a snížit tak přenosovou rychlost, či poškodit zařízení. Řešením je například omezit maximální počet vnořených hlaviček. V případě, kdy paket přesáhne toto množství, firewall paket jednoduše zahodí.

1.8.3 Nadbytečný adresní prostor

Ačkoliv IPv6 má z dnešního pohledu dlouhodobě nevyčerpatelný adresní prostor, není vždy nutné a bezpečné tímto prostorem plýtvat. Například pro linky typu point-to-point není třeba plýtvat velkými sítěmi a je vhodné prodloužit masku. Dříve bylo problematické prodloužit masku až na hodnotu 127 [44], avšak zvýšení masky může pomoci při obraně zařízení před útoky DOS zaplněním paměti sousedů [45, 46, 40].

2 Penetrační testování

Penetrační testování je způsob obrany před potenciálními útočníky dříve, než zaútočí. Testování probíhá podobně jako skutečný útok. Rozdílem je však zachování integrity, dostupnosti a důvěrnosti. Penetrační testeři jsou experti na dané technologie a jejich cílem je odhalit zranitelnosti a otestovat hloubku, kam se lze za pomocí chyb v aplikacích, konfiguracích, protokolech a kryptografie dostat. Penetrační testování znamená hledání chyb, které jsou bezpečnostním rizikem.

2.1 Typy hackerů

Pojem hacker je spojován převážně v negativním smyslu označující člověka, který se nabourává s nekalými úmysly do informačních systémů. Ve skutečnosti ale nemusí jít tomuto člověku vůbec o zlý úmysl. Existuje několik druhů hackerů jako White hat, Black hat, nebo Gray hat [47]. Tito hackeři se liší svým motivem a oprávněním prolamovat informační systém.

2.1.1 White hat

White hat hacker je člověk, který se nabourává do systémů, ale pouze po předchozím oprávnění vlastníka systému. Vlastník si objedná, či jinak dohodne penetrační test a hacker, případně penetrační tester následně hledá slabiny v informačním systému [47]. Rozsah, způsob, čas a výsledek testování je vhodné předem domluvit, aby nedošlo k nedostupnosti systému, který je v daném čase pro vlastníka důležitý. Výstupem testu bývá report zranitelností, které byly v síti nalezeny. Tato služba je poskytována zpravidla za finanční odměnu. Některé firmy jsou zapojeny do takzvaných bug bounty programů. Umožňují tedy v rámci jejich definice testování jejich služeb a v případě nahlášené zranitelnosti dle její závažnosti poskytují nálezci odměnu.

2.1.2 Black hat

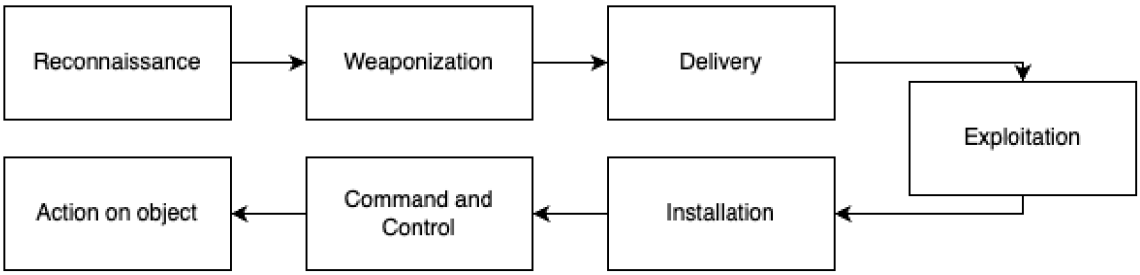
Black hat je pravý opak white hat hackera. Jeho motiv je převážně uškodit, nebo zbohatnout na úkor poškozeného. Black hat hackeři nerespektují bug bounty programy a napadají služby bez oprávnění [47]. V případě úspěšného útoku se snaží služby znepřístupnit, získat data, nebo okrást poškozeného o finance. Ukradená data také často nabízejí na černých tržištích. Může se jednat například o osobní informace, bankovní účty, nebo přístupové údaje k nejrůznějším službám.

2.1.3 Grey hat

Grey hat hacker se jak název napovídá nachází mezi white hat a black hat. Jeho motiv nemusí být nutně uškodit, ale systémy také nabourává bez předchozí dohody [47]. Nalezené zranitelnosti pak může nahlásit vlastníkovi.

2.2 Fáze penetračního testování

Protože penetrační testování simuluje určitým způsobem útok, odvíjí se od útoků i jeho fáze. Ty se snaží být co nejpodobnější skutečnému útoku. Před zahájením testování však musí být se zadavatelem, na rozdíl od skutečného útoku, jasně domluven rozsah testu. Skutečný útok je možné obecně dělit na jednotlivé části pomocí Cyber Kill Chain. Ten definuje 7 částí útoku nazvané Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command and Control a Action on object [48]. Řetězec cyber kill chain je znázorněn na obr. 2.1.



Obr. 2.1: Obecné schéma útoku dle cyber kill chain

2.2.1 Reconnaissance

Fáze reconnaissance je první fáze v cyber kill chain. Fáze se zabývá shromažďováním informací o potenciálním cíli. Reconnaissance se dále dělí na pasivní a aktivní. Při pasivním reconnaissance se informace shromažďují bez vědomí cíle. Může se jednat o veřejné záznamy jako doménová jména, whois, RIPE, APNIC ARIN, nebo sociální sítě, webové stránky, veřejně dostupné dokumenty. U aktivního reconnaissance se může cíl dozvědět o zjišťování informací. Může se například jednat o ping sweep, skenování portů, phishingové e-maily, telefonáty, nebo sociální inženýrství [48].

2.2.2 Weaponization

Ve fázi weaponization se útočník na základě zjištěných informací z fáze reconnaissance připravuje na útok. V této fázi si útočník připravuje vzdálený přístup do zařízení a exploit. Vzdálený přístup je útočníkovi zpřístupněn pomocí *Remote*

Access Tool (RAT) [48]. RAT je software, který útočníkovi umožňuje ovládat napadené zařízení. Zpravidla se RAT snaží o zamaskování v napadeném systému. Exploit je program, který RAT doručí a spustí v systému za pomocí zranitelnosti v systému, nebo softwaru. Exploity často využívají nepozornost uživatele, který vykoná zdánlivě neškodnou operaci jako otevření souboru, který uživateli připadá neškodný, ale ve skutečnosti na pozadí spouští škodlivý kód [48].

2.2.3 Delivery

Poté co útočník připraví exploit a RAT musí škodlivý software oběti doručit. V Cyber kill chain je tato fáze nazvaná delivery. Ve většině útoků je potřeba interakce s uživatelem, existují však útoky, které interakci nepotřebují. Útočníci pro doručení škodlivého softwaru využívají přílohy v e-mailové komunikaci, phisingové útoky, kompromitované stránky, přesměrování na jejich vlastní stránky pomocí DNS, nebo přenosná úložiště. Fáze doručení zanechává stopy a tak se útočníci schovávají pomocí VPN a jiných anonymizačních nástrojů [48].

2.2.4 Exploitation

Po doručení následuje fáze exploitation, kdy je exploit spuštěn. Spuštění může vyvolat uživatel spuštěním softwaru, otevřením souboru ve zranitelném softwaru, nebo může být exploit spuštěn automaticky připojením přenosných úložišť, nebo navštívením kompromitované stránky. Pro úspěšnou exploitaci musí uživatel zranitelnou verzi software, pro kterou byl exploit vytvořen a exploit nesmí být detekován antivirovým softwarem. V případě, kdy útočník cílí na masy, používá exploit kit. Exploit kit je kolekce exploitů cílící na software, který stanice využívá. Například může exploit kit mířit na několik verzí webového prohlížeče a tak je pro útočníka větší pravděpodobnost, že systém úspěšně exploituje [48].

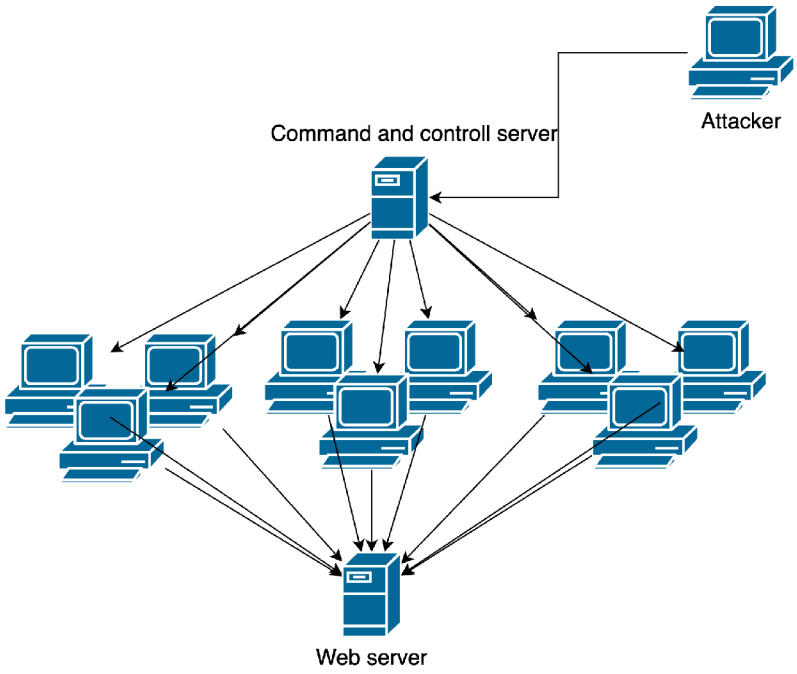
2.2.5 Installation

Po exploitaci následuje fáze instalace exploitu. Útočník například může nastavit operačnímu systému automatické spuštění RAT. Sofistikovanější útoky však využívají droppery a downloadery. Dropper je program, který nainstaluje a spustí škodlivý kód. Droppery se v dnešní době snaží před spuštěním malwaru o vypnutí bezpečnostních mechanismů a skrytí malwaru. Downloader využívá stejný mechanismus jako dropper, ale neobsahuje celý škodlivý kód. Po vypnutí bezpečnostních mechanismů se škodlivý kód stáhne ze serveru [48]. Antivirové softwary se snaží chování dropperů a downloaderů detekovat pomocí signatur a anomálií.

2.2.6 Command and Control

Po nainstalování škodlivého softwaru potřebuje útočník tento systém vzdáleně ovládat. To útočník dělá pomocí *Command and Control* (C&C) systémů. C&C mohou mít různou architekturu. Hlavní tři architektury jsou centralizovaná, decentralizovaná a založená na sociálních sítí.

Centralizovaná architektura je založená na systému klient a server viz obr. 2.2. Klient se připojí na server a ten mu oznámí, co má udělat. Nevýhoda této architektury je v problému single point of failure. V případě, kdy totiž zamezíme komunikaci se serverem, například prostřednictvím firewallu, nemůže útočník stanici ovládat. Server také musí mít dostatečný výpočetní výkon, aby zvládl obsloužit všechny klienty [48]. IP adresy jsou také možnou signaturou útoku a bezpečnostní operační centra monitorují listy podezřelých adres pro detekci napadeného zařízení.



Obr. 2.2: Centralizovaná architektura botnetu

Distribuovaná architektura je založená na peer to peer síti. Napadené stanice komunikují mezi sebou a odpadá tak zátěž na jeden server. Stanice jsou dále zodpovědné za část botnetu, což útočníkovi přináší redundanci [48]. Také je náročnější spravovat listy C&C adres.

Architektura založená na sociálních sítí předává informace napadeným systémům prostřednictví příspěvků na sociálních sítí. Vzhledem k vysoké dostupnosti systémů sociálních sítí je pro útočníka výhodné předávat příkazy pro botnet pomocí příspěvků. Stanice pak dotazují vybrané příspěvky s instrukcemi, kterými se řídí [48]. Z pohledu bezpečnostního operačního centra je detekce takových botnetů

více problematická. Napadené stanice se totiž dotazují na adresy běžných služeb a komunikace je zpravidla šifrovaná.

2.2.7 Action on object

Poslední fází cyber kill chain je samotná akce napadených systémů. Systémy se v této fázi chovají podle útočníkových příkazů. Útočník může mít konkrétní cíl, nebo cílí na množství stanic. V případě cíleného útoku zpravidla postupuje útočník opatrněji a cílem je získání tajných informací ze systému. To mohou představovat například bankovní účty, nebo tajné dokumenty. V případě hromadných útoků se snaží útočník napadnout co největší množství systémů. Skupiny napadených systémů se nazývají botnety [48]. Ty následně může útočník využít na distribuovaný útok odepření služeb (DDoS), těžení kryptoměn, nebo okradení uživatelů systémů o jejich bankovní účty.

3 Využité technologie

Při testování funkčnosti i bezpečnosti IPv6 je vhodné používat specializované nástroje, které umožňují testerovi lepší manipulaci s pakety a vytvořením specifických situací pro ověření chování protokolu. Nejzajímavější nástroje z hlediska bezpečnostního testování jsou sniffery, generátory provozu a automatické testery. Sniffery naslouchají pro veškerý síťový provoz a odposlechnutý provoz následně analyzují a vizualizují použité protokoly. Generátory provozu testerovi umožní generovat specifický provoz, který by za normálních okolností bylo složité, nebo nemožné vygenerovat. Lze tak vyzkoušet scénáře útoku případného útočníka a zkontrolovat tak, zda síť není náchylná na různé útoky. Jiné nástroje umožňují automatické testy některých zranitelností. Nástroj je připojen do sítě a následně síť skenuje na známé zranitelnosti. Výhodou těchto nástrojů je rychlost, avšak oproti manuálním testům detekují automatické zpravidla menší množství zranitelností.

3.1 Wireshark

Wireshark je jeden z nejpoužívanějších snifferů. Jedná se o grafický nástroj, který analyzuje provoz na síťové kartě přepnuté do režimu odposlouchávání [49]. Wireshark obsahuje v základním rozhraní tři okna. V prvním okně lze vidět přehledně veškerý síťový provoz, který byl programem zachycen. Další okno obsahuje při rozkliknutí konkrétní datové jednotky podrobnější informace o využitých protokolech a použitých hodnotách. Poslední okno obsahuje obsah datové jednotky. Obsah může být prezentován různými způsoby. Například může být obsah zakódován v hexadecimální podobě, nebo dekódován do znaků dle ASCII tabulky.

3.2 GNS3

Graphical Network Simulator 3 (GNS3) je opensource síťový simulační a emulační nástroj. GNS3 umožňuje vytváření virtuálních sítí, jak s mezilehlými uzly, tak s koncovými stanicemi. Vzhledem k tomu, že GNS3 není pouze simulátor, na rozdíl například od packet traceru, ale i emulátor umožňuje skutečný tok dat sítí a skutečné chování operačních systémů [50]. Nástroj je tedy vhodný pro studenty zabývající se informačními sítěmi, IT architekty, síťové administrátory, nebo penetrační testery.

GNS3 se skládá z klientské části a serverové. Klientská část představuje grafické uživatelské rozhraní. V uživatelském rozhraní lze pracovat s projekty. Projekty pak mají svoji topologii, kterou lze nastavit v hlavní části programu. Dále se v uživatelském rozhraní nachází nastavení simulátorů, emulátorů a serveru GNS3 [50]. GNS3 server je zodpovědný za běh simulátorů a emulátorů. GNS3 nabízí 3 možnosti běhu

serveru. První možností je local GNS3 server. Tato možnost je vhodná pro malé sítě, ale neumožňuje plný potenciál využití GNS3. Výhodou je, že není zapotřebí GNS3 VM. Druhá možnost je local GNS3 VM. GNS3 VM je virtuální stroj, který se stará o emulaci zařízení. V tomto případě jsou některé zařízení emulovány samotným GNS3 a jiné v GNS3 VM. Tuto možnost GNS3 doporučuje. Poslední možností je remote GNS3 VM. V této variantě se GNS3 nenachází na stejném stroji jako GNS3 klient, ale na vzdáleném stroji. Nevýhodou může být vyšší latence. Výhodami pak jsou možnost vyššího výkonu vzdáleného virtualizačního stroje, nebo možnost více serverů.

3.3 Python

Python je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý koncem 80. let 20. století Guidoem van Rossumem. Na rozdíl od jazyka C nebo C++, které jsou kompilované, je Python interpretovaný. Python tedy není nutné prvně zkompilovat do binárního kódu, ale lze ho spouštět postupně, což usnadňuje vývoj [51].

Přístup interpretovaného jazyka má své výhody a nevýhody. Nevýhodou je větší náchylnost k runtime errorům. Tím, že Python není kompilovaný, nekontroluje kompilátor v kódu datové typy. Oproti kompilovaným jazykům je však velkou výhodou možnost spouštět kód postupně a rychlejší vývoj bez čekání na kompilaci. Program lze testovat i spuštěním jednotlivých příkazů, postupným vypisováním a měněním kódu přímo za běhu [51]. Python je díky své jednoduchosti a rychlosti psaní kódu rozšířený v oblasti informační bezpečnosti.

3.3.1 Scapy

Scapy je interaktivní knihovna pro programovací jazyk Python a *Read Eval Print Loop* (REPL) umožňující manipulaci paketů. Scapy dokáže zakódovat a dekódovat většinu standardních síťových protokolů a následně je zachytávat a odesílat na síťové kartě systému [60]. Díky možnosti vytvářet vlastní pakety a zachytávat příchozí je Scapy užitečný nástroj pro síťové bezpečnostní testování.

3.4 Operační systémy

Důležitou roli hraje v oblastní informační bezpečnosti operační systém. Jedná se o software, který zprostředkovává komunikaci mezi hardwarem a aplikačním softwarem systému [53]. Z hlediska bezpečnosti je to často právě operační systém, který

zařizuje základní síťové funkcionality. Existují různé operační systémy dle případu jejich užití.

3.4.1 Kali linux

Kali linux je linuxová distribuce založená na distribuci Debian zaměřená na penetrační testovaní. Správu a vývoj distribuce zajišťuje firma Offensive Security věnující se informační bezpečnosti. Firma také poskytuje certifikace v oblasti informační bezpečnosti a stará se o databázi zranitelnosti Exploit Database [54].

Kali linux lze nainstalovat na platformu x86, arm, jako linuxový subsystém operačního systému windows, live boot flash disk, cloud, nebo v podobě nethunter na mobilní telefon s android. Další variantou je Kali Purple, což je systém určený pro purple team [54]. Výhodou Kali linuxu oproti ostatním operačním systémům je předinstalace užitečných nástrojů pro penetrační testování. Kali linux není pouze nástroj pro prolamování, ale i užitečná sada nástrojů pro systémové administrátory. Existují i další alternativy operačních systémů předurčených pro penetrační testování jako je ParrotOS, BlackBox, Fedora Security Lab, Black Arch a další. Pro tuto práci byl zvolen Kali linux jako nástroj pro testování a vývoj nástroje pro testování bezpečnostních slabin z důvodu jeho obsáhlé komunity a předchystaným nástrojům.

3.4.2 RouterOS

RouterOS je operační systém od firmy Mikrotik určený pro síťové prvky. Původně firma vytvářela pouze operační systém RouterOS. V roce 2002 však začala vytvářet i vlastní hardware RouterBOARD, na který tento systém dodává [55]. RouterOS je založený na linuxovém jádře a podporuje architektury ARM, ARM64, MIPSBE, MMIPS, SMIPS, TILE, PPC, x86 a Cloud Hosted Router [56]. Systém je licencovaný šesti úrovněmi licencí lišící se funkcionalitou. Mimo standardní licence existuje i trial verze platná 24 hodin a funkčně omezená verze zdarma [57]. RouterBOARD je již dodán s jednou ze standardních licencí.

RouterOS poskytuje řadu základních i pokročilých funkcí mezi které patří konfigurace IPv4 a IPv6, statické i dynamické směrování, nastavení *Virtual Local Area Network* (VLAN), tunelovacích protokolů jako *Point to Point Protocol* (PPP), *Layer 2 Tunneling Protocol* (L2TP), OpenVPN, Wireguard a dalších, bezdrátových síťových rozhraní, přechodné mechanismy mezi protokoly IPv4 a IPv6, front a stavového firewallu [56]. Benefit RouterOS je v jeho obsáhlých konfiguračních možnostech umožňujících pokročilou konfiguraci. Z těchto důvodů byl systém zvolen pro tuto práci jako testované zařízení. Útoky jednotlivých scénářů budou na zařízení potlačovány změnou konfigurace.

3.5 Testovací prostředí

Pro vývoj a testování programu na testování byl vybrán nástroj GNS3. GNS3 je vhodný zejména díky jeho vlastnosti emulace zařízení a tedy prostředí velmi podobného reálným sítím. Testy v jednotlivých scénářích jsou dvou typů. U prvního typu testů je zapotřebí pouze jedno zařízení. Toto zařízení bude vysílat a přijímat data a na jejich základě vyhodnocovat zranitelnosti sítě. V druhém případě je zapotřebí dvou zařízení. Zařízení mezi sebou navážou obousměrný komunikační kanál, pomocí kterého bude hlavní stanice vysílající data posílat příkazy druhému zařízení, které bude naslouchat provozu na síti.

3.5.1 Topologie

V GNS3 byly vytvořeny 3 základní topologie pro testování. Jako první byla vytvořena síť pouze se dvěma hosty. Tato síť není určena pro reálné testování, ale pro vývoj aplikace. Síť se bude využívat v rámci diplomové práce pro testování správné funkčnosti skriptů. Druhá topologie představuje lokální síť. V topologii je směrovač připojený pouze do lokální sítě přepínačem a hosti, kteří jsou propojeni také přepínačem. Topologie bude v rámci diplomové práce sloužit pro testování lokálních IPv6 protokolů. V poslední topologii se nachází 2 sítě propojené směrovačem. Každá síť obsahuje přepínač, ke kterému jsou připojení hosté.

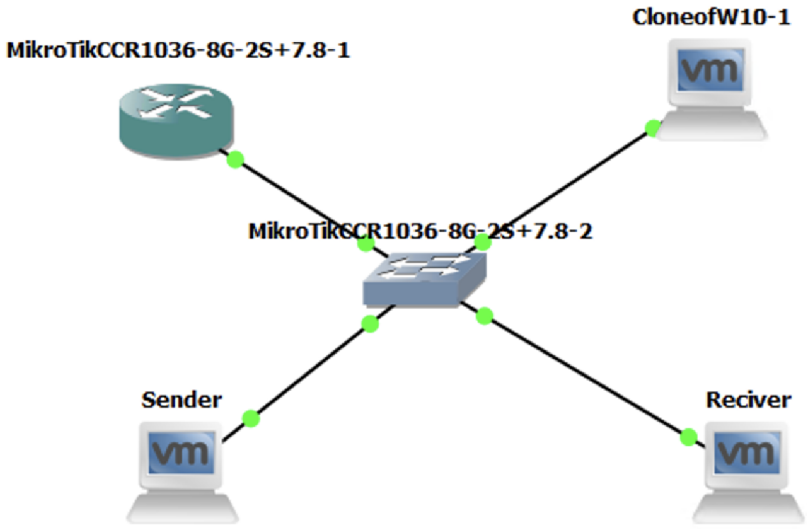
Host to host

Síť byla vytvořena pouze pro účely testování aplikace. Hosti jsou připojeni pouze mezi sebou. Host A zastupuje v této topologii dvě funkcionality. První funkcionalitou hosta je odesílání uměle vytvářených paketů. Druhou funkcionalitou je odesílání příkazů pro hosta B. Host B dostává příkazy od hosta A. Dle příkazu hledá konkrétní paket, případně pakety. V případě, kdy na síťové rozhraní hosta B dorazil předem definovaný paket, host B informuje hosta A. Informování probíhá přeposláním paketu hostovi A prostřednictvím sestaveného komunikačního kanálu. Případně lze hostovi A poslat, zda byl definovaný paket zaznamenán a v jakém počtu.

Single LAN

Topologie se zaměřuje na testování protokolů v rámci lokální sítě. V rámci práce jsou definovány 2 typy testů. První typ potřebuje k testování pouze jednu stanici. Testování probíhá naslouchánim na lince, případně zasláním sestaveného paketu do sítě a následným naslouchánim pro odpověď. Druhý typ potřebuje ke své funkcionalitě dvě stanice. První odesílá předem připravené pakety a druhá naslouchá na

síti zda jsou pakety doručeny. Následně stanice informuje vysílač. Dle použitého scénáře je následně vyhodnoceno, zda je síť zranitelná na testovanou zranitelnost. Tuto topologií znázorňuje obrázek 3.1. Při testování přepínače jsou v topologii využity pouze zařízení sender, reciever a switch.



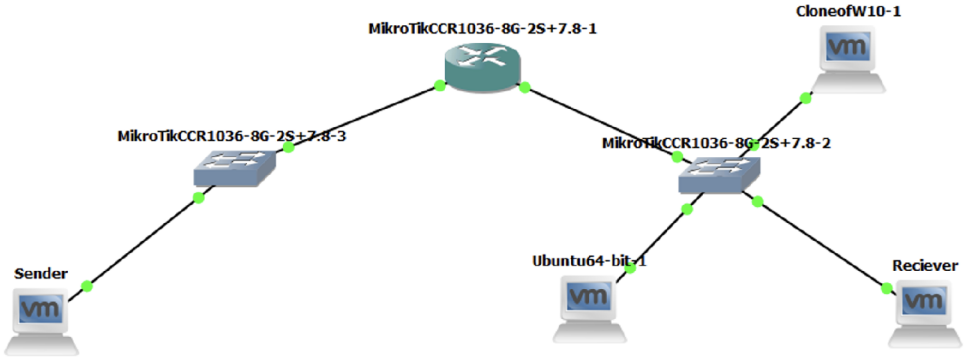
Obr. 3.1: Schéma topologie LAN sítě testující přepínač

Dvě LAN sítě

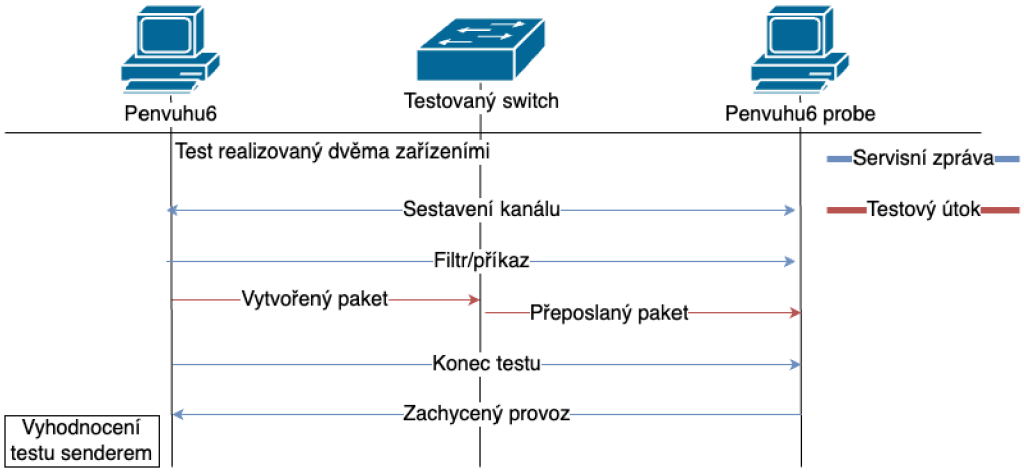
Topologie dvou sítí připojených pomocí směrovače umožňuje testovat zranitelnosti zařízení, špatnou konfiguraci směrovače a další mezisíťové útoky. Opět mohou být v rámci topologie použity 2 typy testů. První typ opět využívá pouze jednu stanici, která vysílá sestavené pakety a naslouchá pro jejich odpověď. V případě dvou stanic je opět jedna v roli vysílače a druhá v roli přijímače. Zařízení mají mezi sebou sestavený komunikační kanál, který musí být povolen na směrovači. V případě testování routeru vyžaduje pouze stanice sender, reciever a samotný router. Pro simulaci reálné sítě bylo do topologie zapojeny i další zařízení viz obr. 3.2

3.5.2 Schéma testů

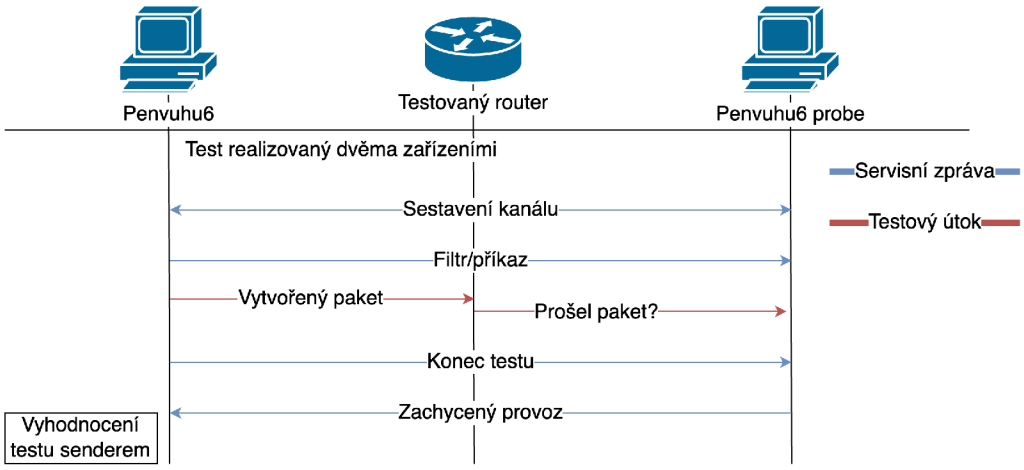
V případě testování pouze jedním zařízení je po spuštění programu vždy zahájen test na konkrétní zranitelnost. Tento test následně vyšle "n" paketů a zároveň naslouchá na příchozí pakety. Následuje vyhodnocení testu a spuštění dalšího testu v pořadí. Po posledním testu je vypsán seznam nalezených zranitelností, možné dopady a doporučení jejich mitigace.



Obr. 3.2: Schéma topologie dvou spojených LAN sítí testující směrovač



Obr. 3.3: Schéma komunikace testu LAN sítě



Obr. 3.4: Schéma komunikace testu dvou LAN sítí

V případě testování dvěma zařízeními je po spuštění vysílače ustanoveno spojení mezi vysílačem a přijímačem. Následuje spuštění testu. Na začátku vysílač odešle přijímači nastavení práce s pakety. Přijímač může pakety zasílat přijímači přes ustanovený kanál, zasílat pouze vybrané pakety přijímači, případně zaslat informaci, kolik paketů splňující podmínku bylo přijato, nebo přijímač vyšle paket potřebný pro test. Následně vysílač vyšle n paketů. Po skončení testu vysílač zašle tuto informaci přijímači a vyžádá od něj výsledek testu, na základě kterého test vyhodnotí. Po vyhodnocení vysílač spouští další testy. Po skončení posledního testu je vypsán seznam nalezených zranitelností, možné dopady a doporučení jejich mitigace. Příklad komunikace je vyobrazen pro test přepínače na obr. 3.3 a pro test směrovače na obr. 3.4.

4 Vývoj nástroje pro IPv6 testování

V rámci diplomové práce byl vyvinut nástroj pro testování bezpečnostních zranitelností sítí využívajících protokol IPv6. Tento nástroj byl v rámci diplomové práce nazván `Penvuhu6`. `Penvuhu6` umožňuje testovat lokální i vzdálené sítě a to koncové stanice i síťové prvky.

Nejčastějším způsobem testování IPv6 sítí je modifikace pole, nebo polí na některé z vrstev TCP/IP modelu. Následně je zkoumáno, jak je s takovým paketem zacházeno z pohledu síťových prvků i koncových stanic. Pro tyto účely nástroj `Penvuhu6` umožňuje generaci libovolného paketu a zachycení na lokální instanci programu `Penvuhu6` nebo vzdálené sondě. Při využití vzdálené sondy je sestaveno servisní IPv6/TCP spojení se vzdálenou sondou.

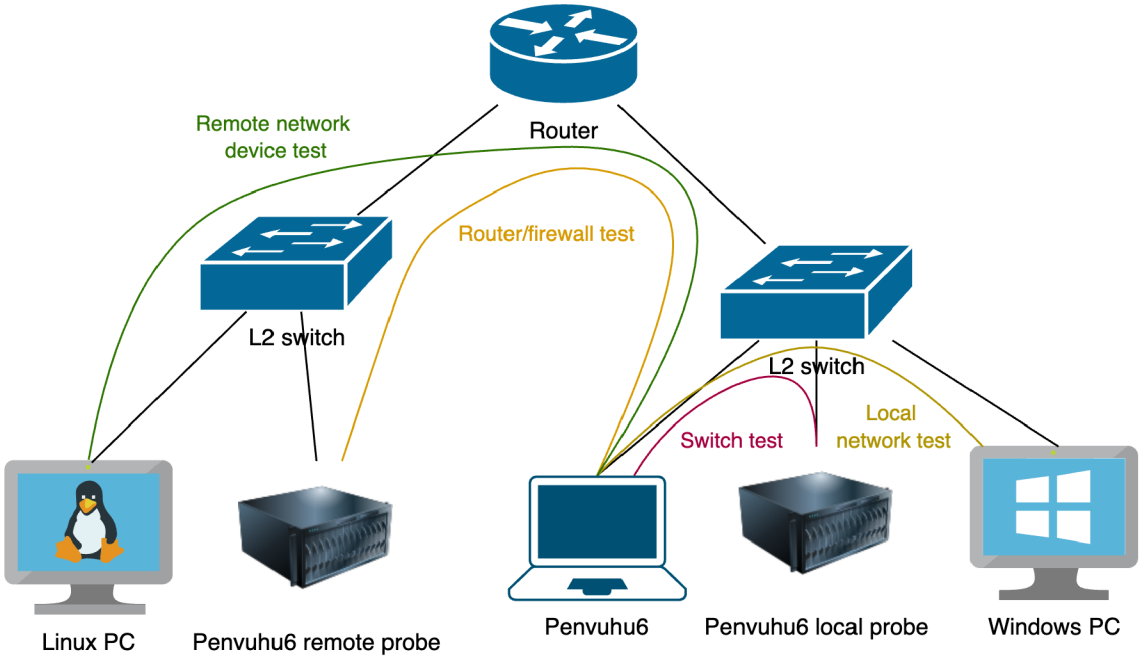
4.1 Architektura

Vzhledem k různorodosti zranitelností a skutečnosti neustálého objevování nových zranitelností byl nástroj `Penvuhu6` navržen pro podporu co největší míry univerzálnosti. Od této myšlenky se odvíjí i architektura nástroje a bylo myšleno na možnost přidání vlastních testů a jejich různorodost. Nástroj lze tedy používat samostatně, nebo za pomocí sond, které zvyšují možnost detekce zranitelností.

V plném zapojení se nástroj skládá z hlavního testovacího zařízení `Penvuhu6` znázorněného na obr. 4.1, lokální sondy `Penvuhu6` local probe a vzdálené sondy `Penvuhu6` remote probe. Pomocí zapojení pouze s hlavním testovacím zařízením lze testovat konkrétní zařízení z hlediska jejich reakce na vyslané zprávy ze zařízení `Penvuhu6` (Remote network device test a Local network test na obr. 4.1). Zapojením s lokální a vzdálenou sondou lze pak dosáhnout testových případů, kdy je zkoumán průchod sítí chybných a nebezpečných paketů (Router/firewall test a Switch test na obr. 4.1). Při použití sond musí být pro cestu sítí povolen libovolný port protokolu TCP.

Nástroj `Penvuhu6` se skládá z několika souborů, přičemž dle nasazení nástroje není vždy třeba všech souborů. Pro nasazení sondy jsou například potřeba pouze soubory message.py, ipv6connection.py a probe.py dle obr. 4.2. Konkrétní závislost je však závislá na využitých souborových testech 4.4. Ty lze vytvářet dvěma způsoby a to ve formátu JSON 4.4.1, nebo využít funkcionality nástroje `Penvuhu6` pro napsání vlastního testu v jazyku Python 4.4.2. Soubory obsažené v nástroji `Penvuhu6` a jejich účel je následující:

- penvuhu6.py - Hlavní program nástroje `Penvuhu6`. Obstarává výpis menu, definici testů, kontrolu vstupních parametrů testů, jejich spuštění a zpracování výsledků.

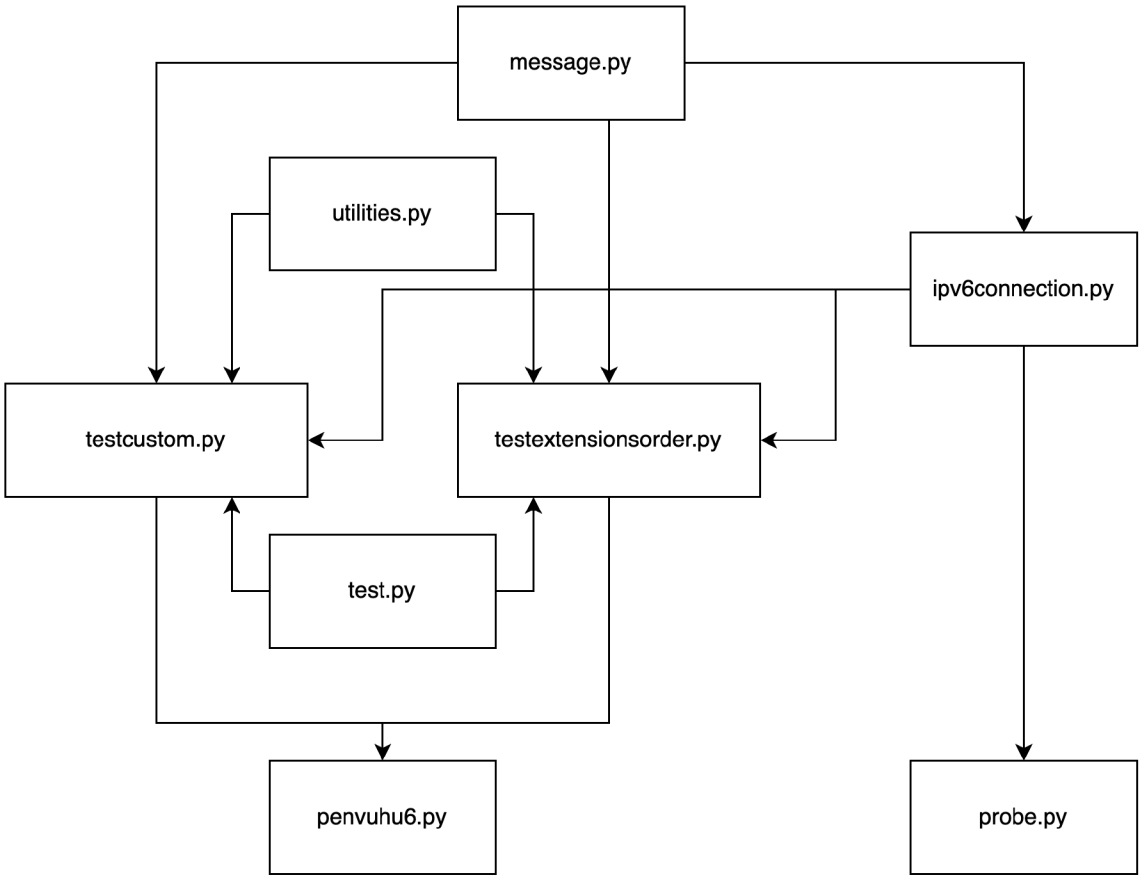


Obr. 4.1: Příklad síťového zapojení nástroje Penvuhu6

- probe.py - Hlavní program sondy `Penvuhu6`. Obstarává výpis menu sondy, spuštění síťového soketu, příjem a zpracování servisních zpráv, zachytávání a odeslání zachyceného síťového provozu.
- ipv6connection.py - Slouží pro zpracování požadavků pro sondu a definuje metody pro odeslání a příjem dat.
- message.py - Obsahuje definici formátu zpráv.
- test.py - Obsahuje rodičovskou třídu pro všechny testy.
- utilities.py - Obsahuje mezi testy sdílené funkcionality pro testování.
- testcustom.py - Obsahuje třídu zpracovávající uživatelské testy.
- testextensionsorder.py - Testový soubor definující třídy pro testování pořadí a repetetivnost záhlaví.
- testfragmentoverlap.py - Testový soubor definující třídy pro testování překrývajících se fragmentů.
- testRA_DHCPv6_guard.py - Testový soubor definující třídy pro testování RA guard a DHCPv6 guard.

4.2 Používání nástroje Penvuhu6

Pro spuštění programu je nejprve potřeba připravit prostředí. V případě operačního systému Kali linux by měly být již všechny potřebné nástroje připraveny. Pro jiné systémy může být potřeba nainstalovat interpretr programovacího jazyka Py-



Obr. 4.2: Souborová závislost nástroje Penvuhu6

thon, správce balíčků pro Python Pip, nebo potřebné knihovny definované v souboru *requirements.txt*. Ty lze díky nástroji Pip jednoduše nainstalovat pomocí příkazu *pip install -r requirements.txt*.

Program interaguje se síťovou kartou včetně zachytávání přijatých paketů, proto je třeba spouštět hlavní program pomocí příkazu *sudo*, který spouští program pod uživatelem root, tedy administrátor systému. Pro spuštění je tedy třeba provést pomocí příkazu *sudo python penvuhu6.py*. V případě přítomnosti vícero verzí Python je vhodné vybrat konkrétní interpreter. Některé verze Kali linux také nemusí obsahovat správnou verzi knihovny Scapy, je tedy vhodné doinstalovat verzi pomocí správce balíčků pip. Po zadaní příkazu je vypsána nápověda programu (obr. 4.3) pomocí knihovny ptlibs [58], která je součástí projektu Penterep.

Nápověda obsahuje popis účelu programu, způsob jakým se program používá včetně příkladu konkrétního použití, seznam přepínačů a předvytvořených testů. Povinným parametrem je interface, který má `Penvuhu6` využívat a odpovídá názvu síťového adaptéru v operačních systémech založených na jádru linux. Po názvu rozhraní následuje žádný až všechny vypsané testy, případně lze využít přepínače *-R* pro výběr všech testů se vzdálenou sondou a *-L* pro výběr všech testů s lokální son-

Obr. 4.3: Nápověda nástroje Penvuhu6

dou. V případě, kdy není definován žádný test, se předpokládá, že uživatel využívá vlastní testy ze souborů JSON, které lze kombinovat i s vytvořenými testy. Custom testy se zadávají pomocí přepínače *–custom*, který je nutné zadat jako poslední. Po tomto přepínači následuje seznam souborů respektující "Unix style pathname pattern expansion"a lze tedy využít jak výpis jednotlivých souborů, tak i wildcard

masky jako je *"*"*, nebo *"?"* a další [59]. Validní jsou však pouze soubory *".json"*.

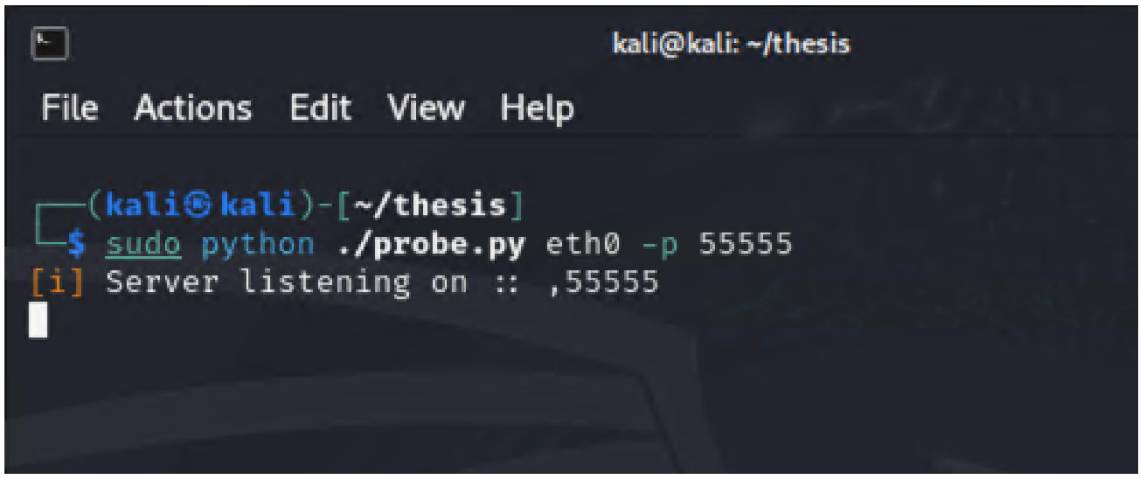
Jak již bylo zmíněno nástroj `Penvuhu6` se skládá z hlavního programu a ze sond. Hlavní program jako první při spuštění kontroluje, zda jsou zadané parametry správě zadané. Existují tři druhy testů typu Python a dva druhy testů typu JSON. Typy Python testů jsou remote probe, local probe a others. Testy v kategorii remote probe se připojují na sondu definovanou pomocí přepínačů *-raddr* a *-rport*. Podobně pro local probe testy je sonda definována přepínači *-laddr* a *-lport*. Před spuštěním testů vyžadující sondu je třeba ji spustit na vzdáleném stroji. Sonda nemusí obsahovat všechny součásti hlavního testeru. Potřebné soubory jsou dle obr. 4.2 *probe.py*, *ipv6connection.py* a *message.py*, je však potřebné rovněž nainstalovat knihovnu Scapy a ptlibs [58]. Spuštění sondy se provádí obdobně jako spuštění hlavního nástroje `Penvuhu6`. Sonda potřebuje ke své funkčnosti rovněž naslouchat na síťovém rozhraní a tak je třeba opět program spustit jako uživatel root pomocí příkazu *sudo*. Vypsání nápovědy sondy je provedeno pomocí příkazu *sudo python probe.py* viz obr. 4.4. Sonda opět musí mít nastaveno síťové rozhraní, na kterém naslouchá podobně jako je tomu u hlavního programu. Dále může být nastavený libovolný port, nebo omezení naslouchá pro příchozí spojení pouze pro konkrétní IPv6 adresu sondy. Po spuštění sonda vypíše adresu, na které naslouchá spolu s TCP portem viz obr. 4.5. Tento port musí být na síťových zařízeních mezi sondou a `Penvuhu6` povolen po celou dobu testování.

Poslední dva přepínače v nápovědě `Penvuhu6` jsou *-dnppp* a *-dnppd*. První přepínač znamená *Do Not Print Passed Packets* a vypíná zobrazení paketů, které `Penvuhu6` považuje za nevalidní a jejich příchod považuje jako indikaci zranitelnosti. Druhý přepínač představuje *Do Not Print Passed Difference* a umožňuje uživateli vypnout konzolové porovnání rozdílů odeslaného a přijatého paketu.

4.3 Princip nástroje

Nástroj `Penvuhu6` pracuje především na principu porovnávání odeslaných a přijatých paketů. Při jeho spuštění jsou jako první zpracovány vstupní parametry a jejich formát. Pokud uživatel zadal některý z parametrů chybně, nástroj vypíše nápovědu. Pokud je vše v pořádku, `Penvuhu6` vytvoří instance testových tříd dle vložených parametrů a následně v těchto třídách spustí metodu *run()* viz obr. 4.6a. Testy se liší dle jejich typu. V případě testu s pomocí sondy jsou prvně připraveny testové pakety. Po jejich přípravě se `Penvuhu6` připojí k sondě a nastaví sondu pro naslouchání, případně filtr pro zachytávání paketů a vyšle testové pakety pro sondu. Po ukončení odesílání pakety `Penvuhu6` ukončí zachytávání sondy a příjme servisním kanálem pakety, které sonda přijala. Následuje odpojení sondy. Přijaté pakety jsou následně opět filtrovány a porovnávány s odchozími pakety. V případě, kdy byl paket po

Obr. 4.4: Nápověda sondy Penvuhu6



Obr. 4.5: Spuštěná sonda čekající na spojení

cestě změněn, se tato změna vypíše, ovšem některá převážně dynamická pole, jako je hop limit `Penvuhu6` záměrně nevypisuje pro přehlednost. Druhým případem je test, který nepotřebuje sondu. V takovém případě jsou opět v první fázi připraveny

pakety k odeslání. Dále je zapnuto zachytávání na rozhraní `Penvuhu6` a odeslány pakety. Po časové prodlevě je zachytávání ukončeno a `Penvuhu6` porovná, které pakety souvisí s odeslanými. Následně je vypsán výsledek testu. Po dokončení všech testů je vypsáno shrnutí všech testů a zda byly úspěšné či nikoliv. Dále je v případě nalezených zranitelností vygenerován *.pcap* soubor s pakety, které neměly být propuštěny. Vývojový diagram obou případů testů je znázorněn na obrázku 4.6b.

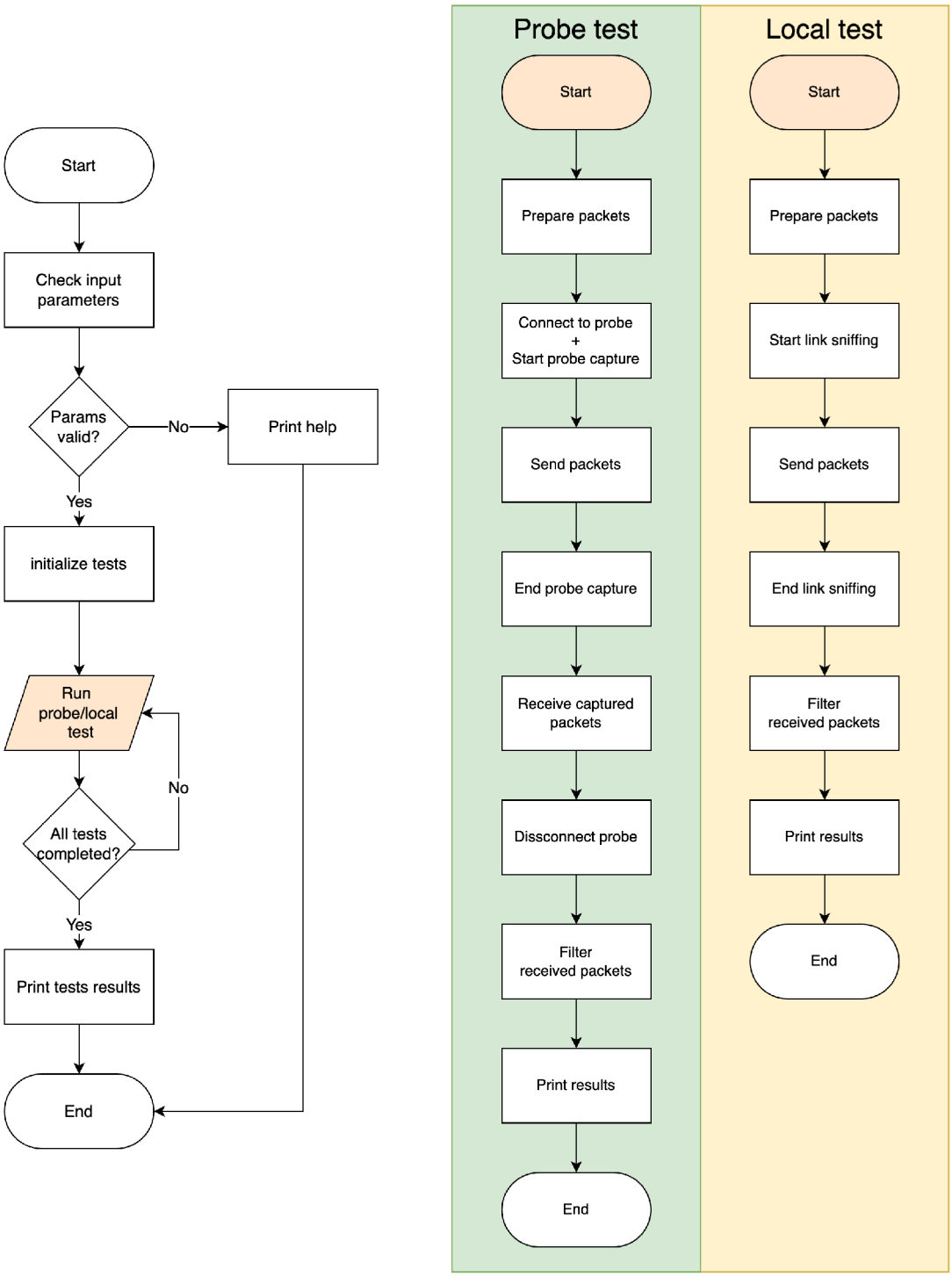
Podobně jako je tomu u `Penvuhu6` i sonda na začátku porovnává vstupní parametry a v případě neúspěchu vypíše uživateli nápovědu. V případě, že jsou parametry validní, začne sonda naslouchat na definovaném rozhraní pro příchozí spojení. Při spojení sonda spustí podprogram, který obstarává komunikaci s připojeným klientem. Ten má na výběr ze šesti operací. První operací je spuštění zachytávání síťového provozu na rozhraní odpovídající operaci *start*. Pro některé může být vhodné omezit přeposílání paketů servisním kanálem a tak lze nastavit *filter*, kterým sonda před odesláním vyřadí nepotřebné pakety. Tento filtr lze resetovat pomocí operace *reset*. Některé testy potřebují větší spolupráci se sondou a lze tak sondě zaslat paket, který by měla vyslat pomocí operace *send*. Tato funkce je užitečná při testování DHCPv6, nebo SLAAC, kdy umožňuje vyslat ze sítě solicit zprávy, na které následně `Penvuhu6` odpoví. Předposlední operace *stop* zastavuje zachytávání na rozhraní a po aplikování filtru odesílá přijaté pakety servisním kanálem zpět k `Penvuhu6`, který je dále zpracovává. Pokud je test u konce, `Penvuhu6` se odpojí od sondy pomocí operace *exit*, která zastaví spuštěná zachytávání a ukončí spojení. Podprogram je tímto bodem ukončen a sonda je opět připravena na další spojení. Pro ukončení sondy lze zadat *ctrl + c*. Hlavní program je znázorněn na obrázku 4.7a a podprogram sondy představuje obrázek 4.7b.

4.4 Tvoření testů

Nástroj `Penvuhu6` byl vytvořen tak, aby bylo možné jej rozšířit o testové případy. Testy jsou definovány dvěma způsoby. Prvním způsobem je vytvoření JSON souboru, který následně nástroj přečte a dle definice vytvoří testové pakety. Druhým způsobem rozšíření testů je vytvoření testu v programovacím jazyce Python a využití funkcionalit nástroje `Penvuhu6`.

4.4.1 Vlastní testy JSON

Nejjednodušším způsobem pro tvoření nových testů je využití souborů JSON. Struktura testového JSON je znázorněna ve výpisu 4.1 a obsahuje atributy *name*, *description*, *type*, *packets*. Atribut *name* obsahuje jméno testu, které je použito při výpisech a ukládání výstupního *.pcap* souboru. Atribut *description* obsahuje popis testu, který

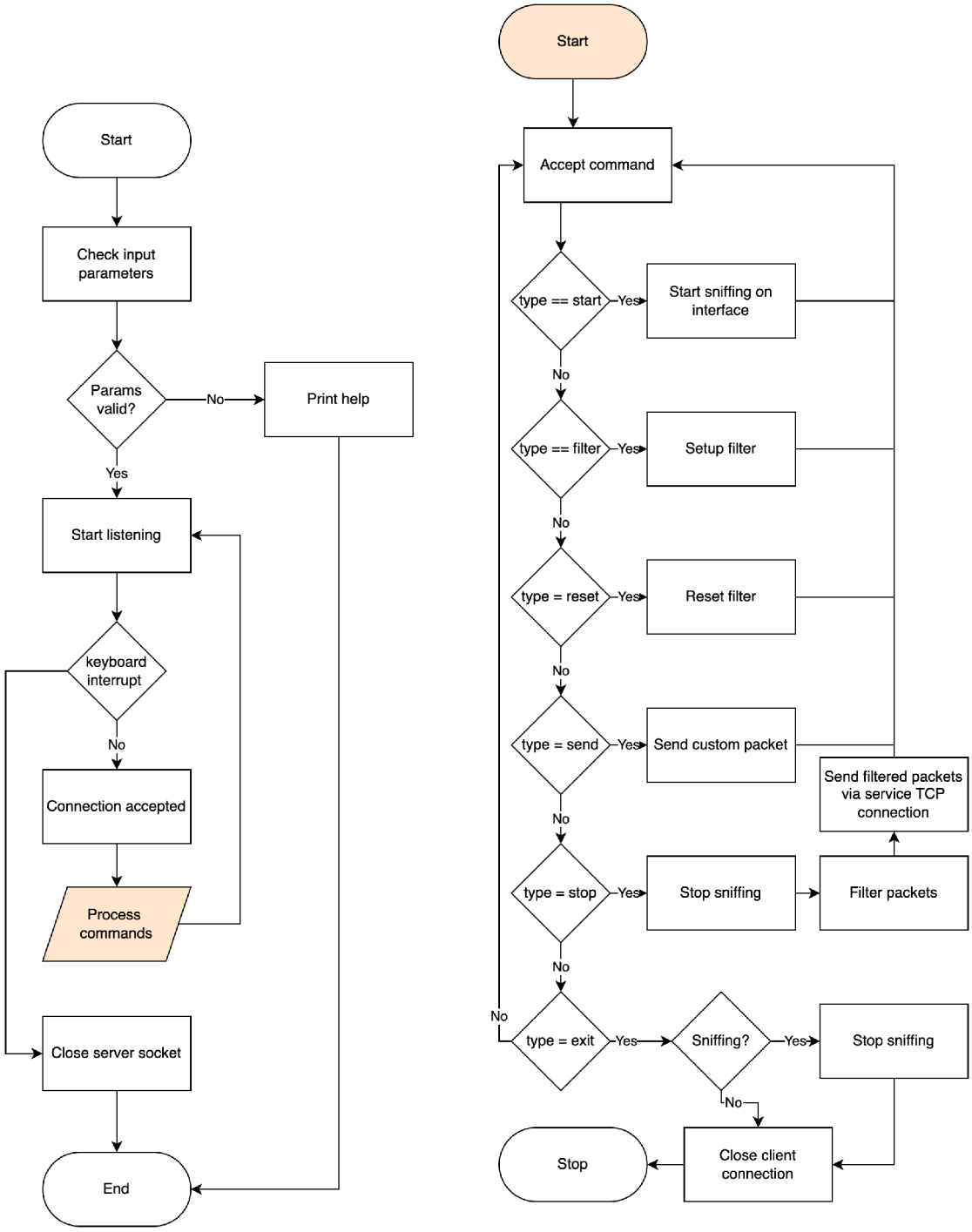


(a) Vývojový diagram Penvuhu6

(b) Vývojový diagram lokálního testu a testu se sondou

Obr. 4.6: Vývojové diagramy hlavního nástroje Penvuhu6

je vypsán při spuštění. Atribut *type* definuje, zda se jedná o lokální test, nebo test se sondou (*probe*). Posledním a nejdůležitějším atributem je *packets*. Tento atribut



(a) Vývojový diagram sondy (probe)

(b) Vývojový diagram operací sondy

Obr. 4.7: Vývojové diagramy sondy

obsahuje pole paketů, které budou v rámci testu vyslány.

Paket je tvořen následujícími atributy *type*, *attributes*, *filter* a *payload*. Atribut *type* odpovídá třídě paketu dle knihovny scapy. Díky tomuto způsobu je zajištěna

vysoká míra customizace, jelikož tímto způsobem jdou vytvořit všechny třídy, již jsou potomky třídy Scapy obsažené ve *scapy.all* [60]. Atribut *attributes* obsahuje atributy třídy, které jsou rovněž definovány v knihovně Scapy. Dalším atributem je *filter*, který na každé vrstvě udává, která pole budou využita pro filtrování, zda paket přišel ve formě textového řetězce odděleného mezerami, může se tedy například jednat o flow label (*fl*). Pokud filter není vůbec přítomen, není vyžadováno při filtraci toto záhlaví, pokud je filter přítomen, ale je prázdný, kontroluje se pouze záhlaví. Pokud je obsažen filter i atributy, kontroluje shoda atributů i záhlaví. Posledním atributem je *payload*, který obsahuje další vnořená záhlaví, rovněž dědící ze třídy packet. V případě, že je atribut třídy *options*, musí být atribut zadán jako pole obsahující jednotlivé *options* s atributem *type* odpovídající názvu třídy v knihovně scapy a *attributes* odpovídající atributu dané třídy 4.2. Příklad celého souboru je uveden v příloze A.

```
{
    "name": "Test_name",
    "description": "Test_description",
    "type": "probe",
    "risk": "Risk␣of␣vulnerability",
    "recommendation": "Advice␣to␣mitigation",
    "packets": [
    ]
}
```

Výpis 4.1: Struktura JSON testu

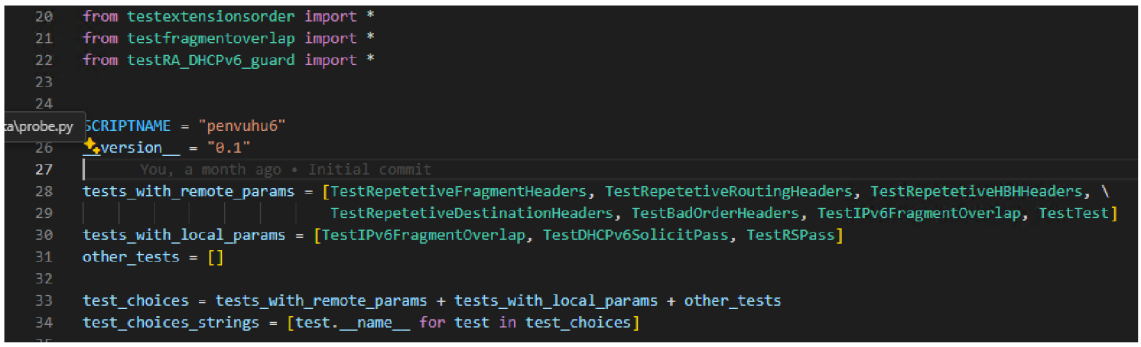
```
{
    "type": "IPv6",
    "attributes": {
        "src": "2001:db8:85a3::8a2e:370:7334",
        "dst": "2001:2::5cfa:ce92:769d:f88e",
        "fl": 67673
    },
    "filter": "src␣dst␣fl",
    "payload": {
        "type": "IPv6ExtHdrHopByHop",
        "attributes": {
            "options": [
                {
                    "type": "PadN",
                    "attributes": {
                        "optdata": "test"
                    }
                }
            ]
        }
    }
}
```

Výpis 4.2: Struktura paketu v JSON testu

4.4.2 Python test

V případě vytvoření vlastního testového souboru v jazyce Python je třeba daný soubor přidat do `Penvuhu6`. To lze zajistit importováním daného souboru pomocí *from <nazev_souboru> import ** a následně přidáním dané testové třídy do Python listu *tests_with_remote_params*, *tests_with_local_params*, nebo *other_tests* podobně jako je znázorněno na obrázku 4.8.

Třída testu musí v případě varianty se sondou obsahovat konstruktor s argumenty *probe_address*, *probe_port* a textitoptions. V případě testu bez sondy jsou *probe_address* a *probe_port* vynechány. Třída by měla dědit třídu *Test*, ale lze spustit test i bez této vazby. Konstruktor (metoda ___init___) by měla zmíněné argumenty nastavit jako self. Dále musí testová třída obsahovat metodu *run()*. Tato metoda nesmí brát žádné parametry a vše potřebné by již měla mít nastaveno z konstruktoru a případně z dictionary *options*. Tělo třídy *run* může být imple-



Obr. 4.8: Definice testů v souboru penvuhu6.py

mentováno dle uvážení autora testu, avšak je možné kód dělit jak je naznačeno metodou *generate_packets* na obrázku 4.9, kde jsou vytvářeny záhlaví a následně jsou využity funkce ze souboru *utilities.py*. Více příkladů viz příloha A. Metoda *add_transport_layer* přiřazuje k vytvořeným záhlavím transportní protokol a funkce *uni_run* se stará o vytvoření IP záhlaví s unikátním flowlabel, který je následně využit jako filtr. Každá metoda *run* musí vracet tři hodnoty, a to textový řetězec *(string)* obsahující název testu, pravdivostní hodnotu *(bool)* obsahující výsledek a textový řetězec *(string)* obsahující název souboru se zachycenými pakety.

```
class TestExample(Test):
    description = "\tThis is simple test for connection testing. Sends 4 packets \
        with destination options extension header."
    code = r""" ···
    long_description = f""" ···
    risk = "There is no risk if this test pases as it is just verification of the \
        connection between the test device and theprobe."
    recomendation = "There is no recomendation for this test as it is just verification \
          of the connection between the test device and the probe."

    def __init__(self, probe_address, probe_port, options={}) -> None: ···

    def run(self):
        def generate_packets():
                headers = []
                headers.append(IPv6ExtHdrDestOpt())
                headers.append(IPv6ExtHdrDestOpt())
                headers.append(IPv6ExtHdrDestOpt())
                headers.append(IPv6ExtHdrDestOpt())
                return headers # return header list

        udp = UDP(sport=get_dynamic_port_number(), dport=53)
        tcp = TCP(sport=get_dynamic_port_number(), dport=80)
        fce = lambda : add_transport_layer(
            (generate_packets()), [udp, tcp]) # transport layer addition
        result, filename = uni_run(fce, self.__class__.__name__, self.probe_address,
                                   self.probe_port, options=self.options)
        return self.__class__.__name__, result, filename,  self.risk , self.recomendation
```

Obr. 4.9: Příklad implementace vlastní testové třídy

5 Scénáře

V rámci diplomové práce byly zpracovány tři testové scénáře, pro různé typy útoků. První ze scénářů se zaměřuje na propustnost firewallu nevalidních paketů s repetetivním, nebo špatně seřazeným pořadím rozšiřujících záhlaví. Druhý scénář se zaměřuje na překrývající se fragmenty a jejich propuštění skrz síťový prvek. Poslední ze scénářů se věnuje problematice Router advertisement a DHCPv6 advertisement guard. Tyto scénáře byly zpracovány pomocí `Penvuhu6` a jeho variantě testů v Python. Detekční mechanismus zranitelnosti pracuje primárně se skutečností, zda paket dorazil, či nedorazil na naslouchající zařízení, v případě kdy je paket při cestě sítí zahozen z jiného důvodu, než je chování a nastavení síťových prvků, nemusí být zranitelnost správně detekována. Například při zahození paketu z důvodu chyby sítě nemusí být přítomná zranitelnost detekována. Ve scénářích jsou použity třídy UDP a TCP ve výchozím nastavení knihovny Scapy. Třída TCP se ve výchozím nastavení snaží navázat spojení pomocí příznak SYN, je tedy možné zahození paketu z důvodu vyhodnocení SYN flood. Naopak v případě, kdy není nastavena hodnota SYN nemusí paket propustit stavový firewall, který o spojení nemá informace. V případě potřeby lze tuto hodnotu přepsat ve funkci *generate_packets* u jednotlivých testů. Nástroj `Penvuhu6` pro všechny testy využívající vytvořené funkce generuje *.pcap* soubory obsahující všechny pakety považované testem za problematické.

5.1 Extension header scénář

Scénář testuje, jak síťový prvek nakládá s pakety, které nerespektují normu RFC 8200 [23], která udává, v jakém pořadí by měly být zapouzdřené rozšiřující záhlaví IPv6. Vzhledem k povaze testu nemá smysl testovat, zda tyto pakety propouští zařízení pracující na 1. vrstvě modelů TCP/IP. Test se tedy zaměřuje prvky pracující na druhé vrstvě TCP/IP a jedná se především o routery a firewally. Testy se zaměřují zejména na opakování záhlaví, nebo přeuspořádání pořadí rozšiřujících záhlaví nevyhovující RFC 8200. RFC 8200 také udává povinnost přijmutí a pokusu o zpracování jakémukoliv uzlu IPv6, avšak z bezpečnostního hlediska lze takové pakety považovat za nebezpečné, jelikož umožňují potenciálnímu útočníkovi zpomalit, zahltit, obejít, případně znepřístupnit směrovač, nebo firewall. Pro předejití těmto útokům je vhodné, aby síťové prvky takové pakety zahazovaly [40].

5.1.1 Repetetivní fragment extension header

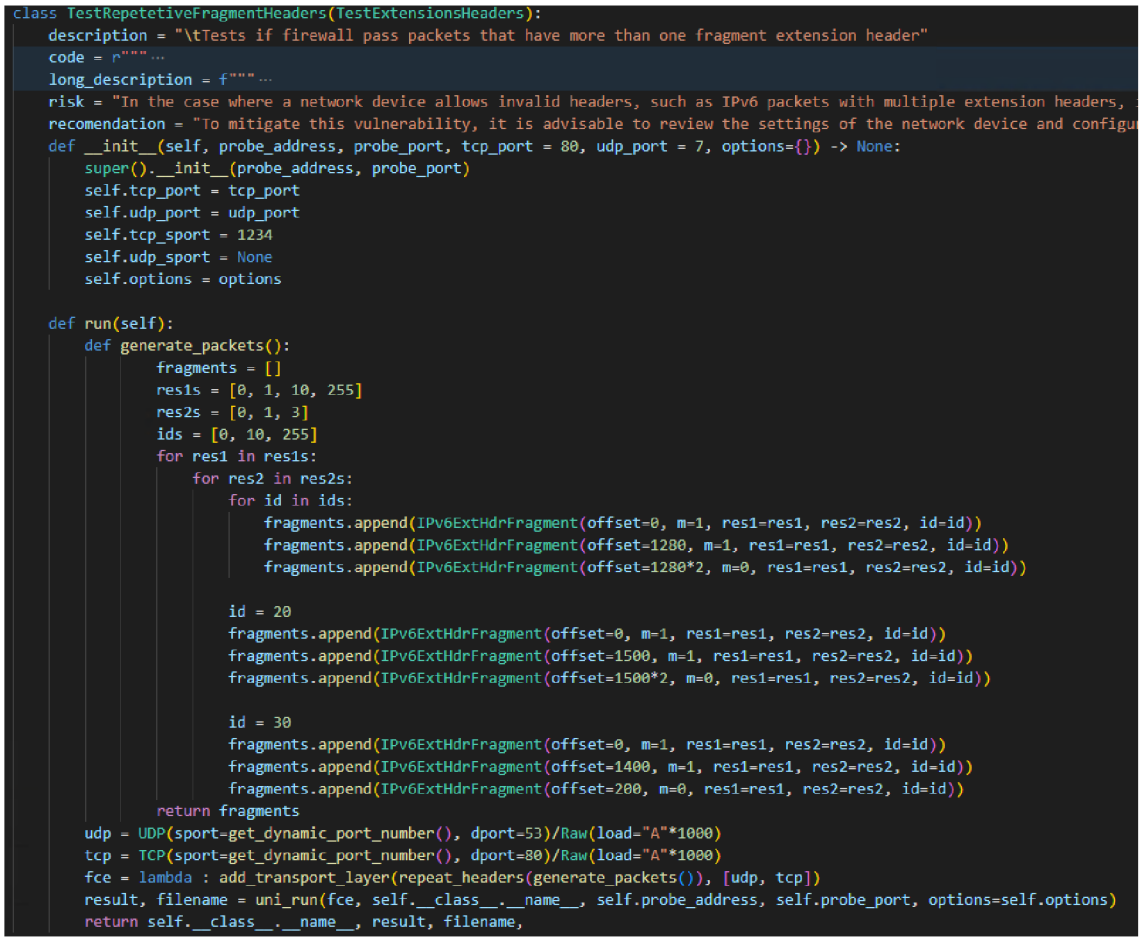
Rozšiřující záhlaví Fragment header v IPv6 oproti fragmentování IPv4 probíhá pouze jednou a v případě, že je třeba po trase menší *Maximum transmission unit* (MTU),

je packet zahozen a vrácena ICMPv6 zpráva o důvodu zahození. Vzhledem k tomuto principu je nesmyslná kombinace dvou a více fragmentačních záhlaví IPv6 a tento stav je chybný. Pokud pakety obsahující dvou a vícenásobné záhlaví Fragment header prochází přes síťový prvek, znamená to, že tento prvek dostatečně nekontroluje procházející pakety. Představuje tak potenciální hrozbu v případě, kdy je takový paket doručen stanici, která nepředpokládá tento chybný stav.

Pro testování opakujících se fragmentačních záhlaví byla v rámci nástroje `Penvuhu6` vytvořena třída *TestRepetetiveFragmentHeaders*. Část generace fragmentačních paketů je vyobrazena na obrázku 5.1. Vzhledem k tomu že není možné otestovat všechny možné varianty, byly vybrány případy, které by mohly software síťového prvku zmást. Ve funkci *generate_packets* jsou generovány záhlaví obsahující hodnoty 0, 1, 10 a 255 v prvním rezervovaném poli a 0, 1 a 3 ve druhém rezervovaném poli. Dále se mění pole identifikace na hodnoty 0, 10, 20, 30 a 255. Tyto hodnoty byly vybrány náhodně. Po generaci fragment záhlaví jsou tato záhlaví vložena za sebe pomocí funkce *repeat_headers*. Následně jsou fragmenty rozšířeny o transportní protokol UDP, nebo TCP a protokol IPv6 s vybranou hodnotou pole *flow label* pro jednodušší filtrování paketu. Funkce *uni_run* je univerzální funkcí pro provedení základního testu dle vývojových diagramů v kapitole 4.3. Výsledek testu je reprezentován informační výpisem viz obr. 5.3 a záznamem přijatých paketů obr. 5.2. V tomto případě bylo odesláno 800 a přijato 334 paketů.

5.1.2 Repetetivní Routing extension header

Test *TestRepetetiveRoutingHeaders* generuje podobně jako test *TestRepetetiveFragmentHeaders* rozšiřující záhlaví, tentokrát však Routing header. Jako délky testovaných záhlaví byly vybrány hodnoty 0, 2 a 255. Hodnota 0 je vybrána, protože takto dlouhý Routing header by neměl vůbec existovat a testuje tak, jak se síťový prvek zachová při tomto chybném stavu. Hodnota 2 je validní hodnota pro délku Routing header. Poslední hodnota 255 je maximální možná délka rozšiřujícího záhlaví. Dále byly vybrány typy rozšiřujícího záhlaví Routing header 0, 2, 7, 50 a 255. Hodnota 0 je typ Source Route, který je již ve stavu depricated. Typ 2 dopovídá Type 2 Routing Header a jedná se o validní hodnotu. Typy 7 a 50 jsou z rozsahu nepřiřazených typů. Poslední hodnota 255 odpovídá rezervovanému typu Routing header. Pole segments left je nastaveno na hodnoty 0 pro žádný následující segment, nebo 1, 2, 255 odpovídající zbývajícím segmentům. Hodnoty 1 a 2 mohou odpovídat reálným stavům. Hodnota 0 je odlišná od ostatních a proto je testována stejně jako maximální hodnota pole 255. Dále byly nastaveny home address jako 2001:2::5cfa:ce92:769d:f88e, 2001:5::5cfa:ce92:769d:f88e a ::1. Hodnota ::1 odpovídá localhost adrese. Ostatní hodnoty byly vybrány náhodně. Kombinací všech těchto



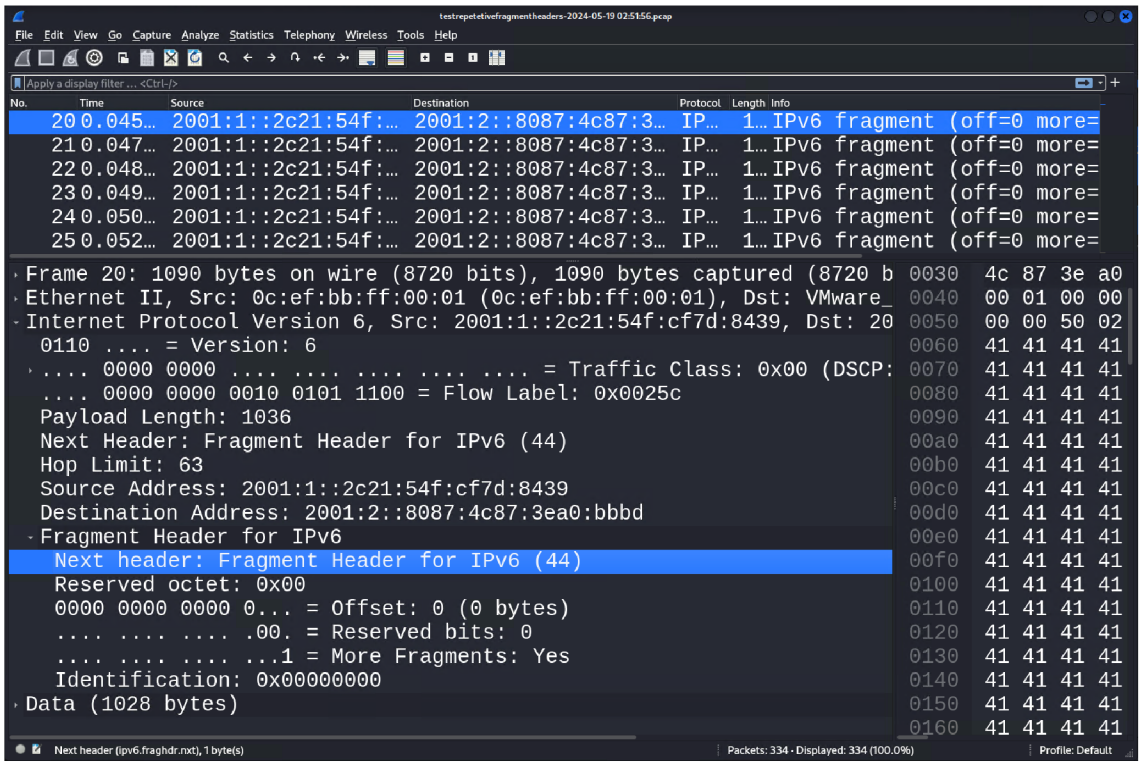
```
class TestRepetetiveFragmentHeaders(TestExtensionsHeaders):
    description = "\tTests if firewall pass packets that have more than one fragment extension header"
    code = r"""...
    long_description = f"""...
    risk = "In the case where a network device allows invalid headers, such as IPv6 packets with multiple extension headers,
    recomendation = "To mitigate this vulnerability, it is advisable to review the settings of the network device and configu
    def __init__(self, probe_address, probe_port, tcp_port = 80, udp_port = 7, options={}) -> None:
        super().__init__(probe_address, probe_port)
        self.tcp_port = tcp_port
        self.udp_port = udp_port
        self.tcp_sport = 1234
        self.udp_sport = None
        self.options = options

    def run(self):
        def generate_packets():
                fragments = []
                res1s = [0, 1, 10, 255]
                res2s = [0, 1, 3]
                ids = [0, 10, 255]
                for res1 in res1s:
                    for res2 in res2s:
                        for id in ids:
                            fragments.append(IPv6ExtHdrFragment(offset=0, m=1, res1=res1, res2=res2, id=id))
                            fragments.append(IPv6ExtHdrFragment(offset=1280, m=1, res1=res1, res2=res2, id=id))
                            fragments.append(IPv6ExtHdrFragment(offset=1280*2, m=0, res1=res1, res2=res2, id=id))

                        id = 20
                        fragments.append(IPv6ExtHdrFragment(offset=0, m=1, res1=res1, res2=res2, id=id))
                        fragments.append(IPv6ExtHdrFragment(offset=1500, m=1, res1=res1, res2=res2, id=id))
                        fragments.append(IPv6ExtHdrFragment(offset=1500*2, m=0, res1=res1, res2=res2, id=id))

                        id = 30
                        fragments.append(IPv6ExtHdrFragment(offset=0, m=1, res1=res1, res2=res2, id=id))
                        fragments.append(IPv6ExtHdrFragment(offset=1400, m=1, res1=res1, res2=res2, id=id))
                        fragments.append(IPv6ExtHdrFragment(offset=200, m=0, res1=res1, res2=res2, id=id))
                return fragments
        udp = UDP(sport=get_dynamic_port_number(), dport=53)/Raw(load="A"*1000)
        tcp = TCP(sport=get_dynamic_port_number(), dport=80)/Raw(load="A"*1000)
        fce = lambda : add_transport_layer(repeat_headers(generate_packets()), [udp, tcp])
        result, filename = uni_run(fce, self.__class__.__name__, self.probe_address, self.probe_port, options=self.options)
        return self.__class__.__name__, result, filename,
```

Obr. 5.1: Generace paketů pro TestRepetetiveFragmentHeaders

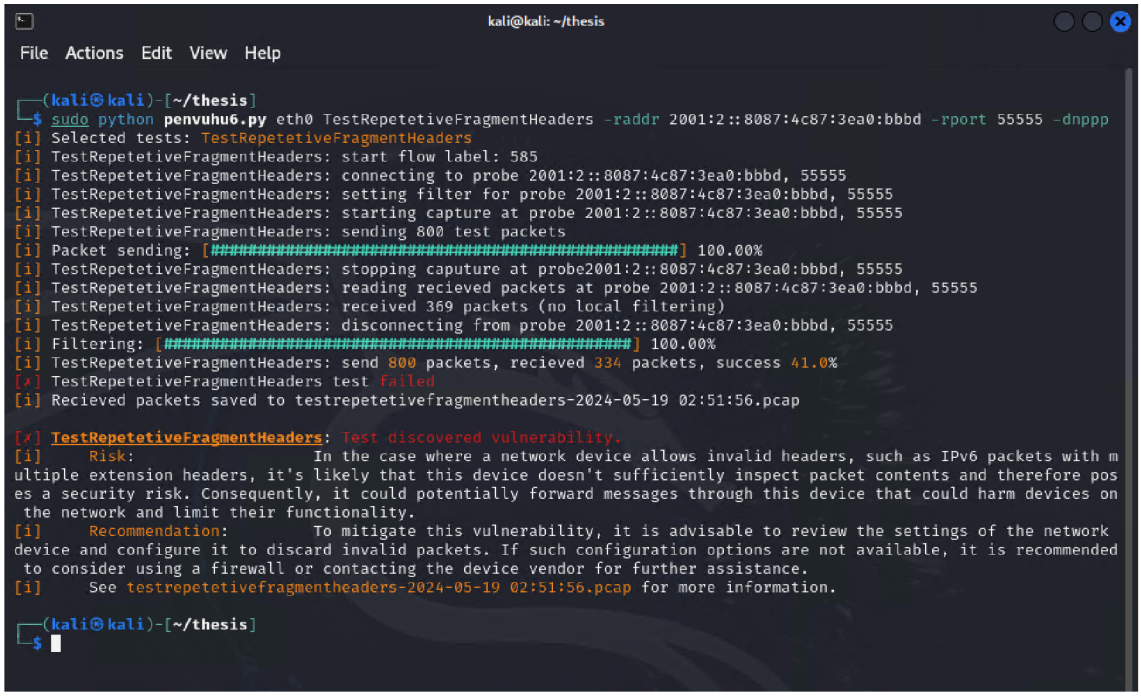
možností jsou vygenerována záhlaví Routing header, která jsou následně vložena za sebe. Podobně jako je tomu u *TestRepetetiveFragmentHeaders* jsou následně připojena transportní záhlaví a IPv6 záhlaví s flow label. Posledním krokem testu je odeslání paketů k sondě a jejich následná analýza stejně jako je tomu u testu repetetivních fragmentujících záhlaví. Výstup testu je znázorněn na obr. 5.5 a příklad průchozího paketu na obr. 5.4. Router v tomto případě propustil 48% odeslaných paketů až k sondě.

5.1.3 Repetetivní Hop-by-hop extension header

Záhlaví Hop-by-hop neobsahuje mnoho polí ke změnám, zato však umožňuje vložení options. Vybrané options jsou Pad1, PadN a RouterAlert s hodnoty 0,1,2,5 a 40000. Následně jsou vygenerována dvě různá záhlaví options header. V prvním případě záhlaví obsahuje pouze jednu option a ve druhém kombinaci dvou výše zmíněných. V případě, kdy options nemají požadovanou velikost, jsou doplněny záhlavím PadN. Hodnota 0 router alert reprezentuje MLD message, hodnota 1 RSVP

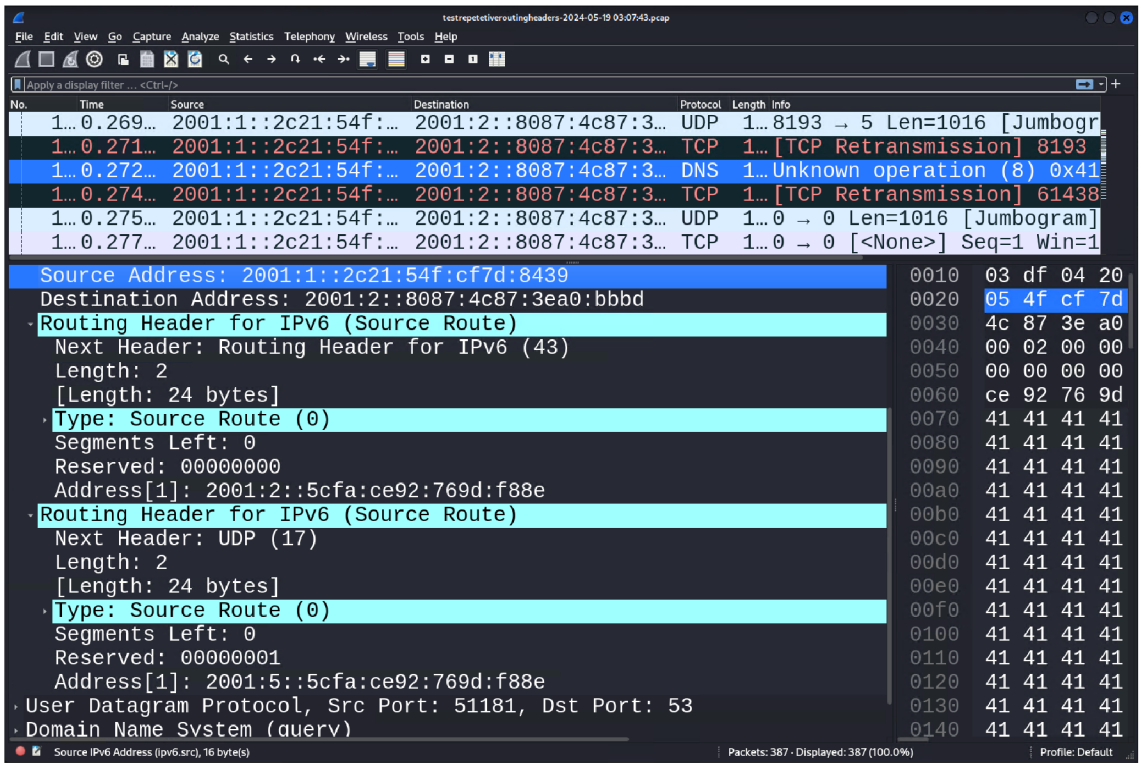


Obr. 5.2: Nevyhovující zachycené pakety testu TestRepetetiveFragmentHeaders

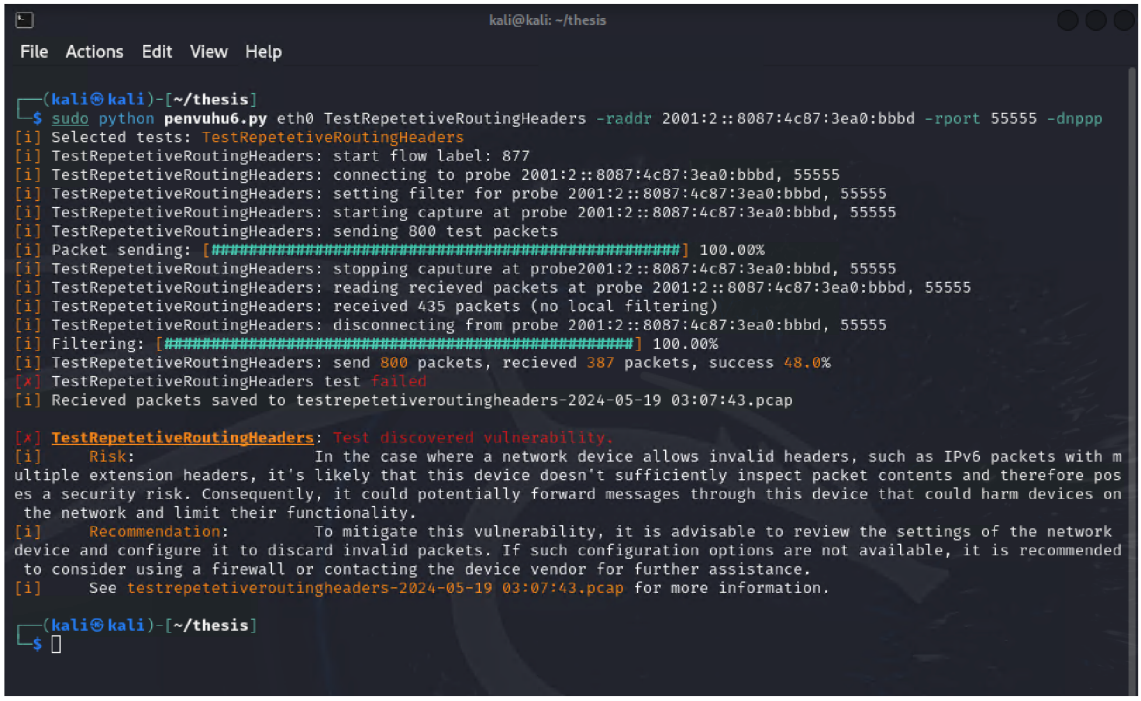


Obr. 5.3: Výsledek testu TestRepetetiveFragmentHeaders

message a hodnota 2 indikuje Active Networks message [33]. Hodnoty 5 a 40000 byly vybrány náhodně z rezervovaného rozsahu. Následně jsou ze záhlaví vytvořeny



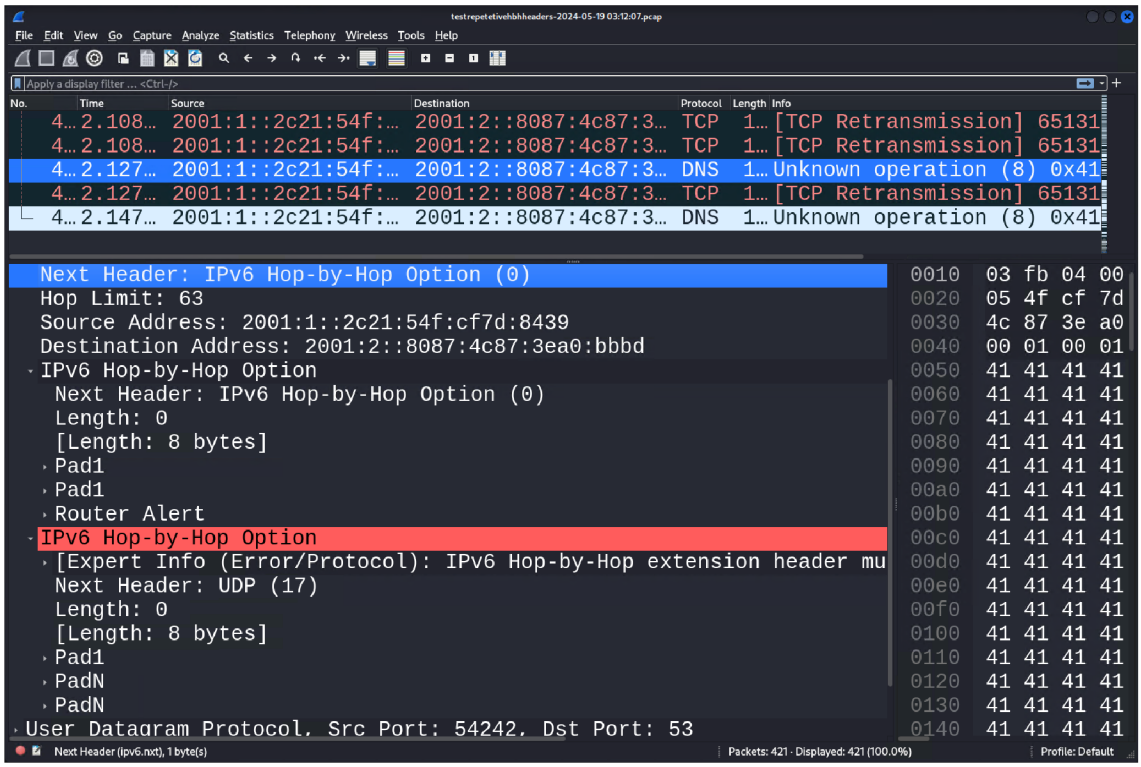
Obr. 5.4: Nevyhovující zachycené pakety testu TestRepetetiveRoutingHeaders



Obr. 5.5: Výsledek testu TestRepetetiveRoutingHeaders

kombinace rozšířené o transportní a IPv6 záhlaví. Detekce opět probíhá za pomocí pole flow label v IPv6 záhlaví. Následně je spuštěno zachytávání sondy a vytvořené

pakety odeslány k sondě. Po odeslání paketů je zachytávání zastaveno a sonda servisním kanálem zašle zachycené pakety zpět k analýze. V případě kdy síť propustí paket obsahující dvě Hop-by-hop header, vyhodnotí test síť jako zranitelnou pro nekontrolování nevalidních paketů. Tato zranitelnost může vést ke zpomalení sítě, nebo propuštění nechtěných zpráv skrz síťový prvek. Výsledek testu je zobrazen na obr. 5.7, příklad průchozího paketu na obr. 5.4. v Případě tohoto testu router dokonce propustil až 52% z celkem 800 odeslaných paketů.



Obr. 5.6: Nevyhovující zachycené pakety testu TestRepetetiveHBHHeaders

5.1.4 Repetetivní Destination options extension header

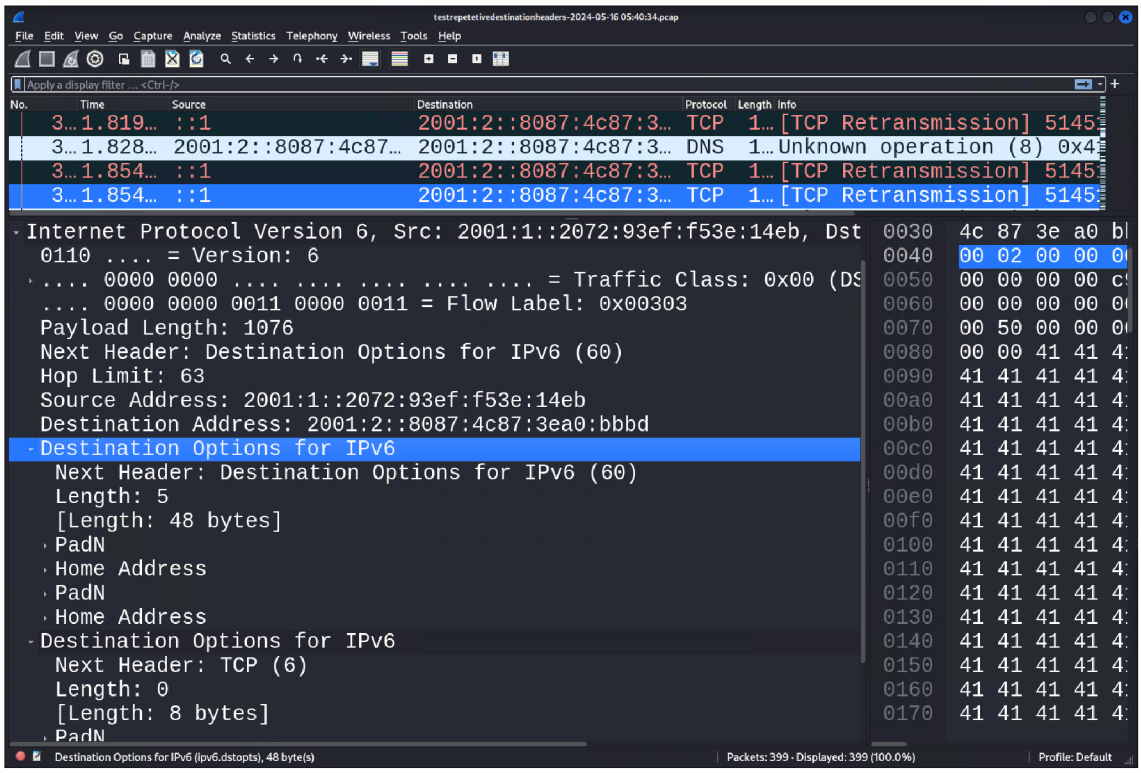
Záhlaví Destination options neobsahuje mnoho polí, ale stejně jako záhlaví Hop-by-hop je určené k přenášení options. Pro scénář byly vybrány vyplňující options Pad1 a PadN spolu s option Home Address. Pro pole Home Address byly vybrány IPv6 adresy odpovídající local host adrese a adrese sondy použité v testu. Podobně jako v testu repetetivních Hop-by-hop jsou vytvořeny dva druhy záhlaví a to s jednou, nebo kombinací dvou definovaných options. V případě, kdy rozšiřující záhlaví neodpovídá definované velikosti, je velikost zarovnána pomocí PadN option. Vytvořená záhlaví jsou zkombinována za sebe a rozšířena o transportní a IPv6 záhlaví s identifikací pomocí pole flow label. Po spuštění zachytávání sondy jsou na ni pakety poslány. Následně je sonda zastavena a výsledky po přeposlání analyzovány.

Obr. 5.7: Výsledek testu TestRepetetiveHBHHeaders

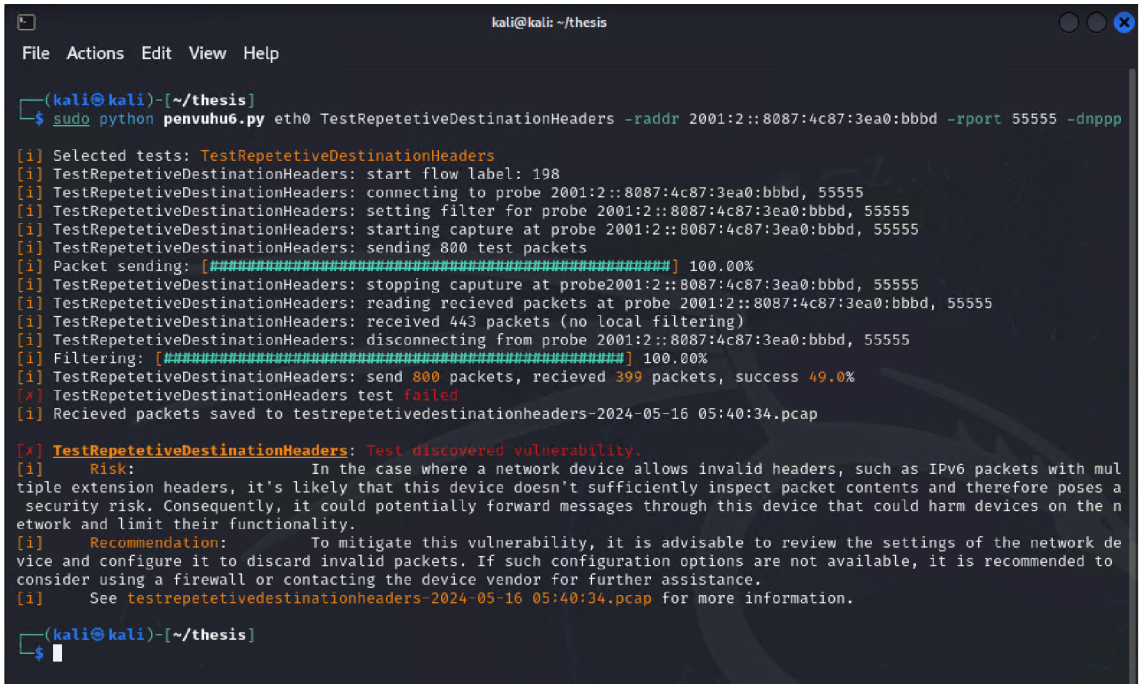
Oproti testu repetetivního Hop-by-hop povoluje RFC 8200 [23] paket obsahující dvě rozšiřující záhlaví Destination options. Tento stav je však vázán převážně k použití tunelů, kde první záhlaví je určeno ke zpracování první stanicí v prvním IPv6 záhlaví a druhé záhlaví Destination options je zpracováno skutečnou koncovou stanicí. Tento test však nevyužívá tunelující záhlaví *Encapsulating Security Payload header* a byť není tento stav přímo v RFC 8200 definován, neměl by být validní. Příklad průchozího paketu při testování nástroj je zobrazen na obr. 5.8. Obrázek vypovídá o propuštění paketu s dvěma Destination header zahlavími obsahující dokonce dokonce dvakrát option Home Address. Výsledek celého testu je znázorněn na obr. 5.9

5.1.5 Špatně seřazená rozšiřující záhlaví

Kromě počtu rozšiřujících záhlaví v paketu definuje RFC 8200 [23] doporučené pořadí rozšiřujících záhlaví. V IPv6 paketu nemusí být nutně přítomna všechna záhlaví, ale v případě kdy je záhlaví obsaženo, je doporučené respektovat pořadí následovně. Po záhlaví IPv6 následuje Hop-by-hop extension header a destination header, který je zpracován stanicí definovanou v destination address IPv6 záhlaví a cíli definovanými v Routing header. Následuje Routing header, Fragment header, Authentication header, Encapsulating Security payload header a opět Destination header. V tom případě je však Destination header určen koncové stanici. Po Destination header pokračuje protokol transportní vrstvy.



Obr. 5.8: Nevyhovující zachycené pakety testu TestRepetetiveDestinationHeaders



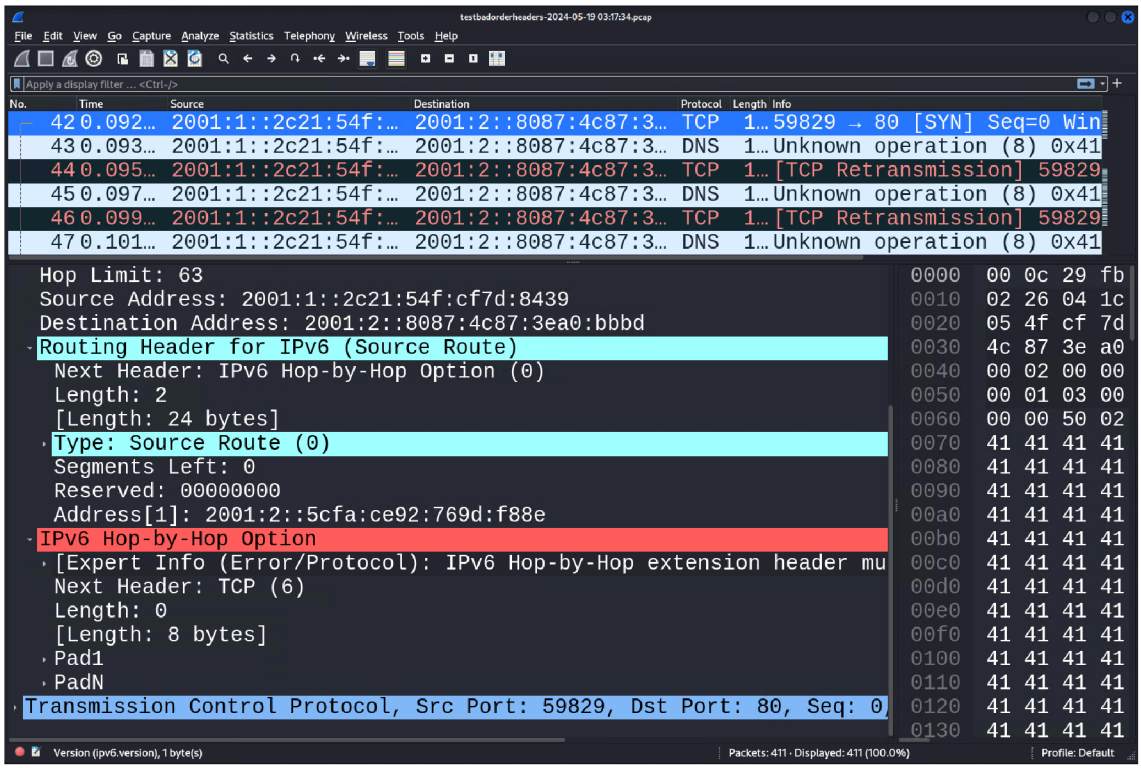
Obr. 5.9: Výsledek testu TestRepetetiveDestinationHeaders

Třída *TestBadOrderHeaders* testuje zda síťový prvek propouští pakety, které obsahují špatné pořadí extension headers. Z důvodu vysokého počtu možných kombi-

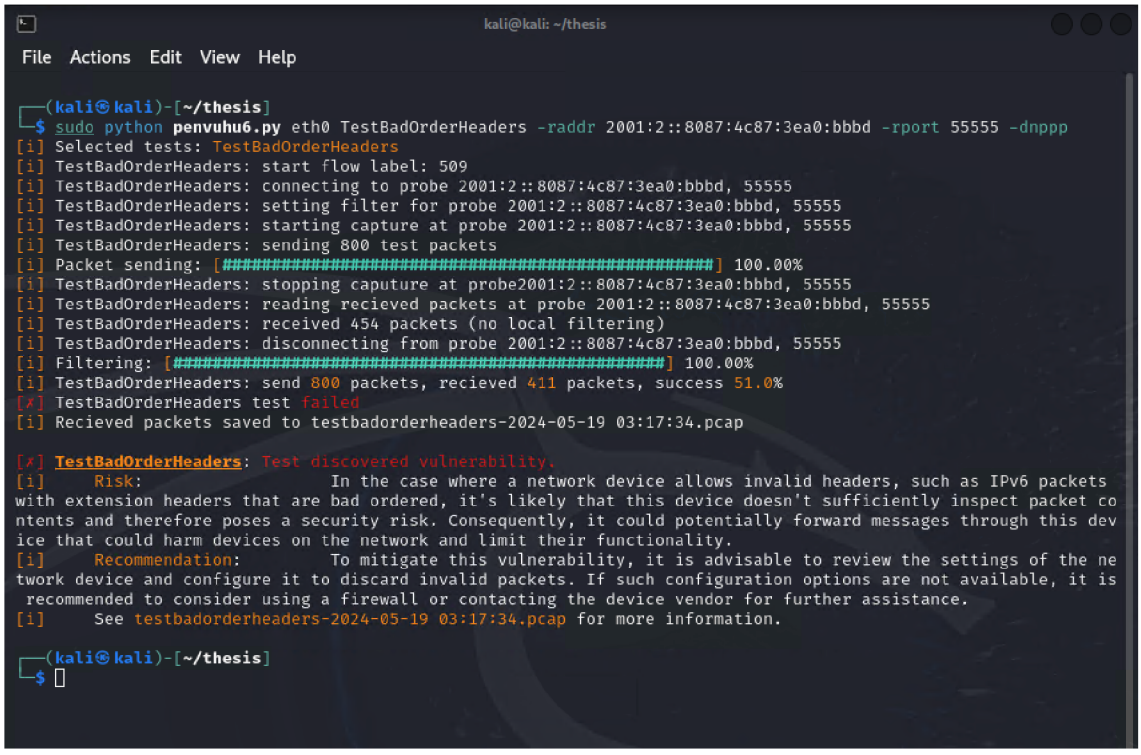
nací testuje *TestBadOrderHeaders* pouze část možných případů. Pro testování byla vybrána rozšiřující záhlaví Hop-by-hop options, Routing a Destination options header. Oproti předchozím testům zaměřeným na opakování záhlaví byla generace variant těchto záhlaví z důvodu vysokého počtu paketů omezena na následující případy. Hop-by-hop options nabývají čtyř různých options a to Pad1, PadN a Router alert s hodnotou 1 a 2. Routing header nabývá délek 0, 2 a 255, typů 0,2 a 255, segments left 0, 1, 2 a rezervované pole nabývá hodnot 0, 1 a 3. Jako home adresy byly zvoleny localhost ::1 a náhodně adresa 2001:2::5cfa:ce92:769d:f88e. Pro Destination options byly zvoleny options Pad1, PadN a Home address option nabývající hodnot home address localhost ::1 a adresy sondy. Po generaci všech zmíněných variant jsou vytvořeny dvojice záhlaví Routing header následované option hob-by-hop header, která by měla předcházet záhlaví Routing header a dvojice Destination options header následovaný hob-by-hop header, který by opět měl předcházet Destination options header. Tyto kombinace jsou rozšířeny o záhlaví transportních protokolů TCP, nebo UDP a protokolu IPv6 s unikátním flow label pro následnou filtraci. Po této generaci je spuštěno zachytávání na sondě, následované odesláním paketů, ukončením zachytávání a zasláním zachycených paketů zpět hlavnímu programu. Ten vyfiltruje jím vyslané pakety, které značí, že síťový prvek pakety nezahodil. V takovém případě test vyhodnotí síť jako zranitelnou pro zasílání špatného pořadí rozšiřujících záhlaví. V případě obr. 5.10 byl propuštěn mimo jiné paket obsahující Routing header následovaný Hop-by-hop header, který by měl být správně prvním rozšiřujícím záhlavím. Celý výsledek testu je na obr. 5.11.

5.2 Překrývající se fragmenty

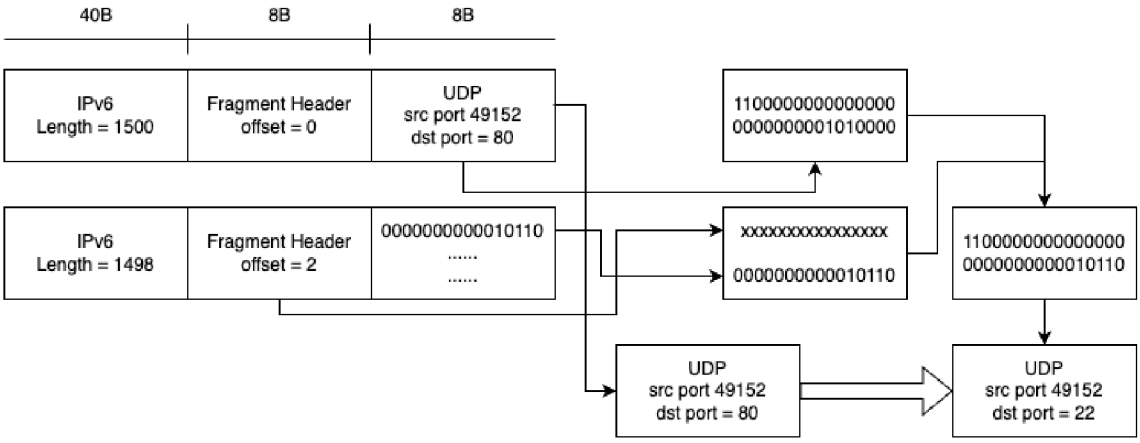
Tento scénář se zaměřuje na testování propouštění překrývajících se fragmentů síťovými prvky. Překrývající se fragmenty, anglicky overlapping fragment je paket s fragmentačním záhlavím, který díky své hodnotě v poli offset zasahuje do předešlého fragmentu. Překrytí fragmentů nastane například v případě, kdy je vyslán první fragment s hodnotou offset 0 a délkou 1500 bytů následovaný fragmentem s hodnotou offset 1000 a dlouhý 1500 bytů. V případě prvního fragmentů je z původního paketu díky rozšiřujícímu záhlaví Fragment header vysláno prvních 1492 bytů. Druhý paket v případě hodnoty offset 1000 tedy přenáší 492 bytů, které již byly přeneseny. IPv6 zařízení nesmějí dle RFC 5722 [42] a RFC 6946 [41] překrývající se pakety generovat a v případě detekce překrytí při sestavování paketu musí takový paket zahodit, avšak obzvláště starší systémy nemusí packet zahodit. V případě, kdy implementace seskládá paket, může dojít k přepsání původních polí. Například je-li na firewall povolen pouze cílový port 80 a první fragment obsahuje v záhlaví cílový port UDP 80, ale následující fragment přepíše tento port na hodnotu 22,



Obr. 5.10: Nevyhovující zachycené pakety testu TestBadOrderHeaders



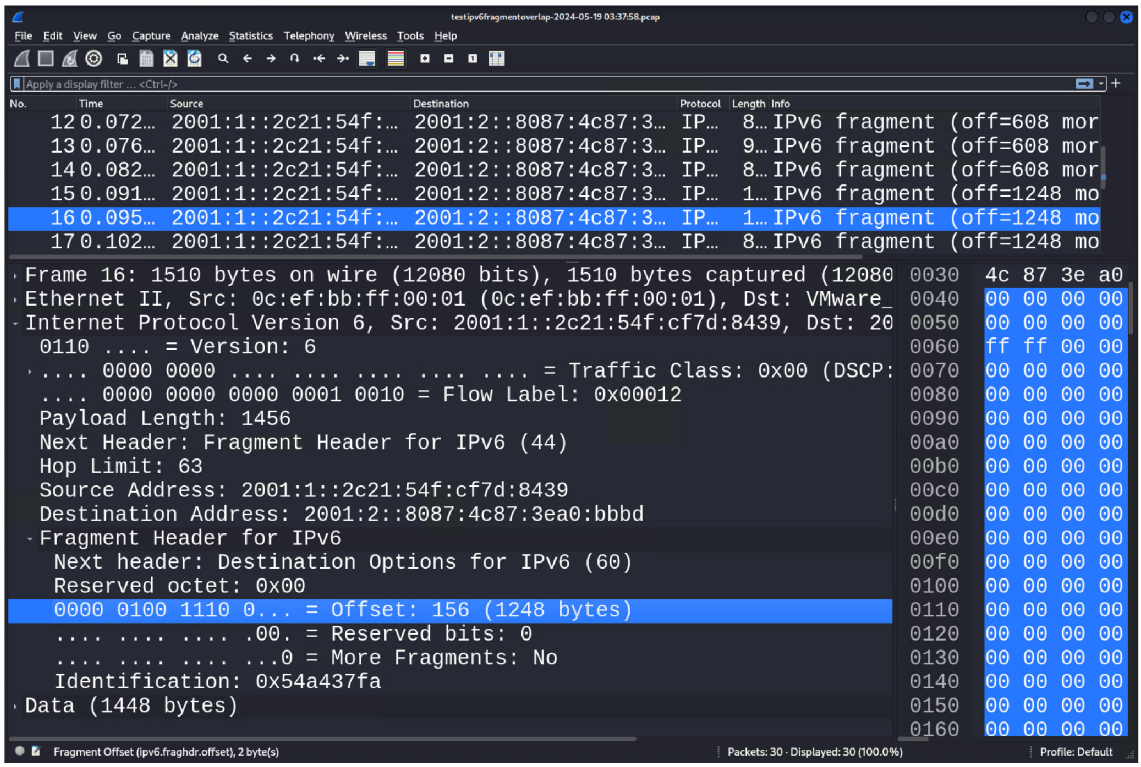
Obr. 5.11: Výsledek testu TestBadOrderHeaders



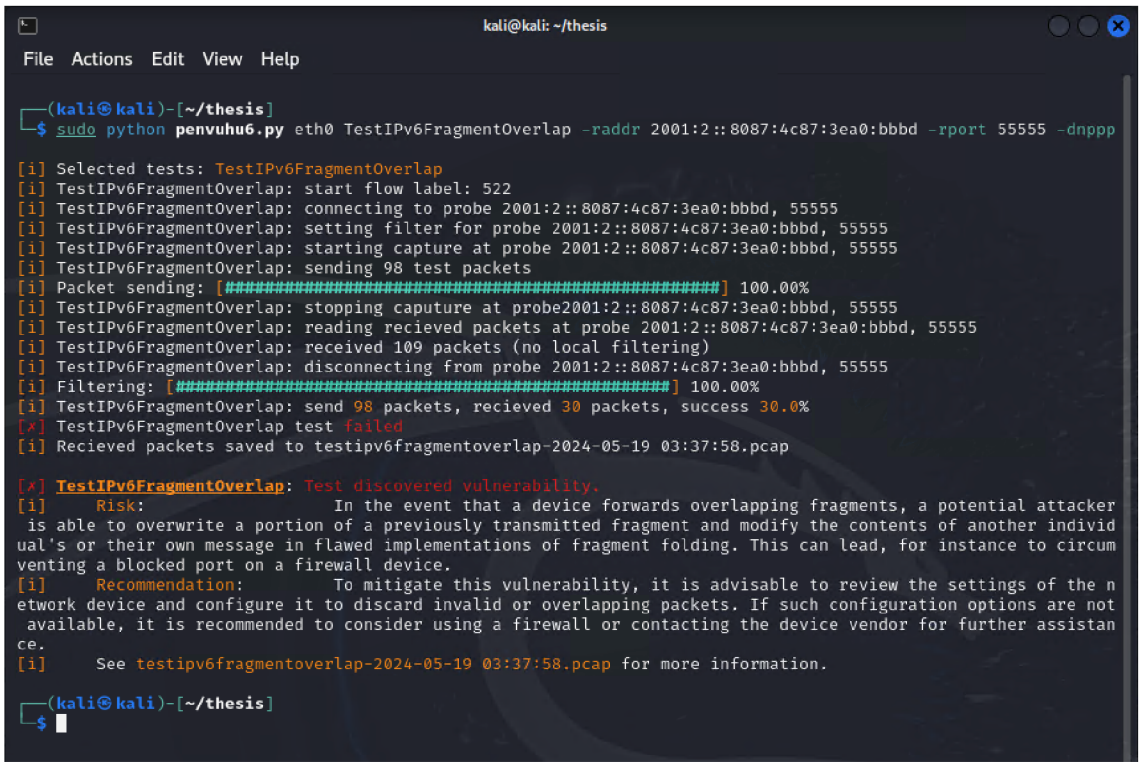
Obr. 5.12: Znázornění principu překrývajících se fragmentů

útočník může získat možnost přihlášení ke službě SSH. Tento princip je znázorněn na obrázku 5.12. Aby k přepsání nedošlo v první fragmentu využívá se pro odsunutí transportní a vyšších vrstev rozšiřujících záhlaví IPv6 s volitelnými options jako Hop-by-hop, nebo Destination options. Díky naplnění těchto rozšiřujících záhlaví lze transportní protokol přesunout do jiných fragmentů, což stěžuje síťovým prvkům schopnost detekce těchto útoků. Fragmentaci lze provést i v rámci jedné sítě, avšak zpravidla se mezi sítovými prvky nenachází zařízení, které by fragmenty mezi dvěma stanicemi kontrolovalo. Primárně je tedy scénář určen k testování směrovačů a firewallů mezi sítěmi, lze ho však použít i v rámci lokální sítě, pokud mezi `Penvuhu6` a sondou filtrující zařízení. Při testování pomocí směrovače s RouterOS bylo odesláno 98 a přijato 30 paketů viz obr. 5.14. Paket na obrázku 5.13 udává offset 1248 bytů, což není ani minimální MTU IPv6 a značí tak náchylnost na propouštění překrývajících se fragmentů.

Test *TestIPv6FragmentOverlap* využívá k odsazení rozšiřující záhlaví Destination options. Vzhledem k tomu že je záhlaví Destination options určeno pro koncovou stanici, je díky němu více pravděpodobné, že se tímto záhlavím nebude síťový prvek zabývat. Výchozí TCP a UDP pro zdrojové i cílové porty u tohoto testu jsou nastaveny na hodnotu 1234. V případě špatné implementace sestavování fragmentů mohou být v případě UDP i TCP přepsány zdrojový i cílový port na hodnotu 65535. Pro MTU byla vybrána téměř minimální hodnota v IPv6 sítích 1300 a maximální velikost datové části ethernet rámce 1500 bytů. Pro odsazení transportního záhlaví byly využity délky 1280 a 640. Záhlaví transportního protokolu je tedy při testu vně i mimo první fragment. Kromě UDP záhlaví jsou přepsány i hodnoty v datové části až po velikost fragmentu. Délka těchto hodnot byla náhodně zvolena na 1600, nebo 800. V případě rámce ethernet se tedy jedná v případě hodnoty výplně 1600 vždy o maximální možné MTU. Jako byty výplně byly zvoleny hodnoty 0. Všechny



Obr. 5.13: Nevyhovující zachycené pakety testu TestIPv6FragmentOverlap



Obr. 5.14: Výsledek testu TestIPv6FragmentOverlap

hodnoty však lze v případě potřeby změnit ve zdrojovém souboru *testfragmentoverlap.py*, který je součástí přílohy A, ve funkci *generate_packets*.

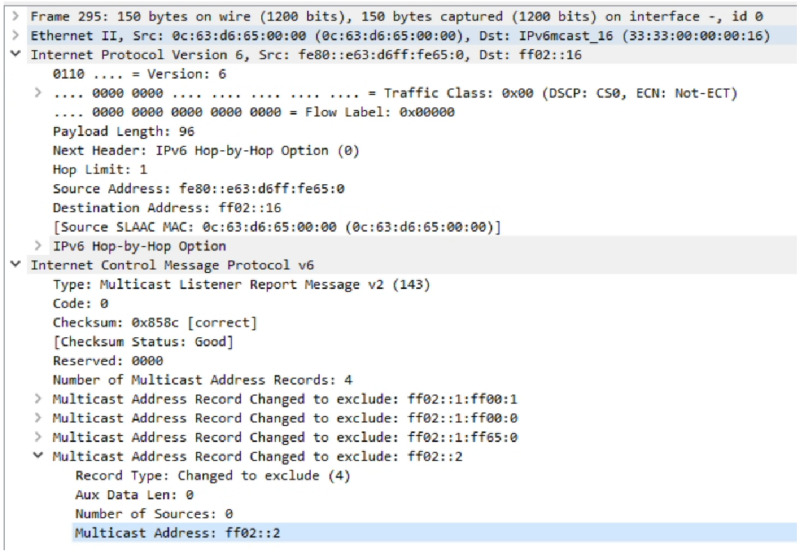
5.3 Router advertisement a DHCPv6 guard

Stejně jako je tomu u IPv4 je i v IPv6 problematický výchozí stav připojené stanice. Nově připojená stanice se teprve musí dozvědět informace o síti. Právě v při připojení je nejjednodušší novou stanici zmást, protože ještě netuší kdo je v síti důvěryhodný a může začít věřit nebezpečným stanicím, které mohou novou zmást chybnou konfigurací sítě. Tento problém je řešen v IPv6 obdobně jako je tomu u IPv4 pomocí guardů. Guardi hlídají tok dat na rozhraních síťových prvků a v případě zachycení škodlivé zprávy tuto zprávu zahodí a upraví nastavení zdrojového rozhraní dle nastavených politik.

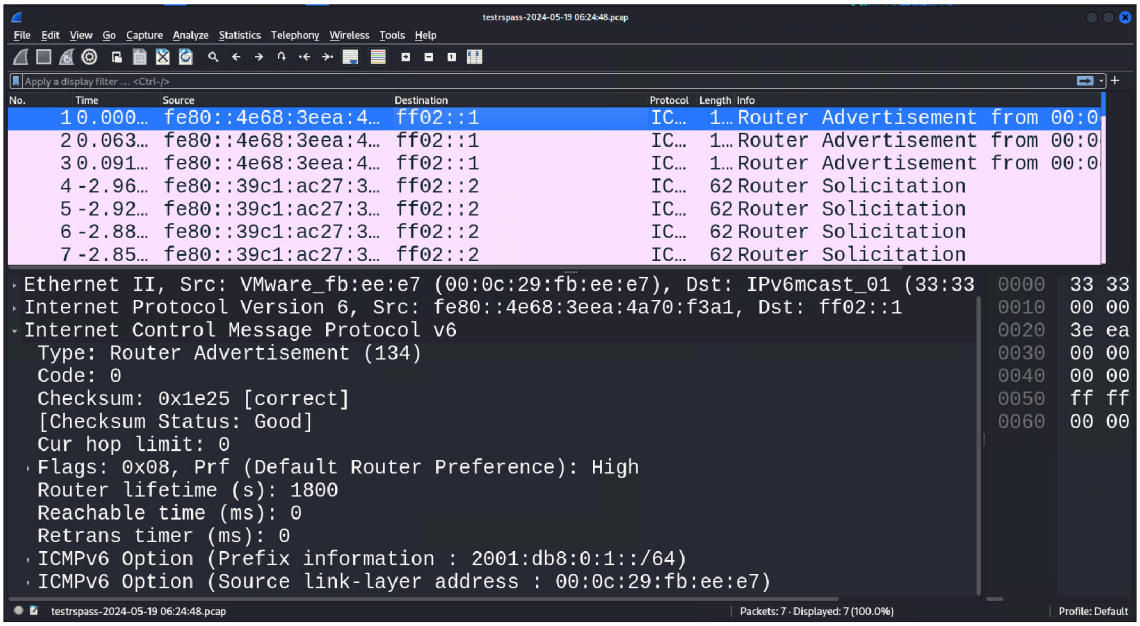
5.3.1 RA guard

Router advertisement zpráva je jedna z nejdůležitějších zpráv IPv6 sítí. Zpráva informuje koncové stanice o síti a jejich parametrech. V případě podvržení této zprávy může být koncová stanice napadena útokem MitM, nebo DoS. Možnou bezpečností ochranou proti podvržení RA zprávy je RA guard, který je spuštěn na přepínači a povoluje tyto zprávy pouze z rozhraní nastavených jako důvěryhodné. Pro správnou funkci testu není vhodné nastavit politiku RA guard na blokování rozhraní, ale pouze nepropuštění paketu, případně výpisu logu o zachycení zprávy. Hodnota hop limit zprávy RA je nastavena na 1 a tedy zpráva neprojde do jiných sítí. Test je tedy zaměřen na testování prvků v lokální síti, především přepínačů.

Při spuštění testu se `Penvuhu6` snaží připojit do multicastové skupiny všech směrovačů pro příjem zpráv router solicitation viz. obr 5.15. Následně jsou vygenerovány zprávy RS a servisním kanálem zaslány k vyslání sondou. `Penvuhu6` by v ideálním případě neměl zprávu RS obdržet. Pokud zprávu obdrží, síť poskytuje potenciálním útočníkům nadbytek informací. Posledním krokem je ověření, zda může `Penvuhu6` vyslat zprávu RA. Na sondě je tedy nastaveno zachytávání a následně vyslány hlavním programem RA pro náhodně zvolené prefixy sítě. Pokud sonda tyto pakety zachytila, potenciální útočník může mást stanice nastavením nevalidních prefixů, změnit výchozí bránu sítě, jmenné servery, nebo odposlouchávat stanice pomocí útoku MitM. Jak je vidět na obrázku 5.17, při testu bylo odeslány i přijato 7 paketů. Zařízení tedy tyto zprávy nefiltrovalo. Obrázek 5.16 obsahuje příklad průchozí zprávy Router Advertisement.



Obr. 5.15: Příklad MLD přihlášení do skupiny všech routerů

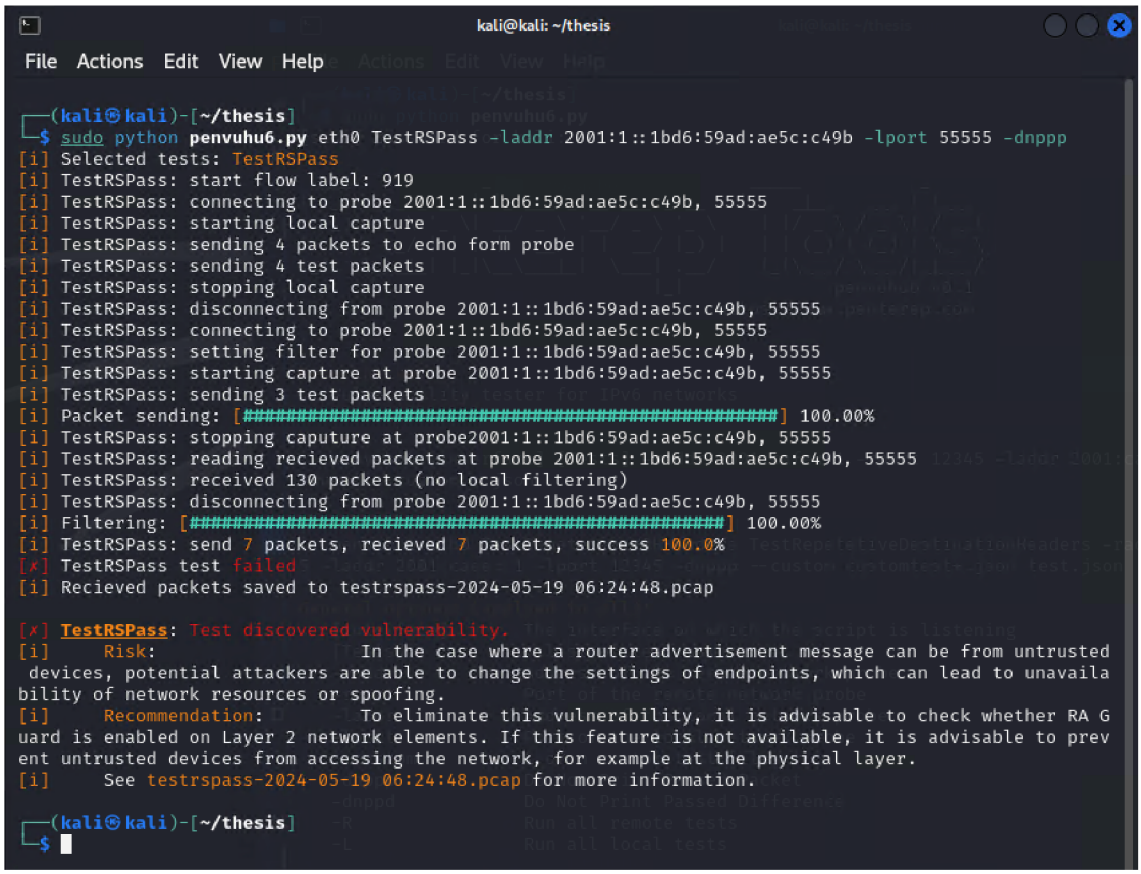


Obr. 5.16: Nevyhovující zachycené pakety testu TestRSPass

5.3.2 DHCPv6 guard

Výchozí způsob získání informací o síti je v IPv6 sítích zpravidla SLAAC, ale lze využít i DHCPv6 a to ve dvou variantách. DHCPv6 může fungovat jako stavové i bezstavové. V obou případech existuje riziko útočníka, který se vydává za legitimní DHCPv6 server. Podobně jako je tomu u zpráv RA je útočník schopný realizovat útok typu DoS, nebo MitM.

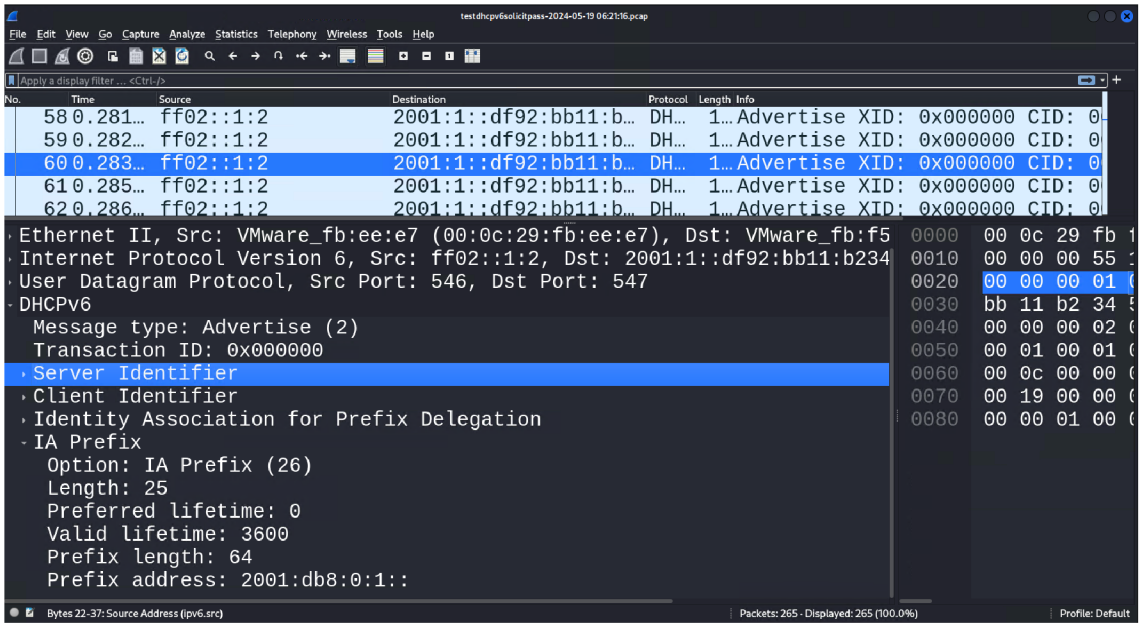
Test *TestDHCPv6SolicitPass* se opět zaměřuje pouze na lokální síť a neřeší testo-



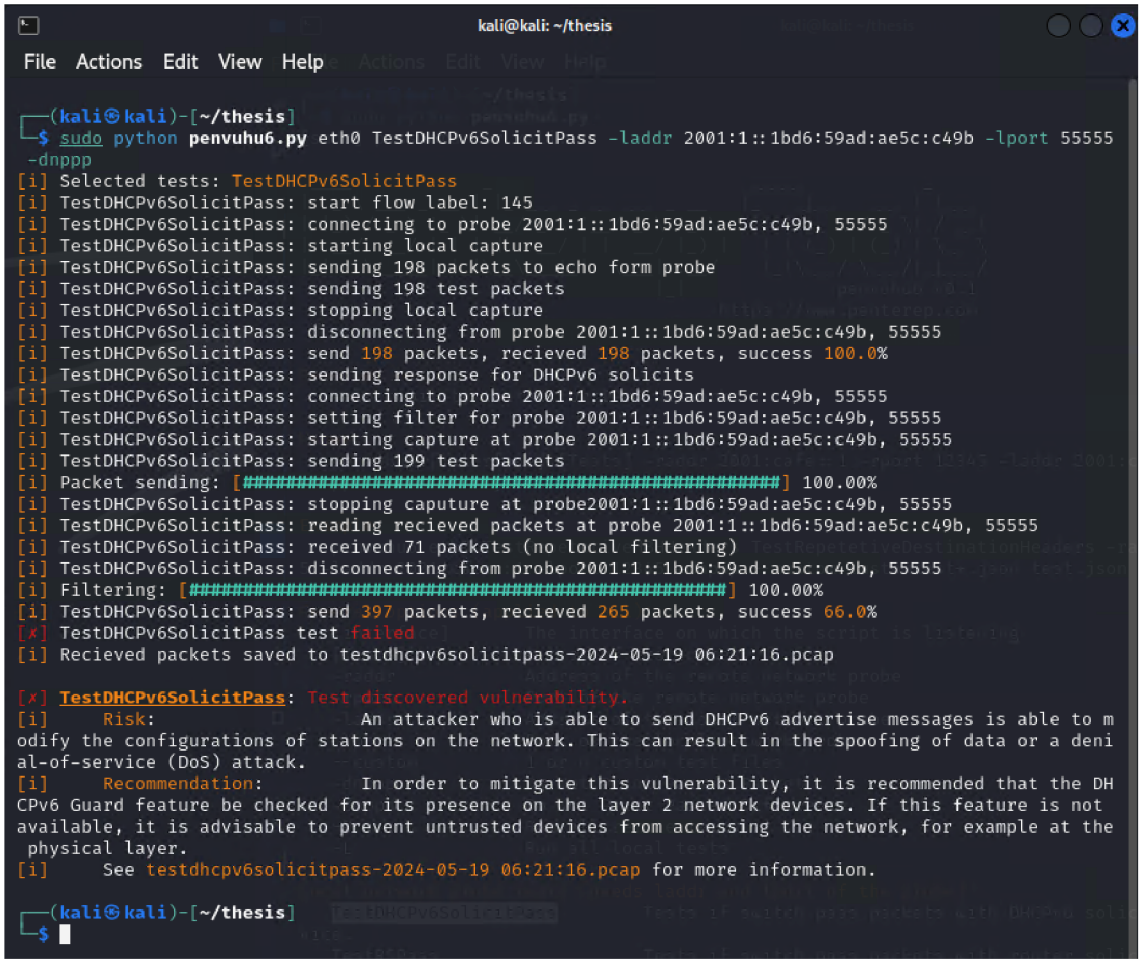
Obr. 5.17: Výsledek testu TestRSPass

vání DHCPv6 relay agenta. Jako první se v testu vygenerují DHCPv6 solicit zprávy. Jako source adresy byly vybrány náhodné MAC adresy a adresa obsahující samé nuly. Pro MAC adresy byly vybrány adresy broadcast, multicast adresa všech směrovačů a fyzická adresa rozhraní `Penvuhu6`. Dále byly náhodně vybrány časy použité při generaci *DHCPv6 Unique Identifier* (DUID) 0, 10, 1000, 100000, 100000000. Pro generaci DUID pomocí enterprise number byly náhodně vybrány hodnoty 0 a 311 a pro generaci pomocí *Universally Unique IDentifier* (UUID) byly vybrány hodnoty 1 a 5. Pro identifikaci a filtraci paketů bylo vybráno transaction ID 0x1234. Vygenerované DHCPv6 solicit zprávy jsou všechny kombinace výše zmíněných hodnot. Po generaci se `Penvuhu6` pomocí MLD Multicast Address Record pokusí přihlásit do multicastové skupiny všech site-local DHCPv6 serverů a linkl-local všech DHCPv6 serverů a relay agentů. Po přihlášení se do skupiny `Penvuhu6` zašle vygenerované pakety sondě k přeposlání a začne naslouchat na svém rozhraní. V případě, kdy `Penvuhu6` přijme zprávu DHCPv6 solicit, znamená to že síť poskytuje zařízením více informací než je třeba. Po odeslání paketů ze sondy je spuštěno zachytávání na rozhraní sondy a `Penvuhu6` se pokusí vyslat DHCPv6 advertise zprávu. V případě, kdy `Penvuhu6` přijalo DHCPv6 solicit vyšle i DHCPv6 advertise na tuto zprávu.

Pokud sonda zachytí zprávy DHCPv6 advertise, síť je zranitelná na podvrhnutí legitimního DHCPv6 serveru. Při testování bylo vysláno 397 a přijato 265 paketů viz obr. 5.19. Zařízení tedy propustilo přibližně 66% vyslaných zpráv. Obrázek 5.18 zobrazuje příklad průchozí zprávy DHCPv6 advertise.



Obr. 5.18: Nevyhovující zachycené pakety testu TestDHCPv6SolicitPass



Obr. 5.19: Výsledek testu TestDHCPv6SolicitPass

6 Testování nástroje

Testování nástroje probíhalo v prostředí GNS3 za pomocí dvou virtuálních strojů s nainstalovaným operačním systémem Kali linux a virtuálního síťového prvku se systémem RouterOS. Hlavním cílem nebylo otestovat síťový prvek, ale nástroj `Penvuhu6`. Testování probíhalo vždy se základní konfigurací a následně s pozměněnou. Zároveň z důvodu omezených výpočetních zdrojů byly testy omezeny na maximální počet 800 odeslaných paketů.

6.1 Testování remote probe testů

Při testování remote probe testů byly na síťovém prvku nakonfigurovány rozhraní ether1 s IPv6 adresním rozsahem 2001:1::/64 a ether2 s rozsahem 2001:2::/64. Pro obě tyto adresy síťový prvek zasílá zprávy Router Advertisement viz obr. 6.1.

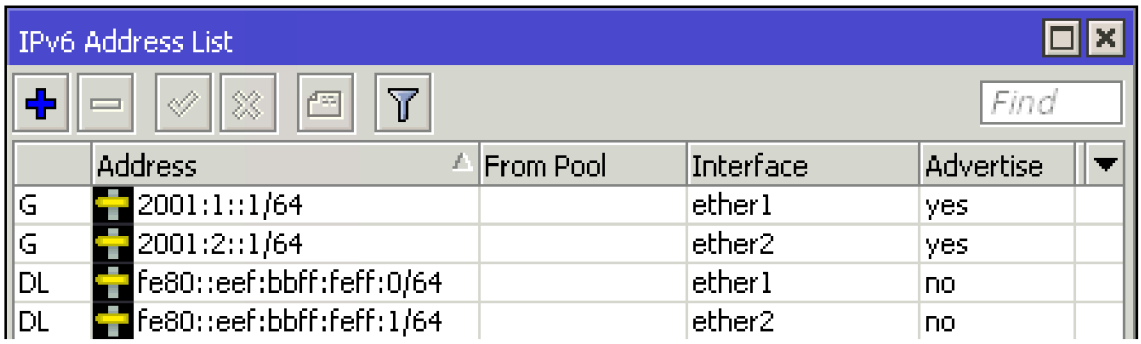
Jako první byly na nástroji `Penvuhu6` spuštěny všechny remote testy pomocí přepínače *-R*. Výpisy těchto testů jsou zobrazeny na obr. 6.2. Výsledek testu *TestBadOrderHeaders* musel být přegenerován, proto neodpovídá časová značka v názvu souboru.

Po dokončení všech testů bylo na síťovém prvku pomocí nastavení firewall zakázáno přeposílání paketů s invalid connection state obr. 6.3. Toto nastavení se zaměřuje především na sledování stavů spojení paketů, paketů se špatným sekvenčním číslem, nebo nadměrně využívajícím prostředků síťového prvku [56].

Po nastavení zahazování invalid connection state byly znovu spuštěny všechny remote testy viz obr. 6.4. Hodnoty odeslaných a přijatých paketů v případech bez i při filtraci pomocí firewallu byly zaneseny do tabulky Tab. 6.1.Při porovnání zejména počtu přijatých paketů vzdálenou sondou si lze všimnou zásadní změny u testů *TestRepetetiveFragmentHeaders* a *TestIPv6FragmentOverlap*. Z výsledků vyplývá, že nastavením firewallu na zahazování paketů jde docílit menšího počtu propuštěných paketů obsahující vícenásobné záhlaví Routing header a překrývajících se fragmentů. Dalším vyčnívajícím testem je test *TestRepetetiveRoutingHeaders*, u kterého se projevilo snížení o 47 přijatých paketů. Pro ostatní testy se zdá být tato filtrace neefektivní.

6.2 Testování local probe testů

Pro testování lokální sítě byly zvoleny stanice s operačním systémem Kali linux a síťový prvek se systémem RouterOS. Při lokálních testech byly nastaveny rozhraní RouterOS *ether1* a *ether2* do bridge *bridge1* s adresním rozsahem 2001:1::/64 viz obr. 6.5. Síťový prvek je tedy zapojen jako L2 přepínač. Přepínač má nastaveno



Obr. 6.1: RouterOS nastavení IPv6 adres na rozhraních ether1 a ether2

```
[i] TestRepetetiveFragmentHeaders: send 800 packets, recieved 623 packets, success 77.0%
[✗] TestRepetetiveFragmentHeaders test failed
[i] Recieved packets saved to testrepetetivefragmentheaders-2024-05-19 05:59:22.pcap
[i] TestRepetetiveRoutingHeaders: send 800 packets, recieved 403 packets, success 50.0%
[✗] TestRepetetiveRoutingHeaders test failed
[i] Recieved packets saved to testrepetetiveroutingheaders-2024-05-19 05:59:49.pcap
[i] TestRepetetiveHBHHeaders: send 800 packets, recieved 457 packets, success 57.0%
[✗] TestRepetetiveHBHHeaders test failed
[i] Recieved packets saved to testrepetetivehbhheaders-2024-05-19 06:00:29.pcap
[i] TestRepetetiveDestinationHeaders: send 800 packets, recieved 452 packets, success 56.0%
[✗] TestRepetetiveDestinationHeaders test failed
[i] Recieved packets saved to testrepetetivedestinationheaders-2024-05-19 06:01:06.pcap
[i] TestBadOrderHeaders: send 800 packets, recieved 404 packets, success 50.0%
[✗] TestBadOrderHeaders test failed
[i] Recieved packets saved to testbadorderheaders-2024-05-19 06:56:55.pcap
[i] TestIPv6FragmentOverlap: send 98 packets, recieved 30 packets, success 30.0%
[✗] TestIPv6FragmentOverlap test failed
[i] Recieved packets saved to testipv6fragmentoverlap-2024-05-19 06:01:49.pcap
```

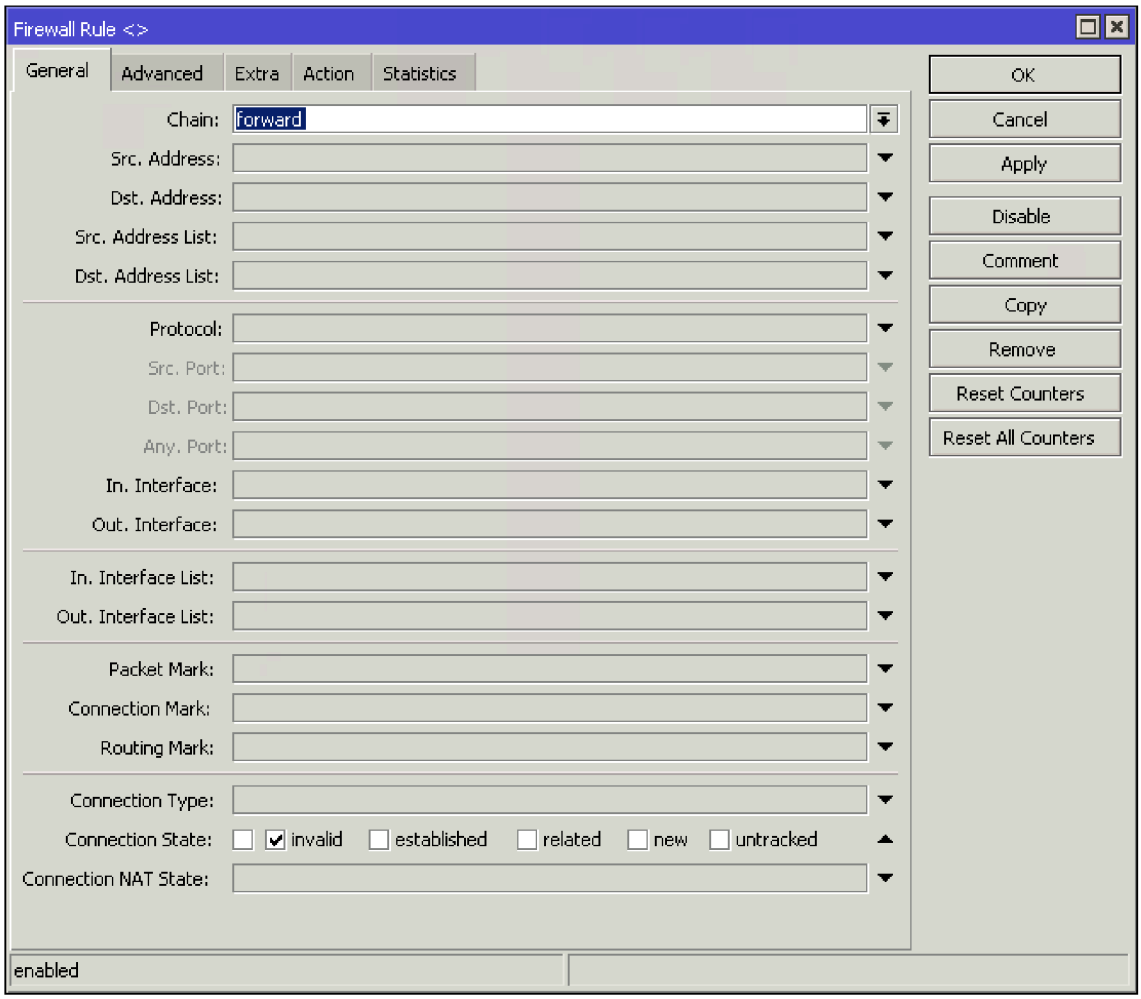
Obr. 6.2: Výsledky remote testů se základní konfigurací

Jméno testu	Od. zákl.	Při. zákl.	Od. filtr.	Př. filtr.
TestRepetetiveFragmentHeaders	800	623	800	60
TestRepetetiveRoutingHeaders	800	403	800	357
TestRepetetiveHBHHeaders	800	457	800	464
TestRepetetiveDestinationHeaders	800	452	800	454
TestBadOrderHeaders	800	404	800	411
TestIPv6FragmentOverlap	98	30	98	0

Tab. 6.1: Počty paketů remote testů dle nastavení směrovače.

zasílání rozsahu pomocí zprávy Router Advertisement pro základní funkcionalitu sítě IPv6.

Jako první jsou nástrojem `Penvuhu6` spuštěny všechny local probe testy, bez jakékoliv další konfigurace přepínače. Výsledky testů jsou znázorněny na obrázku obr. 6.6. Po ukončení testů je pro rozhraní *bridge1* nastaveno IGMP Snooping a DHCP Snooping, přičemž porty *ether1* i *ether2* jsou ponechány jako untrusted

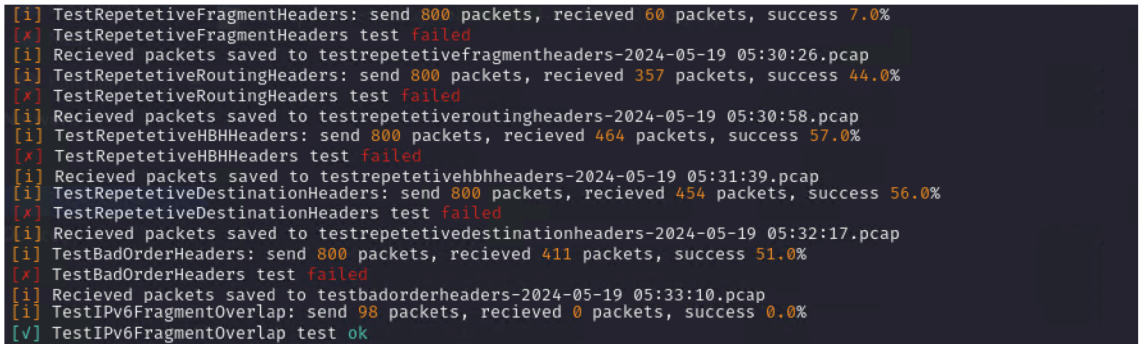


Obr. 6.3: RouterOS nastavení zahazování invalidních spojení

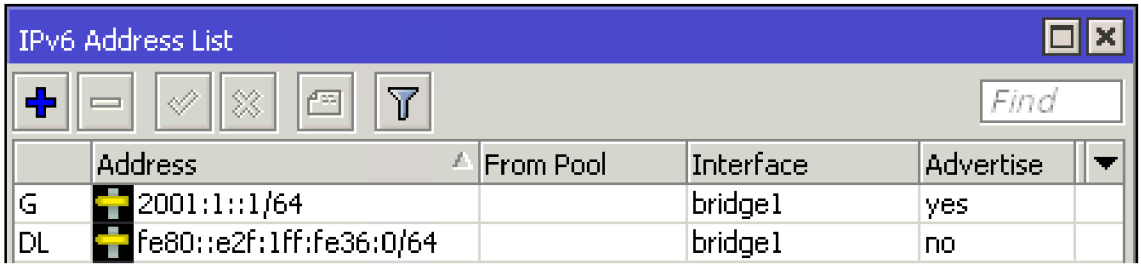
viz obr. 6.8. Všechny lokální testy jsou po této úpravě znovu spuštěny přepínačem *-L* viz obr. 6.7. Výsledky těchto testů byly zaneseny do tabulky Tab. 6.2. Chybějící paket u testu *TestRSPass* obsahoval v poli reserved hodnotu 0, avšak při opakování testu již bylo obdrženo všech 7 paketů. Dle testu tedy IGMP snooping i DHCP snooping nefiltrují ani DHCPv6 zprávy, ani Router Advertisement zprávy z untrusted portů. Nabízí se ještě použití volby Use IP Firewall, avšak v případě definice konkrétního rozhraní RouterOS vyžaduje nadřazený port, tedy *bridge1*.

Jméno testu	Odes. zákl.	Přij. zákl.	Odes. filtr.	Přij. filtr.
TestRSPass	7	7	7	6
TestDHCPv6SolicitPass	397	265	397	265

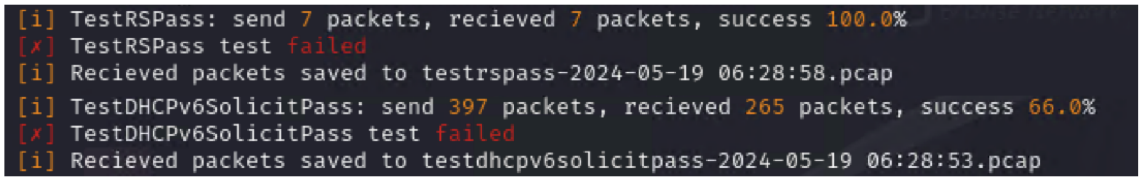
Tab. 6.2: Počty paketů local testů dle nastavení směrovače.



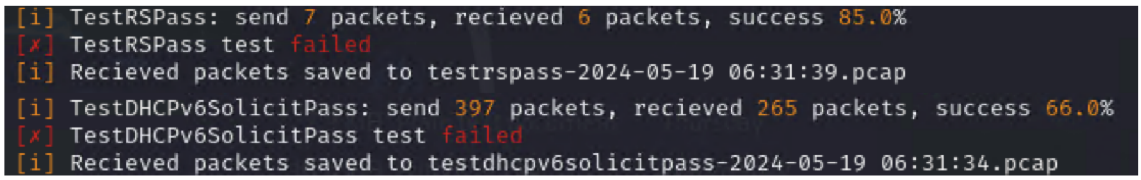
Obr. 6.4: Výsledky remote testů při filtrování invalid spojení



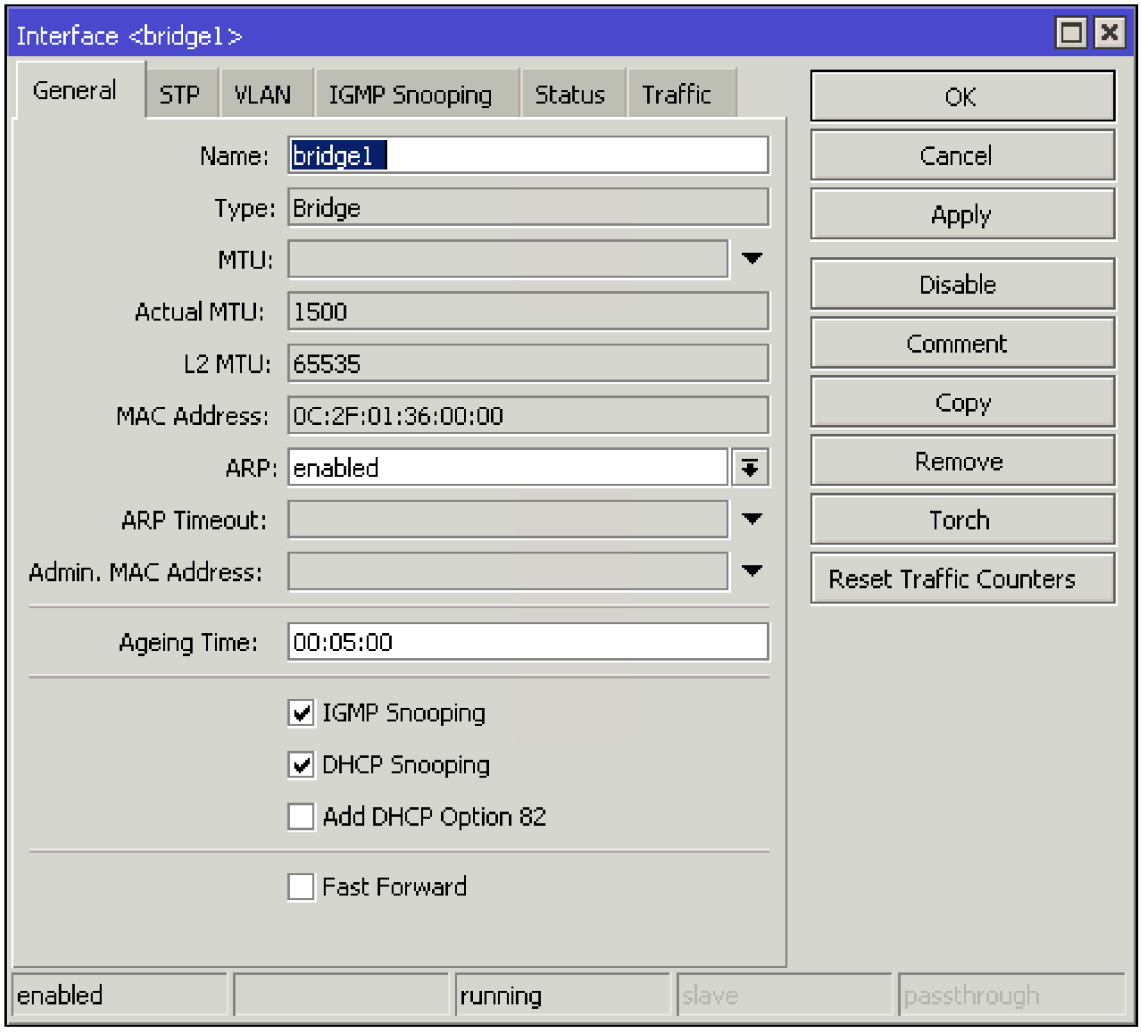
Obr. 6.5: RouterOS nastavení IPv6 adres na rozhraní bridge1 (ether1 a ether2



Obr. 6.6: Výsledky local testů se základní konfigurací



Obr. 6.7: Výsledky local testů při filtrování invalid spojení



Obr. 6.8: RouterOS nastavení snooping

Závěr

Tato diplomová práce se zabývala problematikou bezpečnosti IPv6 a souvisejících protokolů jako je ICMPv6, SLAAC, DHCPv6, Mobility support, NAT64, nebo DNS64. Součástí práce je nástroj pro automatické testování sítí využívajících protokol IPv6. Nástroj byl vytvořen s možností vytvoření vlastních testových scénářů. Součástí vytvořeného nástroje jsou také tři předpřipravené scénáře testující propustnost IPv6 paketů s opakujícím se, nebo repetetivním rozšiřujícím záhlavím, překrývajících se fragmentů a funkčnost Router advertisement a DHCPv6 guard. Nástroj tyto testy vyhodnocuje na základě propuštěných paketů a při detekci zranitelnosti vypíše její popis spolu s možným postupem mitigace.

V první kapitole byl rozebrán důvod vzniku IPv6, jeho porovnání s protokolem IPv4 a detailní popis záhlaví protokolu IPv6 spolu s významem jednotlivých polí. Dále byla probrána adresa IPv6 sítí spolu s typy adres dle jejich použiti na unicast, multicast a anycast. Také byly představeny rozšiřující záhlaví Hop-by-hop options, Routing, Fragment a Destination options spolu s možností No Next Header. Nedílnou součástí protokolu IPv6 je protokol ICMPv6, který obstarává základní funkce jako dynamickou konfiguraci adres, servisní zprávy, automatické nastavení sítě, nebo správu multicastového vysílání. Byla představena funkce Mobility, která umožňuje stanicím zůstat dostupnými i v případě změny sítě, ke které jsou připojeny. Poslední část byla zaměřena na bezpečnostní problémy IPv6, zejména pak nově vzniklé útoky použitím rozšiřujících záhlaví, fragmentací, nebo rizika spojená z dnešního pohledu nevyčerpatelného adresního rozsahu.

Práce poskytla úvod do penetračního testování. Popsala rozdělení útočníků do tří kategorií dle jejich motivace a chování na White hat, Black hat a Gray hat. Následně byly popsány fáze penetračního testování. Tyto fáze se skládají ze shromažďovaní informací (reconnaissance), přípravy (weaponization), doručení exploitu (delivery), spuštění exploitu (exploitation), instalace backdooru (installation), ovládání napadených strojů (command and control).

Praktická část diplomové práce se věnovala vývoji nástroje pro automatické testování síťových prvků a zařízení `Penvuhu6`. Při vývoji bylo využito programu GNS3, sloužícího k emulaci sítě a síťových prvků. Nástroj byl vyvinut v programovacím jazyce Python spolu s využitím knihovny pro vytváření a zachytávání paketů Scapy. `Penvuhu6` byl spouštěn a testován na operačním systému Kali linux. Nástroj umožňuje kromě předem vytvořených scénářů testerům vyvíjet vlastní testové scénáře dvěma způsoby. Prvním z nich je definování testu pomocí souboru JSON. V něm se dále definují pakety k odeslání dle tříd knihovny Scapy. Druhým způsobem je vyvinutí vlastního testu za pomocí jazyka Python a následné jednoduché upravení hlavního programu `Penvuhu6`.

Po vývoji nástroje se práce zaměřila na předpřipravené testové scénáře. Tyto se dělí na local a remote. Local scénáře testují síťová zařízení v lokální síti, zatímco remote testují zařízení mimo lokální síť. Local testy se zaměřují na správnou funkci DHCPv6 guard a RA guard na přepínačích. Remote testy zkoumají propouštění nebezpečných paketů síťovými prvky. Zejména se jedná o pakety se špatně seřazenými, nebo opakujícími se rozšiřujícími záhlavími a také překrývající se fragmenty.

Poslední kapitola se věnovala testování vyvinutého nástroje spolu s emulovaným zařízením s operačním systémem RouterOS. Při jeho prvním použití byl síťový prvek nakonfigurován pouze základní konfigurací. Následně byla testována změna při aplikování restriktivních pravidel. Obě varianty byly následně porovnány a vyhodnoceny.

Vyvinutý nástroj je funkční a zdokumentovaný pro možné budoucí rozšíření o nové testy a funkcionality.

Literatura

[1] POSTEL, J. RFC 760. *RFC Editor* [online]. 1980 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc760

[2] DEERING, S. a R. HINDEN. RFC 1883. *RFC Editor* [online]. 1995 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc1883

[3] JEŘÁBEK, Jan. *Pokročilé komunikační techniky*. 2023. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací Technická 12, 616 00 Brno, 2008. ISBN 978-80-214-4636-6.

[4] POSTEL, J. RFC 791. *RFC Editor* [online]. 1981 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc791

[5] POSTEL, J. RFC 790. *RFC Editor* [online]. 1981 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc790

[6] REYNOLDS, J.K. a J. POSTEL. RFC 870. *RFC Editor* [online]. 1983 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc870

[7] FULLER, V., T. LI, J. YU a K. VARADHAN. RFC 1338. *RFC Editor* [online]. 1992 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc1338

[8] FULLER, V., T. LI, J. YU a K. VARADHAN. RFC 1519. *RFC Editor* [online]. 1993 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc1519

[9] FULLER, V. a T. LI. RFC 4632. *RFC Editor* [online]. 2006 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc4632

[10] REKHTER, Y., B. MOSKOWITZ, D. KARRENBERG a G. DE GROOT. RFC 1597. *RFC Editor* [online]. 1994 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc1597

[11] REKHTER, Y., B. MOSKOWITZ, D. KARRENBERG, G. J. DE GROOT E. LEAR a E. LEAR. RFC 1918. *RFC Editor* [online]. 1996 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc1918

[12] EGEVANG, K. a P. FRANCIS. RFC 1631. *RFC Editor* [online]. 1994 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc1631

[13] SRISURESH, P. a K. EGEVANG. RFC 3022. *RFC Editor* [online]. 2001 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc3022

[14] RAJAHALME, J., A. CONTA, B. CARPENTER a S. DEERING. RFC 3697. *RFC Editor* [online]. 2004 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc3697

[15] AMANTE, S., B. CARPENTER, S. JIANG a J. RAJAHALME. RFC 6437. *RFC Editor* [online]. 2011 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc6437

[16] HINDEN, R. a S. DEERING. RFC 4291. *RFC Editor* [online]. 2006 [cit. 2024-05-19]. Dostupné z: https://www.rfc-editor.org/info/rfc4291

[17] RIPE. IPv6 Address Allocation and Assignment Policy. *RIPE Network Coordination Center* [online]. 2020 [cit. 2024-05-19]. Dostupné z: https://www.ripe.net/publications/docs/ripe-738/

[18] THOMSON, S., T. NARTEN a T. JINMEI. RFC 4862. *RFC Editor* [online]. 2007 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc4862

[19] THOMSON, S. a T. NARTEN. RFC 2462. *RFC Editor* [online]. 1998 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc2462

[20] HABERMAN, B. a D. THALER. RFC 3306. *RFC Editor* [online]. 2002 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc3306

[21] SAVOLA, P. a B. HABERMAN. RFC 3956. *RFC Editor* [online]. 2004 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc3956

[22] IANA. Internet Protocol Version 6 (IPv6) Parameters. *Internet Assigned Numbers Authority* [online]. [cit. 2024-05-20]. Dostupné z: https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml

[23] DEERING, S. a R. HINDEN. RFC 8200. *RFC Editor* [online]. 2017 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc8200

[24] KENT, S. RFC 4302. *RFC Editor* [online]. 2005 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc4302

[25] KENT, S. RFC 4303. *RFC Editor* [online]. 2005 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc4303

[26] HINDEN, R. a G. FAIRHURST. RFC 9268. *RFC Editor* [online]. 2022 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc9268

[27] CONTA, A., S. DEERING a M. GUPTA, ED. RFC 4443. *RFC Editor* [online]. 2006 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc4443

[28] PERKINS, ED., C., D. JOHNSON a J. ARKKO. RFC 6275. *RFC Editor* [online]. 2011 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc6275

[29] GONT, F. RFC 7739. *RFC Editor* [online]. 2016 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc7739

[30] NARTEN, T., E. NORDMARK, W. SIMPSON a H. SOLIMAN. RFC 4861. *RFC Editor* [online]. 2007 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc4861

[31] DEERING, S., W. FENNER a B. HABERMAN. RFC 2710. *RFC Editor* [online]. 1999 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc2710

[32] VIDA, ED., R. a L. COSTA, ED. RFC 3810. *RFC Editor* [online]. 2004 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc3810

[33] PARTRIDGE, C. a A. JACKSON. RFC 2711. *RFC Editor* [online]. 1999 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc2711

[34] FENNER, W. RFC 2236. *RFC Editor* [online]. 1997 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc2236

[35] BAO, C., C. HUITEMA, M. BAGNULO, M. BOUCADAIR a X. LI. RFC 6052. *RFC Editor* [online]. 2010 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc6052

[36] BAO, C., X. LI, F. BAKER, T. ANDERSON a F. GONT. RFC 7915. *RFC Editor* [online]. 2016 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc7915

[37] BAGNULO, M., P. MATTHEWS a I. VAN BEIJNUM. RFC 6146. *RFC Editor* [online]. 2011 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc6146

[38] BAGNULO, M., A. SULLIVAN, P. MATTHEWS a I. VAN BEIJNUM. RFC 6147. *RFC Editor* [online]. 2011 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc6147

[39] MAWATARI, M., M. KAWASHIMA a C. BYRNE. RFC 6877. *RFC Editor* [online]. 2013 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc6877

[40] VYNCKE, É., K. CHITTIMANENI, M. KAEO a E. REY. RFC 9099. *RFC Editor* [online]. 2021 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc9099

[41] GONT, F. RFC 6946. *RFC Editor* [online]. 2013 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc6946

[42] KRISHNAN, S. RFC 5722. *RFC Editor* [online]. 2009 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc5722

[43] DEERING, S. a R. HINDEN. RFC 2460. *RFC Editor* [online]. 1998 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc2460

[44] SAVOLA, P. RFC 3627. *RFC Editor* [online]. 2003 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc3627

[45] KOHNO, M., B. NITZAN, R. BUSH, Y. MATSUZAKI, L. COLITTI a T. NARTEN. RFC 6164. *RFC Editor* [online]. 2011 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc6164

[46] GEORGE, W. RFC 6547. *RFC Editor* [online]. 2012 [cit. 2024-05-20]. Dostupné z: https://www.rfc-editor.org/info/rfc6547

[47] KASPERSKY. Black hat, White hat, and Gray hat hackers — Definition and Explanation. *Www.kaspersky.com* [online]. [cit. 2024-05-20]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/hacker-hat-types

[48] YADAV, Tarun a Arvind Mallari RAO. Technical Aspects of Cyber Kill Chain. *Security in Computing and Communications* [online]. Cham: Springer International Publishing, 2015, 2015-08-08, 438-452 [cit. 2024-05-20]. Communications in Computer and Information Science. ISBN 978-3-319-22914-0. Dostupné z: doi:10.1007/978-3-319-22915-7_40

[49] SHARPE, Richard, Ed WARNICKE a Ulf LAMPING. Wireshark User-s Guide. *Wireshark* [online]. [cit. 2024-05-20]. Dostupné z: https://www.wireshark.org/docs/wsug_html_chunked/

[50] GNS3. Getting Started with GNS3. *GNS3 Documentation* [online]. [cit. 2024-05-20]. Dostupné z: https://docs.gns3.com/docs/

[51] GOWRISHANKAR, S. a A. VEENA. *Introduction to Python programming* [online]. Boca Raton, [2019] [cit. 2024-05-20]. ISBN 978-0815394372.

[52] STATCOUNTER. Desktop Operating System Market Share Worldwide. *StatCounter Global Stats* [online]. [cit. 2024-05-20]. Dostupné z: https://gs.statcounter.com/os-market-share/desktop/worldwide/

[53] UNIVERSITY OF WOLLONGONG. Understanding operating systems. *University of Wollongong* [online]. [cit. 2024-05-20]. Dostupné z: https://www.uow.edu.au/student/support-services/academic-skills/online-resources/technology-and-software/operating-systems/

[54] G0TMI1K. What is Kali Linux? *Kali Linux* [online]. [cit. 2024-05-20]. Dostupné z: https://www.kali.org/docs/introduction/what-is-kali-linux/

[55] MIKROTIK. Manual: RouterOS FAQ. *MikroTik Wiki* [online]. [cit. 2024-05-20]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:RouterOS_FAQ

[56] MIKROTIK. RouterOS. *MikroTik Documentation* [online]. [cit. 2024-05-20]. Dostupné z: https://help.mikrotik.com/docs/display/ROS/RouterOS

[57] MIKROTIK. Manual: License: License Levels. *MikroTik Wiki* [online]. [cit. 2024-05-20]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:License

[58] PENTEREP. Ptlibs 1.0.4. *PyPI* [online]. [cit. 2024-05-20]. Dostupné z: https://pypi.org/project/ptlibs/1.0.4/

[59] PYTHON. Glob - Unix style pathname pattern expansion. *Python documentation* [online]. [cit. 2024-05-20]. Dostupné z: https://docs.python.org/3/library/glob.html

[60] BIONDI, Philippe a SCAPY COMMUNITY. Welcome to Scapy-s documentation!. *Readthedocs* [online]. [cit. 2024-05-20]. Dostupné z: https://scapy.readthedocs.io/en/latest/index.html

Seznam symbolů a zkratek

ISO	International Organization for Standardization
OSI	Open Systems Interconnection
IPv6	Internetový Protokol verze 4
IPv6	Internetový Protokol verze 6
IPv6	Classless Inter-domain Routing
NAT	Network Address Translator
IANA	Internet Assigned Numbers Authority
RIR	Regional Internet Registries
NIR	National Internet Registries
LIR	Local Internet Registries
ISP	Internet Service Provider
RP	Rendezvous Point
SLAAC	Stateless Address Autoconfiguration
RA	Router Advertisement
DAD	Duplicate Address Detection
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
ARP	Address Resolution Protocol
IGMP	Internet Group Management Protocol
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6
NDP	Neighbor Discovery protocol
DoS	Denial of service
GNS3	Graphical Network Simulator 3

RAT	Remote Access Tool
C&C	Command and Control
MITM	Man in the middle
Dos	Denial of Service
VLAN	Virtual Local Area Network
PPP	Point to Point Protocol
L2TP	Layer 2 Tunneling Protocol
REPL	Read Eval Print Loop
MLD	Multicast Listener Discovery
ICMP	Internet Group Management Protocol
MTU	Maximum transmission unit
DUID	DHCPv6 Unique Identifier
UUID	Universally Unique IDentifier

A Obsah elektronické přílohy

Celá aplikace je v elektronické příloze a adresářová struktura je popsaná níže.

- `/` ..... kořenový adresář přiloženého archivu
  - `ptlibs/` ..... knihovny nástroje Penterep, použita verze 1.0.4, ke stažení zde [58]
  - `penvuhu6.py` ..... hlavní program
  - `probe.py` ..... hlavní program sondy
  - `ipv6connection.py` ..... třída správy spojení
  - `message.py` ..... datový typ zprávy
  - `test.py` ..... mateřská třída testů
  - `testextensionsorder.py` ..... testy rozšiřujících záhlaví
  - `testfragmentoverlap.py` ..... testy překrývajích se fragmentů
  - `testRA_DHCPv6_guard.py.py` ..... testy RA a DHCPv6 Guard
  - `utilities.py` ..... pomocné funkce
  - `testcustom.py` ..... třída custom JSON testů
  - `requirements.txt` ..... potřebné python knihovny
  - `CustomExample.json` ..... příklad custom JSON testu
  - `capture/` ..... adresář obsahující průchozí testové pakety
    - `testbadorderheaders-2024-05-19 03-17-34.pcap`
    - `testipv6fragmentoverlap-2024-05-19 03-37-58.pcap`
    - `testrepetetivedestinationheaders-2024-05-16 05-40-34.pcap`
    - `testrepetetivefragmentheaders-2024-05-19 02-51-56.pcap`
    - `testrepetetivehbhheaders-2024-05-19 03-12-07.pcap`
    - `testrepetetiveroutingheaders-2024-05-19 03-07-43.pcap`
  - `diff/` ..... adresář obsahující výsledky testů 6. kapitoly
    - `local/` ..... adresář obsahující zaznamenané local testy
      - `testdhcpv6solicitpass-2024-05-19 06-28-53.pcap`
      - `testdhcpv6solicitpass-2024-05-19 06-31-34.pcap`
      - `testrspass-2024-05-19 06-28-58.pcap`
      - `testrspass-2024-05-19 06-31-39.pcap`
    - `remote/` ..... adresář obsahující zaznamenané remote testy
      - `testbadorderheaders-2024-05-19 05-33-10.pcap`
      - `testbadorderheaders-2024-05-19 06-01-43.pcap`
      - `testbadorderheaders-2024-05-19 06-56-55.pcap`
      - `testipv6fragmentoverlap-2024-05-19 06-01-49.pcap`
      - `testrepetetivedestinationheaders-2024-05-19 05-32-17.pcap`
      - `testrepetetivedestinationheaders-2024-05-19 06-01-06.pcap`
      - `testrepetetivefragmentheaders-2024-05-19 05-30-26.pcap`
      - `testrepetetivefragmentheaders-2024-05-19 05-59-22.pcap`
      - `testrepetetivehbhheaders-2024-05-19 05-31-39.pcap`
      - `testrepetetivehbhheaders-2024-05-19 06-00-29.pcap`
      - `testrepetetiveroutingheaders-2024-05-19 05-30-58.pcap`
      - `testrepetetiveroutingheaders-2024-05-19 05-59-49.pcap`