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

GENEROVÁNÍ ICMPV6 A IPV6 PAKETŮ PRO ZÁTĚŽOVÉ TESTOVÁNÍ POMOCÍ NÁSTROJE JMETER

GENERATE ICMPV6 AND IPV6 PACKETS FOR LOAD TESTING USING JMETER TOOL

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Samuel Šulka

AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Dvořák, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Samuel Šulka ***ID:*** 241116

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Generování ICMPv6 a IPv6 paketů pro zátěžové testování pomocí nástroje JMeter

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte z literatury problematiku fungování protokolů IPv6, ICMPv6, DHCPv6 a NAT64. Dále nastudujte problematiku nástroje JMeter a existujících rozšiřujících modulů. V rámci bakalářské práce vytvořte minimálně čtyři scénáře testování zátěže zejména pomocí protokolu ICMPv6 v prostředí bez IPv4. Vytvořte rozšiřující modul pro nástroj JMeter a jeho grafické rozhraní, který umožní nastavení potřebných parametrů testu. Kompletní specifikace musí být předem schválena vedoucím práce. Vytvořené scénáře otestujte a získané výsledky vhodně zpracujte a komentujte.

DOPORUČENÁ LITERATURA:

[1] RODRIGUES, Antonio Gomes, Bruno DEMION a Philippe MOUAWAD. Master Apache JMeter - From Load Testing to DevOps: Master performance testing with JMeter. Packt Publishing Ltd, 2019.
[2] ATAR, Afsana. Mastering JMeter 5.0. Packt Publishing, 2020.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: Ing. Jan Dvořák, Ph.D.

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

ABSTRAKT

Bakalárska práca sa zaoberá záplavovými útokmi pomocou generovania ICMPv6 paketov v prostredí bez IPv4 protokolu. Bol vytvorený Trafgen konfiguračný súbor, ktorý mal za úlohu posielanie ICMPv6 paketov. Pre ostatné scenáre bol využitý Scapy v pythone. Pre každý scenár bol vytvorený zásuvný modul do softwaru Apache JMeter. Scenáre boli otestované na reálnom zariadení a výsledky boli zdokumentované v písomnej i vizuálnej podobe.

KĽÚČOVÉ SLOVÁ

ICMPv6, IPv6, JMeter, testovanie, záplavové útoky, DoS

ABSTRACT

The Bachelor Thesis deals with flood attacks using generating of ICMPv6 packets in an environment without IPv4 protocol. A Trafgen configuration file was created, which had the task of sending ICMPv6 packets. For other attacks was used Scapy in Python. A plug-in modul for software Apache JMeter was created for each scenario. Scenarios were tested on real device and the results were documented in write and visual form.

KEYWORDS

ICMPv6, IPv6, JMeter, testing, flood attacks, DoS

ŠULKA, Samuel. *Generování ICMPv6 a IPv6 paketů pro zátěžové testování pomocí nástroje JMeter*. Bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedúci práce: Ing. Jan Dvořák, Ph.D

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Samuel Šulka
VUT ID autora:	241116
Typ práce:	Bakalárska práca
Akademický rok:	2023/24
Téma závěrečnej práce:	Generování ICMPv6 a IPv6 paketů pro zátěžové testování pomocí nástroje JMeter

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som poďakoval vedúcemu práce pánovi Ing. Janovi Dvořákovi, Ph.D. za odborné vedenie a užitočné rady a mojej rodine za podporu.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V dnešnej dobe sú záplavové útoky čoraz častejšie a potencionálni útočníci ich vykonávajú čím ďalej, tým viac sofistikovane, a preto je z rôznych bezpečnostných dôvodov dobré vedieť, ako tieto útoky fungujú, akú záťaž naše servery dokážu zniesť a zaistiť tak do istej miery bezpečnosť serveru. Záplavové útoky s využitím ICMPv6 protokolu sú o to viac nebezpečnejšie, lebo sa im dá ťažšie predísť z dôvodu vysokej závislosti protokolu IPv6 na protokole ICMPv6. K tomuto pochopeniu útokov a otestovaniu koncového zariadenia nám pomôže Software Apache JMeter. Výhodou tohoto záťažového testovania je možnosť overiť fungovanie a výkon systému pri viacerých používateľoch naraz. Bakalárska práca sa bude zaoberať konkrétne zaťažovaním systému s ICMPv6 paketmi.

Cieľom bakalárskej práce je vytvorenie štyroch testovacích scenárov záplavového útoku s ICMPv6 a IPv6 protokolom a modulu s grafickým rozhraním.

Na úvod bakalárskej práce je vysvetlené, ako fungujú protokoly IPv6, ICMPv6, DHCPv6, NAT64, DNS64 a nakoniec program JMeter, Trafgen a Scapy. Zároveň je dôležité spomenúť rôzne bezpečnostné a výkonové faktory, ktoré ovplyvňujú fungovanie týchto protokolov.

V praktickej časti bakalárskej práce je v úvode všeobecne vysvetlené, ako fungujú testovacie scenáre, aké nástroje a knižnice sú najviac využité. Následne je vymenovaný hardware, ktorý je použitý v topológií. V nadchádzajúcej sekcií sú popísané detailne kroky inštalácie a všetko, čo je potrebné k spusteniu modulov. Nasleduje kompletný popis modulu, konfiguračného súboru Trafgen a popis grafického rozhrania pre jednoduchý ICMPv6 záplavový útok. Tri scenáre sú vytvorené pomocou Python knihovni Scapy. Knihovňa Scapy je využitá na manipuláciu s paketmi a na generovanie paketov. V ďalšej sekcii je rozobratá samotná topológia testovacieho prostredia. Na záver praktickej časti sú prezentované testovacie scenáre podrobným popisom.

Výsledky sú následne detailne zdokumentované a vizuálne podložené.

Cieľ práce

Cieľom bakalárskej práce je pochopenie príslušných protokolov, ktoré boli využité v rámci praktickej časti na vývoj modulu a grafických rozhraní pre rôzne testovacie scenáre na generovanie ICMPv6 a IPv6 paketov, za účelom záplavových útokov. Cieľom je vytvorenie štyroch testovacích scenárov a ich implementovanie do modulov, pomocou ktorých dokáže užívateľ spustiť tieto testy. Cieľom je taktiež vyskúšať rôzne druhy útokov a či tieto útoky majú nejaký vplyv na obeť.

1 Teoretická časť

V teoretickej časti sú popísané iba relevantné protokoly k riešeniu praktickej časti bakalárskej práce. Táto kapitola obsahuje podrobný popis protokolov použitých v tejto práci, ich funkčnosť a využitie v dnešnej dobe. Na úvod je potrebné si predstaviť model TCP/IP a jeho vrstvy. Nasledujúce kapitoly sú orientované výhradne na protokol IPv6 a protokoly, ktoré používajú IPv6.

1.1 Protokoly sieťovej vrstvy

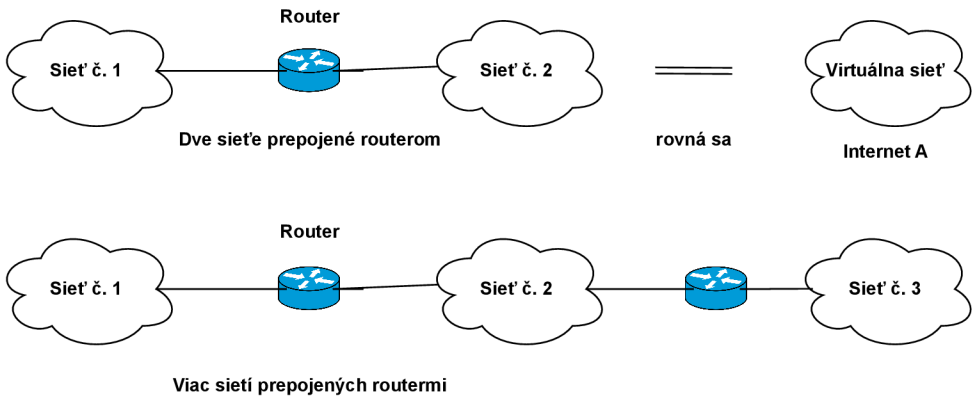
V tejto časti bakalárskej práce sú popísané protokoly sieťovej vrstvy aj z pohľadu možných DoS útokov. Predovšetkým sú popísané protokoly IPv6, ICMPv6.

1.1.1 TCP/IP

ARPAnet bol vytvorený v roku 1969 s cieľom vytvorenia veľkej siete, kde bude možné prepínať pakety a tento celý veľký projekt siete podporila ARPA. Podarila sa im vybudovať sieť, ktorá bola používaná každý jeden deň a sieť sa stala operatívnou kvôli jej úspešnosti. V roku 1983 museli užívatelia TCP/IP konvertovať z tohoto štandardu na novšie protokoly, pretože sa TCP/IP uchytila ako vojenský štandard. [8]

Vznik internetu sa priamo viaže k vzniku štandardu TPC/IP, pretože až vtedy vznikol pojem internet. Dnešný internet obsahuje viac ako stovky tisícok sietí po celom svete a tieto menšie siete tvoria internet ako celok. Internet dnes nie je závislý len na jednej „jadrovej“ sieti, ale internet poskytujú rôzni internetoví poskytovatelia a títo poskytovatelia ponúknu užívateľovi svoje služby pomocou lokálneho prístupu k internetu. Poskytovateľov služieb je po svete veľa. Spoločnosti si neskôr začali všímať TCP/IP štandardu a začali zisťovať rôzne výhody, ako napríklad rôzne sieťové aplikácie. Transmission Control Protocol/Internet Protocol (TCP/IP) je súbor štandardizovaných pravidiel, vďaka ktorým je umožnená komunikácia medzi počítačmi a pozostáva z dvoch protokolov: TCP (Transmission Control Protocol) a IP (Internet Protocol). Názov TCP/IP odkazuje na komplexný „balík“ dátových protokolov, ktoré sú komunikačného charakteru. TPC/IP je známy a častokrát označovaný pod názvom Internet Protocol suite. I keď je OSI model dostatočne užitočný a funkčný, TCP/IP model má inú štruktúru vrstiev, a teda sa tieto modely od seba odlišujú. Od modelu OSI sa oddeľuje počtom vrstiev, ktoré má iba 4 a model OSI má týchto vrstiev 8. V TCP/IP vykonáva jedna vrstva veci, ktoré museli v OSI modely vykonávať napríklad dve vrstvy. TCP/IP mala primárne spájať siete a i cez ich nerovnomernosť štruktúry fyzickej siete, doručiť servis pre komunikáciu. Existujú rôzne

druhy sietí ako napríklad takzvané *Backbones*, ktoré sú veľmi veľkých rozmerov a ich primárnou funkciou je prepájať siete. *Backbones* môžu slúžiť aj ako prístupové body. Ďalej existujú malé regionálne siete, ktoré napríklad prepájajú rôzne univerzity medzi sebou alebo objekty. V neposlednom rade sú tzv. komerčné siete, ktoré zákazníkovi poskytnú prístup k veľkým sieťam ako napríklad, *Backbones* siete. Ako posledné je treba spomenúť aj lokálne siete, ktoré sú väčšinou využívané v domácnostiach alebo rôznych objektoch. Väčšinou sa určuje počet užívateľov, ktorý sa pripoja k sieti, dopredu podľa geografickej veľkosti siete, kedy je napríklad Ethernet sieť obmedzovaná práve touto veľkosťou. Aby mohli dvaja užívatelia komunikovať medzi sebou bez ohľadu na to, že je každý z inej siete, je potreba prepojiť veľké množstvo sietí hierarchicky. [14]



Obr. 1.1: Dve prepojené siete.

TCP/IP model je rozdelený do štyroch vrstiev. Ďalšie modely, ktoré sú rozdelené na vrstvy, sú napríklad SNA (Systems Network Architecture) alebo vyššie spomínaný OSI (Open System Interconnection). Tieto modely by sa však nemali porovnávať medzi sebou, pretože sú tam rozdiely vo vrstvových modeloch a používajú rôzne protokolové balíky.

Aplikačná vrstva

Aplikačné programy priamo komunikujú s transportnou vrstvou a aplikačná vrstva je najvyššou vrstvou v modeli TCP/IP. Na rozdiel od modelu ISO/OSI si aplikácie na aplikačnej vrstve musia relačné a prezentačné služby vykonávať samé. [15] Používa sa program, ktorým TPC/IP komunikuje. Aplikáciu možno definovať ako bežiaci proces, ktorý spolupracuje s iným procesom od druhého hostiteľa. Aplikácie ako Telnet a *File Transfer Protocol* (FTP) alebo *Simple Mail Transfer Protocol* (SMTP) sú najčastejším príkladom týchto aplikácií. Rozhranie, ktoré umožňuje komunikáciu

dvoch vrstiev, a to komunikačnej a aplikačnej, je definované portom a soketmi a toto rozhranie sa nachádza medzi aplikačnou vrstvou a transportnou vrstvou. [14]

Transportná vrstva

Druhá vrstva po aplikačnej vrstve je používaná na prenos dát ku peerovi, ktorý je vzdialený aplikácii, pomocou end-to-end modelu. Veľkou výhodou je, že dokáže obsluhovať viacero aplikácií naraz. Používaný je primárne TCP protokol, ktorý kontroluje celý tok dát, zaručuje spoľahlivé údaje orientované hlavne na pripojenie a prebieha eliminácia duplikátne posielaných údajov. Ďalej sa vyskytuje v tejto vrstve UDP (User Datagram Protokol), ktorý nie je až tak efektívny a spoľahlivý ako predchádzajúci TCP protokol. Je vo väčšine prípadov nespoľahlivý a ponúka takzvaný *Best-effort service*, čo znamená, že tento protokol nespĺňa podmienky pre spoľahlivú službu, i keď sa o to snaží s najväčším úsilím. Keďže niektoré aplikácie tento protokol využívajú, musia si zaobstarať kontrolu toku a všetky ostatné funkcie, ktoré UDP nedoručí. Ak sú ale aplikácie, ktorým nevadí menšia strata dát a potrebujú rýchly prenos, je pre nich protokol UDP ideálny na použitie. [14]

Sieťová vrstva

Tretia vrstva TCP/IP hierarchie, nazývaná ako internetová vrstva, poskytuje pre systém doručovanie dát do ostatných zariadení na sieti. Najviac používaný a pochopiteľne najviac významný protokol je *Internet Protocol* alebo IP. Tento protokol neposkytuje kontrolu toku alebo podobné funkcie, ktoré boli doručované vo vyšších vrstvách. Tento protokol funkciou smerovania dopomáha k prenosu správ do ich cieľa. Ďalej sa v „internetovej“ vrstve vyskytuje protokol *Reverse Address Resolution Protocol*(RARP), *Internet Control Message Protocol*(ICMP), *Address Resolution Protocol*(ARP) a *Internet Group Management Protocol*(IGMP). [14]

Vrstva sieťového rozhrania

Vrstva sieťového rozhrania alebo inak nazývaná i „linková vrstva“, je rozhranie pre sieťový hardvér. Rozhranie môže byť paketovo alebo streamovo orientované a nie vždy je spoľahlivé čo sa týka doručovania. Model TPC/IP vyznačuje flexibilitu IP vrstvy z dôvodu možnosti použitia v podstate ktoréhokoľvek sieťového rozhrania, ktoré je aktuálne k použitiu. Dostatočne spoľahlivými príkladmi sú IEEE 802.2 alebo aj menej známy X.25. Ako bolo už vyššie spomenuté, tak žiadne protokoly sieťovej vrstvy nie sú modelom TCP/IP štandardizované, a teda štandardizovaný je iba postup ako sa k rôznym protokolom pristupuje priamo zo sieťovej vrstvy. [14]

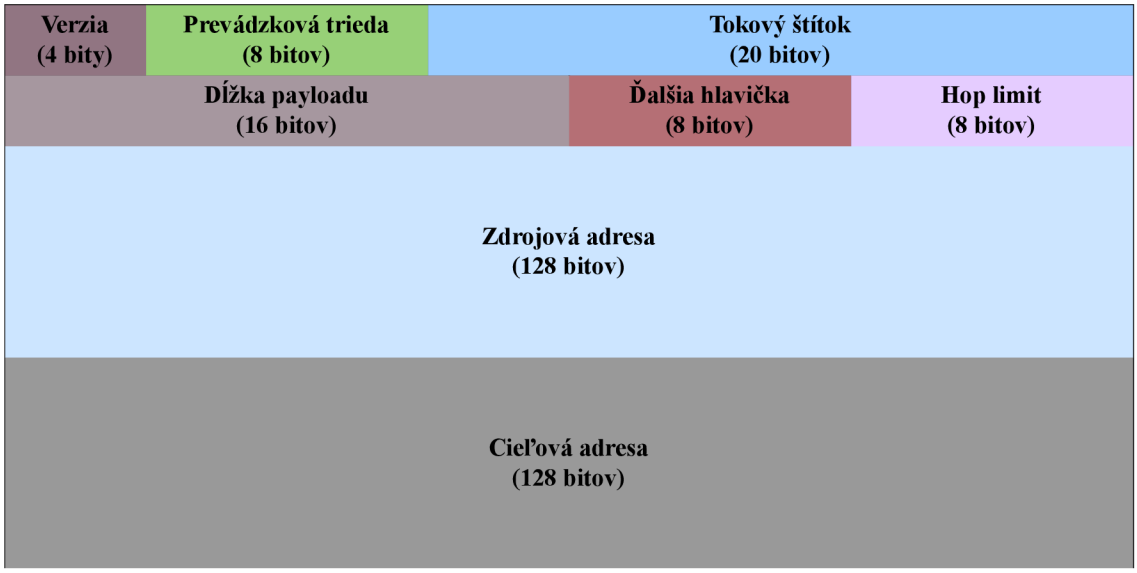
1.1.2 Protokol IPv6

Internet protocol verzie 6 (IPv6) je následovník plošnejšie používaného a aktuálneho protokolu *Internet protocol* verzie 4 (IPv4). Na jeho vývoji majú najväčší podiel Steven Deering a Robert Hinden, ktorí napokon vydali súbor viacerých dokumentov RFC (*Internet Protocol, Version 6(IPv6) Specification*), ktoré vymedzujú základné špecifikácie protokolu IPv6. [20]

Migrácia adries z IPv4 na IPv6 bude trvať ešte dlhé roky, nie je vymedzený začiatok tejto migrácie, kedy sa vlastne celá začala. Oba protokoly spadajú do sieťovej vrstvy OSI a väčšinu zraniteľností, ktoré sú objavené v IPv4, budú pravdepodobne škodlivé aj v protokole IPv6. Jednou z týchto zraniteľností je napríklad aj útok „Denial of service“, inak známy ako DoS alebo tiež známy Man-in-the-middle útok. [18]

Hlavička IPv6 protokolu sa skladá z:

– Verzie (Version), ktorej hodnota je 6
– Prevádzková trieda má za úlohu identifikovať priority paketu a servisnú triedu tohoto paketu
– Tokový štítok slúži k identifikácií paketov v rámci rôznych unikátnych tokov
– Dĺžka payload sa určuje v oktetoch a určuje dĺžku paketu, ktorý nasleduje
– Pole ďalšej hlavičky označuje, aká hlavička nasleduje po IPv6 hlavičke
– Pole Hop limit počíta paketu počet hopov, ktoré slúžia na cestovanie paketu a táto hodnota sa odpočítava po prejdení paketu cez nejaké sieťové zariadenie - náhrada TTL.



Obr. 1.2: IPv6 Hlavička

Protokol IPv6 sa vyvíjal veľmi dlhé roky a napriek tomuto faktu, sa považuje za mladý protokol spomedzi všetkých ostatných protokolov. Hlavným dôvodom pre vývojárov k vývoju protokolu IPv6 bol nepochybne malý adresný priestor v protokole IPv4, ktorý bol limitáciou. Dá sa predpokladať, že protokol IPv4 dokáže podporovať niečo okolo 4 miliardy adries, avšak toto číslo sa môže trochu líšiť v praxi. V praxi sa odhaduje, že protokol IPv4 by dokázal uniesť 250 miliónov adries, ktoré sú celkom unikátne. [18]

Adresa protokolu IPv6 je zväčša zapisovaná v hexadecimálnej podobe, pretože pozostáva zo šestnástich bajtov a každé dva bajty (tzn. štyri znaky adresy) sú oddelené dvojbodkou. Adresný priestor IPv6 je 128 bitov, a teda užívatelia IPv6 majú na výber z $3,4 \cdot 10^{38}$ adries. Podla výsledkov APNIC používa IPv6 veľa svetových krajín a celkovo sa svet približuje k celosvetovému používaniu protokolu IPv6 k 20 percentám. Z tých väčších krajín vedú USA s Indiou (skoro 50%). V rámci Európy je prvé Belgicko (52%), ďalej Nemecko (38%) a Grécko (34%). Česká republika si s takmer 10 percentami nevedie oproti iným krajinám najlepšie. [20]

1.1.3 IPv4

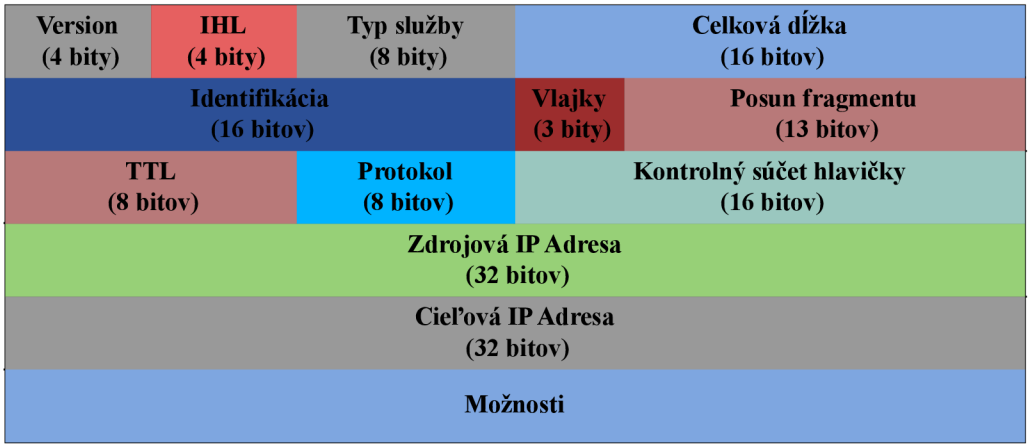
Internet protocol verzie 4 je predchodcom IPv6 ale v tejto dobe je oproti nemu viac využívaný a použiteľnejší ako IPv6 protokol. Používa 32 bitov pre adresovanie, a to je najväčšou nevýhodou oproti IPv6 protokolu. Keďže ľudská populácia má tendenciu rásť každým rokom tak, USA správne dedukuje, že do roku 2050 by mohol počet ľudí na svete vzrásť až na hranicu 10.9 miliardy, čo by mohlo znamenať problém

pre internet a celkovo sieťové odvetvie. Teda matematicky je jasné, že protokol IPv4 nebude k tomuto účelu dostatočný. IPv4 je v službe už viac ako 41 rokov a po celý čas sa považuje za celkovo spoľahlivý protokol. Do produkcie v ARPAnete sa dostal v priebehu roku 1983. IPv4 potrebuje protokol NAT pre adresné preklady a prekladá súkromné adresy na verejné, funguje to i naopak. Cieľom používania NAT pri IPv4 bolo dočasne rozšíriť adresnú kapacitu. [18]

Existuje tzv. „Stateful NAT64“ (stavový), ktorý funguje ako mechanizmus prenášajúci pakety z IPv6 do IPv4 a z IPv4 do IPv6. Prekladanie prebieha prekladaním priamo hlavičiek paketov podľa už určeného algoritmu pre preklad IP/ICMP. [11]

Hlavička IPv4 (viď obr. 1.3) protokolu sa skladá z:

– Verzie, ktorej hodnota je 4
– IHL prezentuje dĺžku internetovej hlavičky
– Pole typ služby usmerňuje parametre, ktoré sú abstraktné a tieto parametre slúžia k výberu tých skutočných parametrov pre službu
– Celková dĺžka je uvádzaná v oktetoch a udáva dĺžku celého datagramu aj s dátami a s internetovou hlavičkou. Minimálna dĺžka je 576 oktetov
– Pole identifikácia slúži pri zostavovaní fragmentov a priraďuje sa odosielateľovi
– Pod polom vlajky alebo inak príznaky sú schované kontrolné vlajky, ktoré sa týkajú fragmentácie
– Posun fragmentu určuje, kde sa má fragment nachádzať v rámci datagramu
– Pole TTL je čas, ktorý označuje, koľko môže datagram stráviť v systéme. Zahadzuje sa, ak je hodnota nula
– Protokol označuje aký protokol sa používa v internetovom datagram v dátach
– Kontrolný súčet hlavičky existuje kvôli zmenám polí v rámci hlavičky, ktoré musí zaznamenávať
– Nasledujú IP adresy
– Možnosti sú variabilné a užívateľ ich nemusí použiť. [17]



Obr. 1.3: IPv4 Hlavička

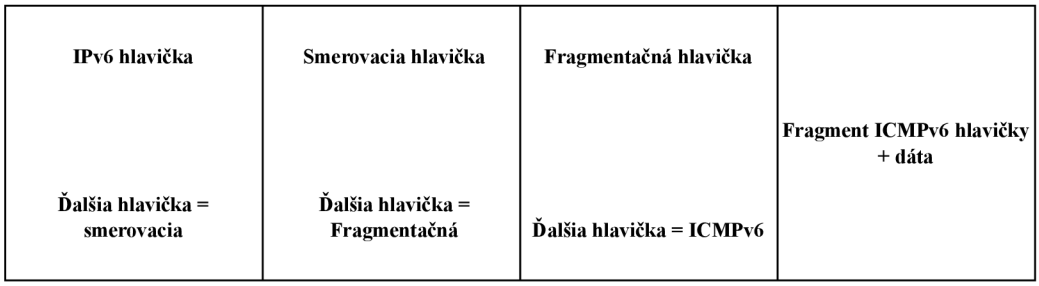
1.1.4 Dôvody prechodu na IPv6

I keď nie je IPv6 dostupná plne pre všetkých a všade, tak tento fakt sa postupom času mení a viac poskytovateľov služieb podporuje IPv6 protokol dnes. Je dosť pochopiteľné, že pre niektoré firmy sa neoplatí IPv6 protokol zaviesť z ekonomických alebo iných rôznych dôvodov. Je potrebné ale zhrnúť dôvody k prechodu na IPv6, ktoré by mohli tieto firmy presvedčiť. Samozrejme prvým a najdôležitejším dôvodom je oveľa väčší adresný priestor, pretože IPv6 prešlo z 32 bitového na 128 bitové adresovanie, presnejšie $3,4 \cdot 10^{38}$ unikátnych adries, čo je mnohonásobok toho, čo ponúka starší protokol verzie 4. Lokálne IPv4 adresy sa vyskytujú častokrát v kombinácii s prekladačom adries NAT, ktorý ich mení na verejné adresy, a týmto poskytuje užívateľovi prístup k internetu. NAT tu vykonáva mapovanie adries a portov transportnej vrstvy a celá užívateľova lokálna sieť je schovaná pod adresu, ktorá je potom viditeľná na verejnom internete. [20] Napríklad pri NAT64 je možné komunikáciu spustiť bez obmedzenia len zo strany protokolu IPv6 a z opačnej strany je komunikácia obmedzená [16]. Pri IPv6 nie je vôbec potrebné používať NAT a nemá to žiaden väčší zmysel. Kľúčovým dôvodom je taktiež zložitá infraštruktúra pri používaní IPv4 adresy, pretože IPv4 nemá dostatok adries a používanie NAT neverejných adries k tejto zložitosti tiež prispieva. [20]

1.1.5 Rozšírené hlavičky IPv6

V rámci tejto kapitoly sú zhrnuté všetky využité rozšírené hlavičky protokolu IPv6, ktoré boli využité na záplavové útoky spolu s protokolom ICMPv6.

Rozšírené hlavičky sú označené v poli ďalšia hlavička (Next Header) a počet týchto hlavičiek je obmedzený. Rozpoznávanie týchto hlavičiek je vyriešené pomocou Next Header hodnoty, ktorá je pre každú hlavičku unikátna. Čísla týchto rozšírených hlavičiek sú identické ako pre protokol IPv6, tak pre protokol IPv4. V rámci jedného IPv6 paketu sa môže vyskytnúť niekoľko rozšírených hlavičiek, ale i jedna alebo vôbec žiadna a každá sa musí nachádzať v poli ďalšej hlavičky s vlastnou priradenou hodnotou. Uzle nedokážu spracovať, vložiť a dokonca ani vymazať tieto rozšírené hlavičky ak paket ešte len prichádza do destinácie, avšak hop-by-hop je výnimkou. Tieto úkony dokáže uzol vykonať hneď ako paket príde k cieľovému uzlu, ktorý bol v poli cieľovej adresy označený. Naopak pri hlavičke hop-by-hop je možné, že niektorý z uzlov ju preskúma už počas cesty k uzlu v poli cieľovej adresy a taktiež ju môže náhodný z uzlov aj spracovať. Táto hlavička musí bezprostredne nasledovať za hlavičkou IPv6 protokolu a pritom je táto rozšírená hlavička identifikovaná hodnotou 0 v poli ďalšej hlavičky. [5]
Protokol IPv6 ponúka nasledujúce rozšírené hlavičky:
– Hop-by-Hop (Hop-by-Hop Options header)
– Fragmentačná hlavička (Fragmentation header)
– Možnosť destinácie (Destination Options header)
– Smerovacia hlavička (Routing header)
– Overovacia hlavička (Authentication header)
– Zapúzdrovacia hlavička (Encapsulating Security Payload header) [5]



Obr. 1.4: Príklad viacerých rozšírených hlavičiek po sebe

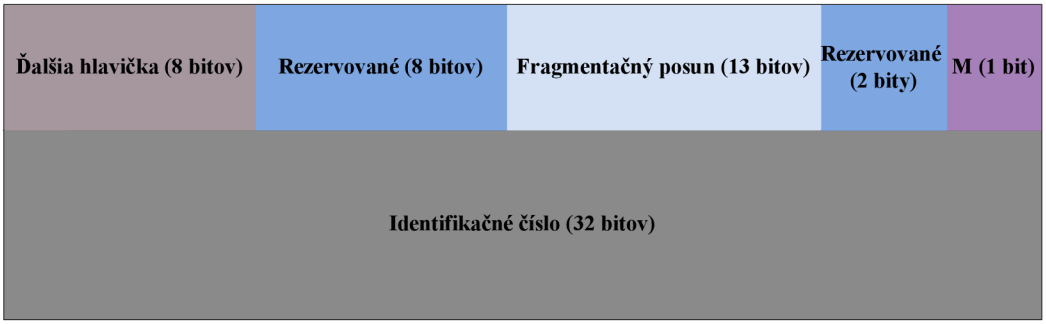
Rozšírené hlavičky majú striktne odporúčané poradie podľa ktorého sa radia v pakete:
1. začínajúc hlavičkou IPv6
2. nasleduje hlavička Hop-by-Hop
3. hlavička Možnosti destinácie
4. Smerovacia hlavička

5. Fragmentačná hlavička
6. Overovacia hlavička
7. Zapúzdrovacia hlavička
8. Hlavička hornej vrstvy

Všetky vyššie spomenuté hlavičky je možné použiť jedenkrát, avšak hlavička Možnosti destinácie je použiteľná dvakrát ako jediná. [5]

Fragmentačná hlavička

Protokol IPv6 používa túto rozšírenú hlavičku fragmentácie na odosielanie veľkých paketov, ktoré by neprešli sieťovým filtrom MTU. V prípade IPv6 sa fragmentácia nevykonáva v rámci smerovaču, cez ktorý paket prechádza, ale fragmentácia je vykonávaná pomocou zdrojových uzlov. Hlavička je číselne označená v poli ďalšej hlavičky protokolu IPv6 hodnotou 44. [5]



Obr. 1.5: Formát fragmentačnej hlavičky

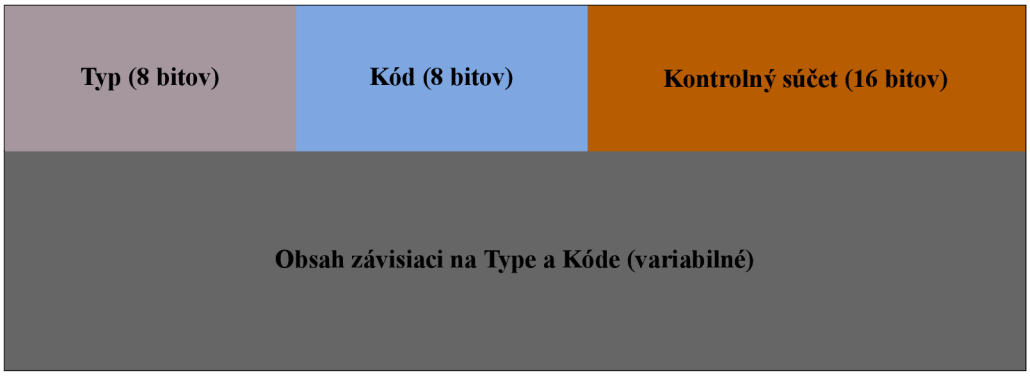
Formát fragmentačnej hlavičky:

– Pole Ďalšia hlavička má dĺžku 8 bitov a jeho úlohou je rozpoznať druh prvej hlavičky a časť paketu, ktorú je možné rozdeliť
– Pole Rezervované s dĺžkou 8 bitov je uvedené pre prenos a to s hodnoutou 0
– Pole Fragmentačný posun s dĺžkou 13 bitov, tieto bity sú v celých číslach. Toto pole je určené na posun dát, ktoré nasledujú za touto hlavičkou. Prihliada sa na začiatok časti paketu, ktorú je možno fragmentovať
– Pole Rezervované s dĺžkou 2 bitov je uvedené pre prenos a to s hodnotou 0
– Toto pole M je informácia, či príde ďalší fragment po tomto alebo nie. Ak nasleduje fragment, hodnota je 1 a ak nie, hodnota je 0
– Pole Identifikačného čísla s dĺžkou 32 bitov [5]

1.1.6 Protokol ICMPv6

Ak nie je uvedené inak, zdrojom informácií pre túto sekciu je *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification.* [7]

Protokol ICMPv6 a ICMP majú veľmi podobné fungovanie v oboch verziách protokolov IP či už verzie 6 alebo verzie 4, avšak sú viditeľné menšie zmeny vo fungovaní. Hodnotou ďalšej hlavičky v protokole IPv6 je číslo 58 označované ako ICMPv6. Pri spracúvaní paketov sa môže častokrát vyskytnúť nejaký problém alebo chyba a presne to je úlohou ICMPv6 protokolu, ktorý využíva IPv6 uzly na detekovanie a následné nahlásenie týchto chýb. Pomocou echo-requestu alebo inak nazývaný ICMPv6 ping je možné diagnostikovať sieť. Je potrebné, aby každý uzol IPv6 mohol implementovať ICMPv6 protokol. Pre porovnanie je hodnota ICMPv4 ďalšej hlavičky číslo 1. Kódy a typy správ ICMPv6 sa odlišujú od ICMPv4 nielen číslami, ale i chybovými správami, teda nie všetky ICMPv6 správy majú ekvivalent v ICMPv4. Pred každou jednou správou ICMPv6 protokolu sa nachádza hlavička IPv6 obsahujúca žiadnu alebo viacero rozšírených hlavičiek, napr. „Hop-by-Hop“ s kódom 0 alebo podobné. ICMPv6 hlavička je vždy označená v poslednej hlavičke, ktorá je pred ňou, a teda v prípade rozšírených hlavičiek je označená v poslednej rozšírenej hlavičke. [7]



Obr. 1.6: Formát ICMPv6 správy

Popis formátu ICMPv6 správy (viď obr. 1.6):

Pole „Typ“ určuje typ správy a definuje údaje, ktoré nasledujú a ich celú štruktúru. Pole „Kód“ je priamo závislé od typu správy a poskytuje ďalšie detaily o obsahu chybovej alebo informačnej správy k dosiahnutiu prídavnej úrovni detailnosti správy. Pole „Kontrolný súčet“ identifikuje poškodenie údajov v ICMPv6 správach a rôznych

častiach IPv6 hlavičky. [7]
Existujú viaceré ICMPv6 správy a rozdeľujú sa medzi chybové správy a informačné.

Ako bolo spomenuté v začiatku kapitoly, ICMPv6 je primárne na prenos chybových správ a informačných správ, ktoré vznikli pri spracovaní paketov, ale i na diagnostiku (*ICMPv6 "ping"*). Týchto správ je mnoho druhov a všetky používanejšie sú spomenuté v tab. 1.2, tab. 1.3 a tab. 1.1. [7]

Tab. 1.1: ICMPv6 Chybové správy - typy

Typ	**Názov**	**Preklad**
1	Destination Unreachable Message	Nedosiahnuteľná destinácia
2	Packet Too Big Message	Príliš veľký paket
3	Time Exceeded Message	Čas prekročený
4	Parameter Problem Message	Problém s parametrom

Tab. 1.2: ICMPv6 Chybové správy - kódy

Typ	**Kód**	**Názov**	**Preklad**
1	0	No route to destination	Žiadna cesta k cieľu
1	1	Communication administratively prohibited	Komunikácia administratívne zakázaná
1	2	Beyond scope of source address	Mimo rozsahu zdrojovej adresy
1	3	Address unreachable	Adresa nedosiahnuteľná
1	4	Port unreachable	Port nedosiahnuteľný
1	5	Source address failed ingress/egress policy	Zdrojová adresa nevyhovela vstupu/výstupu
1	6	Reject route to destination	Odmietnutá cesta k cieľu
2	0	Packet Too Big	Príliš veľký paket
3	0	Hop limit exceeded in transit	Prekročený limit skokov počas prenosu
3	1	Fragment reassembly time exceeded	Čas na zloženie fragmentov prekročený
4	0	Erroneous header field encountered	Nájdené chybné pole v hlavičke
4	1	Unrecognized Next Header type encountered	Nájdený nerozpoznaný typ nasledujúcej hlavičky
4	2	Unrecognized IPv6 option encountered	Nájdená nerozpoznaná možnosť IPv6

Tab. 1.3: ICMPv6 Informačné správy - kódy

Typ	**Kód**	**Názov**	**Preklad**
128	0	Echo Request	Žiadosť o ozvenu
129	0	Echo Reply	Odpoveď ozveny
130	0	Multicast Listener Query (MLD)	Dotaz poslucháča multicastu
131	0	Multicast Listener Report (MLD)	Správa poslucháča multicastu
132	0	Multicast Listener Done (MLD)	Ukončenie poslucháča multicastu
133	0	Router Solicitation (NDP)	Žiadosť o router (NDP)
134	0	Router Advertisement (NDP)	Reklama na router (NDP)
135	0	Neighbor Solicitation (NDP)	Žiadosť o suseda (NDP)
136	0	Neighbor Advertisement (NDP)	Reklama na suseda (NDP)
137	0	Redirect Message (NDP)	Presmerovacia správa (NDP)

1.2 Protokoly aplikačnej vrstvy

1.2.1 Protokol DHCPv6

Protokol IPv6 používa protokol DHCPv6 na konfiguráciu a primárne na priraďovanie adries k zariadeniam a aktívne má kompetenciu na spravovanie adries pre užívateľov. Priraďuje globálne IPv6 adresy. Pre kontrolu prístupu používa protokol DHCPv6 v porovnaní so SLAAC stavový režim a týmto správu veľmi uľahčuje. DHCPv6 je veľmi používané a jeho využiteľnosť má veľký rozsah.

Pri vývoji DHCPv6 sa príliš nezamýšľalo nad samotnou bezpečnosťou protokolu a to viedlo k mnoho rôznym útokom ako je napríklad tzv. nečestný server, ktorý dokáže zhodiť veľké množstvo sieťových pripojení, ktoré spravuje tento DHCPv6 server. DHCPv6 správy dokážu byť pre užívateľa nebezpečné ak sú z čistého textu. Tieto správy dokážu o užívateľovi vyzradiť citlivé informácie o jeho identite. Útočník dokáže zneužiť i sledovanie procesu konfigurácie samotnej IPv6 adresy a tak dokáže sledovať čo užívateľ na sieti vykonáva alebo si môže útočník vystopovať užívateľovu polohu. [10]

1.2.2 Protokol DNS64

DNS64 sa využíva na syntézu zdrojových záznamov, označených ako AAAA, z prostriedkov záznamu A. Hostiteľom s IPv6 poskytuje kvalifikované doménové meno uzla IPv4 pre začiatok komunikácie. Pri začiatku komunikácie iniociovanej od uzla, ktorý potom čaká na IPv6 cieľovú adresu pomocou správy AAAA RR. Tu sa potom využije funkcia DNS64, kedy sa syntetizuje AAAA záznam zo záznamu A, aby sa uzol IPv6 bol schopný naučiť adresu cieľa. Je potrebné ale spomenúť, že DNS64 je iba doplnkom pre DNS. Ak pri prekladači, ktorý má nastavený a povolený DNS64, príde dotaz na AAAA RR od uzla s IPv6 tak DNS64 hľadá priamo AAAA RR a ak AAAA RR neexistuje, potom sa posiela dotaz na A. Následne po nájdení A DNS64 po pridaní Prfe64::/n pomocou NAT64 vytvorí syntetické AAAA RR a vytvorí to pre odpovedajúcu IPv4 adresu iba za podmienky, že je n menšie ako 96. Následne sa AAAA RR vráti k IPv6, pričom ak toto bolo splnené, tak môže začať komunikácia medzi IPv6 a IPv6 pričom druhá IPv6 adresa je len preložená z IPv4 a následne k nej priradená. [2]

1.3 Záplavové útoky

Táto práca sa zaoberá záplavovými útokmi, ktoré sú v tejto kapitole popísané a sú v rámci praktickej časti otestované a využité. Útokov je mnoho druhov a každý útočí na niečo iné. Známy v tejto oblasti je i SYN, ICMP(v6) alebo UDP záplavový útok. V tejto kapitole sú opísané predovšetkým ICMPv6 záplavové útoky a sú spomenuté ich výhody.

1.3.1 ICMPv6 Záplavové útoky

Hlavným účelom záplavového útoku je preťaženie siete, na ktorú útočník útočí s takým množstvom paketov, ktoré preťažia kapacitu siete alebo zariadenia, na ktorom beží nejaká služba.

ICMPv6 záplavové útoky sú využívané primárne na zhodenie služieb poskytovaných užívateľovi protokolu IPv6. Protokol ICMPv6 má funkcionality, pre ktoré je neodmysliteľnou súčasťou IPv6 protokolu a tými funkcionalitami sú napríklad správy pre zisťovanie smerovača (router discovery), alebo správy na tzv. objavenie suseda (neighbor discovery). Protokol ICMPv6 má celú radu správ, ktoré sa dajú zneužiť na záplavové útoky. Záplavové útoky vykonané pomocou ICMPv6 protokolu sú jednoduché na vykonanie a princíp tohoto útoku je poslanie veľmi veľkého počtu paketov na adresu iba jednému uzlu, čo môže byť počítač alebo router, ktorý stojí medzi zariadeniami. Jedným z mnohých týchto útokov môže byť tzv. reklama smerovača (router advertisement) a princípom tohoto útoku je neustále vyžadovanie novej

IPv6 adresy od uzlu, ktorý sa veľmi pravdepodobne preťaží kvôli snahe generovať tieto adresy pomocou prefixu.

V predošlej verzií IPv4 fungovali ICMP záplavové útoky principiálne rovnako ako v IPv6 avšak IPv4 nebola tak závislá na ICMP protokole tak ako novšia IPv6, a teda na IPv4 mohli byť tieto správy častokrát blokované, aby sa vyhlo takýmto útokom. Pri protokole IPv6 to možné nie je, pretože ICMPv6 má základné funkcionality, bez ktorých by IPv6 ako taká fungovať nemohla. ICMPv6 záplavové útoky sú viac nebezpečné a menej zabezpečené ako v prípade klasických ICMP záplavových útokov v protokole IPv4. [13]

1.4 Nástroje a knižnice použité pre testovanie a generovanie prevádzky

1.4.1 Trafgen

Trafgen dokáže rýchlo generovať pakety a prevádzku v sieti bez toho, aby bolo vidieť na zdroji paketov odchádzajúca prevádzka a to znamená, že trafgen má tzv. nulovú kópiu. Pomocou Trafgenu je možné testovať výkon a následne ho zhodnotiť. Zároveň Trafgen používa techniku tzv. „fuzz-testovanie“, kedy sa vkladajú náhodne vstupy pri testovaní. Avšak je Trafgen užitočný, nie je s ním možné dosiahnuť simuláciu plného prúdu. Trafgen je možné použiť v mnoho variantách testovania kde sa primárne testuje záťaž aby sme analyzovali stabilitu serveru a prípadne môže slúžiť ako nástroj v rámci prevencie pred DoS útokmi a poukazovať na nedostatky systémov. Trafgen spúšťa procesy v limite, podľa toho, koľko CPU mu je poskytnutých, priamo dostupných a vytvorí prstencovú vyrovnávaciu pamäť po procese pripojenia týchto poskytnutých CPU ku svojim vlastným CPU a tento celý proces vykonáva so zostaveným zoznamom paketov. [3]

Trafgen je známy kvôli svojmu silnému konfiguračnému jazyku a jeden z dôvodov jeho sily a výkonnosti sú aj makrá prebrané z preprocesora jazyku C. Pri vytváraní konfiguračného súboru je potrebné nastaviť isté parametre pre správne fungovanie prevádzky. Parametre použité v praktickej časti a ich stručný popis sú zobrazené v tabuľkách nižšie (viď tab. 1.4, 1.5, 1.6). [3]

Tab. 1.4: ICMPv6 zoznam parametrov

Parameter	**Popis**
ICMPv6 Message	icmp6|icmpv6(type=<číslo>, echorequest, echoreply, code=<číslo>, csum=<číslo>)
type	Typ správy (predvolene: 0)
code	Kód správy (predvolene: 0)
echorequest	ICMPv6 echo (ping) žiadosť
echoreply	ICMPv6 echo (ping) odpoveď
csum	Kontrolný súčet správy (automaticky vypočítané)

Tab. 1.5: IPv6 zoznam parametrov

Parameter	**Popis**
IPv6 Header	ip6|ipv6(ver=<číslo>, class=<číslo>, flow=<číslo>, len=<číslo>, nexthdr=<číslo>, hoplimit=<číslo>, da=<ip6_addr>, sa=<ip6_addr>)
ver|version	Verzia (predvolene: 6)
tc|tclass	Prevádzková trieda (predvolene: 0)
fl|flow	Hodnota toku (predvolene: 0)
len|length	Dĺžka záťaže (vypočítaná automaticky)
nh|nexthdr	Typ ďalšej hlavičky (predvolene: 0)
hl|hoplimit|ttl	Omeškanie, čiže čas života (predvolene: 0)
sa|saddr	Zdrojová adresa IPv6 (predvolene: adresa zariadenia IPv6)
da|daddr	Cieľová adresa IPv6 (predvolene: 0:0:0:0:0:0:0:0)

Tab. 1.6: Ethernet zoznam parametrov

Parameter	**Popis**
Ethernet Frame	eth(da=<mac>, sa=<mac>, type=<číslo>)
da|daddr	Cieľová MAC adresa (predvolene: 00:00:00:00:00:00)
sa|saddr	Zdrojová MAC adresa (predvolene: MAC adresa zariadenia)
etype|type|prot|proto	Ethernet typ (predvolene: 0)

1.4.2 TcpReplay

Tcpreplay sa využíva primárne k prehrávaniu sieťovej prevádzky a úprave tejto prevádzky. TcpReplay nie je samostatný nástroj, ale je to celá sada nástrojov ako TcpReplay-edit, TpLiveplay. TcpReplay funguje až do 2 vrstvy a spolupracuje so sieťovým hardvérom. [9] Dokáže prepísať na druhej, tretej alebo štvrtej vrstve všetky pakety, alebo i túto sieťovú prevádzku klasifikovať či už ako klient alebo server a následnej je možné túto prevádzku naspäť prehrať pomocou rôznych sieťových zariadení do siete. [1]

1.4.3 Scapy

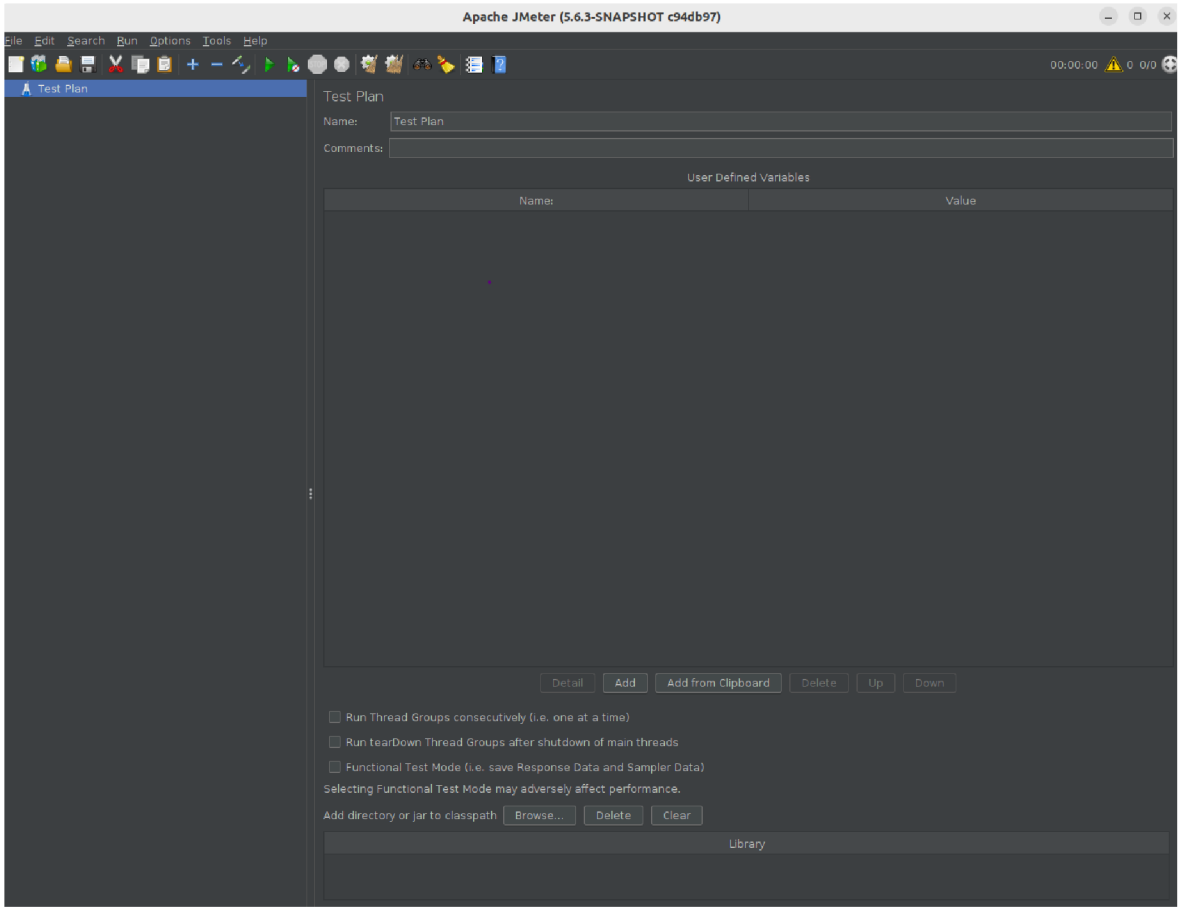
Scapy je program alebo inak povedané nástroj, ktorý je napísaný v Pythone a je využívaný výhradne na manipuláciu s paketmi, ktoré prechádzajú sieťou. Manipulácia s týmito paketmi môže byť rôzna, cez posielanie paketov, tzv. čuchanie paketov až po sofistikované falšovanie paketov. Využitie tohoto nástroja je primárne určené na vytváranie paketov, avšak je využiteľný na dešifrovanie paketov, alebo ich odchytávanie a mnoho ďalších zaujímavých a veľmi užitočných vecí, ktoré sa dajú využiť k vytvoreniu záplavových útokov. Scapy nemožno použiť zadávaním príkazov, ale je potrebné použiť priamo triedy a funkcie ako napríklad **send(), sendp(), sendpfast(), sniff(), sr()** a mnohé iné..., ktoré je možné rôzne využiť priamo pre vytvorenie paketu alebo na iné použitie. V rámci vytvárania paketov je možné pracovať s protokolmi, ako napríklad ARP, ICMP, IMCPv6, STP, DNS, Ethernet, IPv6 a IPv4 a Scapy ich ponúka oveľa viac ako tieto vypísané. [21]

1.4.4 Program JMeter

Software Apache JMeter je nástroj používaný na záťažové testovanie serverov, webových aplikácií, statických alebo dynamických zdrojov, kedy simuluje viacero pou-

žívateľov naraz, a to pomocou distribuovaného módu a títo užívatelia sú virtuálny. JMeter je možné používať na zariadeniach, kde sa dá použiť Java, a teda sú plne kompatibilné s jazykom Java, ako napríklad z tých známejších Windows, Linux alebo macOS. JMeter je možné spustiť s užívateľským rozhraním alebo pomocou príkazovej riadky(pri záťažovom testovaní sa odporúča príkazový riadok). Software je prispôsobený, aby zvládol testovať z rôznych zdrojov, čo znamená, že užívateľ dokáže spustiť testy z dynamického alebo statického zdroja.

Na úvodnej obrazovke (viď obr. 1.7) je možné zvoliť meno testu, prípadne pridať komentáre k tomuto testu. Horná lišta ponúka možnosti ako spustenie testu (zelená šípka) alebo uloženie si aktuálneho testu (sivá disketa) a podobné možnosti. Vľavo sa nachádza testovací plán a bez tejto položky nie je možné zostrojiť testovací plán.



Obr. 1.7: Ukážka úvodnej obrazovky softwaru JMeter

Typy záťažových testov:

– Prebieha kontrola, ktorá závisí na počte pripojených simulovaných užívateľoch a jej výsledky sú priamo úmerné tomuto číslu a odpovedajú času odpovede apliká-

cie, ktorá je webová
– Tá istá aplikácia, ktorej rozdiely sa testujú v rôznych prostrediach a detailne sa sleduje ako sa mení jej správanie v inom prostredí
–Testovanie pomocou dopredu vytvorených chýb, ktoré sú priamo spájané so záťažou
– Testovanie krajných limitov aplikácie a zisťuje akú veľkú záťaž aplikácia prežije pri zapojení sa veľkého množstva užívateľov naraz
– Testovanie priamo v integračnom procese a tieto testy sú automatizované
– Testovanie veľkých alebo malých databáz, FTP servery alebo LDAP servery a ďalšie iné. [19]

Konkrétne testy:

– Záťažové testy (testuje sa výkon)
– Náporové testy (testovanie počas najväčšej záťaže aplikácie a testuje sa odozva aplikácie)
– Testy výdrže (hodnotí sa stabilita a prípadné problémy, ktoré mohli nastať počas dlhšej doby testovania)
– Test odolnosti (testovanie robustného kódu aplikácie, ktorá je vystavená rôznym zlyhania počas chodu testu)
– Hrotový test (testujú sa problémy spojené so zmenami príchodu prevádzky, ktorá sa z času na čas zmení o veľké hodnoty). [19]

Dôležité pri záťažovom testovaní je si určiť:

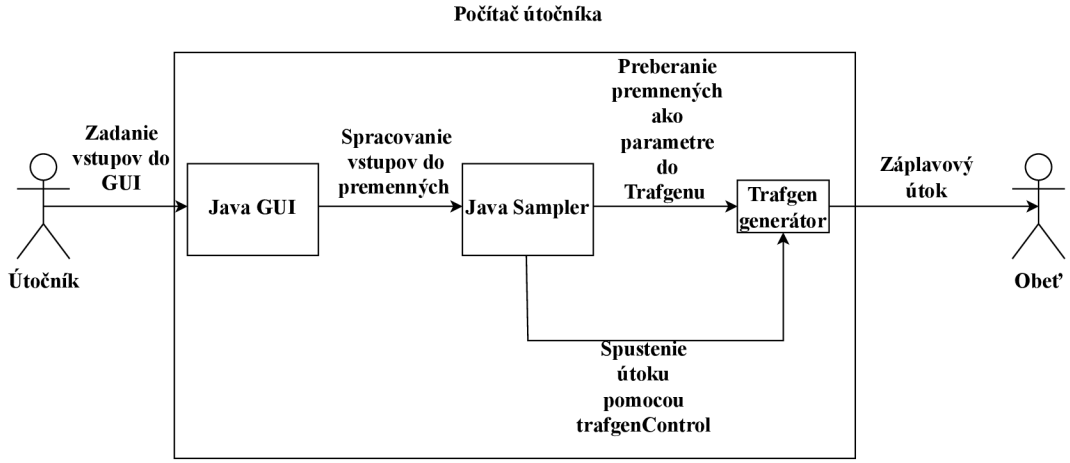
– čo je potrebné a určené k testovaniu (či už server alebo webovú aplikáciu)
– čoho chceme testom dosiahnuť alebo zistiť (či chceme dohnať server až ku spadnutiu alebo chceme otestovať jeho maximálne hranice prípadne porovnať dve verzie medzi sebou). [19]

2 Riešenie praktickej časti

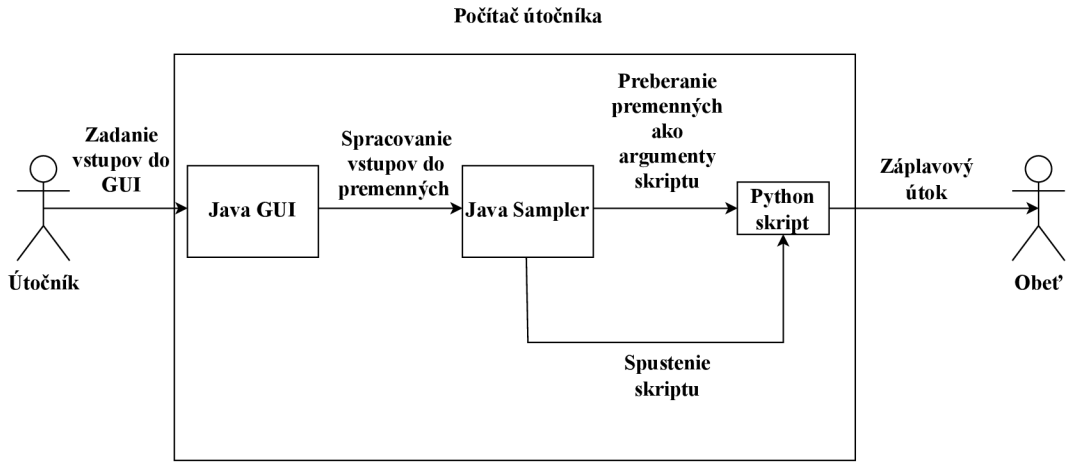
V rámci praktickej časti bolo prioritou vývoj modulu, pomocou ktorého bolo možné posielať ICMPv6 pakety s možnosťou zvolenia chybovej alebo informačnej správy, a to presného typu a kódu správy. Modul bol vytvorený v jazyku Java. K vypracovaniu praktickej časti boli využité vedomosti získané z teoretickej časti. Súčasťou vývoja modulu bolo aj vytvorenie konfiguračného súboru, ktorý slúžil na nastavenie správneho odosielania paketov a generovanie prevádzky pomocou Trafgenu. Trafgen je generátor dátovej prevádzky a pochádza zo sady sieťových nástrojov netsniff-ng. Ďalej bolo potrebné k modulu vytvoriť grafické užívateľské rozhranie, do ktorého boli pridané prvky potrebné pre správne spustenie záťažového testu. Trafgen bol použitý v rámci prvého testovacieho scenára, avšak na ostatné testovacie scenáre možnosti Trafgenu nebolo možné využiť a bol využitý nástroj Scapy, ktorý bol využívaný v skriptoch, ktoré generujú prevádzku. V rámci riešenia praktickej časti bola inšpiráciou na vytváranie útokov webová stránka, ktorá sa zaoberala prevenciou a detekovaním DoS alebo DDoS útokov [12].

Pre každý testovací scenár bolo vytvorené vlastné grafické rozhranie, pretože každý z týchto scenárov disponoval inými parametrami testu a týmto spôsobom sa dosiahla priehľadnosť v daných scenároch. Bolo vytvorených celkovo 5 scenárov, avšak autor sa jeden z nich rozhodol nevybrať a nevložiť do bakalárskej práce. Jednalo sa o scenár zaoberajúci sa Jumbogramami, ktorý nebolo možné otestovať na danej topológii, avšak jeho implementáciu je možné vidieť na GitLabe autora [22].

Pre správne fungovanie prevádzky pri Python skriptoch bol použitý nástroj TcpReplay.



Obr. 2.1: Diagram priebehu útoku vnútri modulu s použitím Trafgen generátoru



Obr. 2.2: Diagram priebehu útoku vnútri modulu s použitím Scapy

2.1 Použitý hardware testovacieho prostredia

Koncové zariadenie útočníka, ktoré bolo použité na generovanie paketov a generovaniu prevádzky je virtuálny stroj s operačným systémom Ubuntu 22.04.4 LTS, ktorý je na notebooku s Windows 11 (host). Tento host notebook disponuje RAM operačnou pamäťou s 16 GB, z ktorých sa virtuálnemu stroju poskytlo 10 GB. Procesor Intel i7-1165G7 s frekvenciou 2,80 GHz. Disponuje taktiež grafickou kartou NVIDIA GeForce MX350.

O prepojenie koncových zariadení bol využitý switch CUDY, ktorý pozostáva z 8 portov RJ45 s rýchlosťou 1Gb/s. Prepínacia rýchlosť je 16 Gb/s.

Koncové zariadenie obete, ktoré prijímalo pakety a generovanú prevádzku bol mikropočítač Raspberry 4b. Disponuje 2 GB operačnej pamäte RAM a procesorom so 4 jadrami.

2.2 Inštalácia

Pred samotnou inštaláciou ddos modulu je potrebné stiahnuť knihovňu IPAddress, ktorá je verejne dostupná na GitHube. Knihovňa pracuje s IP adresami a snaží sa zvládnuť manipuláciu s nimi. Knihovňu IPAddress je potrebné vložiť do adresára JMetru /jmeter/lib/ext. Pravdepodobne bude potrebné príkazom **sudo apt install tcpreplay** nainštalovať TcpReplay, pretože bez bez tohoto nástroja nebude fungovať funkcia sendpfast() v Scapy. Pre nainštalovanie modulu ddos do programu JMeter je nutné stiahnutú prílohu Ddos.jar vložiť do adresára JMetru /jmeter/lib/ext. Po presunutí modulu je možné spustiť JMeter, ktorý obsahuje aj vyššie spomenuté moduly. Pred spustením je potrebné sa presvedčiť, že python dokáže spúšťať scripty bez administrátorských práv sudo. Modul bol vyvíjaný a testovaný v jazyku JAVA 17 a operačnom systéme Ubuntu 22.04.4 LTS. Pre Raspberry (mikropočítač, na ktorý sa posielajú útoky) je Ubuntu špeciálne vytvorené, avšak verzia je rovnaká a to 22.04.4 LTS.

Pre správne fungovanie všetkých scenárov, ktoré používajú scapy je potrebné v príkazovom riadku zrušiť požadovanie administratívnych práv v systéme (sudo), pretože sa pri spúšťaní testov z JMetru spúšťa execute python skriptu. Je potrebné zadať do príkazovej riadky nasledovné príkazy: **sudo apt-get install libcap2-bin** a následne **sudo setcap cap_net_raw+ep($which python3)**. Taktiež je potrebné povoliť práva k príkazu python pomocou **sudo chmod +s "$(command -v python3.10)"** a tento príkaz nastaví tzv. „setuid“ bit na súbor, ktorý je výstupom príkazu. Tento príkaz je odporúčaný použiť len za účelom spustenia záťažového testu, pretože v iných prípadoch môže byť tento príkaz nebezpečne zneužitý.

Následne je potrebné premiestniť sa do JMeter adresára pomocou **~/(cesta k JMetru)/jmeter/bin** a odtiaľ sa spustí JMeter pomocou **sudo /.jmeter**. Po kliknutí pravým tlačidlo na test plan v kolónke vľavo, je potrebné vybrať **Add > Threads > DDoS Simple Thread Group**. Po týchto krokoch je možné kliknutím na DDoS Simple Thread Group pravým tlačidlom vybrať jeden z mnohých záťažových testov z ponuky. Po každom záťažovom teste, ktorý používa Python skript, je potrebný reštart softwaru JMeter.

2.3 Vývoj modulu

Pri vývoji modulu sa muselo zamerať na všetky potrebné veci k prevádzke ICMPv6 paketov v prostredí IPv6. Bolo vytvorené grafické užívateľské rozhranie s nevyhnutnými elementami pre nastavenie testového plánu.

Pre správne fungovanie odosielania ICMPv6 paketov, bola vytvorená premenná, ktorá obsahuje názvy a typy kódov. Používateľ si môže zvoliť, ktorú chybovú alebo informačnú správu pošle. Boli pridané všetky typy ICMPv6 správ. V užívateľskom rozhraní sú tieto správy zobrazované pomocou rolovacieho menu, kde si užívateľ zvolí zo sady ICMPv6 správ.

Bol vytvorený konfiguračný súbor Trafgen pre nastavenie odosielania prevádzky ICMPv6. Tento súbor nesie názov AbstractIcmpv6Flood.cfg a nachádza sa v adresári s DDOS modulom. K nastaveniu tohoto súboru s konfiguráciou bol využitý Trafgen manuál.

2.3.1 Nastavenie Trafgen konfiguračného súboru

Konfiguračný súbor bolo potrebné vytvoriť ešte pred buildom modulu a vložením do Apache JMeter. Konfiguračný súbor (viď obr. 2.3) bol vložený do **jmeter-ddos-plugin\src\main\resources\trafgen_cfg**. Súbor bol vytvorený z dôvodu fungovania odosielania paketov na druhé zariadenie alebo server.

- ETH-P-IP – je symbolická konštanta, ktorá reprezentuje EtherType
- 0x86DD – fialový rámček na obrázku (viď obr. 2.3) – hexadecimálna hodnota, ak je hlavička nižšej úrovne Ethernet, EtherType je nastavený na 0x86DD (IPv6)
- eth – červený rámček na obrázku (viď obr. 2.3) – bola nastavená cieľová a zdrojová MAC adresa, ktoré sa berú z kódu ako premenné a užívateľ si môže nastaviť v užívateľskom rozhraní vlastné adresy
  –Predvolené je nastavená pre **ssadr** adresa zariadenia, z ktorého sa posielajú pakety.
  –**daddr** má predvolenú adresu 00:00:00:00:00:00
- ipv6 – oranžový rámček na obrázku (viď obr. 2.3) – nastavili sa IPv6 adresy, ktoré sa preberajú z užívateľského rozhrania.
  – **ttl** (time to live) je prednastavená na 0. Užívateľ nastaví pomocou premennej z užívateľského rozhrania
  – **version** je predvolené nastavená na hodnotu 6
  – **daddr** je prednastavená na 0:0:0:0:0:0:0:0
  – **ssadr** je prednastavená na IPv6 adresu zariadenia a nám sa tu preberá hodnota z užívateľského rozhrania

Ďalšie parametre ako **lenght** (vypočíta sa sama), **flow** je nastavený v predvoľbe ako 0 alebo **nexthdr** je tiež nastavený v predvoľbe s hodnotou 0

- icmpv6 – žltý rámček na obrázku (viď obr. 2.3) – bol nastavený typ (**type**) a kód (**code**) aby bral hodnotu z parametru, ktorý v kóde reprezentuje všetky ICMPv6 správy, takže si užívateľ znova zvolí podľa seba.
  Kontrolný súčet (**checksum**) sa vypočíta automaticky, takže nie je potreba tento parameter používať v konfiguračnom súbore
- fill – modrý rámček na obrázku (viď obr. 2.3) – nastavuje sa obsah paketov, pomocou parametru payload a tak môže užívateľ zvoliť veľkosť v bajtoch pomocou užívateľského rozhrania. [3]

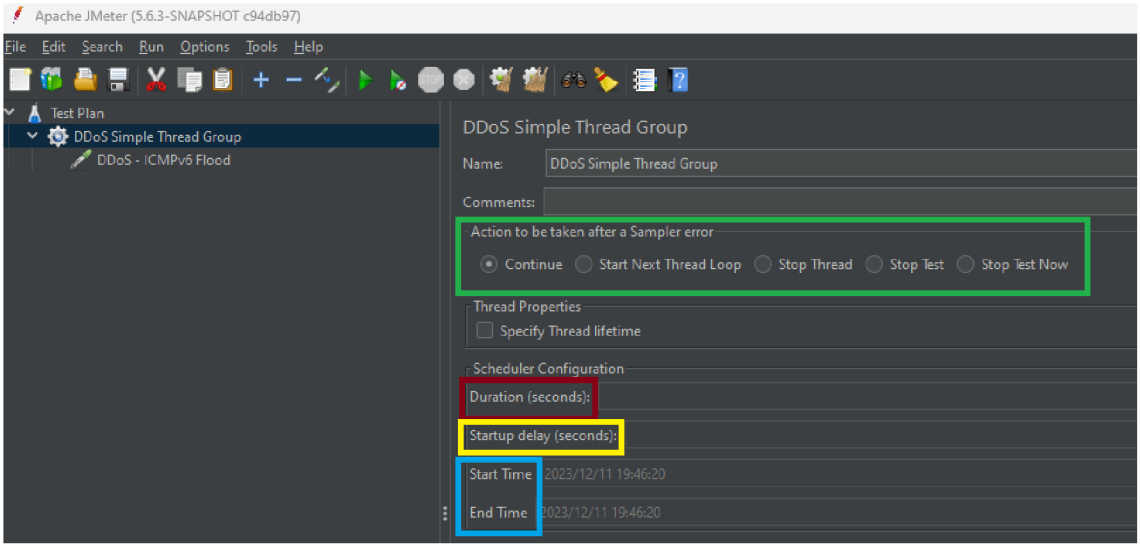
```
#define ETH_P_IP    0x86DD

{
    eth(daddr=##Icmpv6FloodSampler.dmac, saddr=##Icmpv6FloodSampler.smacTG, proto=ETH_P_IP),
    ipv6(ttl=##Icmpv6FloodSampler.ttl, ver=6, da=##Icmpv6FloodSampler.targetIPv6, sa=##Icmpv6FloodSampler.sourceIPv6),
    icmpv6(type=##Icmpv6FloodSampler.typeTG, code=##Icmpv6FloodSampler.codeTG),
    fill('B',##Icmpv6FloodSampler.payload),
}
```

Obr. 2.3: Konfiguračný súbor Trafgen

2.3.2 DDoS ICMPv6 flood v Apache JMeter

V tejto sekcii je vysvetlené, ako vytvoriť testovací scenár pomocou modulu.

Pred vytvorením záplavového útoku pomocou ICMPv6 modulu je potrebné najprv vytvoriť vláknovú skupinu vytvorenú pre DDoS s názvom *DDoS Simple Thread Group* (viď obr. 2.4), kde je možnosť nastaviť trvanie testu (červený štvorček), po koľkých sekundách od stlačenia tlačidla spustiť sa, test sa spustí (žltý štvorček) alebo sa dá vopred nastaviť dátum a čas spustenia testu (modrý štvorček). Modul má názov *ICMPv6Flood*. Ak nastane chyba v samplery, dá sa nastaviť, ako sa má JMeter zachovať (zelený štvorček).



Obr. 2.4: Tvorba vlákna

Ďalej je potrebné kliknutím pravým tlačidlom na *DDoS Simple Thread Group* zvoliť *DDoS - ICMPv6 Flood* pre zvolenie užívateľského rozhrania pre ICMPv6 záplavový útok. Po zvolení tejto možnosti je potreba nastaviť (viď sek. 2.3.3) všetky potrebné parametre pre spustenie záplavového útoku. Nakoniec, je potrebné kliknúť na zelenú šípku vľavo hore a tým sa spustí test.

2.3.3 Užívateľské rozhranie

V tejto časti sú opísané časti grafického užívateľského (viď obr. 2.5) rozhrania a ich funkcia.

Ako prvá časť je opísaná *Network Interface* kde si užívateľ zvolí rozhranie, ktoré sa využije pri posielaní paketov. Modul automaticky detekuje všetky dostupné rozhrania v zariadení.

V ďalšej časti, označené **v žltom rámčeku**, je tzv. *linková vrstva*, kde si užívateľ dokáže nastaviť MAC adresu cieľového zariadenia a zdrojového zariadenia a tieto parametre sa nastavia v už vyššie spomínanom *Trafgen* konfiguračnom súbore. Zdrojová MAC adresa sa dá nastaviť aj v rozsahu adries, ktoré užívateľ zvolí.

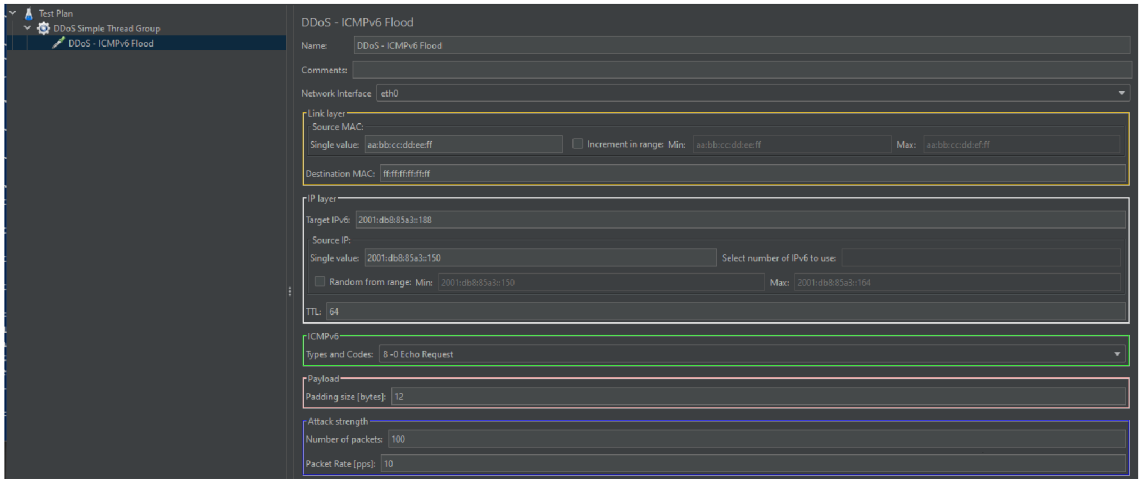
V bielom rámčeku je IP vrstva, kde sa využíva IPv6 protokol a užívateľ zvolí cieľovú adresu zariadenia. Ďalej je možné zadať zdrojovú IPv6 adresu i s rozsahom a počtom adries. Nastavenie *hoplimitu/ttl* je tiež možnosťou užívateľského rozhrania.

V zelenom rámčeku je možné nastaviť ICMPv6 informačnú alebo chybovú správu, ktorá sa bude posielať na cieľové zariadenie.

Oranžový rámček reprezentuje *payload*, kedy sa nastaví veľkosť dát v bajtoch. Treba myslieť ale na to, že nie všetky vyššie spomenuté informačné a chybové

správy majú payload, napríklad pri *echo requeste* sú dáta payloadu nulové alebo sú nastavené na určitú hodnotu, ktorá sa očakáva pri odpovedi.

V poslednom **modrom rámčeku** je možnosť nastavenia počtu paketov a ako často sa pošle paket *pps – packet per second*.



Obr. 2.5: Užívateľské rozhranie

2.4 Kompozícia testovacieho prostredia

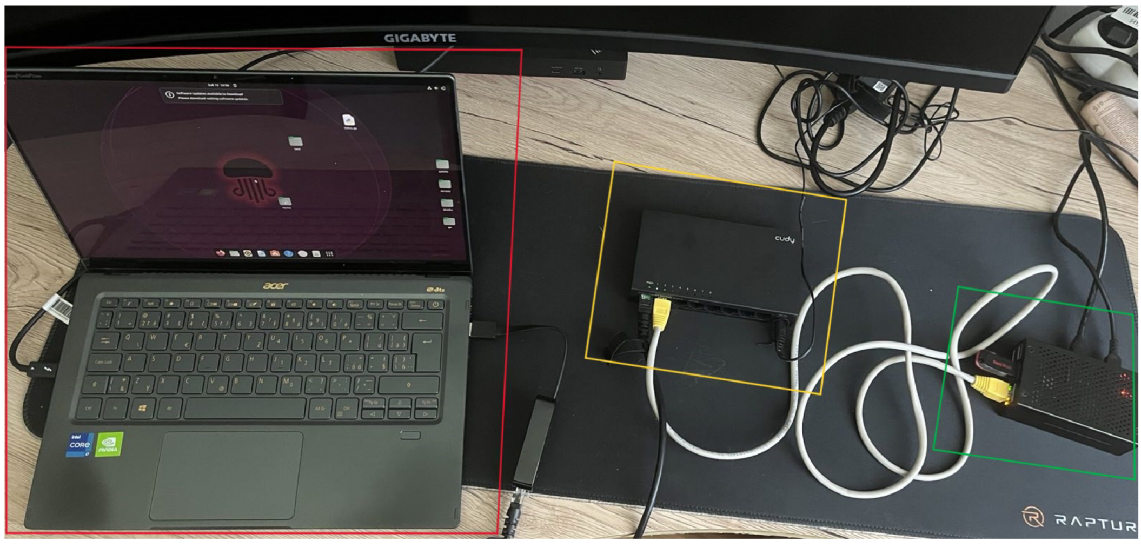
V rámci testovacieho prostredia boli dve koncové zariadenia. Jedno zariadenie bolo využívané ako útočník a druhé bolo využité ako obeť. Zariadenie, z ktorého prebiehal útok bol virtuálny stroj na notebooku. Tento virtuálny stroj používal systém Ubuntu 22.04.4 LTS, kde boli nainštalované všetky potrebné programy, ktoré boli potrebné na realizáciu scenárov a vývoj modulov. V zariadení obete nebolo nainštalované nič, okrem programu wireshark na monitorovanie priebehu útoku a program net-tools na správu IP adries zariadenia.

Obr. 2.6: Mikropočítač využitý ako zariadenie obete

K mikropočítaču Raspberry (viď obr. 2.6) bolo pripojené napájanie, HDMI mikro adaptér do ktorého sa napája HDMI kábel, spúšťacie USB na ktorom je UBUNTU a nakoniec Ethernet kábel, ktorý vedie do prepínača.

Testovacie laboratórium pozostávajúce z dvoch koncových užívateľov, z ktorých je jeden útočník a druhý obeť, bolo rozšírené o jeden prepínač s gigabitovými ethernetovými prípojkami. Raspberry Pi má tiež gigabitovú ethernet prípojku takže pripojenie bolo možné využiť naplno. Toto testovacie laboratórium malo teda prepojenie: zariadenie - prepínač - zariadenie. Celá topológia bola prepojená ethernetovými káblami zo strany útočníka i zo strany obete. Po použití tejto topológie bolo testovacie prostredie kompletne izolované od internetu a teda žiadne vonkajšie vplyvy by nemali naše testovanie ovplyvňovať. Vo všetkých testovacích scenároch bola použitá rovnaká kompozícia testovacieho prostredia.

Návrh tejto topológie bol následne prevedený do reality pomocou vyššie spomenutých komponentov a sieťových zariadení (viď obr. 2.7).



Obr. 2.7: Topológia testovacieho prostredia prevedená do reality

Popis obrázku 2.7:

–V červenom rámčeku je označený laptop a teda útočník, z ktorého boli prevedené útoky a bola z neho generovaná prevádzka, ktorá zaplavovala zariadenie v zelenom rámčeku
–Žltý rámček označuje prepínač s rýchlosťou portov 1 Gbit/s
– Zelený rámček označuje obeť (Raspberry), na ktorú sa posielali pakety zo zariadenia v červenom rámčeku.

2.5 Prvý testovací scenár

V rámci tejto práce bol vytvorený testovací scenár pomocou vytvoreného modulu. Mal za úlohu zaplaviť druhé koncové zariadenie pomocou ICMPv6 paketov.

Niektoré obrázky bolo potrebné odfotografovať telefónnym zariadením, pretože autor nechcel zasahovať do výkonu mikropočítača alebo mikropočítač bol neschopný spraviť snímky obrazovky pod záťažovým testom.

2.5.1 Príprava testovacieho prostredia

Testovanie bolo vykonané pomocou vyššie uvedenej topológie Spustenie Apache JMeter vykonáme pomocou príkazu **./jmeter.sh** v adresári **/bin**. Bolo potrebné pridať globálne IPv6 adresy pomocou príkazu:
sudo ip -6 addr add 3ffe:0:0:0:200:ff:fe00:1/64 dev enp0s3

Pre druhé zariadenie:
sudo ip -6 addr add 3ffe:0:0:0:200:ff:fe00:2/64 dev enp0s3
Test konektivity pomocou príkazu:
ping -6 3ffe:0:0:0:200:ff:fe00:2

```
sami@sami-VirtualBox:~$ ping -6 3ffe:0:0:0:200:ff:fe00:2
PING 3ffe:0:0:0:200:ff:fe00:2(3ffe::200:ff:fe00:2) 56 data bytes
64 bytes from 3ffe::200:ff:fe00:2: icmp_seq=1 ttl=64 time=1.41 ms
64 bytes from 3ffe::200:ff:fe00:2: icmp_seq=2 ttl=64 time=0.878 ms
64 bytes from 3ffe::200:ff:fe00:2: icmp_seq=3 ttl=64 time=0.693 ms
64 bytes from 3ffe::200:ff:fe00:2: icmp_seq=4 ttl=64 time=0.755 ms
64 bytes from 3ffe::200:ff:fe00:2: icmp_seq=5 ttl=64 time=0.694 ms
64 bytes from 3ffe::200:ff:fe00:2: icmp_seq=6 ttl=64 time=0.850 ms
```

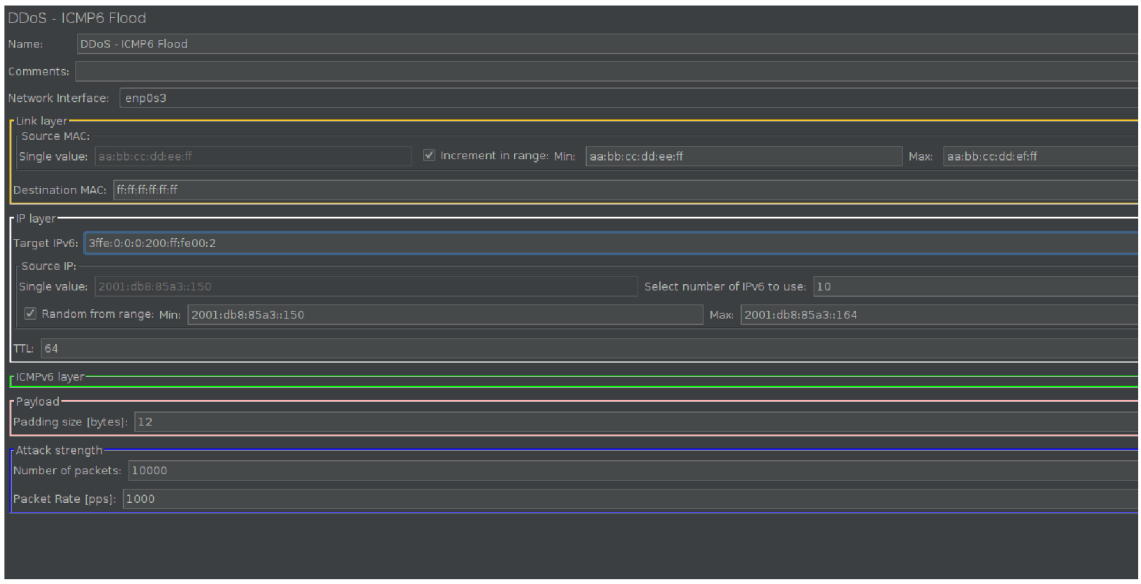
Obr. 2.8: Ukážka testu konektivity

Pomocou testu konektivity bola overená vzájomná viditeľnosť oboch koncových zariadení a bolo možné vykonať záplavový útok pomocou Apache JMeter.

2.5.2 Realizácia prvého testovacieho scenára

Na realizáciu testu bolo potrebné vyplniť všetky parametre, spomenuté v podkapitole 2.3.3. Do políčka Source MAC adresy nebolo potrebné písať nič, pretože ako bolo písané v podkapitole 2.3.1, tak bola táto hodnota prevzatá automaticky zo zariadenia. Políčku Destination MAC address bola pridelená adresa druhého stroja. Tak isto bolo postupované pri IPv6 adresách kedy bolo ponechané odosielanie z jednej IPv6 adresy v tomto scenári. Time To Live je ponechané na hodnote 64. V rámci testu boli posielané echo-request informačné ICMPv6 správy. Veľkosť payloadu bola nastavená na 12 bajtov a nakoniec sa nastavila sila útoku. Nastavený bol aj počet posielaných paketov na hodnotu 10000 a počet paketov za sekundu bol nastavený na 1000.

Na druhom koncovom zariadení bol zapnutý wireshark, ktorý očakával ICMPv6 pakety.



Obr. 2.9: Grafické rozhranie prvého testovacieho scenára

Po spustení testu sa na virtuálnej mašine začala generovať prevádzka a bola odosielaná cez switch až do mikro počítača Raspberry, ktorý slúžil ako obeť útokov v tomto testovacom prostredí. V počítači obete bol zaplutý wireshark, v ktorom boli vyfiltrované ICMPv6 pakety a bol zobrazený prichádzajúci flood z virtuálnej mašiny. Jednalo sa o ICMPv6 pakety a konkrétne echo-request správy, ktoré zahltili mikropočítač obete. V tomto scenári išlo primárne o zahltenie zariadenia pomocou ICMPv6 protokolu, a to sa v rámci možností podarilo. V rámci tejto práce bol otestovaný ICMPv6 echo-request flood a echo-reply.

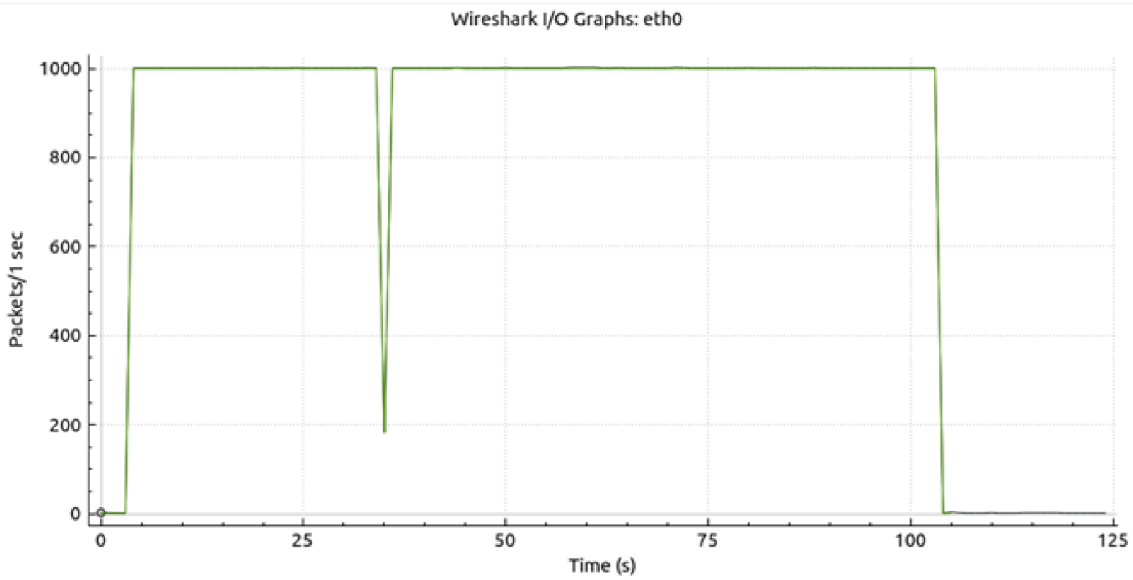
Keďže netsniff-ng nemá kompletne doimplementovanú ICMPv6 v Trafgene, tak sa rôzne typy a kódy v tomto scenári nedali naplno využiť. Bolo možné použiť iba generovanie echo-request a echo-reply správ. Typy a kódy sú však implementované v kóde a GUI je pripravené posielať rôzne typy správ, ak budú v Trafgene opravené. Error pri napísaní kódu a typu do konfiguračného súboru Trafgenu bol: **trafgen_parser.y:476: proto_field_expr_eval: assertion '0' failed**.

2.5.3 Testovanie prvého scenára

Testovací scenár je určený k zahlteniu zariadenia pomocou ICMPv6 echo-request správ.

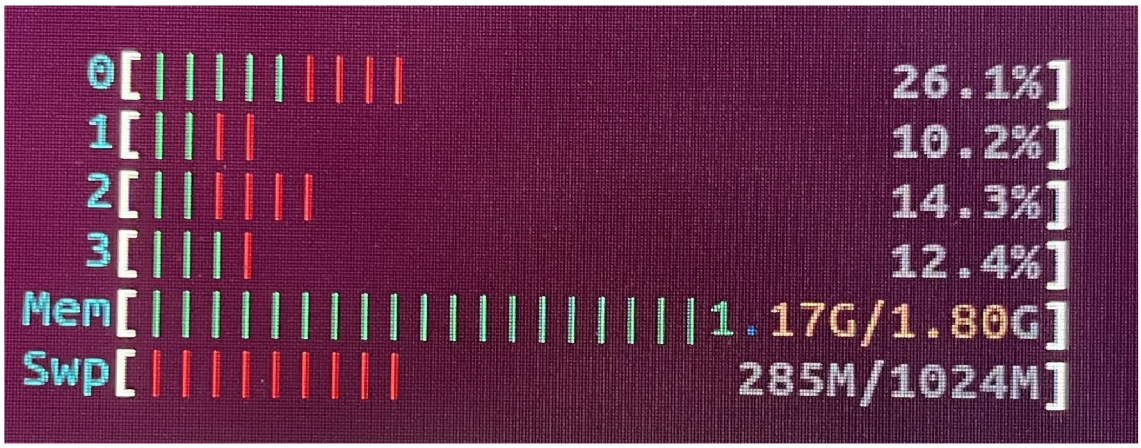
213	22.268579856	2001:db8:85a3::151	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
214	22.268580216	2001:db8:85a3::158	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
215	22.268580236	2001:db8:85a3::156	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
216	22.268580254	2001:db8:85a3::15e	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
217	22.268580273	2001:db8:85a3::161	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
218	22.268580292	2001:db8:85a3::156	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
219	22.268580310	2001:db8:85a3::154	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
220	22.268580330	2001:db8:85a3::158	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
221	22.268597020	2001:db8:85a3::162	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
222	22.268597038	2001:db8:85a3::152	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
223	23.271228066	2001:db8:85a3::15e	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
224	23.271228409	2001:db8:85a3::161	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
225	23.271228432	2001:db8:85a3::15e	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li
226	23.271228454	2001:db8:85a3::162	3ffe::200:ff:fe00:2	ICMPv6	70	Echo (ping) request id=0x4242, seq=16962, hop li

Obr. 2.10: Úspešný ICMPv6 záplavový útok pomocou echo-request správy z viacerých IPv6 adries



Obr. 2.11: Graf popisujúci pakety za sekundu v prvom scenári

Na grafe 2.11 je viditeľné, že útok prebiehal konzistentne od začiatku útoku až po koniec. V rámci tohoto útoku boli posielané ICMPv6 echo-request správy bez odpovede, takže na grafe je viditeľný počet paketov za sekundu s hodnotou 1000.



Obr. 2.12: Zaťaženie mikropočítača Raspberry pri záplave s ICMPv6 echo-request správami

Obrázok (viď obr. 2.12) ukazuje záťaž pri klasickom ICMPv6 echo-request útoku. Bolo veľmi obtiažne zahltiť mikropočítač iba s ICMPv6 paketmi. Limitácia iba na echo-request správy bola v tomto prípade obmedzujúca, a preto sa nepodarilo dosiahnuť vyšších hodnôt. Za túto limitáciu mohla sada netsniff-ng, ktorá nemala pre trafgen dostatočne implementovaný ICMPv6 protokol a jediné využiteľné správy boli echo-request a echo-reply. Najvyššia hodnota zaťaženia bola 26.1 %. Pri spočítaní priemeru všetkých štyroch jadier bolo priemerné zaťaženie 15.75 %. Pamäť sa pohybovala okolo hodnoty 1.17 G.

2.6 Druhý testovací scenár

V rámci tohoto scenára bol využitý vytvorený modul. Mal za úlohu posielanie fragmentovaných paketov. Účelom tohoto scenára bolo zaplavenie zariadenia s využitím fragmentačnej hlavičky a použitia fragmentov, ktoré sa v skripte manuálne rozdeľujú na maximálnu hodnotu MTU. Boli posielané ICMPv6 echo-request správy.

2.6.1 Príprava testovacieho prostredia

Dôležitým faktorom v tomto testovacom scenári bola hodnota sieťového filtru MTU, ktorá býva zvyčajne nastavená na hodnotu 1500 a nemožno ju u mnohých zariadení zmeniť. Minimálna MTU hodnota protokolu IPv6 je 1280 bajtov, čo je vyššia hodnota v porovnaní s protokolom IPv4, kde je táto minimálna hodnota iba 576 bajtov, pričom východzia hodnota je pri oboch protokoloch rovnaká a to 1500 bajtov. Pri niektorých zariadeniach funguje príkaz **ip link set dev <názov siete> mtu 9000**, čo vlastne zvýši hodnotu MTU na 9000 bajtov, a teda sieťou môžu

prechádzať pakety o veľkosti 9000 bajtov. Všeobecne sa však neodporúča táto veľkosť hodnoty MTU, a teda v tomto scenári bola MTU nastavená na 1500 bajtov a pakety boli rozfragmentované na túto veľkosť. Po tomto príkaze je možné hodnotu MTU skontrolovať pomocou **ip link show <názov siete>**. Avšak v tomto scenári bolo využité fragmentovanie paketov, takže nebolo potrebné meniť hodnotu MTU. V prípade veľkosti celkového payloadu, bola možnosť poslať maximálne 65535 bajtov, ktoré boli následne rozfragmentované skriptom na fragmenty s maximálnou veľkosťou 1500 bajtov.

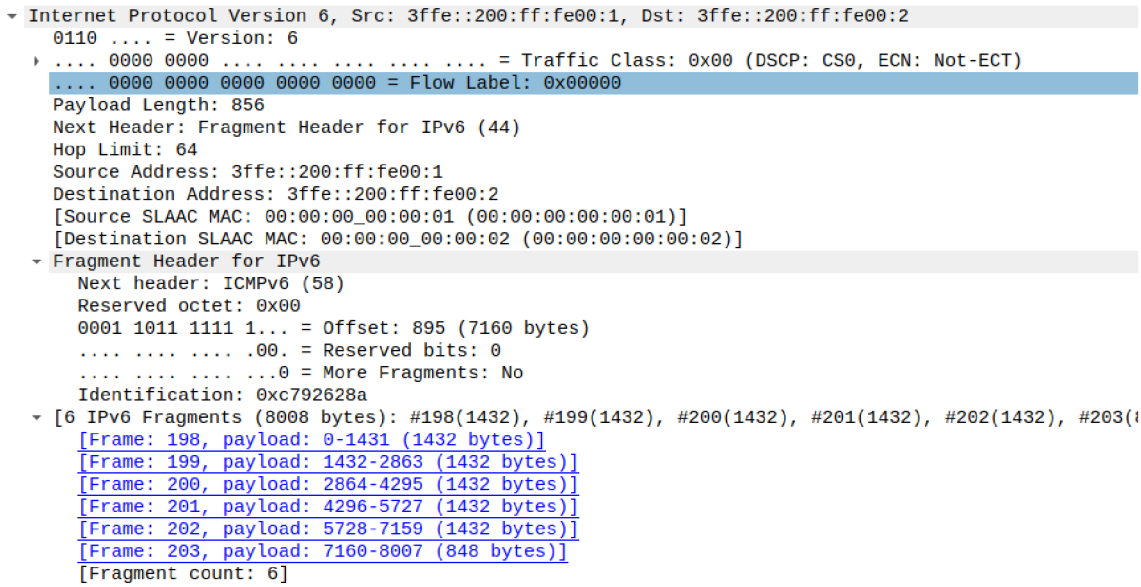
Bol použitý Python skript, ktorý bol spúšťaný samplerom a v tomto skripte bola využitá knižnica Scapy. Knižnica Scapy použitá v Pythone potrebuje administrátorské oprávnenia z dôvodu operácií na nižšej úrovni siete. Príkazom **sudo apt-get install libcap2-bin** bol nainštalovaný package, ktorý obsahoval utilitu **setcap**. Príkazom **sudo setcap cap_net_raw+ep($which python3)** sa nastavilo, že python interpreter povolil skriptom bežať bez potreby administrátorských privilégií (sudo príkaz).

2.6.2 Realizácia druhého testovacieho scenára

V tejto podkapitole je rozobraná realizácia testovacieho scenára, ktorý vykonáva záplavový útok pomocou IPv6 fragmentácie.

V rámci testovacieho scenára bol použitý protokol ICMPv6 a konkrétne boli využité echo-request správy. V tomto scenári, ako bolo vyššie spomenuté, nebol použitý Trafgen, pretože na manipuláciu paketmi tohoto typu a ich vytváranie bolo priateľskejšie a efektívnejšie pomocou použitia nástroja Scapy.

Bolo vytvorené užívateľské grafické rozhranie, s potrebnými testovacími parametrami. V tomto scenári nebol využitý Trafgen ako generátor prevádzky. Keďže sa v tomto útoku využívala fragmentácia, na ktorú bolo potrebné podrobnejšie zostaviť paket, ktorý bol posielaný, bolo potrebné využiť Python, pomocou ktorého bol vytvorený skript. Tento skript vytvoril manuálnu fragmentáciu paketu, ktorý prekročil filtračnú hodnotu siete MTU, ktorá je predvolene nastavená na väčšine sietí s hodnotou 1500. Užívateľ bol schopný teda vložiť payload hodnoty väčšej ako je toto MTU (napr. 8000 bajtov) a tento skript bol schopný tento payload rozložiť na fragmenty, ktoré spĺňali túto MTU hodnotu.



Obr. 2.13: ICMPv6 paket s fragmentačnou hlavičkou

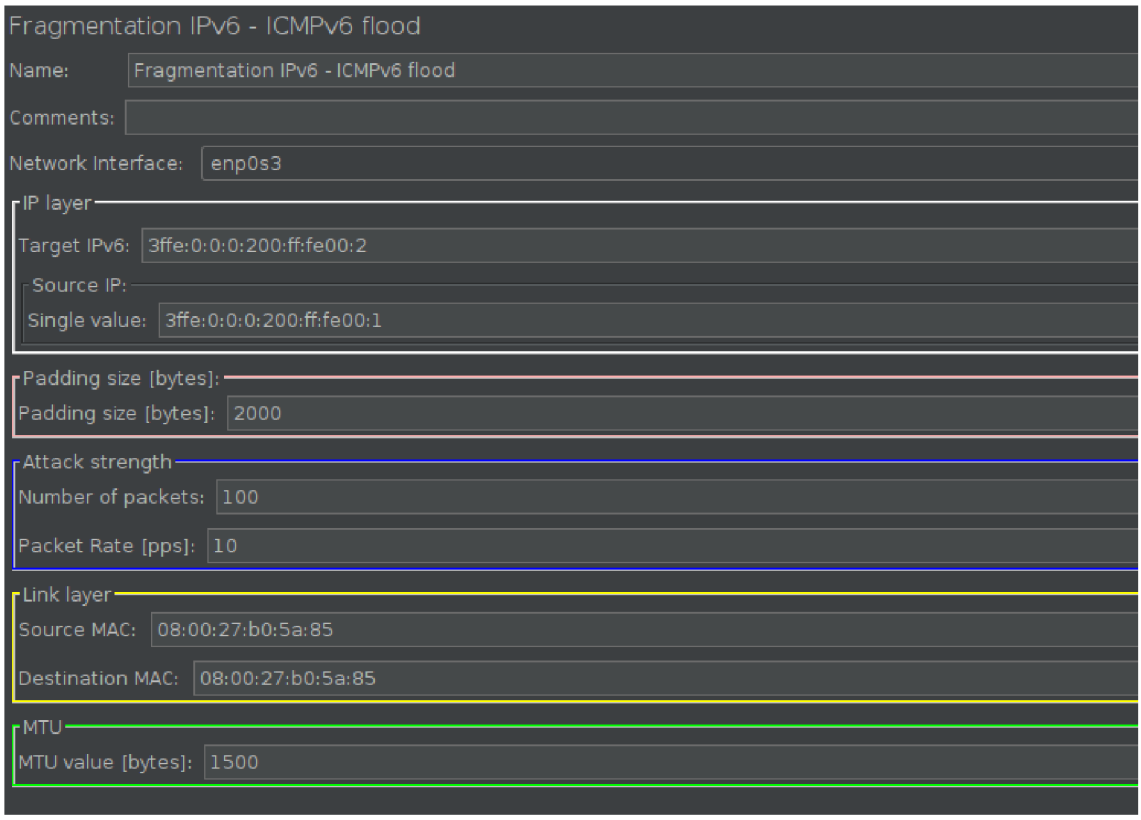
Na obrázku (viď obr. 2.13) je fragmentačný paket, na ktorom vidieť že po tomto fragmente už žiadne ďalšie pakety nenasledujú. Taktiež možno vidieť všetky fragmenty a ich celkový počet.

Celý proces spustenia testu prebiehal tak, že užívateľ vložil parametre testu ako IP adresy, payload, počet paketov za sekundu a počet paketov celkovo poslaných do GUI. Z GUI sa tieto parametre prevzali v Samplery, kde dochádzalo k spracovaniu týchto parametrov do premenných a tieto premenné sa používali ako argumenty pri spúšťaní skriptu, ku ktorému dochádzalo tiež priamo v Samplery.

```
1 packet = Ether(src=mac_src, dst=mac_dest) /
2 IPv6(src=ip_src, dst=ip_dest) /ICMPv6EchoRequest() /Raw(load=payload)
3 fragments = fragment6(packet, mtu) #MTU value most
4 #of the time set on 1500 so I used it here
```

Obr. 2.14: Ukážka kódu, ktorý rozdeľuje paket na fragmenty

V tomto kóde (viď obr.2.14) z python skriptu je vytvorenie samotného paketu so všetkými potrebnými hlavičkami a následne sa veľkosť tohoto paketu rozfragmentuje na menšie fragmenty s maximálnou veľkosťou hodnoty MTU, ktorú užívateľ zadá v GUI.



Obr. 2.15: Grafické rozhranie pre fragmentačný záplavový útok

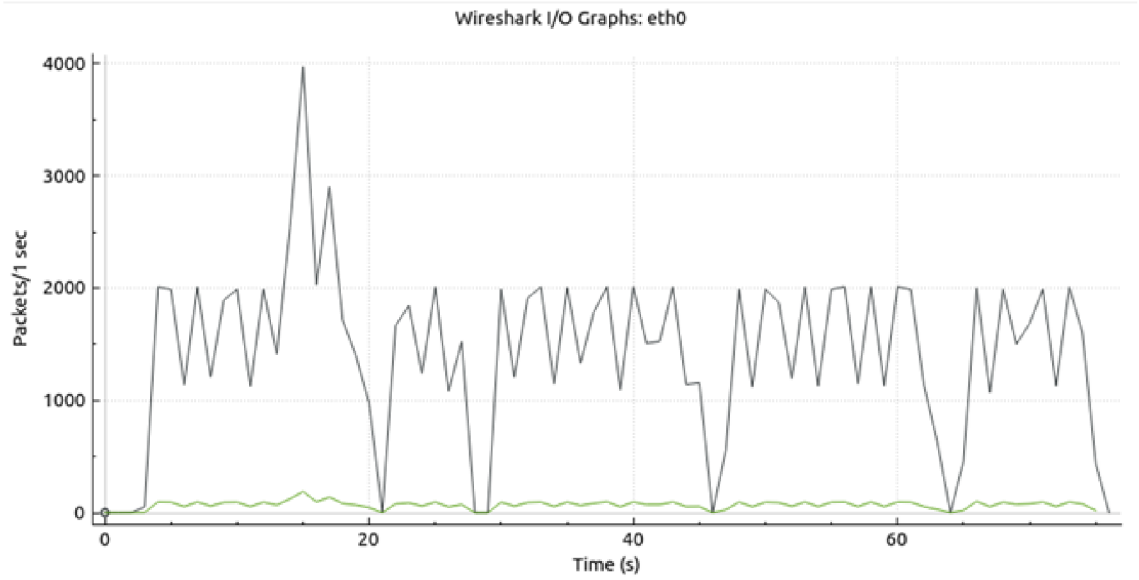
Grafické rozhranie je v tomto prípade obohatené o textový rámček, do ktorého užívateľ môže zadať hodnotu MTU, pretože na každej sieti sa táto hodnota môže líšiť a je to dôležitý faktor pri tomto útoku s využitím fragmentácie.

2.6.3 Testovanie druhého scenára

Testovací scenár je určený k zahlteniu zariadenia pomocou fragmentácie ICMPv6 echo-request správ.

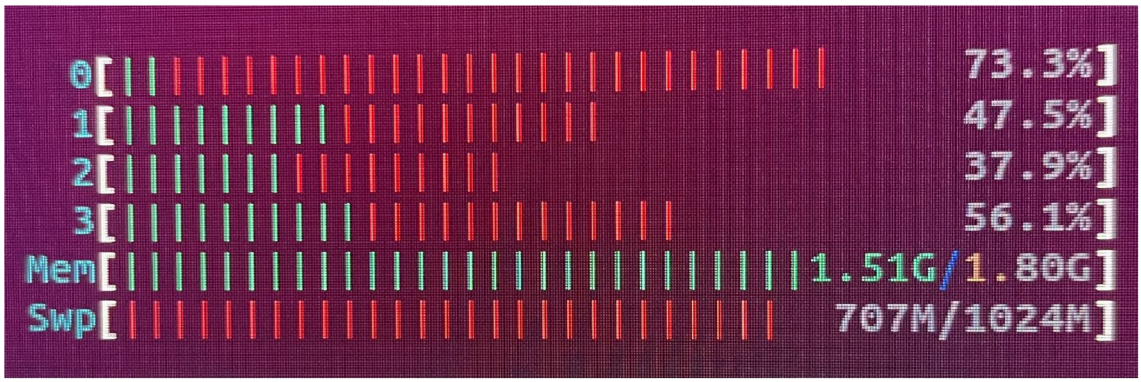
17589	27.053530730	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	774	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
17590	27.054527950	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=0 more=y ident=0xbc289ca7 nxt=58)
17591	27.055535008	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=1432 more=y ident=0xbc289ca7 nxt=5
17592	27.056536874	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=2864 more=y ident=0xbc289ca7 nxt=5
17593	27.057798747	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	774	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
17594	27.058996375	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=0 more=y ident=0xbc289ca7 nxt=58)
17595	27.059427459	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=1432 more=y ident=0xbc289ca7 nxt=5
17596	27.060715410	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=2864 more=y ident=0xbc289ca7 nxt=5
17597	27.061481723	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	774	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
17598	27.062467895	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=0 more=y ident=0xbc289ca7 nxt=58)
17599	27.063511779	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=1432 more=y ident=0xbc289ca7 nxt=5
17600	27.065970573	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=2864 more=y ident=0xbc289ca7 nxt=5
17601	27.065970838	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	774	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
17602	27.066741426	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=0 more=y ident=0xbc289ca7 nxt=58)
17603	27.067472253	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=1432 more=y ident=0xbc289ca7 nxt=5
17604	27.068411186	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=2864 more=y ident=0xbc289ca7 nxt=5
17605	27.069603399	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	774	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
17606	27.070447918	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=0 more=y ident=0xbc289ca7 nxt=58)
17607	27.072012130	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=1432 more=y ident=0xbc289ca7 nxt=5
17608	27.072474132	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=2864 more=y ident=0xbc289ca7 nxt=5
17609	27.073443645	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	774	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
17610	27.074420358	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=0 more=y ident=0xbc289ca7 nxt=58)
17611	27.075723392	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=1432 more=y ident=0xbc289ca7 nxt=5
17612	27.076422494	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=2864 more=y ident=0xbc289ca7 nxt=5
17613	27.077457915	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	774	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
17614	27.079266837	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=0 more=y ident=0xbc289ca7 nxt=58)
17615	27.079266946	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=1432 more=y ident=0xbc289ca7 nxt=5
17616	27.081035775	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	IPv6	1494	IPv6 fragment (off=2864 more=y ident=0xbc289ca7 nxt=5
17617	27.081427549	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	774	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n

Obr. 2.16: Úspešný ICMPv6 záplavový útok pomocou fragmentácie



Obr. 2.17: Graf popisujúci pakety za sekundu v druhom scenári

Na grafe (viď obr. 2.17) je vidieť, že samotné ICMPv6 echo-request správy nedosiahli rýchlosti paketov 1000 za sekundu a je to z dôvodu, že väčšinu šírky sieťového pásma zabrali fragmenty, ktoré celkovo dosahujú (vrátane ICMPv6 echo-request paketov) až 2000 paketov za sekundu. Z tohoto vychádza, že záleží na veľkosti payload, ktorý dáme rozfragmentovať. Čím menší payload, tým viac echo-request paketov za sekundu sa dokáže odoslať.



Obr. 2.18: Zaťaženie mikropočítača Raspberry pri fragmentačnom záplavovom útoku

Na obrázku vyššie (viď obr. 2.18) je zobrazená záťaž mikropočítača Raspberry chvíľu pred úplným zamrznutím systému. Fragmentačný útok vyčerpal naplno swap priestor až do hodnoty 1024 M a systém začal byť nestabilný a neovládateľný. Hodnoty CPU dosahovali 20 % až 73.3 %. Pri spočítaní priemeru všetkých štyroch jadier bolo priemerné zaťaženie 53.7 %. Pri konci testu, keď bol swap priestor plne vyčerpaný a systém začal zasekávať, boli hodnoty CPU vyššie ako keď swap priestor ešte nebol vyčerpaný. Je vidno i vyššia spotreba pamäte v hodnote 1.51 G oproti pokojnému stavu.

Fragmentačný útok zároveň najviac dokázal zahltiť šírku sieťového pásma.

2.7 Tretí testovací scenár

V tejto časti bakalárskej práce bol vytvorený scenár, ktorý využíval rozšírené hlavičky protokolu IPv6 ako smerovacia hlavička, Hop-by-Hop, hlavička destinácie alebo fragmentačná hlavička. Účelom tohoto scenára bolo zaplaviť druhé zariadenie spracovávaním prídavných hlavičiek, čo je častokrát veľmi zdĺhavý proces pre sieťové zariadenia a celkovo zariadenia. Boli posielané ICMPv6 echo-request správy.

2.7.1 Príprava testovacieho prostredia

Pri príprave testovacieho prostredia nebolo potrebné vykonať zvlášť nastavenia. Prerekvizitou na fungovanie tohoto scenára bolo správne nastavenie druhého testovacieho scenára, a teda odstránenie potreby mať administrátorské práva pri spúšťaní Python skriptov.

2.7.2 Realizácia tretieho testovacieho scenára

V tejto podkapitole je rozobraná realizácia testovacieho scenára, ktorý vykonával záplavový útok pomocou IPv6 rozšírených hlavičiek.

Útok bol realizovaný tak, aby sa pomocou rozšírených hlavičiek zaťažilo druhé zariadenie, prípadne sieť, ktoré tieto pakety musí spracovávať a kontrolovať hlavičku po hlavičke. Všetky tieto hlavičky boli pridané postupne v poradí, v ktorom sa odporúča a toto poradie je možné nájsť v teoretickej časti v kapitole 1.1.5 o rozšírených hlavičkách protokolu IPv6.

V tomto útoku boli použité konkrétne 4 rozšírené hlavičky z tohoto zoznamu a to: Hop-by-Hop hlavička, hlavička Možnosti destinácie, Smerovacia hlavička, Fragmentačná hlavička. Hlavičky Hop-by-Hop a možnosti destinácie boli doplnené o tzv. „padding“ pomocou PadN výplne, ktorá slúži na vyplnenie týchto hlavičiek neaktívnymi bajtmi, aby sa veľkostne zarovnali tieto hlavičky. Každopádne vyplnenie pomocou PadN taktiež prispieva k preťaženiu zariadenia, pretože tieto bajty musia byť tiež spracované a skontrolované zariadením, a teda musia byť skontrolované i neaktívne bajty. Pad1 vyplní hlavičku o jeden bajt, pričom PadN sa dokáže využiť na vyplnenie n bajtmi.

```
▾ IPv6 Hop-by-Hop Option
    Next Header: Routing Header for IPv6 (43)
    Length: 10
    [Length: 88 bytes]
  ▸ Pad1
  ▸ PadN
  ▸ PadN
▾ Routing Header for IPv6 (Source Route)
    Next Header: Fragment Header for IPv6 (44)
    Length: 4
    [Length: 40 bytes]
  ▸ Type: Source Route (0)
    Segments Left: 2
    Reserved: 00000000
    Address[1]: 3ffe::200:ff:fe00:1
    Address[2]: 3ffe::200:ff:fe00:2
▾ Fragment Header for IPv6
    Next header: Destination Options for IPv6 (60)
    Reserved octet: 0x00
    0000 0000 0000 0... = Offset: 0 (0 bytes)
    .... .... .... .00. = Reserved bits: 0
    .... .... .... ...0 = More Fragments: No
    Identification: 0x00000000
▾ Destination Options for IPv6
    Next Header: ICMPv6 (58)
    Length: 20
    [Length: 168 bytes]
  ▸ Pad1
  ▸ PadN
  ▸ PadN
```

Obr. 2.19: ICMPv6 paket s rozšírenými hlavičkami

Na obrázku (viď obr. 2.19) je vidno viacero rozšírených IPv6 hlavičiek, pričom sa začína hlavičkou Hop-by-Hop, kde je daná ďalšia hlavička, dĺžka hlavičky a ďalej

iba výplň Pad1 a PadN.

Ďalej nasleduje smerovacia hlavička, ktorá pozostáva z ďalšej hlavičky, dĺžku, typu smerovacej hlavičky, počet segmentov (ktoré je potrebné ešte navštíviť), rezervovaných bitov a dvoch adries v určenej trase.

Fragmentačná hlavička obsahuje ďalšiu hlavičku, rezervovaný oktet, offset (udáva, či ide o prvý fragment), rezervované bity, či ešte nasledujú nejaké ďalšie fragmenty a nakoniec identifikácia na spätné zostavenie fragmentov.

Potom nasledujú možnosti destinácie s ďalšou hlavičkou, dĺžkou a výplňou Pad1 a PadN. Tieto výplne Pad1 a PadN sú použiteľné iba pri Hop-by-Hop hlavičke a pri možnostiach destinácie.

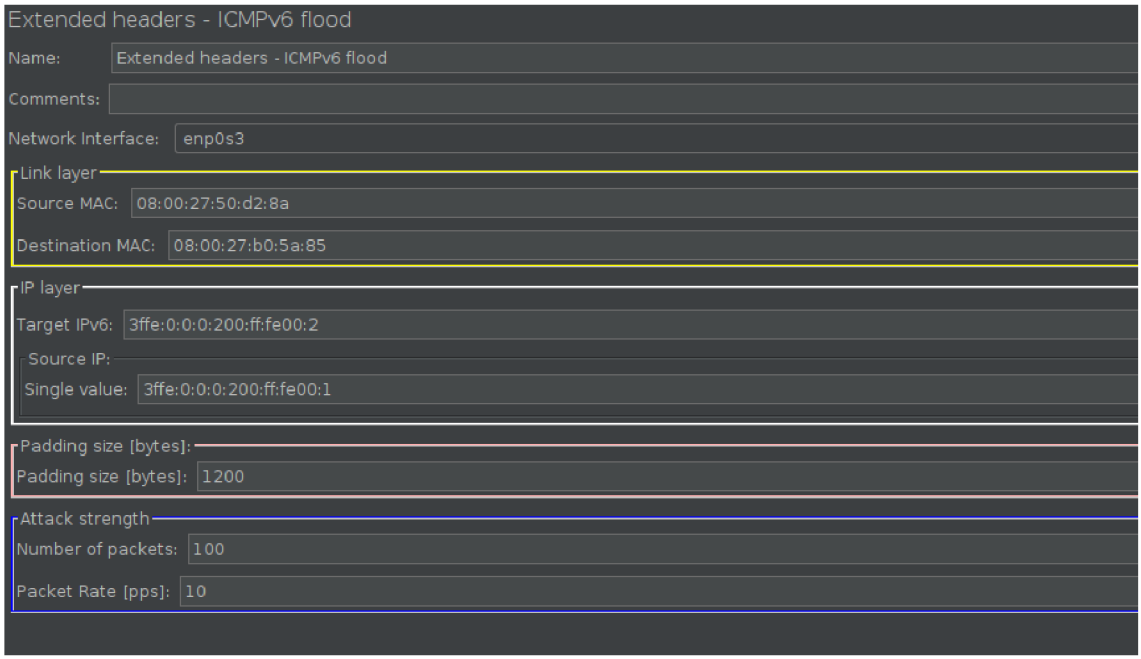
V skripte, ktorý používa knihovňu Scapy boli vytvorené pakety, ktoré sa skladajú z vyššie spomínaných hlavičiek a nakoniec bola pridaná IMCPv6 echo-request správa. Bola obmedzená možná veľkosť payloadu vložená používateľom z dôvodu veľkosti samotných rozšírených hlavičiek a bolo potrebné, aby sa veľkosť paketu zmestila do MTU hodnoty, ktorá bola nastavená na hodnotu 1500 bajtov.

```
1  def ipv6_packet(ip_src, ip_dest, packet_id, payload_size,
2  mac_src, mac_dest):
3      packet =  Ether(src=mac_src, dst=mac_dest)
4      /IPv6(src=ip_src, dst=ip_dest)
5      hopbyhop_options = [Pad1(), PadN(optdata=b"X"*80)]
6      hopbyhop_hdr = IPv6ExtHdrHopByHop(options=hopbyhop_options)
7      packet /= hopbyhop_hdr
8      routing_hdr = IPv6ExtHdrRouting(addresses=[ip_dest, ip_src])
9      packet /= routing_hdr
10     fragment_hdr = IPv6ExtHdrFragment()
11     packet /= fragment_hdr
12     destination_options = [Pad1(), PadN(optdata=b"XD"*80)]
13     destination_hdr = IPv6ExtHdrDestOpt(options=destination_options)
14     packet /= destination_hdr
15     packet /= ICMPv6EchoRequest(id=packet_id, seq=packet_id,
16     data=b"X" * payload_size)
17
18     return packet
```

Obr. 2.20: Ukážka kódu vytvorenia paketu s rozšírenými hlavičkami

V kóde (viď 2.20) z Python skriptu je znázornená funkcia ipv6_packet, ktorá

skladá obyčajný ICMPv6 echo-request, ale pred ICMPv6 hlavičkou je vložená rada rozšírených hlavičiek ako Hop-by-Hop, Smerovacia hlavička, Fragmentačná hlavička a Destinačná hlavička. Nakoniec sa je pridaná ICMPv6 hlavička, ktorá má v sebe premennú payload_size, ktorý vkladá užívateľ do GUI avšak táto hodnota je v samplery obmedzená na maximálnu hodnotu okolo 900 bajtov, aby sa paket zmestil do filtru MTU aj s rozšírenými hlavičkami.



Obr. 2.21: Grafické rozhranie pre záplavový útok s pomocou rozšírených hlavičiek

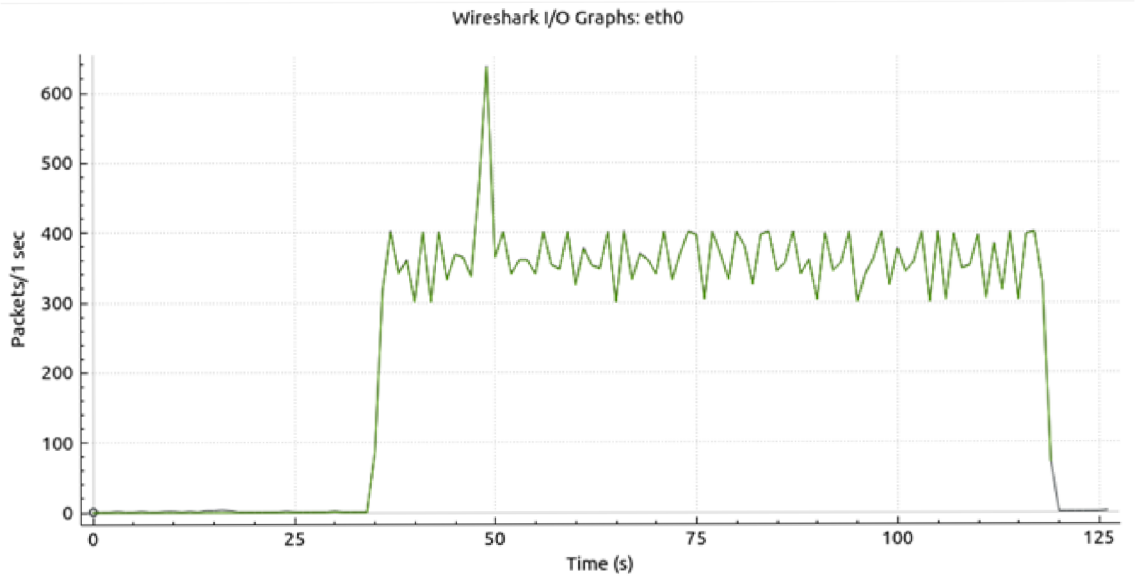
Grafické rozhranie (viď obr. 2.21) bolo orientované na jednoduchosť v orientácií pre užívateľa. Ako bolo už vyššie spomenuté, rámček payload je obmedzený na hodnotu 900 bajtov, takže ak užívateľ zadá vyššie číslo, tak žiadne pakety neprejdú na druhé zariadenie. Ak by aj táto funkcionalita nebola obmedzená, tak by pakety i tak neprešli, kvôli obmedzeniu filtru MTU.

2.7.3 Testovanie tretieho scenára

Testovací scenár je určený k zahlteniu zariadenia pomocou správ s rozšírenou hlavičkou.

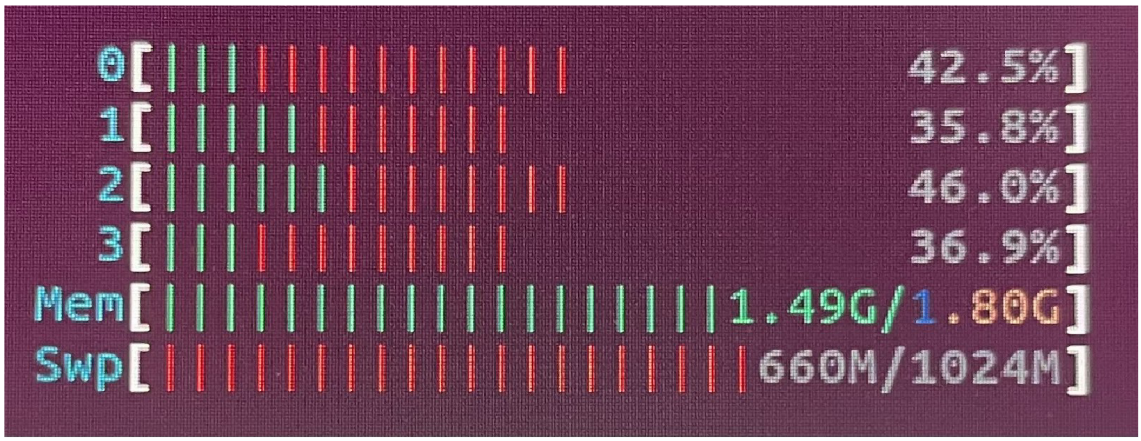
50908	248.669500796	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x0026, seq=38, hop li
50909	248.769564957	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x0027, seq=39, hop li
50910	248.869308519	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x0028, seq=40, hop li
50912	249.072787033	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x0029, seq=41, hop li
50913	249.173116158	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x002a, seq=42, hop li
50914	249.273100914	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x002b, seq=43, hop li
50915	249.373769696	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x002c, seq=44, hop li
50916	249.473643803	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x002d, seq=45, hop li
50917	249.573115953	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x002e, seq=46, hop li
50918	249.673230394	3ffe::200:ff:fe00:2	3ffe::200:ff:fe00:2	ICMPv6	1266	Echo (ping) request id=0x002f, seq=47, hop li

Obr. 2.22: Úspešný ICMPv6 záplavový útok pomocou rozšírených hlavičiek



Obr. 2.23: Graf popisujúci pakety za sekundu v treťom scenári

Útok nedosahoval plnej rýchlosti paketov zrejme kvôli tomu, že obsahuje rozšírené hlavičky. Takéto pakety zariadenie dlhšie kontrolovalo, a tak rýchlosť paketov za sekundu bola výrazne obmedzená.



Obr. 2.24: Zaťaženie mikropočítača Raspberry pri záplave rozšírenými hlavičkami

Zaťaženie bolo v tomto prípade priemerné, ale v rámci možností stále efektívne a škodlivé. Najväčšie zaťaženie bolo 46 % ako je možno vidieť na obrázku (viď obr. 2.24). Pri spočítaní priemeru všetkých štyroch jadier bolo priemerné zaťaženie 40.3 %. Pamäť sa pohybovala okolo 1.49 G.

2.8 Štvrtý testovací scenár

Štvrtý záťažový test bol mierený na záplavu pomocou rôznych ICMPv6 správ. Účelom tohoto scenára bolo zaplavenie zariadenia s náhodným odosielaním správ. Správy ako ICMPv6 echo-request, Neighbor Solicitation, Router Advertisement, Router Solicitation.

2.8.1 Príprava testovacieho prostredia

Príprava testovacieho prostredia nebola potrebná, pretože bola prevedená v minulých testovacích scenároch.

Testovacie prostredie bolo zostavené podľa vyššie uvedenej topológie.

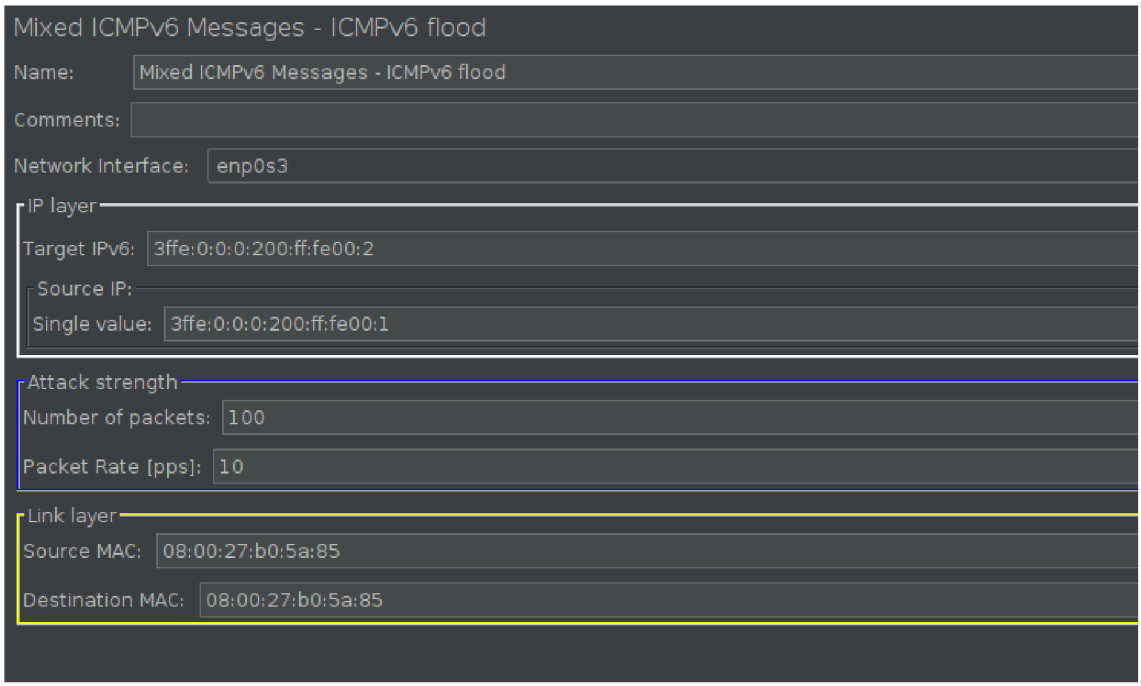
2.8.2 Realizácia štvrtého testovacieho scenára

V tomto testovacom scenári išlo predovšetkým o vyvolanie „chaosu“ použitím rôznych ICMPv6 správ. Boli použité správy informačné, ale aj chybové. Správy ako Echo-request, Neighbor Solicitation, Router Advertisement, Router Solicitation, Neighbor Advertisement a Parameter problem. Každá z týchto správ je zasielaná s úplnou náhodnosťou, a teda systém to môže zmiasť.

```
def create_packets(type, ip_src, ip_dest, mac_src, mac_dest):
#will create the packets with different ICMPv6 message
    try:
        if type == "echo-request":
            return Ether(src=mac_src, dst=mac_dest) /
            IPv6(src=ip_src, dst=ip_dest) / ICMPv6EchoRequest() /
            Raw(load=b"X" * 1400)
        elif type == "ns":
            return Ether(src=mac_src, dst=mac_dest) /
            IPv6(src=ip_src, dst=ip_dest) / ICMPv6ND_NS(tgt=ip_dest)
        elif type == "router":
            return Ether(src=mac_src, dst=mac_dest) /
            IPv6(src=ip_src, dst=ip_dest) / ICMPv6ND_RS()
        elif type == "advert":
            return Ether(src=mac_src, dst=mac_dest) /
            IPv6(src=ip_src, dst=ip_dest)
            /ICMPv6ND_NA(R=0, S=1, O=1, tgt="fe80::4")
        elif type == "routAdv":
            return Ether(src=mac_src, dst=mac_dest) /
            IPv6(src=ip_src, dst=ip_dest) / ICMPv6ND_RA()
        elif type == "param":
            return Ether(src=mac_src, dst=mac_dest) /
            IPv6(src=ip_src, dst=ip_dest) /
            ICMPv6ParamProblem(code=0, ptr=40)
            #ptr is error offset
    except Exception as e:
        return None
```

Obr. 2.25: Ukážka kódu viacerých náhodných ICMPv6 správ

Vyššie zobrazený kód z Python skriptu mal za úlohu vytvoriť niekoľko ICMPv6 paketov s rôznymi správami v rámci funkcie create_packets. Každý paket sa skladal z hlavičky Ether(), IPv6 hlavičky a konkrétnej ICMPv6 správy.



Obr. 2.26: Grafické rozhranie pre záplavový útok s náhodných ICMPv6 správ

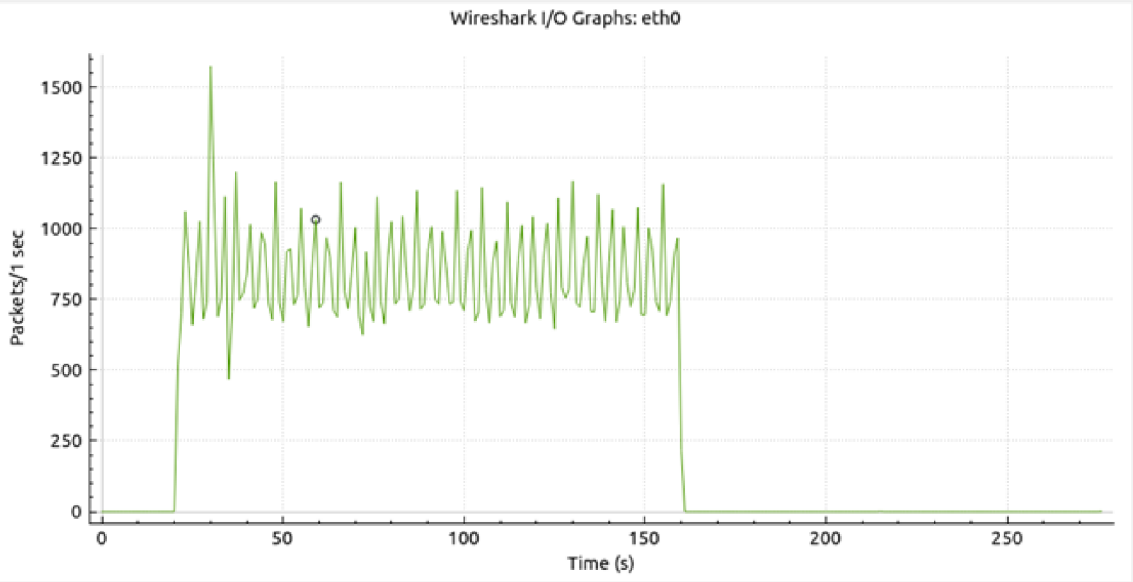
Pri tomto (viď obr. 2.26) grafickom rozhraní neboli prevedené žiadne veľké zmeny, sú tam iba dôležité textové rámčeky pre útok s náhodnými správami. Keďže pri tomto útoku by bolo obtiažne, aby užívateľ zadal veľkosť payload-u pre každý jeden typ tejto správy, tak payload každej ICMPv6 správy bol zadaný manuálne autorom práce priamo do skriptu, kde boli vytvorené tieto ICMPv6 správy.

2.8.3 Testovanie štvrtého scenára

Testovací scenár je určený k zahlteniu zariadenia pomocou náhodných správ. Scenár obsahuje informačné aj chybové správy. Postupnosť týchto správ je celkom náhodná. Z chybových správ bola pridaná ICMPv6 správa Parameter Problem a z informačných správ boli pridané Echo-request, Neighbor Solicitation, Router Advertisement, Router Solicitation, Neighbor Advertisement.

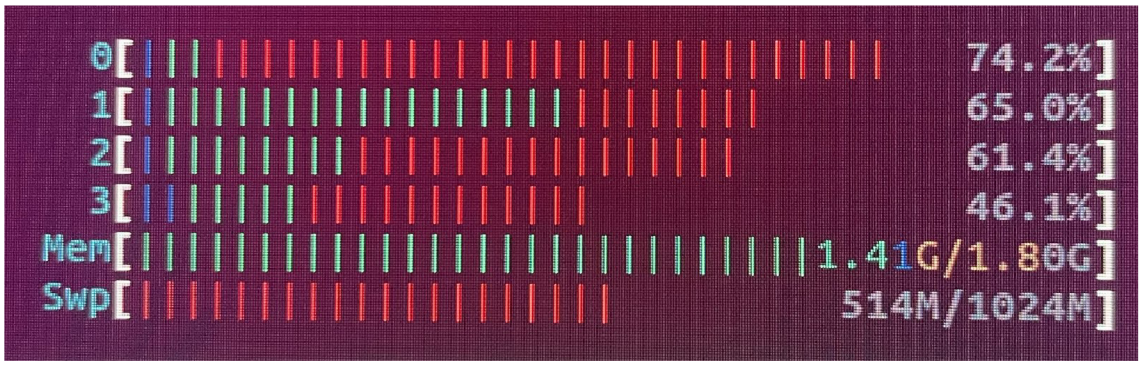
50098	127.092976299	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	1462	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
50099	127.193419248	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50100	127.295873686	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Parameter Problem (erroneous header field encountered
50101	127.393001183	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	1462	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
50102	127.492984676	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50103	127.593035863	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	1462	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
50104	127.693229539	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	1462	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
50105	127.793127896	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50106	127.893411098	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Router Solicitation
50107	127.993070824	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Parameter Problem (erroneous header field encountered
50109	128.092999969	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Router Solicitation
50110	128.193112484	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Router Solicitation
50111	128.293029700	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	70	Router Advertisement
50112	128.393054710	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50113	128.493114344	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	1462	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
50114	128.593062563	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50116	128.693173681	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	1462	Echo (ping) request id=0x0000, seq=0, hop limit=64 (n
50117	128.796522881	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50118	128.893899831	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	70	Router Advertisement
50119	128.993005815	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50120	129.092967718	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Solicitation for 3ffe::200:ff:fe00:2
50121	129.193151194	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Parameter Problem (erroneous header field encountered
50122	129.293171023	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Parameter Problem (erroneous header field encountered
50123	129.392937801	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Router Solicitation
50124	129.493235685	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50125	129.593026022	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Parameter Problem (erroneous header field encountered
50126	129.693205321	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Router Solicitation
50127	129.792950054	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Parameter Problem (erroneous header field encountered
50128	129.892929771	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	78	Neighbor Advertisement fe80::4 (sol, ovr)
50129	129.993045037	3ffe::200:ff:fe00:1	3ffe::200:ff:fe00:2	ICMPv6	62	Router Solicitation

Obr. 2.27: Úspešný ICMPv6 záplavový útok pomocou náhodných ICMPv6 správ



Obr. 2.28: Graf popisujúci pakety za sekundu vo štvrtom scenári

Útok (viď obr. 2.28) dosahoval určenú rýchlosť paketov za sekundu a z grafu je vidieť konzistentnosť útoku. Občasné odchýlky od určenej rýchlosti za sekundu spôsobujú ICMPv6 echo-reply správy, ktoré sa v grafe taktiež zobrazujú.



Obr. 2.29: Zaťaženie mikropočítača Raspberry pri záplavovom útoku pomocou náhodných správ

Pri útoku s náhodnými správami sa podarilo dosiahnuť celkom vysokú záťaž systému. Systém nezamrzol, ale hodnoty CPU sa pohybovali rôzne od 20 % až do vyššie (viď obr. 2.29) zobrazených hodnôt. Maximálna hodnota prvého jadra dosiahla 74.2 %. Pri spočítaní priemeru všetkých štyroch jadier bolo priemerné zaťaženie 61.675 %. Pamäť sa pohybovala okolo 514 M.

Záver

V rámci bakalárskej práce boli preštudované a popísané potrebné protokoly k praktickej časti bakalárskej práce.

Boli vytvorené štyri scenáre a ku každému z nich samostatný modul. Pri prvom scenári tvoril modul Java Sampler a GUI s Trafgenom, pri ďalších troch bol modul vytvorený z Java Sampler-u a GUI, prevádzka bola generovaná pomocou Python skriptu, ktorý využíval Scapy knihovňu. Pre každý scenár bol vytvorený samostatný modul. Tieto scenáre, ich tvorba a príprava k ich vytvoreniu bola popísaná v rámci praktickej časti.

Bola navrhnutá najefektívnejšia topológia pre testovanie vytvorených scenárov. Následne bola táto topológia zrealizovaná a nastavená. K testovaniu scenárov bolo v topológií zakomponované reálne zariadenie a tým bol mikropočítač Raspberry 4b.

Na záver praktickej časti boli otestované scenáre a vizuálne zdokumentované a popísané všetky dosiahnuté výsledky. Tieto scenáre boli otestované na reálnom zariadení Raspberry. Pri všetkých scenároch bolo zasielaných 1000 paketov za sekundu.

V týchto scenároch bol najúspešnejší fragmentačný záplavový útok, ktorý vyťažil mikropočítač natoľko, že ďalšia manipulácia s počítačom nebola možná a musel byť reštartovaný. Zároveň tento scenár dokázal najviac zaťažiť šírku sieťového pásma. Pomocou tohoto scenára sa podarilo zaťažiť najviac na 73.3 %. Pri spočítaní priemeru všetkých štyroch jadier bolo priemerné zaťaženie 53.7 %.

Scenár s náhodnými ICMPv6 správami dosiahol celkom veľké záťažové hodnoty a účel testovania splnil. Pomocou tohoto scenára sa podarilo zaťažiť mikropočítač najviac na 74.2 %. Pri spočítaní priemeru všetkých štyroch jadier bolo priemerné zaťaženie 61.675 %.

Pri scenári s rozšírenými hlavičkami spomaľovala kontrola všetkých hlavičiek rýchlosť paketov za sekundu a zaťaženie mikropočítača nebolo na vysokých číslach. Pomocou tohoto scenára sa podarilo zaťažiť najviac na 46 %. Pri spočítaní priemeru všetkých štyroch jadier bolo priemerné zaťaženie 40.3 %.

Scenár s posielaním obyčajných ICMPv6 správ mal veľkú limitáciu z dôvodu nedostatočného vývinu protokolu ICMPv6 v Trafgene a bolo možné posielať iba echo-request informačné správy. Pomocou tohoto scenára sa podarilo zaťažiť najviac na 26.1 %. Pri spočítaní priemeru všetkých štyroch jadier bolo priemerné zaťaženie 15.75 %.

Literatúra

[1] AppNeta. appneta. tcpreplay. Online. Dostupné po prihlásení z: https://github.com/appneta/tcpreplay?tab=readme-ov-file. [cit. 2023-05-22]

[2] BAGNULO Marcelo, GARCIA-MARTINEZ Alberto, VAN BEIJNUM Iljitsch. The NAT64/DNS64 tool suite for IPv6 transition Online. Dostupné po prihlásení z:https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6231295. [cit. 2023-05-21]

[3] BORKMANN, Daniel. TRAFGEN(8) - Linux manual page online | Administration and privileged commands. Linux manual page online. 2013, 2013(1), 1. Online. Dostupné z: https://www.venea.net/man/trafgen(8). [cit. 2023-12-11]

[4] Foreign Policy. *Setting Protocol*.Online. no. 129, 2002, 97. JSTOR, Dostupné po prihlásení z: http://www.jstor.org/stable/3183403. [cit. 2023-10-28].

[5] DEERING, S. a HINDEN, R. Internet Protocol, Version 6 (IPv6) Specification. Online. RFC 8200. July 2017. Dostupné z: https://doi.org/https://doi.org/10.17487/RFC8200. [cit. 2024-05-20].

[6] GORDON, RJ. *DDoS Attack Simulation to Validate the Effectiveness of Common and Emerging Threats*. Journal of Information Warfare. Peregrine Technical Solutions, 2017, 2017(vol. 16, 1), 49–63. ISSN 14453312, 14453347. Dostupné z: https://www.jstor.org/stable/26502876. [cit. 2023-11-24]

[7] GUPTA Mukesh, CONTA Alex.*Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*.Online. 2006, Dostupné z: https://datatracker.ietf.org/doc/html/rfc4443#section-2.[cit. 2023-11-20].

[8] HUNT, Craig.*TPC/IP Network Administration*. Third edition. O'Reilly Media, Inc.2002. Dostupné z: https://books.google.cz/books?hl=cs&lr=&id=wqabAgAAQBAJ&oi=fnd&pg=PT6&dq=HUNT,+Craig.TPC/IP+Network+Administration.+Third+edition.+O%E2%80%99Reilly+Media,+Inc.2002.&ots=zTPRBTuoPd&sig=uh787pRERVhvUizKDOX8BpzhJcA&redir_esc=y#v=onepage&q=HUNT%2C%20Craig.TPC%2FIP%20Network%20Administration.%20Third%20edition.%20O%E2%80%99Reilly%20Media%2C%20Inc.2002.&f=false.[cit. 2023-11.3]

[9] KLASSEN Fred. AppNeta. Tcpreplay. Online. Dostupné z:https://tcpreplay.appneta.com/. [cit. 2023-05-08]

[10] LI Lishan, REN Gang, LIU Ying , WU Jianping. Secure DHCPv6 mechanism for DHCPv6 security and privacy protection. Tsinghua Science and Technology, vol. 23, no. 1, pp. 13-21, Feb. 2018. Online. Dostupné po prihlásení z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8293069. [cit. 2023-05-10]

[11] MATTHEWS, Philip, VAN BEIJNU Iljitsch a BAGNULO Marcelo. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146. RFC Editor, 2011, 2011(6146), 45. Online. Dostupné z: https://www.rfc-editor.org/info/rfc6146. [cit. 2023-12-10]

[12] MyF5. BIG-IP Systems: DoS Protection and Protocol Firewall Implementations/ Detecting and Preventing System DoS and DDoS Attacks. 2014-01-31. Online. Dostupné z: https://techdocs.f5.com/kb/en-us/products/big-ip-afm/manuals/product/dns-dos-firewall-implementations-11-5-0/2.html#conceptid.[cit. 2023-05-13]

[13] OMAR E. Elejla, BELATON Bahari, ANBAR Mohammed, ALNAJJAR Ahmad. A Reference Dataset for ICMPv6 Flooding Attacks. Journal of Engineering and Applied Sciences 11(3): 476-481, 2016. ISSN: 1816-949X. Medwell Journals, 2016. Online. Dostupné po prihlásení z: https://www.researchgate.net/profile/Omar-Elejla/publication/305402583_A_Reference_Dataset_for_ICMPv6_Flooding_Attacks/links/578dce6e08ae59aa66816653/A-Reference-Dataset-for-ICMPv6-Flooding-Attacks.pdf. [cit. 2023-05-07]

[14] PARZIALE, Lydia, Dr. Wei LIU, Carolyn MATTHEWS, Nicolas ROSSELOT, Chuck DAVIS, Jason FORRESTER a David T. BRITT. *TCP/IP Tutorial and Technical Overview*. Eight edition. IBM Redbooks, 2006, 974 s. 8. ISBN 0738494682. Dostuné z: https://www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf. [cit. 2023-11-27]

[15] PIRKL Tomáš: Web server pro vestavěné aplikace, diplomová práce, Brno, FIT VUT v Brně, 2007. [cit. 2023-10-10]

[16] SATRAPA, Pavel. Vyšla definice NAT64. Lupa.cz. Dostupné z: https://www.lupa.cz/clanky/vysla-definice-nat64/. [cit. 2023-10-17]

[17] POSTEL J. INTERNET PROTOCOL. RFC 791. RFC Editor. 1981. Dostupné z: https://www.rfc-editor.org/rfc/rfc791. [cit. 2024-05-22]

[18] VAN HEERDEN, R.P., I.M. BESTER a I.D. BURKE.*A Review of IPv6 Security Concerns*.ISSN 14453312,14453347. Journal of Information Warfare. Online. 2012, 2012(vol. 11, 3), 25–38. Dostupné z: https://www.jstor.org/stable/26486877.[cit. 2023-10-30]

[19] RODRIGUES, GOMES Antonio, DEMION Bruno a MOUAWAD Philippe. Master Apache JMeter - From Load Testing to DevOps: Master performance testing with JMeter. Packt Publishing Ltd, 2019. Dostupné z: https://books.google.cz/books?hl=cs&lr=&id=D_amDwAAQBAJ&oi=fnd&pg=PP1&dq=RODRIGUES,+Antonio+Gomes,+Bruno+DEMION+a+Philippe+MOUAWAD.+Mas-+ter+Apache+JMeter+-+From+Load+Testing+to+DevOps:+Master+performance+testing+with+JMeter.+Packt+Publishing+Ltd,+2019&ots=iVhuMFKE4E&sig=EajKriR9Wwv9cWQkl31PRk335dc&redir_esc=y#v=onepage&q&f=false. [cit. 2023-10-30]

[20] SATRAPA, Pavol. *IPv6*. Online. 4. aktualizované a rozšírené vydanie. Praha: CZ.NIC, 2019. ISBN 978-80-88168-46-1. Dostupné z: https://www.nic.cz/files/edice/IPv6-2019.pdf. [cit. 2023-10-24].

[21] S, Rohith Raj; R, Rohith; MOHARIR, Minal a G, Shobha. SCAPY- A powerful interactive packet manipulation program. Online. 2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS). 2018, s. 1-5. ISBN 978-1-5386-7949-4. Online. Dostupné z: https://doi.org/10.1109/ICNEWS.2018.8903954. [cit. 2024-05-06].

[22] ŠULKA Samuel. ict-tester. jmeter-ddos-plugin. Online. Dostupné po prihlásení z: https://gitlab.utko.feec.vutbr.cz/ict-tester/jmeter-plugins/jmeter-ddos-plugin/-/tree/sulka__Bp?ref_type=heads. [cit. 2023-05-20]

[23] Websupport. htop – sprievodca nástrojom. Online. Dostupné z: https://www.websupport.sk/podpora/kb/htop-manual/. [cit. 2023-05-22]

Zoznam symbolov a skratiek

DDoS	Distributed denial of service
DoS	Denial of service
MTU	Maximum Transmission Unit
SYN	Synchronize
FTP	File Transfer Protocol
IPv6	Internet Protocol version 6
IPv4	Internet Protocol version 4
NAT64	Network address translation
NAT-PT	Network Address Translation, Protocol Translation
TCP/IP	Transmission Control Protocol/ Internet Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
DNS64	Domain Name System 64
DNS	Domain Name System
ICMPv6	Internet Control Message Protocol version 6
ICMP	Internet Control Message Protocol
GUI	Graphical User Interface
CPU	Central Processing Unit
GHz	Giga hertz
RAM	Random-access memory
ETH	Ethernet
ARP	Address Resolution Protocol
STP	Spanning Tree Protocol
LDAP	Lightweight Directory Access Protocol

USB	Universal Serial Bus
HDMI	High-Definition Multimedia Interface
SLAAC	Stateless Address Autoconfiguration
ARPA	Advanced Research Projects Agency
TTL	Time to live
RFC	Request for Comments
SNA	Systems Network Architecture)
OSI	Open System Interconnection)
SMTP	Simple Mail Transfer Protocol)

Zoznam príloh

A Obsah elektronickej prílohy

Príloha je v podobe .zip súboru a obsahuje všetky nižšie spomenuté súbory.

```
/ ..... koreňový adresár priloženého archívu
├── java ..... adresár s java triedami
│   ├── icmpv6flood
│   │   ├── Icmpv6FloodGui.java
│   │   └── Icmpv6FloodSampler.java
│   ├── fragmentipv6
│   │   ├── FragmentGui.java
│   │   └── FragmentSampler.java
│   ├── extendedheaderfloodipv6
│   │   ├── ExtendedHeaderFloodGui.java
│   │   └── ExtendedHeaderFloodSampler.java
│   └── mixedicmpv6flood
│       ├── MixedICMPv6FloodGui.java
│       └── MixedICMPv6FloodSampler.java
├── skripty ..... adresár s Python skriptami
│   ├── hdr.py
│   ├── fragmentIPv6.py
│   └── mixedICMPv6.py
├── trafgen ..... adresár s Trafgen config súborom
│   └── AbstractIcmpv6Flood.cfg
└── Ddos.jar
```