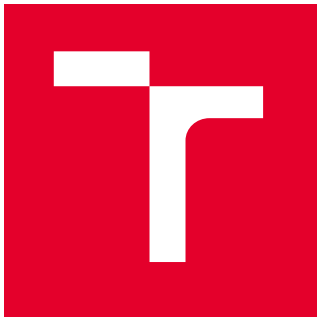


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

NÁSTROJ PRO GENEROVÁNÍ LOW-RATE DOS ÚTOKŮ

GENERATOR OF LOW-RATE DOS ATTACKS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

Michal Kaiser

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Marek Sikora

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Michal Kaiser ***ID:*** 221034

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Nástroj pro generování low-rate DoS útoků

POKYNY PRO VYPRACOVÁNÍ:

Low-rate útoky jsou poměrně novou a sofistikovanou podskupinou útoků typu DoS, které se snaží běžným uživatelům znemožnit přístup k Internetové službě (nejčastěji webovým stránkám). Oproti klasickým záplavovým DoS generují low-rate útoky jen velmi málo paketů, které však vhodně zneužívají slabin serverů a způsobí zaplnění vstupních front serverů a zablokování přístupu ostatním legitimním uživatelům. Úkolem této bakalářské práce je podrobně nastudovat mechanismy low-rate útoků NewShrew a LoRDAS, vytvořit funkční generátor obou útoků a případně útoky optimalizovat vzhledem k aktuálním bezpečnostním opatřením počítačových systémů. Dalším úkolem bude navrhnout a vytvořit virtuální experimentální pracoviště, ve kterém budou útoky otestovány a jejich funkčnost ověřena. Dalším úkolem práce bude navrhnout a ověřit metodu jejich prevence, detekce a filtrace pro ochranu webových serverů.

DOPORUČENÁ LITERATURA:

[1] LUO, Jingtang a Xiaolong YANG. The NewShrew attack: A new type of low-rate TCP-Targeted DoS attack. 2014 IEEE International Conference on Communications (ICC). IEEE, 2014, 2014, , 713-718. DOI:10.1109/ICC.2014.6883403

[2] MACIÁ-FERNÁNDEZ, Gabriel, Jesús E. DÍAZ-VERDEJO, Pedro GARCÍA-TEODORO a Francisco DE TORO-NEGRO. LoRDAS: A Low-Rate DoS Attack against Application Servers. Critical Information Infrastructures Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, 2008, , 197-209. Lecture Notes in Computer Science. DOI:10.1007/978-3-540-89173-4_17

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Marek Sikora

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

ABSTRAKT

Bakalárska práca sa venuje implementácii a problematike Low-rate DoS útokov, kde sa na základe týchto získaných informácií v priebehu zostavovania práce zostrojia generátory Low-rate DoS útokov s názvom NewShrew a LoRDAS. Čitateľ sa oboznámi všeobecne s problematikou DoS útokov, ich fungovaním a zneužívaním sieťovej komunikácie, kde sa následne tieto základne útoky rozdelia podľa charakteristiky ich priebehu a popíšu sa podrobnejšie Low-rate útoky. Po teoretickom úvode sa popíšu útoky NewShrew a LoRDAS, a zostavia sa generátory týchto útokov v programovacom jazyku Python. Následne sa oboznámi s metódami ich detekcie a obranných techník. Tieto útoky sa následne otestujú na konfigurovanej sieti. Výsledkom práce budú útoky NewShrew a LoRDAS, ktoré sa snažia zneprístupniť webový server, výsledky správneho nastavenia útokov a navrhnuté detekčné a obranné systémy.

KĽÚČOVÉ SLOVÁ

Low-rate DoS, generátor Low-rate DoS, DoS, NewShrew, LoRDAS, aplikačné servery, RTO, Mechanizmus pomalého štartu

ABSTRACT

The bachelor thesis is focused on the implementation and issues of Low-rate DoS attacks, where based on this information obtained during the compilation of the bachelor thesis, Low-rate DoS attack generators called NewShrew and LoRDAS are constructed. The reader will be introduced to the general issues of DoS attacks, their operation, and the exploitation of network communication, where these basic attacks will then be divided according to their network flow characteristics, and the Low-rate attacks will be described in more detail. After the theoretical introduction, the NewShrew and LoRDAS attacks are described. Subsequently, the methods of their detection and defense techniques will be introduced. Generators of these attacks are implemented in the Python programming language. These attacks will then be tested on test networks. The result of the thesis will be NewShrew and LoRDAS DoS attacks that attempt to make a web server inaccessible, results of correct settings of attacks, and also detection and prevention system are proposed.

KEYWORDS

Low-rate DoS, Low-rate DoS generator, DoS, NewShrew, LoRDAS, aplication servers, RTO, Slow Start Mechanism

KAISER, Michal. *Nástroj pro generovaní low-rate DoS útoků*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 99 s. Bakalárska práca. Vedúci práce: Ing. Marek Sikora

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Michal Kaiser
VUT ID autora:	221034
Typ práce:	Bakalárska práca
Akademický rok:	2021/22
Téma závěrečnej práce:	Nástroj pro generovaní low-rate DoS útoků

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. Marekovi Sikorovi za cenné rady, odborné vedenie, konzultácie a trpezlivosť počas tvorby bakalárskej práce. Taktiež chcem poďakovať mojej rodine za ich podporu počas celého štúdia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Bakalárska práca sa zaoberá popisom útokov odopretia služby (angl. Denial of service (DoS)) s názvom Low-rate DoS. Zameriava sa na všeobecnú funkčnosť DoS útokov, opísanie Low-rate útokov, ich rozdelenie a obranné techniky. Na základe týchto znalostí sú zostavené skripty s implementáciou Low-rate útokov, ktoré sa nazývajú NewShrew a LoRDAS. Podľa popisu vrstiev ISO/OSI referenčného modelu sa NewShrew útok zameriava najmä na aplikácie, ktoré komunikujú pomocou Protokolu riadenia prenosu (angl. Transmission Control Protocol (TCP)), najčastejšie webové servery a FTP (angl. File Transfer Protocol) servery. Ďalšou kategóriou sú aplikačné servery, ktorých zraniteľnosti zneužíva LoRDAS útok. DoS útoky a najmä Low-rate útoky sa stávajú vo svete internetu veľkou hrozbou, keďže ich detekcia donedávna nebola navrhnutá a unikali rôznym obranným mechanizmom a filtračným systémom. Tieto útoky prevažne fungujú na periodickom posielaní dát, vďaka ktorým sa snažia zneužiť určitú zraniteľnosť siete a znefunkčniť sieťové zariadenia. Útoky sú ťažšie rozoznateľné od bežných DoS útokov, keďže k ich funkčnosti a zneprístupneniu sieťového zariadenia nie je potrebné posielanie veľkého množstva paketov (v porovnaní so záplavovými útokmi).

Cieľom práce je teoreticky popísať Low-rate útoky, rozdeliť ich, popísať detekčné a obranné systémy a následne na základe popísaných teoretických znalostí implementovať a realizovať funkčný generátor NewShrew Low-rate DoS útoku a LoRDAS Low-rate DoS útoku, ktoré budú implementované v programovacom jazyku Python. Na základe dobrej znalosti TCP sieťovej prevádzky je cieľom útokov webový server v nakonfigurovanej testovacej sieti dostať do stavu odopretia služby. Vytvorená implementácia útokov je naprogramovaná vo forme skriptu, kde má útočník možnosť si navoliť niekoľko parametrov, vďaka ktorým si môže útok prednastaviť priamo na požiadavky siete.

V prvej časti je rozobratá téma sieťovej komunikácie, kde pochopenie jej funkčnosti je potrebné na realizovanie a pochopenie útokov na jednotlivých vrstvách ISO/OSI modelu. Preto sú v tejto časti teoreticky popísané TCP/IP a ISO/OSI referenčné modely.

Druhá časť práce sa venuje všeobecnému popisu DoS útoku. Opisuje jeho funkčnosť, realizáciu a rozdelenie útokov. Zadefinujú sa pojmy ako DoS a distribuované odopretie služby (angl. Distributed Denial of Service (DDoS)) a vývoj v posledných rokoch, ktorý viedol k vytvoreniu nových druhov Low-rate DoS útokov. A následne popíšeme rozdiel medzi klasickými záplavovými (angl. flood) DoS útokmi s novými Low-rate a Slow DoS.

V tretej časti je popísaný útok NewShrew, kde vymedzíme jeho parametre, nastavenie útoku a v poslednej podkapitole tejto časti popíšeme implementáciu tohto

útoku v programovacom jazyku Python s ukážkou na jeho generovanie.

V štvrtej časti je popísaný útok LoRDAS, ktorý ako zraniteľnosť využíva spôsob spracovania požiadavok na aplikačných serveroch, kde sa popíšu spôsoby, akými aplikačné servery, konkrétne webové, spracovávajú požiadavky. Následne sa popíše, akým spôsobom je vhodné nastaviť parametre útoku LoRDAS na jednotlivé spôsoby spracovania požiadaviek. V poslednej podkapitole tejto časti sa popíše implementácia a generovanie LoRDAS útoku.

V piatej časti práce sa popíšu detekčné a obranné systémy voči Low-rate DoS, ktoré bolo potrebné v nedávnej dobe zrealizovať práve kvôli ťažkosti ich detekovania. V tejto časti sa preto opíšu možnosti detekovania Low-rate útokov najmä na základe ich periodicity, ich vlastností alebo na základe frekvenčnej a časovej domény. Na základe toho sú ďalej popísané obrané mechanizmy ako filtrácia, prerozdelenie zdrojov a zlepšenie parametrov siete.

V šiestej časti popíšeme konfiguráciu testovacej siete určenej pre funkčnosť NewShrew DoS, ktorá je potrebná na jeho realizovanie. Útok otestujeme na danej sieti, kde na základe výsledkov testovania dospejeme k správnemu nastaveniu parametrov útoku a jeho efektívnosti a účinnosti na určitý druh sietí. V poslednej podkapitole sú popísané možnosti obrany voči tomuto útoku.

V poslednej časti je LoRDAS útok otestovaný na nakonfigurovanej sieti, kde sa otestuje jeho účinnosť a efektivita na jednotlivé prístupy k spracovaniu požiadaviek webovými servermi, a následne sa teoreticky navrhnú možné detekčné a obranné mechanizmy.

1 Sieťová komunikácia

Rôzne typy DDoS a DoS útokov sa zameriavajú na rôzne časti sieťovej komunikácie. Aby sme teda pochopili, ako fungujú typy DoS útokov, vedeli sa voči nim brániť a mohli tieto typy útokov filtrovať, tak bude potrebné vedieť, ako funguje sieťová komunikácia. Sieťová komunikácia sa skladá z mnohých vrstiev, na ktorých jednotlivo pracuje niekoľko protokolov alebo aplikácií, ktoré vytvárajú spojenie. Tomuto rozdeleniu a opísaniu vrstiev sa venuje niekoľko referenčných modelov. Najčastejšie sa však používajú TCP/IP model a ISO/OSI model [1].

1.1 TCP/IP model

TCP/IP model sa skladá zo štyroch vrstiev, kde prvou najnižšou vrstvou je prístupová vrstva, druhá je internetová vrstva, tretia je transportná vrstva a štvrtou najvyššou vrstvou je aplikačná vrstva. TCP/IP model je vytvorený podľa určitej sady protokolov na rozdiel od ISO/OSI modelu. Pre lepšie porozumenie TCP/IP je teda kľúčové chápať, ako fungujú protokoly na jednotlivých vrstvách [2]. TCP/IP vrstvy môžeme vidieť na tabuľke 1.1.

Vrstvy	TCP/IP
4	Aplikačná vrstva
3	Transportná vrstva
2	Internetová vrstva
1	Spojová vrstva

Tab. 1.1: Prehľad vrstiev TCP/IP modelu.

1.2 ISO/OSI model

Model ISO/OSI pozostáva zo siedmich vrstiev. Pomocou tohto modelu komunikuje medzi sebou niekoľko druhov zariadení, kde tieto zariadenia pracujú na určitých vrstvách podľa použitia. Sieťové zariadenia ako smerovač a prepínač pracujú väčšinou v spodných troch vrstvách ISO/OSI modelu. Počítače a servery väčšinou pracujú na všetkých siedmich vrstvách. Každá vrstva spracováva údaje v rôznych jednotkách, ktoré sa nazývajú Protocol Data Unit (PDU). Na každej vrstve sa prenášané informácie zapúzdrujú do novej PDU, ktorá pridáva buď hlavičku, alebo pätu, ktorá nesie v sebe informácie potrebné k úspešnému riadeniu komunikácie v sieti. Na 1.2 je znázornená tabuľka s jednotlivými vrstvami OSI/ISO modelu. Keďže sieťová prevádzka

na ISO/OSI modeli je viac štrukturalizovaná, tak v ďalších kapitolách sa bližšie pozrieme na funkčnosť týchto vrstiev a rozoberieme, ako na týchto vrstvách fungujú DoS útoky. Tie v nasledujúcej kapitole detailnejšie popíšeme a rozdelíme [3, 4].

Vrstvy	ISO/OSI	PDU
7	Aplikačná vrstva	Data
6	Prezentačná vrstva	Data
5	Relačná vrstva	Data
4	Transportná vrstva	Segment
3	Sieťová vrstva	Paket
2	Spojová vrstva	Rámec
1	Fyzická vrstva	Bity

Tab. 1.2: Prehľad vrstiev ISO/OSI modelu.

1.2.1 Fyzická vrstva

Fyzická vrstva spracováva a prenáša dáta ako sekvenciu bitov. To znamená, že PDU pre fyzickú vrstvu je jeden bit. Povinnosťou fyzickej vrstvy je poskytovať transparentný prenos bitov zo spojovej vrstvy odosielateľa na spojovú vrstvu prijímateľa komunikácie. To docielia elektrické, mechanické a funkcionálne prostriedky na aktiváciu, deaktiváciu a údržbu fyzického spojenia medzi dvomi sieťovými rozhraniami. Okrem prenášania dát musia byť prenášané aj informácie o riadení prenosu. Tieto informácie sú pridané do dátového toku a prenášané na rovnaké kanály ako ostatné dáta. Tento typ prenosu sa nazýva in-line signalizácia. Druhou možnosťou je prenášanie dát pomocou off-line signalizácie, pri ktorej sa dáta prenášajú pomocou samostatného kanálu. Výber spôsobu prenosu riadiacich informácií je ponechaný na použitom protokole. Protokoly fyzickej vrstvy sa líšia v závislosti od typu fyzického média a typu signálu, ktorý sa na ňom prenáša. Signálom môže byť prenášané elektrické napätie cez kábel, svetelný signál prenášaný cez vlákno alebo dokonca elektromagnetický signál prenášaný vzduchom [3].

Útoky na fyzickej vrstve zahŕňajú fyzické zničenie, obštrukciu, manipuláciu a nesprávne fungovanie fyzických médií, čo vedie k jeho nedostupnosti. To vyžaduje opravu, aby boli fyzické rozhrania a média k dispozícii. V tomto prípade nemôžeme hovoriť o bežnom DoS útoku, preto sa mu už bližšie venovať nebudeme, aj keď vo svojej podstate ide o odopretie služby [5].

1.2.2 Spojová vrstva

Vrstva dátového spojenia je zodpovedná za zriadenie, udržiavanie a rozhodovanie o tom, ako preniesť dáta cez fyzické médium. PDU v tejto vrstve je rámec. IEEE 802 štandardy sa používajú ako protokoly na komunikáciu spojovej vrstvy. Veľkosť týchto rámcov sa môže pohybovať od niekoľko desiatok bajtov po niekoľko tisícok bajtov. Spojová vrstva pridáva k paketu, ktorý je na sieťovej vrstve, svoje riadiace informácie vo forme hlavičky a päty. Porovnaním s ostatnými vrstvami má mnoho zložitých funkcií. Jednými z hlavných úloh spojovej vrstvy je riadenie prepojovania dátových okruhov, identifikácia a výmena parametrov, detekcia chýb, odovzdávanie niektorých sieťových konfigurácií a manažment dátovej vrstvy [3].

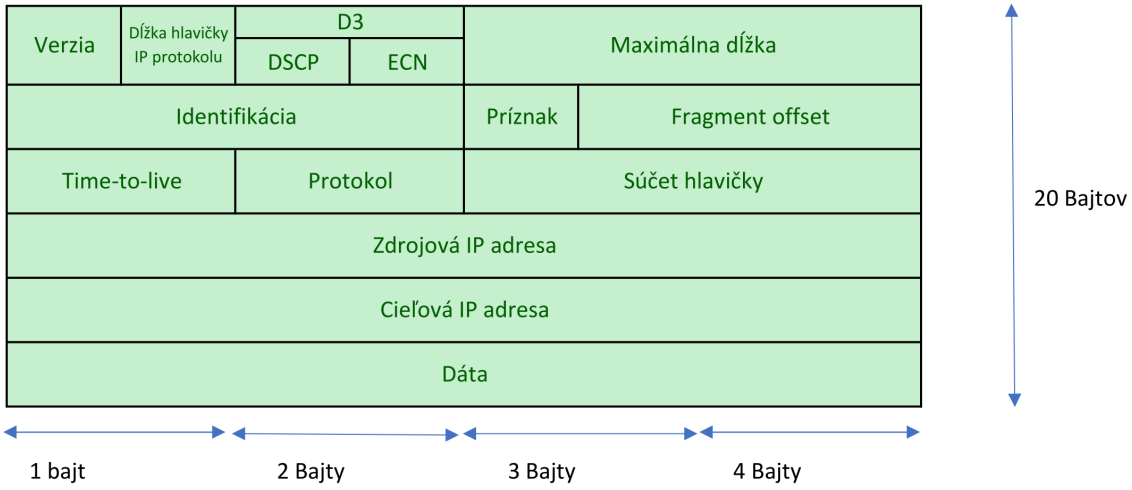
Najkritickejšie útoky na spojovej vrstve zahŕňajú vyčerpanie tabuľky MAC adries, spoofing ARP, spoofing MAC adries, útoky na virtuálne siete LAN (VLAN) a mnohé ďalšie. Tieto útoky vo všeobecnosti narúšajú bežný prevádzkový tok od odosielateľa k príjemcovi [6].

1.2.3 Sieťová vrstva

PDU pre sieťovú vrstvu je paket. Sieťová vrstva je veľmi dôležitá, čo sa týka smerovania údajov z jednej siete do druhej a kontroly podsietí. Smerovanie môže byť zložitá operácia, kde v niektorých prípadoch môže veľa faktorov prispieť k výberu najlepšej trasy od zdroja k cieľu. Jednou z hlavných úloh sieťovej vrstvy je smerovanie a prenos paketov, sieťové pripojenie a multiplexovanie paketov, segmentácia siete, detekcia chýb a obnova chybných paketov, kontrola toku dát, zrýchlenie prenosu dát, resetovanie a obnova sieťového pripojenia, mapovanie IP adries k MAC adresám, a riadenie sieťovej vrstvy [3].

Na sieťovej vrstve sa tok dát riadi pomocou informácii internetového protokolu (IP), kde sú tieto informácie obsiahnuté v hlavičke IP protokolu, ktorá je znázornená na obrázku 1.1.

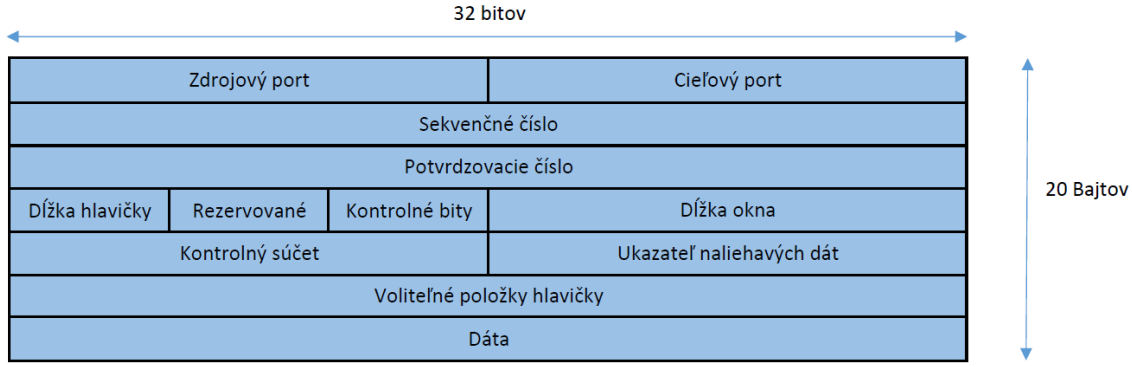
Útoky DoS na sieťovej vrstve zahŕňajú zneužitie siete obete s väčšou premávkou, ako dokáže sama zvládnuť. Ako výsledok nadbytočnej sieťovej prevádzky sa spustí sieť obete, ktorá reaguje pomaly na požiadavky alebo zahodí niektoré pakety. Strata paketov môže spôsobiť záplavový útok opakovaným odosielaním požiadaviek, ktoré ďalej zvyšujú sieťovú prevádzku. Zvýšená sieťová prevádzka saturuje sieť a stáva sa nedostupnou pre legitímnych užívateľov. Napríklad ako v Ping záplavovom útoku, kde ICMP dáta zaplavujú sieť, spotrebúvajú celú šírku pásma a sieť začne odmietať služby. Počas útoku na tejto vrstve je šírka pásma zaplavená dostatočným počtom paketov. Výsledkom útokov bude teda zneprístupnená celá šírka pásma sieťovej linky pre legitímnych užívateľov [7].



Obr. 1.1: Hlavička paketu IPv4.

1.2.4 Transportná vrstva

Transportná vrstva je štvrtou vrstvou v referenčnom modele ISO/OSI, na ktorej PDU je segment. Keďže existujú aplikácie, ktoré môžu poskytovať dva typy služieb, tak transportná vrstva je rozdelená na spojovo orientované pripojenie a na nespojovo orientované pripojenia. Transportná vrstva ma na starosti niekoľko úloh ako ustanovenie a ukončenie pripojovania segmentov, sekvenčnú kontrolu, end-to-end detekovanie a opravovanie chýb, segmentáciu paketov, end-to-end kontrolu toku, monitorovanie parametrov kvality služieb (QoS), a obmedzovanie toku PDU. Transportná vrstva poskytuje veľkú podporu relačnej vrstve, kde sa stará o poskytovanie mechanizmu na rozlíšenie, ktoré údaje patria jednotlivým reláciám [3]. TCP segment je zobrazený na obr. 1.2.



Obr. 1.2: Hlavička TCP segementu.

Spojovo orientovaná komunikácia prechádza v sieťovej prevádzke do troch fáz: založenie prenosu, prenos údajov a ukončenie prenosu. Počas fázy založenia transportná vrstva nastavuje parametre end-to-end komunikácie. Napríklad multiplexovanie relácií do sieťového pripojenia, optimálna veľkosť segmentu a získanie sieťového pripojenia, ktoré zodpovedá potrebám jednotlivých relácií. Po nadviazaní spojenia sa spustí prenos dát a využíva detekciu chýb a mechanizmov korekcie, segmentácie a riadenia toku. Keď sa údaje prenesú, odosielateľ pošle príjemcovi žiadosť o ukončenie spojenia a spojenie sa ukončí [3].

Útoky DoS transportnej vrstvy sú vo všeobecnosti založené na útoku s veľkým objemom dát útočiacim na sieťovú infraštruktúru. Tieto útoky sa snažia o generovanie a preposielanie veľkého množstva sieťovej komunikácie na znefunkčnenie celého radu dostupných sieťových služieb používaných legitímnymi užívateľmi. Takéto útoky najčastejšie zahŕňajú zneužívanie TCP alebo UDP protokolov na zahltenie sieťových prostriedkov. Napríklad protokol TCP používa systém segmentov začatia (angl. request) a potvrdenia (angl. acknowledgement) komunikácie a na udržanie jej správnej funkčnosti. Napríklad pre naviazanie komunikácie pošle klient požiadavku na server a pre potvrdenie prijatia požiadavky pošle naspäť klientovi acknowledgment segment. Po výmene požiadavky a acknowledgment segmentov môžu koncové zariadenia začať komunikáciu. Transportná vrstva však stále posiela požiadavky a acknowledgment segmenty a drží spojenie otvorené, kým zhodujúci sa acknowledgemnt paket nedorazí. Tento stav je často zneužívaný útočníkmi v DoS útokoch, ktoré sa nazývajú SYN záplavové útoky. Pri tomto type útoku je posielané veľké množstvo falošných požiadaviek na server, čoho výsledkom je zahltenie pamäte serveru kvôli držaniu veľkého množstva spojení, ktoré sú otvorené, ale komunikácia nie je započatá. Keď vyprší čas podvodných požiadaviek, server je nedostupný a legitímni užívatelia k nemu nemajú prístup. Najčastejšie využívané protokoly pri útokoch na transportnú vrstvu sú TCP-SYN útok, CHARGEN-UDP [8, 9].

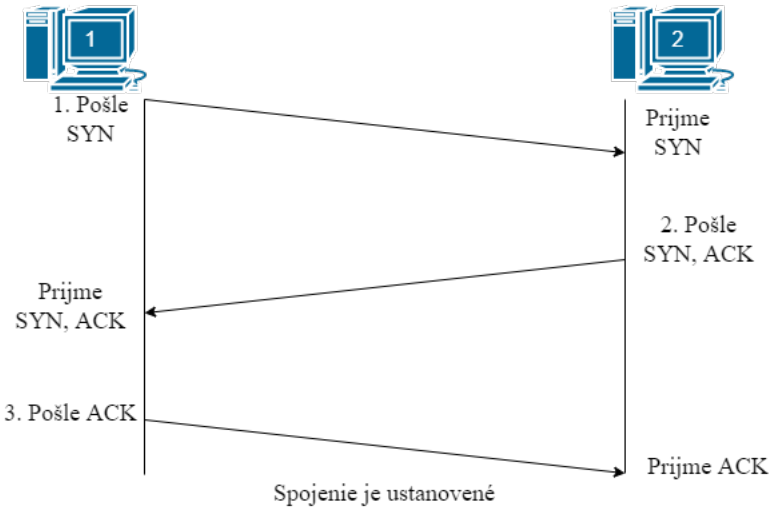
TCP Three-way Handshake

Pri procese Three-way handshake si sieťové zariadenia nadväzujú spojenie, sledujú každý segment v rámci relácií a vymieňajú si informácie o prijatých údajoch pomocou informácií v hlavičke TCP. TCP je plne duplexný protokol, kde každé spojenie predstavuje dve jednosmerné komunikačné relácie. Ako je ukázané na obrázku 1.3, tak Control Bits v TCP hlavičke zobrazuje fázu a status pripojenia. Jednými z hlavných funkcií Three-way Handshaku sú:

- Zisťuje, že cieľové zariadenie je prítomné v sieti.
- Overuje, že cieľové zariadenie má aktívnu službu a prijíma požiadavky na číslo cieľového portu, ktoré má klient v úmysle použiť.

- Informuje cieľové zariadenie, že zdrojový klient zamýšľa nadviazať komunikačnú reláciu na tomto čísle portu.

Po dokončení komunikácie sa relácie zatvoria a spojenie sa ukončí [10].

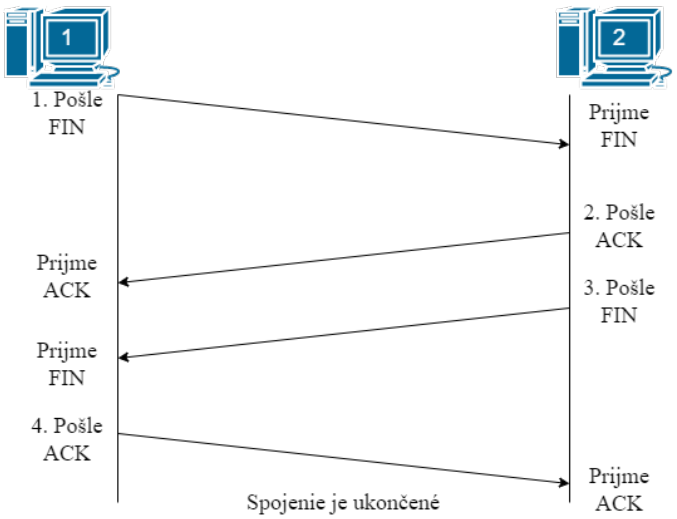


Obr. 1.3: Nadviazanie TCP spojenia.

Nadviazanie spojenia pozostáva z troch krokov. Pri inicializácii spojenia klient pošle požiadavku TCP s Control bitom nastaveným na hodnotu SYN (Synchronizácia), čo je synchronizačný príznak pre začatie client-to-server relácie. Server potvrdí komunikáciu s klientom pomocou TCP segemntu s Control bitom nastaveným na hodnotu ACK (Potvrdenie), čo je potvrdzovací príznak, a odošle požiadavku na server-to-client reláciu pomocou segmentu s Control bitom SYN. Klient potvrdí komunikáciu so serverom pomocou TCP segemntu s Control bitom ACK [10].

TCP ukončenie spojenia

Pri ukončení TCP spojenia zasa proces pozostáva zo štyroch krokov. Ak sa pripojenie ukončuje, v hlavičke segmentu musí byť nastavený ovládací príznak finish (FIN). Na ukončenie každej jednosmernej TCP relácie sa používa obojsmerný handshake, ktorý pozostáva zo segmentu FIN a segmentu ACK. Preto na ukončenie jednej konverzácie podporovanej TCP sú potrebné štyri výmeny na ukončenie oboch relácií. Ukončenie môže iniciovať klient alebo server. Keď klient už nemá žiadne ďalšie údaje na odoslanie, tak odošle segment s nastaveným príznakom FIN. Server odošle ACK na potvrdenie prijatia FIN segmentu na ukončenie relácie od klienta k serveru. Server pošle FIN klientovi, aby ukončil reláciu server-to-client. Klient odpovie ACK na potvrdenie FIN zo servera [10]. Názorné ukončenie spojenia je na obrázku 1.4.



Obr. 1.4: Ukončenie TCP spojenia.

1.2.5 Relačná vrstva

Relačná vrstva poskytuje mechanizmus správy dialógu medzi aplikačnými procesmi koncového používateľa. Poskytuje buď duplexnú, alebo poloduplexnú komunikáciu, zodpovedá za checkpointing, odloženie, ukončenie a reštart spojenia. Táto vrstva nadväzuje a ukončuje TCP/IP relácie.

Jeden z prípadov DoS útokov na relačnú vrstvu je práve na chybný formát žiadostí služby bezpečných socketov (angl. Secure socket layer (SSL)). SSL poskytuje bezpečnosť vo webových službách ako bankovníctvo, online nakupovanie atď. Kvôli bezpečnosti väčšina organizácií migruje na SSL, aby poskytovali lepšie zabezpečenie ich služieb. V súčasnosti je väčšina transakcií vedená cez SSL, no napriek tomu to láka veľké množstvo útočníkov. Najčastejšie sa objavuje zneužívanie three-way handshaku TCP protokolu. Po ukončení TCP handshaku nastáva handshake SSL spojenia. Počas nadviazania spojenia SSL dochádza k výmene správ na overenie pravosti oboch komunikujúcich zariadení. Stanovia šifrovací kľúč a možnosti následnej bezpečnej komunikácie. Napriek vysokej bezpečnosti SSL protokolu existuje niekoľko útokov, ktoré zneužívajú SSL a snažia sa zahltiť zdroje napadnutého zariadenia. Jedným z týchto útokov je SSL traffic záplavový útok, ktorý sa snaží zneprístupniť dostupné sieťové pripojenie, alebo lepšie povedané šírku pásma, ktoré je vytvorené cez zabezpečený kanál [11].

Bez ďalších informácií server s SSL pripojením a veľkou sieťovou prevádzkou nedokáže rozoznať legitímneho užívateľa od útočníka. Okrem toho nemôže ani poslať webovú požiadavku na overenie legitímnosti zdroja, čo vedie k neustálemu obmedzeniu služby, aj keď je spojenie zabezpečené, tak nedokáže poskytnúť dostatočné pripojenie [11].

1.2.6 Prezenčná vrstva

Prezenčná vrstva, podobne ako relačná vrstva, nemá vlastné PDU. Ako naznačuje jej názov, je zodpovedná za spôsob prezentácie údajov v aplikáciách. Na začiatku komunikácie ustanovuje formu údajov. Následne po týchto ustanoveniach môže poskytovať ďalšie služby, ako je kompresia dát, šifrovanie a preklad. Výber použitých prezenčných služieb závisí od jednotlivých aplikácií [3].

1.2.7 Aplikačná vrstva

Aplikačná vrstva je siedmou vrstvou v referenčnom modele ISO/OSI, ktorá je zodpovedná za definovanie služieb prezentovaných na strane užívateľa. Protokoly aplikačnej vrstvy sa líšia podľa konkrétneho typu údajov, ktoré užívateľ chce preniesť. Aplikačná vrstva tiež definuje prijateľné parametre QoS pre každú službu. Napríklad dáta, ktoré prenášajú hlasovú stopu, vyžadujú iné parametre QoS ako prenos dat Hypertextového prenosového protokolu (angl. hypertext transfer protocol (HTTP)). No jednými z hlavných funkcií aplikačnej vrstvy sú identifikácia služieb poskytovaná užívateľovi a definovanie parametrov QoS požadovaných aplikačnými protokolmi. Definovanie bezpečnostných mechanizmov, ktoré sa majú použiť, a synchronizácia komunikujúcich aplikácií [3].

DoS útoky na aplikačnej vrstve sú o niečo komplikovanejšie. Zvyčajne sa tieto útoky používajú proti finančným inštitúciám, aby odklonili IT a bezpečnostných správcov od narušenia bezpečnosti [12].

Tieto druhy útokov sa spúšťajú všeobecne pre konkrétne cielené účely vrátane prerušenia transakcií a prístupu k databázam a vyžaduje porovnateľne menej zdrojov. Tieto útoky sú jedny z najťažších útokov na mitigáciu, pretože napodobňujú správanie legitímnych užívateľov pri interakcii s používateľským rozhraním [12].

HTTP

HTTP je protokol aplikačnej vrstvy používaný na distribuovanú komunikáciu hypermediálnych informačných systémov, pomocou čoho sa môžu prenášať súbory ako xml a html a mnoho ďalších. V dnešnom svete sa využíva v rámci služby World Wide Web (WWW). Protokol funguje na princípe dotaz-odpoveď. Táto komunikácie prebieha medzi klientom a serverom, kde každá komunikácia musí byť vyvolaná klientom. Pre komunikáciu sa využíva TCP protokol, keďže je potrebné zaručiť spoľahlivý prenos dát [13].

V dnešnej dobe je podľa [14] najviac používaná verzia HTTP 1.1, v ktorej sa začalo využívať prezistentné pripojenie, kvôli čomu servery komunikáciu hneď ne-

uzavrú, ale určitý čas čakajú na prípadné ďalšie príkazy [13]. V rámci takéhoto prezistentného spojenia je často používaná keep-alive požiadavka, ktorá je zneužívaná pomocou útoku LoRDAS popísaného v kapitole 4.

HTTP keep-alive pripojenie

Použitie prezistentného pripojenia v rámci TCP protokolu je výhodné z hľadiska zlepšenia výkonu, keďže nie je potreba otvárať a zatvárať TCP pripojenie pre každý blok dát zvlášť. Ako už bolo povedané, tak prezistentné pripojenie bolo zavedené v rámci HTTP verzie 1.1. Takýto typ pripojenia je označovaný ako keep-alive pripojenie. Keep-alive necháva pripojenie otvorené na určitý počet sekúnd (V Apache 2.4+ to je nastavené na 5 sekúnd) pre účel umožnenia odoslania ďalších požiadaviek na server [15]. Funkcia neurčuje, ako dlho by malo byť pripojenie otvorené, a tieto funkcie necháva na klientov a servery, aby zatvorili spojenie, keď uznajú za vhodné. V Apache HTTP serveri je keep-alive pripojenie kontrolované parametrom KeepAliveTimeout, pomocou ktorého sa nastaví počet sekúnd, ktoré server drží pripojenie otvorené, a druhý parameter MaxKeepAliveRequests definuje, koľko požiadaviek keep-alive môže klient poslať počas jedného otvoreného TCP spojenia [15, 16].

2 DoS a DDoS útoky

Denial of Service, v preklade odopretie služieb, je jeden z najčastejších kybernetických útokov na počítačové siete. V poslednej dobe sa stal jedným z hlavných nebezpečenstiev na Internete [17]. DoS útok nastáva, ak legitímni používatelia nemajú prístup k informačným systémom alebo iným zariadeniam v dôsledku vykonania škodlivej kybernetickej hrozby útočníkom. Ovplyvnené služby môžu zahŕňať emaily, webové stránky, online účty alebo iné služby, ktoré závisia od príslušného počítača, serveru. Jednou z hlavných a základných podmienok odmietnutia služby je zaplavenie zacieleného hostiteľa alebo siete s funkčným prenosom, kvôli ktorému cieľ útoku nemôže reagovať na požiadavky od legitímnych užívateľov. Najčastejšie kvôli zníženiu kvality alebo rýchlosti pripojenia, alebo sa jednoducho zahltí systém, ktorý sa zrúti a následne nastáva odopretie služby. Aj keď v dnešnej dobe existuje už niekoľko stratégií na spustenie DoS, väčšina z nich môže byť klasifikovaná ako útoky, ktoré zneužívajú zraniteľnosť alebo ktoré zaplavuje cieľ. Útoky, ktoré zneužívajú zraniteľnosť, sa pokúšajú poslať správu, ktorá je špeciálne upravená, aby využívala určitú zraniteľnosť, a je schopná vyradiť cieľ útoku z prevádzky. Záplavové útoky sa na druhej strane snažia posielať veľký počet správ (paketov), ktoré vyčerpajú niekoľko zdrojov sieťového zariadenia, ako napríklad šírku pásma, procesor alebo pamäť. Pri tomto type útoku môžu byť správy identické ako správy, ktoré sa legitímne používajú pri bežnej komunikácii so sieťovou službou. Nutné je však podotknúť, že môžu existovať služby používajúce obidve stratégie útoku. Záplavové útoky sa často používajú s prostriedkami, ktoré získajú kontrolu nad sieťovými zariadeniami ostatných používateľov, ktorí sú zneužití na zaplavovanie cieľu útoku [18, 19, 20].

Takýto typ útoku sa nazýva DDoS. Zariadenia, ktoré sú začlenené do DDoS útoku, sú najčastejšie infikované malwarom, ktorý povolí útočníkovi na diaľku kontrolovať ich zariadenie. Toto zariadenie sa nazýva bot alebo zombie a skupina takýchto botov sa nazýva botnet [21].

Avšak niektoré DoS útoky, ktoré sa objavili v nedávnej dobe, sa uskutočňujú pomocou nízkeho množstva dát (v radoch KB) na cielené zariadenia, kde napriek tomu dosiahneme odopretie služby. Tento typ útoku sa nazýva Low-rate DoS útok, ktorý bude bližšie preberaný v podkapitole 2.2. Jeden z prvých útokov, ktorý používa tento prístup, sa nazýva Shrew DoS, ktorého nový a vylepšený typ NewShrew bude jednou z hlavných tém tejto práce [18, 21]. V tejto kapitole okrem toho popíšeme funkčnosť low-rate útokov a vymedzíme rozdiel medzi záplavovými, pomalými a low-rate útokmi.

2.1 Záplavové útoky

Základnými a prvotnými typmi DoS útokov sú práve záplavové útoky. Vďaka ich jednoduchosti ich používanosť nabrala na veľkom počte útokov v priebehu celej existencie DoS útokov.

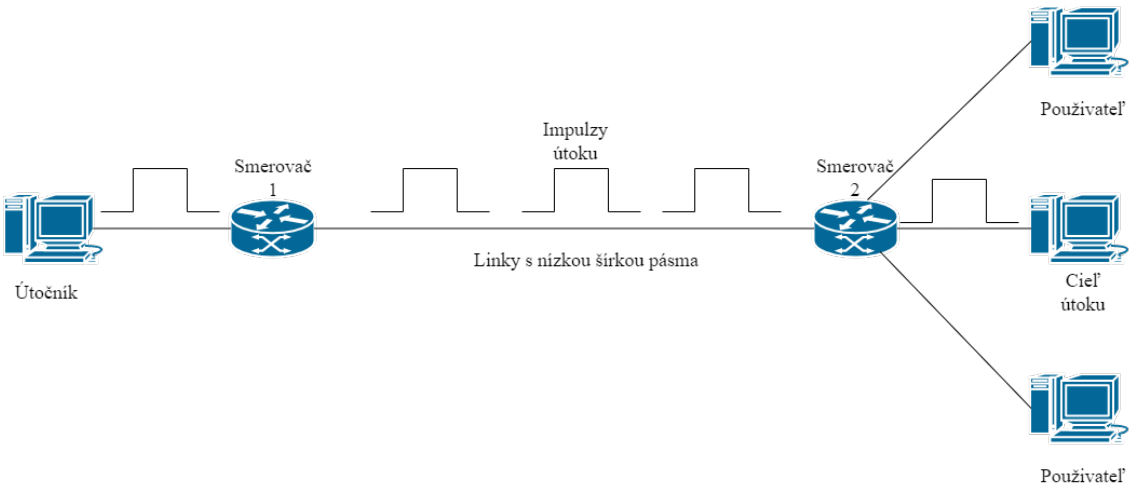
No kvôli ich jednoduchej realizácii a implementácii ich bolo jednoduché aj detegovať a odvrátiť, na rozdiel od Low-rate útokov. Záplavový DoS útok namierený proti jednému alebo viacerým sieťovým zdrojom zaplaví cieľ ohromným počtom dát. V závislosti od zámeru útočníkov a rozsahu a úspechu predchádzajúcich snáh o zhromažďovanie informácií si útočníci môžu vybrať konkrétneho hostiteľa, ako je napríklad server, alebo sa môžu zamerať na náhodných hostiteľov v cieľovej sieti [22].

Každý z týchto prístupov má potenciál znepríjemniť službu jednému hostiteľovi alebo celej sieti v závislosti od toho, aká kritická je úloha obete pre zvyšok siete. Keďže ich hlavným zámerom je zaplaviť zneužitý server čo najväčším počtom paketov, tak ich detekcia spočíva v jednoduchej detekcii nadmerného posielaných dát z jedného počítača útočníka alebo celej botnet siete [23].

2.2 Low-rate útoky

V priebehu vývoja technológií DoS útoky prešli veľkým vývojom, kde sa od jednoduchých záplavových útokov vyvinuli do útokov, ktoré sofistikovane využívajú zraniteľnosti v sieťach. Napriek nízkej priemernej rýchlosti útoku, od niekoľko KB až po jednotky MB, dokážu tieto útoky obmedziť prístup legitímnym používateľom. Tieto útoky možno klasifikovať ako logické a v základom vnímaní a rozdelení by sme ich vedeli rozdeliť na Slow DoS a Low-rate DoS. Avšak práve tieto typy sa v dnešnej dobe DoS útokov zvyknú stále zamieňať. Napriek tomu, že sa táto práca venuje najmä Low-rate útokom, tak je potrebné v tejto kapitole spomenúť aj Slow DoS a v krátkosti opísať rozdiely medzi Low-rate DoS a Slow DoS [24].

U Slow DoS je ich hlavnou charakteristikou veľmi pomalý tok dát. Kvôli neošetreným zraniteľnostiam systémov alebo abnormálnemu používaniu služby tieto útoky môžu preťažiť cieľový server a spôsobiť odmietnutie služby ostatným používateľom iba niekoľkými paketmi. Práve túto vlastnosť majú slow DoS a Low-rate útoky spoločnú. Malý objem dát. Vďaka nízkym dátovým tokom a zvyčajne legitímnemu používaniu všetkých protokolov sú tieto útoky veľmi podobné bežným používateľom s veľmi pomalým internetovým pripojením [25].



Obr. 2.1: Vizualizácia TCP spojenia Low-rate útoku, cez linku s nízkou šírkou pásma.

Low-rate DoS útoky na druhej strane disponujú aj inými vlastnosťami. Ako bolo už spomínané, tak Low-rate DoS útoky podobne ako slow DoS generujú nízku sieťovú prevádzku, pomocou ktorej sa snažia znefunkčniť server. Vo väčšine prípadov je to 10 % až 20 % bežnej sieťovej prevádzky [26]. Aj keď je prevádzka low-rate DoS nízka, tak dokáže znefunkčniť základné sieťové zariadenia a spôsobiť odopretie služby cieľu útoku. Základnou charakteristikou Low-rate DoS útokov je práve tvar periodických impulzov, v ktorých posielajú dáta. Mnohé jednotlivé útočné impulzy Low-rate DoS z rôznych zdrojov útokov s nízkym priemerným tokom dát sú posielané v synchronizovanej podobe, aby vytvorili sled impulzov za určité obdobie. To znamená, že útočník sa snaží obmedziť sieťovú prevádzku pomocou striedania period odosielania veľkého množstva dát a period bez odosielania dát, počas ktorého je napadnuté sieťové zariadenie zneprístupnené v dôsledku zneužitia zraniteľnosti. Tieto impulzy dát cez sieťovú prevádzku vytvárajú v sieti miesto s nízkou prenosovou rýchlosťou (angl. bottleneck link), čo spôsobuje zablokovanie danej linky medzi serverom a sieťovým zariadením, ako napríklad smerovač, alebo vedie k poklesu kvality služieb. Aj keď je priemerná prevádzka útokov Low-rate DoS nízka, jej útočná sila nie je oveľa menšia ako napríklad v prípade záplavových DoS útokov [26].

Ciele Low-rate DoS útokov je možné kategorizovať do dvoch skupín. Jedným z nich je miesto s nízkou prenosovou rýchlosťou a druhou kategóriou sú aplikačné systémy alebo servery. Low-rate DoS často zneužívajú zraniteľnosť protokolov TCP/IP, čo výrazne znižuje priepustnosť linky a môže výrazne znížiť kvalitu napadnutej služby. Dôvod, prečo je prevádzka útoku Low-rate DoS nízka, možno odvodiť z obr. 2.1. Prevádzka útoku Low-rate DoS sa sústreďuje do pravouhlého impulzu, v ktorom sa

v krátkom časovom úseku odošle veľké množstvo paketov a tento proces odosielania sa opakuje pri špecifikovanej frekvencii, t.j. perióde impulzov [27].

Preto sila útoku závisí od amplitúdy a trvania pravouhlého impulzu, to znamená, že prevádzka útoku Low-rate DoS je spriemerovaná v priebehu času v obdĺžnikovom impulze. Preto je priemerná návštevnosť útokov Low-rate DoS malá. Efekt Low-rate DoS útoku sa prejavuje v zníženej kvalite služby napadnutého cieľa. Preto sa nazýva aj útok degradácie kvality. Z toho teda vyplýva, že formou útoku Low-rate DoS je séria periodických impulzov väčšieho množstva paketov v sieti cieľu útoku najčastejšie obsahujúce pásmo s nízkou priepustnosťou [27].

Medzi tieto útoky sa radia NewShrew a LoRDAS. Zatiaľ čo NewShrew útok sa periodickým posielaním dát snaží odstaviť akýkoľvek protokol pracujúci pomocou TCP, kde musia byť dodržané určité podmienky siete pre funkčnosť, na druhej strane LoRDAS útok sa snaží zaplniť frontu aplikačných serverov prezistentnými pripojeniami, ako napríklad keep-alive požiadavky spomenuté v 1.2.7 [28, 29]. Detailnejšie sa však týmto útokom budeme venovať v samostatných kapitolách 3 a 4. V nasledujúcej podkapitole popíšeme útok Shrew, na základe ktorého funguje útok NewShrew s tým, že rieši jeho nedostatky.

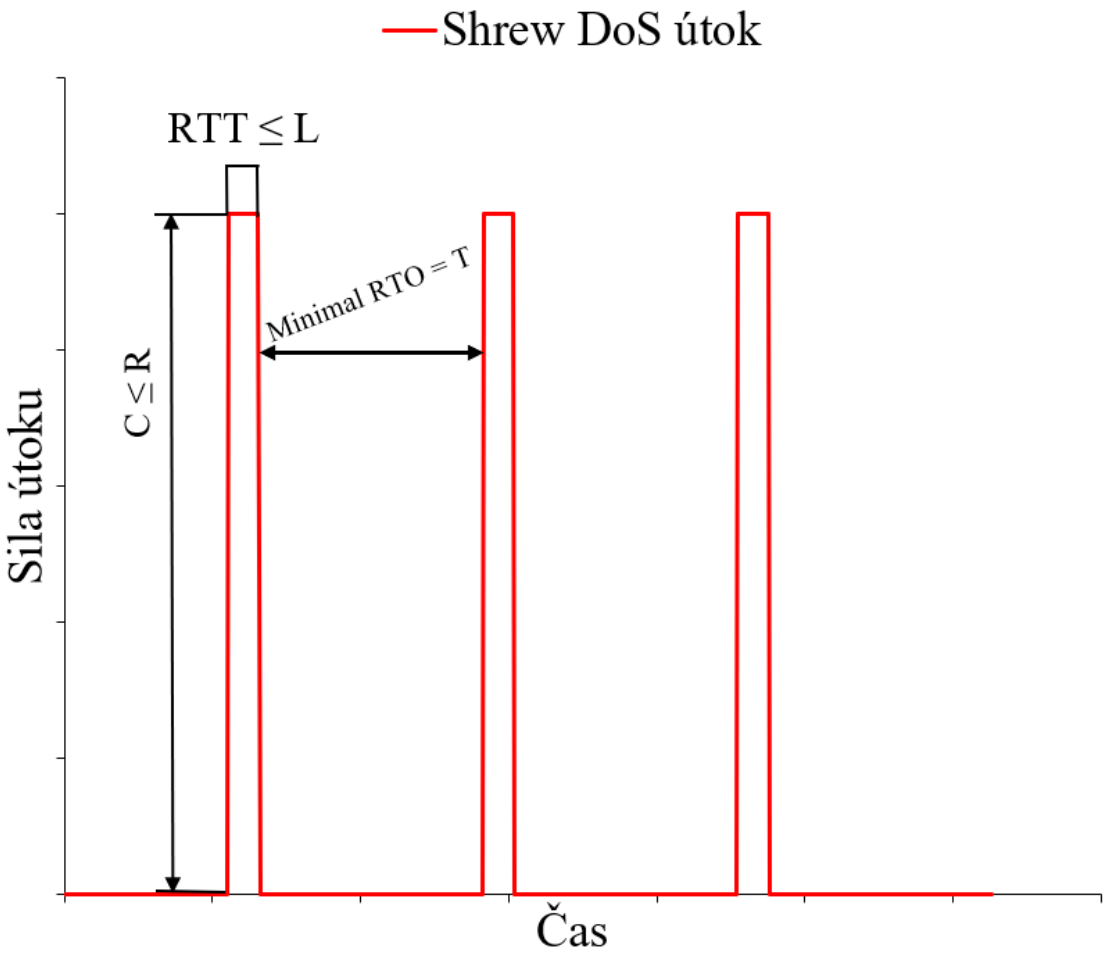
2.2.1 Shrew DoS útok

Jedným z popredných útokov, ktoré majú priebeh Low-rate DoS, je Shrew DoS, ktorý zneužíva Retransmission timeout (RTO). Tento útok využíva homogenitu mechanizmu TCP RTO, kde sa RTO zvyčajne rovná jeho dolnej hranici, a to hodnote minimálneho RTO (minRTO), čo je 1 sekunda [30].

Ak zvážime jeden tok dát TCP a útok Shrew pozostávajúci z periodických impulzov veľkých dát, a ak prenášané dáta majú dostatočné množstvo paketov, aby rýchlo zaplnili buffer smerovača na linke s malou priepustnosťou na dostatočne dlhý čas, čo spôsobí zapnutie RTO na vyvolanie strát paketov. Počas interburst periódy (T) sa neposielajú žiadne dáta a čaká sa na uplynutie minRTO, preto veľkosť interburst periódy je rovná minRTO. Na druhej strane, počas burst periódy (L) sa posiela, čo najväčšie množstvo dát. Veľkosť burst periódy by sa mala odvíjať od veľkosti priemerného RTT v sieti a je buď väčšia, alebo menšia ako round-trip time (RTT). Keď takéto množstvo paketov dorazí na úzke miesto, odosielateľ TCP zastaví prenos paketov a vstúpi do stavu RTO. Po časovom úseku minRTO, zatiaľ čo sa odosielateľ iba pokúša o opätovné odoslanie stratených paketov, príde ďalšia burst perióda, takže odosielateľ bude musieť znova prejsť do RTO. Opakovaním vyššie uvedeného procesu môže správne nakonfigurovaný útok Shrew výrazne obmedziť priepustnosť TCP s nízkou priemernou rýchlosťou útoku [30, 31].

Okrem toho je možné nastaviť útočníkom aj veľkosť sily útoku počas burst periódy. Tento parameter sa nazýva burst magnitúda (R) a mal by byť rovný alebo väčší, ako priepustnosť linky s nízkou priepustnosťou označenou ako C (bottleneck linka). Priebeh Shrew DoS útoku s jednotlivými parametrami je možné vidieť na obrázku 2.2.

Shrew DoS útok je predchodcom NewShrew útoku, ktorého funkčnosť vychádza z rovnakých predpokladov ako zo Shrew útoku, a preto bude vysoká podobnosť z hľadiska nastavovania parametrov útoku.



Obr. 2.2: Priebeh Shrew DoS útoku.

3 NewShrew útok

NewShrew útok vychádza z klasického Shrew útoku, ktorý bol popísaný v kapitole 2.2.1 a rieši niekoľko jeho zraniteľností a nedostatkov. Podobne ako Shrew DoS, tento útok tiež pozostáva z periodického striedania burst periód a interburst periódy a využíva zraniteľnosť RTO. NewShrew teda zahŕňa Shrew ako jednu zo svojich podmnožín a jeho účinnosť závisí okrem RTO aj od zraniteľnosti v mechanizme pomalého štartu (angl Slow start mechanism) protokolu TCP [32]. NewShrew útok úmyselne nechá uplynúť RTO, aby prešiel do fázy pomalého štartu po každej burst perióde. Kvôli mechanizmu pomalého štartu bude exponenciálne rásť rýchlosť odosielania s paradoxne malým počtom úspešne prenesených paketov. Keď potom zneužitý server obnoví rýchlosť odosielania na určitú hodnotu, útočník spustí ďalšiu burst periódu s veľkým množstvom paketov. Pri sieťovej prevádzke nebude útočník potrebovať vysokú veľkosť impulzu burst periódy, aby zaplnil linku s nízkou priepustnosťou smerovača a obnovil časový limit zneužitého serveru. Server medzitým nadobudne slabú priepustnosť, pretože počas fázy pomalého štartu sotva prenesie niekoľko paketov. Využitie nedostatku pomalého štartu prináša NewShrew útoku tri hlavné výhody oproti Shrew. Prvými dvomi sú nižšia priemerná rýchlosť útoku a vyššia účinnosť útoku. Takáto nízka priemerná rýchlosť a vysoká účinnosť znamená, že NewShrew môže drasticky znížiť priepustnosť TCP. Treťou výhodou je vyššia odolnosť proti DoS obranným mechanizmom. To znamená, že NewShrew nemusí byť zdetegovaný prevenčnými a obrannými systémami, napríklad aj tými, ktoré boli navrhnuté pre Shrew DoS. Vyladením jeho parametrov útoku (dĺžka burst periódy, veľkosť impulzu burst periódy a interburst periódy) dokáže NewShrew útok dosiahnuť v sieti s randomizáciou RTO a detekčným systémom Shrew DoS stále vyššiu efektivitu a v priemere až o 11,54 % vyššiu degradáciu výkonu siete [33]. V tejto kapitole bude bližšie popísaný NewShrew útok a najmä ladenie jeho parametrov, ktoré bude dôležité pre správnu funkčnosť. Pre lepšiu predstavu budú popísané RTO aj mechanizmus pomalého štartu [32].

3.1 Roud-trip time

Round-trip time (RTT) je čas v milisekundách, ktorý je potrebný na to, aby sa sieťová požiadavka dostala od zdroja do cieľového zariadenia a späť ku zdroju. RTT je dôležitou metrikou pri určovaní stavu pripojenia v miestnej sieti alebo vo väčších internetových sieťach. Správcovia siete ho bežne využívajú na diagnostiku rýchlosti a spoľahlivosti sieťových pripojení [34] .

Zlepšenie latencie sa môže merať skrátením času cesty požiadavky od zdroja k cieľu a naspäť a odstránením prípadov, keď sú využívané inými protokolmi, ako

napríklad úpravou štandardného handshake TLS/SSL [34].

Okrem toho je RTT využívaný v prednastavení času RTO, kde RTO je náhodne generované aj na základe RTT času, čo bude využité najmä pri zostrojovaní obranných mechanizmov proti NewShrew útoku.

3.2 Rentransmission timeout

RTO môže nastať v niekoľkých prípadoch. Jedným z nich, ktorý sa bude využívať pri NewShrew útoku, je, keď odosielateľovi chýba priveľa potvrdení, teda ACK segmentov, a rozhodne sa, že si vezme čas na ich prijatie a úplne zastaví odosielanie. Po určitom čase, zvyčajne aspoň po jednej sekunde, odosielateľ opatrne začne znova odosielať dáta, pričom najprv otestuje sieť s jedným paketom, potom s dvoma paketmi atď. V dôsledku toho RTO spôsobí minimálne jednosekundové oneskorenie v sieti. Postupným opakovaním tohto deja vieme dosiahnuť RTO, ktoré trvá aj tisíc sekúnd. Tieto časové limity opakovaného prenosu spôsobujú značné problémy s výkonom siete a aplikácií a vyžadujú určité ladenie a optimalizáciu. Tento problém však NewShrew útok dokáže zneužiť a ako bolo spomínané, dokáže bez povšimnutia obmedziť sieťovú prevádzku na niekoľko hodín aj dní [35].

V dnešnej dobe sa však RTO minimalizuje a bežne je v kerneli operačných systémov nastavené na minimálnu hodnotu 200 ms. V určitých aplikáciách ako IP telefónia môžu administrátori minimálnu hodnotu nastaviť ešte nižšie, čo výrazne môže zmeniť priebeh útoku [35]. Vo výpise 3.1 je definované minimálne a maximálne RTO hodnotou HZ, ktorá je rovná 1000 ms, takže hodnota RTO sa pohybuje medzi 200 ms až 120 sekúnd.

```
1 #define TCP_RTO_MAX ((unsigned)(120*HZ))
2 #define TCP_RTO_MIN ((unsigned)(HZ/5))
```

Výpis 3.1: Retransmission timeout definovaný v Linux 2.6+.

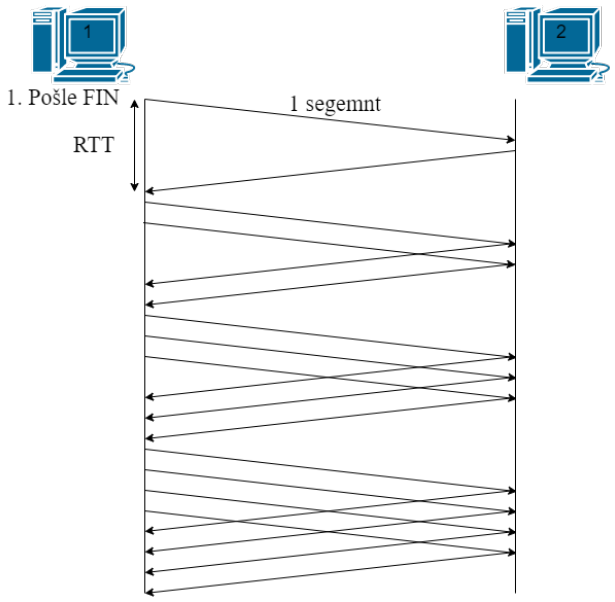
Okrem toho sa hodnota RTO odvíja od počtu opakovaného odosielania nepotvrdených paktov až do maximálnej hodnoty opakovania `tcp_retries`, ktorá je definovaná v sysctl, kde je rovná pätnástim opakovaniam. Časový limit RTO sa postupne exponenciálne navyšuje až do hodnoty `TCP_RTO_MAX`. Po uplynutí pätnástich pokusov o obnovenie spojenia zásobník TCP upozorní vyššie vrstvy o prerušení spojenia a aplikácia pripojenie reštartuje a v prípade trvania zahadzovania paketov začne retransmission timeout plynúť zase od minimálnej hodnoty RTO. V tabuľke 3.1 môžeme vidieť, ako sa navyšuje RTO. U pätnásteho opakovania trvá 804,6 sekúnd, kým protokol TCP oznámi aplikácii vyššej vrstvy prerušenie spojenia [35].

tcp_retries	RTO [ms]	Doba prerušenia spojenia [s]
1	200	0.2
2	400	0.6
3	800	1.4
4	1600	3.0
5	3200	6.2
6	6400	12.6
7	12800	25.4
8	25600	51.0
9	51200	102.2
10	102400	204.6
11	120000	324.6
12	120000	444.6
13	120000	564.6
14	120000	684.6
15	120000	804.6

Tab. 3.1: Navyšovanie RTO.

3.3 Slow Start mechanizmus

Jedným z najbežnejších spôsobov optimalizácie rýchlosti pripojenia je zvýšenie rýchlosti pripojenia (t.j. zvýšenie šírky pásma). Ak sa zariadenie pokúsi odoslať príliš veľa údajov, tak môže preťažiť sieť. Presýtenie spojenia je známe ako preťaženie a môže viesť k pomalej komunikácii alebo dokonca strate údajov. Priebeh pomalého mechanizmu je vidieť na obrázku 3.1. Pomalý štart zabraňuje preťaženiu siete reguláciou množstva dát, ktoré sa cez ňu posielajú. Nadväzuje spojenie medzi odosielateľom a prijímačom definovaním množstva dát, ktoré je možné preniesť s každým paketom, a pomaly zvyšuje množstvo dát, kým sa nedosiahne kapacita siete. To zaisťuje prenos čo najväčšieho množstva dát bez zanášania siete. Tento mechanizmus však NewShrew dokáže zneužívať a ako už nám je známe, tak do tohto stavu sa cieľ útoku dostane práve po vypršaní RTO, kde NewShrew zase spustí periódu posielanie veľkého množstva paketov [36]. Tým spôsobí ďalšiu stratu paketov a prechodu serveru do ďalšej fázy RTO a pomalého mechanizmu. Čo nám dovoľuje predĺžiť dobu interburst periódy.



Obr. 3.1: Slow start mechanizmus.

3.4 Parametre NewShrew útoku

NewShrew útok pozostáva z periodických štvorcových vĺn v čase určenom tromi parametrami, t.j. interburst periódou (T), počas ktorej sa neposielajú dáta, burst periódy (L), počas ktorej sa posiela väčšie množstvo dát a veľkosť dát posielaných počas burst periódy, burst magnitúda, (R). Pre správnu funkčnosť NewShrew útoku je treba zaistiť niekoľko parametrov siete, a to že linka s nízkou priepustnosťou je first in first out (FIFO), čo znamená, že každý prichádzajúci paket bude zahodený, ak je vyrovnávacia pamäť plná. Ďalej, aby sa eliminoval vplyv aplikácie na prenos paketov, predpokladáme, že TCP odosielateľ má vždy údaje z aplikačnej vrstvy na odoslanie. Posledný predpoklad je, že spätný chod linky nie je nikdy preťažený. Ďalej bude ovplyvňovať silu a efektivitu útoku rôzne nastavenie kombinácií veľkosti burst periódy, interbust periódy a burst magnitúdy [37].

Už vieme, že komunikácia TCP po uplynutí RTO prejde do fázy pomalého štartu a exponenciálne obnovuje svoju prenosovú rýchlosť. Počas tejto fázy má sieťová prevádzka zaujímavú vlastnosť. Jej prenosová rýchlosť sa rýchlo zvyšuje, ale celkový počet odosielaných paketov je nízky. Táto funkcia poskytuje príležitosť NewShrew útoku. Keď útočník odošle veľký počet paketov za malú periódu času, tak dokáže zahltiť úzku linku vedúcu k serveru. To spôsobuje hromadné straty paketov a TCP tok dát serveru musí prejsť do RTO a zastaviť prenos paketov. Tento stav bude trvať po dobu času RTO, kde v tomto prípade budeme pre ideálne podmienky útoku uvažovať o minimálnom RTO 1 sekunda. Keďže v dnešnej dobe sa na väčšine sie-

ťových zariadení používa náhodne vygenerované RTO, tak bude priebeh útoku vyzerať inak. Následne tok dát serveru vstupuje do fázy pomalého štartu a začína sa obnovovať rýchlosť prenosu [28]. Kvôli funkcii, ktorú sme spomenuli vyššie, sa po niekoľkých úspešne prenesených paketov zvýši rýchlosť odosielania cieľu útoku na hodnotu, ktorá je dostatočná pre útočníka na základe nastavenia parametrov interburst periódy, burst periódy a magnitúdy útoku. V tomto momente útočník spustí ďalšiu periódu s veľkým objemom dát. Pomocou zneužitého serveru útočník môže ľahko zahltiť linky s nízkou rýchlosťou a zase docieli stav, kedy cieľ útoku prejde do RTO. Medzitým cieľ útoku nadobudne nízku priepustnosť, pretože počas pomalého prenosu s ťažkosťou prenáša dáta. A tento proces sa opakuje každú periódu [35].

Homogenita TCP timeout zaručuje, že takáto stratégia útoku by mohla byť efektívna aj pre heterogénne-RTT toky dát. Dôležitým bodom útoku je, že ak NewShrew útok má v úmysle úplne obmedziť sieťovú prevádzku servera, tak nastavením jeho interburst perióda na minRTO sa môže NewShrew degradovať na Shrew DoS a jeho priemerná sila útoku sa zvýši. Preto je dôležité u NewShrew útoku nejakú chvíľu počkať a nespúšťať burst periódu hneď po uplynutí RTO [28].

3.4.1 Nastavenie parametrov NewShrew útoku

Ako sme už v kapitole 3.4 spomínali, tak tri základné parametre NewShrew útoku na seba vzájomne pôsobia, preto je dôležité tieto parametre dostatočne vyvážiť, aby NewShrew útok dokázal odoprieť službu cielenému serveru. Vyladiť tieto parametre útoku je náročný problém. Za kritické sa považujú iba dva čiastkové problémy ladenia NewShrew útoku, a to, ako sa parametre navzájom ovplyvňujú a ako urobiť kompromis medzi znížením priepustnosti a nákladmi na útok [28].

Prvá časť nastavenia sa venuje interakcii medzi trojicou parametrov útoku. V prvom rade môžeme ľahko odvodiť rozsah parametrov útoku, kde interburst perióda by mala byť väčšia ako minimálne RTO, burst perióda, ktorá by mala byť v rozmedzí RTT a burst magnitúda, ktorá by mala byť väčšia alebo rovná šírke pásma linky úzkeho miesta (C) [28].

Po naplnení vyrovnávacej pamäte sieťového prvku úzkeho miesta by mal NewShrew DoS ďalej trvať určitý čas (označený L_{t0}), tak môže vyvolať dostatočný počet zahodených paketov, aby cieľ útoku TCP komunikácie prešiel do RTO fázy. Perióda medzi začiatkom burst periódy a pretečením vyrovnávacej pamäte linky s nízkou prenosovou rýchlosťou sa rovná ($L - L_{to}$). Obsadená vyrovnávacia pamäť pri spustení burst periódy je označená ako Q. Ako priemer prenosovej rýchlosti (V) počas časového obdobia od začiatku burst periódy do pretečenia vyrovnávacej pamäte smerovača popisuje rovnica 3.1:

$$(L - L_{to}) \cdot (V + R - C) = B \cdot Q \tag{3.1}$$

Kde B je buffer okrajového smerovača, v našom prípade filtračného serveru, a ďalej platí, že $Q \geq B$, $C \geq V$ a $L \geq L_{to}$ [28].

$$R = \frac{B - Q}{L - L_{to}} + C - V \tag{3.2}$$

V tomto prípade Q, V a L_{to} nemajú vplyv na správne nastavenie burst periódy [28].Ak sa perióda T rovná perióde T_0, tak môžeme derivovať tento vzťah pomocou $\frac{\mathrm{d}R}{\mathrm{d}L}$ a dostaneme rovnicu:

$$\frac{\mathrm{d}R}{\mathrm{d}L} = -\frac{B - Q}{(L - L_{t0})^2} \leq 0, \text{ pre } T = T_0 \geq RTO \tag{3.3}$$

Na odvodenie interakcie medzi R a T je predpokladaný spôsob útoku, ktorý má pevnú dĺžku brust periódy t.j. $L = L_0$. Zvýšením veľkosti magnitúdy burstu R môže útočník zaplniť smerovač alebo filtračný server s nízkou priepustnosťou linky vedúcej k napadnutému serveru [28]. V súlade s tým by mal znížiť svoju periódu medzi veľkými zhlukmi dát T, aby zabránil nadmernému nárastu prenosovej rýchlosti cielenému serveru. Odvodením vzťahov získame:

$$\frac{\mathrm{d}T}{\mathrm{d}R} \leq 0, \text{ pre } L = L_0 \geq RTT \tag{3.4}$$

kde $\frac{\mathrm{d}T}{\mathrm{d}R} = 0$ a $T = minRTO$. Tento prípad, kde $\frac{\mathrm{d}T}{\mathrm{d}R} = 0$, môžeme považovať za nastavenie ako u Shrew útoku a nesprávne nastavenie NewShrew parametrov [28]. Pomocou rovnice 3.3 a 3.4 derivujeme vzťah medzi T a L, ak vieme, že $R = R_0$, a dostaneme rovnicu:

$$\frac{\mathrm{d}T}{\mathrm{d}L} \leq 0, \text{ pre } R = R_0 \geq C \tag{3.5}$$

Ďalšie vzťahy zobrazujú, aký ma vplyv zmena parametrov NewShrew útoku, a ako nastaviť správne tieto parametre pre získanie kompromisu medzi efektivitou a účinnosťou. Z rovnice 3.4 vieme, že priemerná rýchlosť útoku $A = R \cdot \frac{L}{T}$ rastie spolu s R, kde L je dané, a vyplýva z toho:

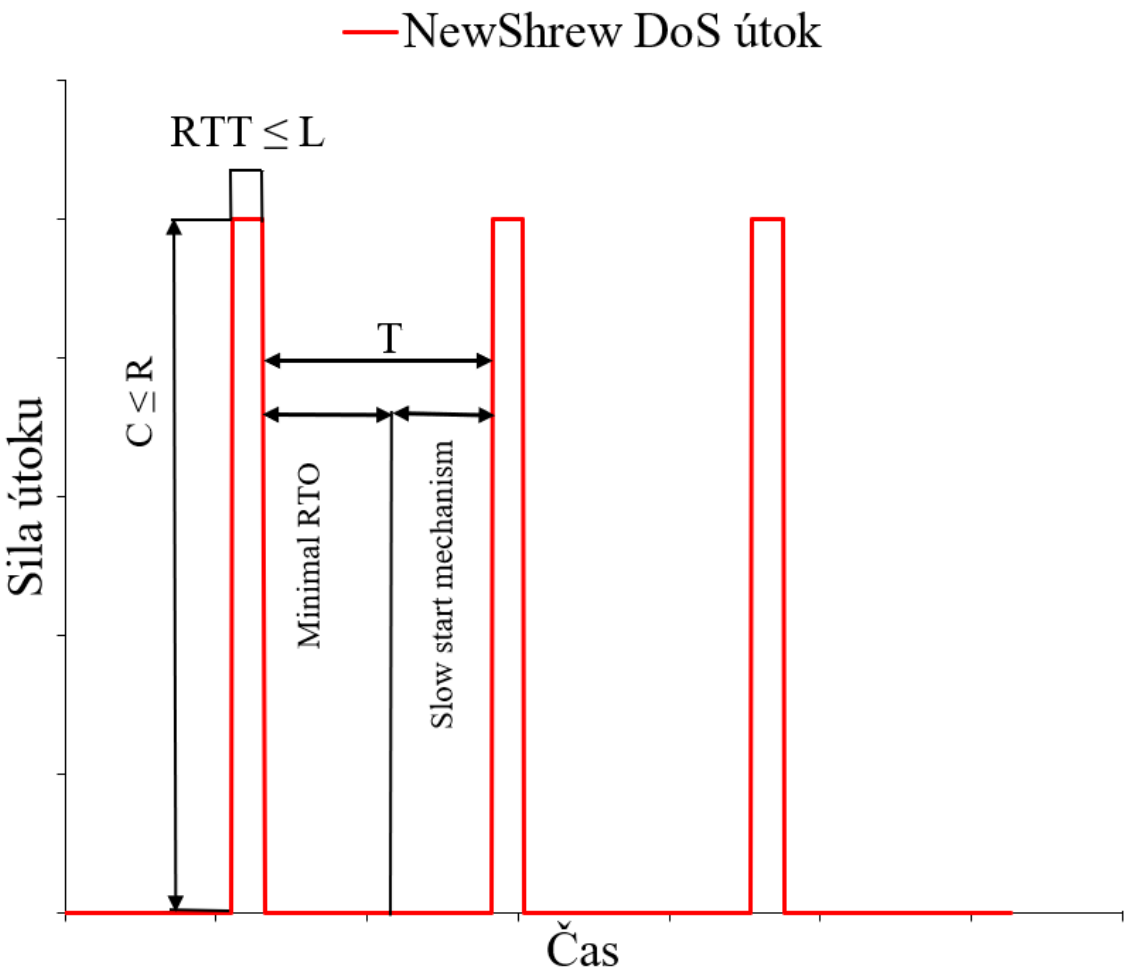
$$\frac{\mathrm{d}A}{\mathrm{d}R} > 0, \text{ pre } L = L_0 \geq RTT \tag{3.6}$$

Naopak, priepustnosť k cielenému serveru, ktorú označujeme ako P, klesá s klesajúcim T, pretože menšie T znamená menej času pre server na obnovenie jeho prenosovej rýchlosti. V kombinácii s 3.4 dostávame rovnicu:

$$\frac{\mathrm{d}P}{\mathrm{d}R} \leq 0, \text{ pre } L = L_0 \geq RTT \tag{3.7}$$

Z rovníc 3.6 a 3.7 je vidieť kompromis medzi zhoršením priepustnosti a nákladmi na útok. Zvýšením jeho veľkosti R môže NewShrew dosiahnuť vyššiu degradáciu priepustnosti za cenu vyšších nákladov na útok. Ak je dané dostatočne veľké R, útok môže dokonca znížiť priepustnosť na nulu. Naopak, útok môže znížiť R na dosiahnutie nižších nákladov na útok za cenu nižšej degradácie priepustnosti [28]. Samozrejme, útočník môže urobiť kompromis medzi zhoršením priepustnosti a inými parametrami, ako napríklad priemernou silou útoku.

Priebeh útoku pomocou nastavených parametrov je možné vidieť na obrázku 3.2, kde je v grafe zakomponovaná dĺžka minRTO, pomalého mechanizmu a burst magnitúda rovná alebo väčšia ako priepustnosť linky s nízkou priepustnosťou.



Obr. 3.2: Priebeh NewShrew DoS útoku.

3.5 Implementácia NewShrew

Generátor NewShrew útoku je implementovaný ako Python3 skript s použitou verziou 3.8 a vyššie. Na správne fungovanie skriptu nie je potrebné nainštalovať žiadne dodatočné moduly Pythonu, keďže skript pracuje so štandardnými modulmi[1]. Program ráta s rôznymi predpokladmi na správnu funkčnosť generátoru a počas vyvíjania generátoru sa prihliadalo na rôzne spôsoby jeho implementácie. Na posielanie periodických burst períód sa používa knižnica Socket, pomocou ktorej sa posielajú pakety na cielený server. V prvom spôsobe sa generátor snažil posielať TCP pakety, to však nezaručovalo jednu z podmienok NewShrew útoku, a to, že spätný chod linky útoku nebude nikdy obmedzený, a preto ako efektívnejšie riešenie posielania burst períód sa použilo posielanie UDP socketov, ktorého tok dát sa síce neriadi, ale zaručuje, že sa pošle dostatočný počet paketov smerom k serveru a dokáže zahltiť úzke miesto v sieti a nezahltí to spätný tok dát smerom k útočníkovi. Zadefinovanie UDP socketu môžeme vidieť vo výpise 3.2. Následne sa pomocou funkcie sendto(), knižnice socket, posielajú UDP segmenty v cykle a na základe časovania definovaného pomocou vstupných argumentov skriptu. Funkcia posielania burst períód je definovaná v súbore `attack.py`.

```
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    logger.connection_establish_log(self.new_shrew_attack.
    ip_address, self.new_shrew_attack.log)
except socket.error:
    logger.failed_soc_create(self.get_attacker_ip(), self.
    new_shrew_attack.ip_address, self.new_shrew_attack.log)
    sys.exit(0)
```

Výpis 3.2: Implementácia generovania UDP socketu.

Spomenuté parametre útoku sa pomocou knižnice argparse odovzdávajú skriptu prostredníctvom argumentov, ktoré užívateľ definuje v termináli spolu so spustením skriptu `NewShrew.py`. Následne sa tieto parametre odovzdajú Python súboru `attack.py`. Pomocou ktorého sa zadefinuje objekt NewShrew, ktorý je potom odovzdávaný ostatným funkciám a má zadefinované všetky parametre. Parametre, pomocou ktorých dokážeme nastaviť NewShrew útok, môžeme vidieť v návode na použitie v prílohe A.

Ako už je vyššie spomínané, tak útok posiela UDP sockety v burst perióde, ktorá je definovaná pomocou premennej burst_length, kde sa v cykle posielajú datagramy. Rozdiel začiatočného času burst periódy a momentálneho času definovaný premennou burst_detla sa porovnáva s burst_length, v prípade, že sa rovná alebo

[1]Zoznam so štandardnými knižnicami je dostupný z `https://docs.python.org/3.8/py-modindex.html`

je väčší, tak sa ukončí cyklus a prejde sa do fázy interburst periódy a NewShrew útok čaká a neposiela dáta až do uplynutia interburst periódy. Tieto periódy sa striedajú a spúšťajú v cykle, v ktorom sa rozdiel momentálneho času a celkového začiatku útoku porovnáva s celkovou dĺžkou útoku, kde po uplynutí celkového času sa ukončí celý útok. V rámci posielania paketov sa taktiež riadi magnitúda, ktorou sa nastavuje množstvo posielajúcich dát.

Čo sa týka nastavenia parametrov, tak má používateľ možnosť nastavovania parametrov v rôznych rozsahoch. Preto je možné používateľom nastaviť parametre, tak aby NewShrew útok degradoval na Shrew útok. Ako bolo už vyššie spomínané, je dôležité, aby útočník nastavil interburst periódu väčšiu ako je minimálna veľkosť RTO, aby komunikácia dokázala prejsť aj do mechanizmu pomalého štartu, a tak zneužila túto zraniteľnosť TCP. Burst periódu je zasa potrebné nastaviť v rozsahu veľkosti RTT a burst magnitúdu, ktorej sila útoku je väčšia ako priepustnosť úzkej linky.

V rámci útoku sme uvažovali aj o použití paralelného programovania, ktoré však nemôže byť použité, keďže útok je zostavený z presného časovania periód v radoch milisekúnd, kde by jednotlivé vlákna neboli dostatočne časovo zosynchronizované. Okrem toho v podstate nie je potrebné dosahovať vysokú silu útoku, keďže sa v rámci burst periódy posielajú dáta v rozsahu 10 KB až 5 MB. Preto pre NewShrew útok nie je potrebné použitie paralelizmu.

Okrem argumentov, ktoré ovplyvňujú samotný beh programu, je v skripte možnosť nastaviť aj argument pre potrebu logovania. Tieto argumenty však nijako neovplyvňujú priebeh útoku, preto ich nebudeme v tejto kapitole nejako významne popisovať. Vytvorili sme Python súbor `logger.py`, ktorej funkcie dodávajú logovacie správy do CLI za behu programu a zároveň ich odosiela do súboru, ktorý je zadefinovaný defaultnou cestou alebo je možnosť ho meniť cez argument $-log$ zadávaný pomocou scriptu spusteného v terminále. Súbor `pars_checkers.py` sa používa na funkcie, ktoré kontrolujú tvar a formát vstupu, ktoré užívateľ zadáva.

3.5.1 Generovanie NewShrew útoku

NewShrew útok dokážeme generovať pomocou spustenia Python skriptu, ktorý je vo výpise 3.3.Tento skript však vyžaduje zadefinovanie niekoľkých základých parametrov, ktoré sú vyžadované ako povinné parametre útoku. Ide najmä o IP adresu, ktorú je potrebné nastaviť v každom prípade. Druhým parametrom je dĺžka útoku, po dobu ktorej chceme, aby NewShrew pôsobil na cieľ útoku. Na tri rozhodujúce parametre NewShrew útoku však bola potreba ich nastavenie cez CLI ako vyžadované, keďže správna konfigurácia týchto troch parametrov a to burst magnitúda, burst perióda a interburst perióda, rozhoduje o správnej funkčnosti NewShrew útoku

a zneprístupnení služieb.

```
python3 NewShrew.py
```

Výpis 3.3: Spustiteľný Python súbor pre NewShrew útok.

S RTO = $1s$, a bottleneck linkou rovnou $100Mb/s$, je treba dĺžku interburst periódy nastaviť na väčšiu hodnotu ako 1 sekundu a burst magnitúda by sa mala rovnať alebo byť väčšia ako priepustnosť bottleneck linky, kde je potreba zobrať do úvahy aj dĺžku burst periódy, ktorej veľkosť bude predmetom testovania v nasledujúcej kapitole. Príkladom útoku s danými parametrami siete môžeme vidieť na výpise terminálu 3.4.

```
python3 NewShrew.py -l 2000 -bl 0.05 -bm 110 -ib 1.8 -p 80 10.1.1.2
```

Výpis 3.4: Spustenie NewShrew útoku s definovanými parametrami.

Ako už bolo spomínané, tak v CLI sa zobrazujú správy o tom, ako momentálne útok prebieha. Vo všeobecnosti, ak pripojenie a odosielanie paketov prebehne správne, tak útok vygeneruje správu, ktorá informuje o tom, že útok z IP adresy útočníka prebieha na IP adresu a port sieťového zariadenia, na ktoré je útok cielený. Následne sa zobrazí správa, ktorá informuje o správnom nadviazaní TCP spojenia a potom sa už periodicky zobrazujú správy o momentálnych periódach útoku. Tieto logovacie informácie môžeme vidieť na výpise terminálu 3.5.

```
2022-05-10 09:02:20.318376: The Neshrew DoS attack from 10.10.10.10 has
    began to target: 10.10.40.10 with port 8080

                           Attack parameters:
                               Burst magnitude:        110.0 Mbit/s
                               Burst period:           0.05 seconds
                               Interburst period:      1.8 seconds

2022-05-10 09:02:20.319229: Connection to 10.10.40.10 is established...

2022-05-10 09:02:24.447479: number of packets sent in 1. period is: 534
    (Amount of data 0.687792 MB)
2022-05-10 09:02:24.447888: Wait for interburst_period time of 1.8
   seconds
2022-05-10 09:02:26.376714: number of packets sent in 2. period is: 534
    (Amount of data 0.687792 MB)
2022-05-10 09:02:26.377085: Wait for interburst_period time of 1.8
   seconds
2022-05-10 09:02:28.304665: number of packets sent in 3. period is: 534
    (Amount of data 0.687792 MB)
2022-05-10 09:02:28.305026: Wait for interburst_period time of 1.8
   seconds
```

```
2022-05-10 09:02:30.230139: number of packets sent in 4. period is: 534
    (Amount of data 0.687792 MB)
2022-05-10 09:02:30.230508: Wait for interburst_period time of 1.8
   seconds
2022-05-10 09:02:32.152219: number of packets sent in 5. period is: 534
    (Amount of data 0.687792 MB)
2022-05-10 09:02:32.153198: Wait for interburst_period time of 1.8
   seconds
2022-05-10 09:02:34.091167: number of packets sent in 6. period is: 534
    (Amount of data 0.687792 MB)
2022-05-10 09:02:34.091488: Wait for interburst_period time of 1.8
   seconds

2022-05-10 09:02:34.673865: NewShrew atack is interrupted and has been
   stopped
```

Výpis 3.5: Zobrazenie logovacích správ NewShrew útoku v termináli.

4 LoRDAS útok

LoRDAS útok je podkategóriou Low-rate útokov, ktoré sú mierené na aplikačné servery. Tieto servery sa vyznačujú tým, že dokážu spracovávať niekoľko požiadaviek súbežne. Toto spracovane požiadaviek prebieha paralelne, v čom je rozdiel napríklad od iteračných serverov, ktoré ich spracovávajú postupne. Väčšina serverov na internete je implementovaných ako aplikačné servery, keďže poskytujú služby používateľom, ktorí sa pripojujú niekoľkí súčasne [38].

Na to, aby dokázal v dnešnom svete obsluhovať veľké množstvo používateľov, musí spúšťať pomocou multiproccesingu niekoľko vlákien naraz. Napriek tomuto je táto možnosť spracovávania požiadaviek súčasne obmedzená na definovanú hodnotu administrátorom alebo technické možnosti serveru. Túto skutočnosť zneužíva LoRDAS útok. LoRDAS útok sa snaží dotazovať a držať maximálny možný počet otvorených pripojení na server. Takto zahltený server sa snaží LoRDAS útok držať neprístupný minimálnym možným množstvom požiadaviek za určitý čas pomocou zneužitia mechanizmov na obsluhovanie požiadaviek [29]. LoRDAS útok sa teda snaží držať plnú frontu požiadaviek na server, ktorý obsluhuje niekoľko požadaviek súčasne. Použitie distribuovaného modelu LoRDAS útoku bude mať rovnaký účinok ako použitie LoRDAS útoku ako DoS. Rozdiel je v tom, ako sa počet požiadaviek prerozdelí medzi botnet.

Vo všeobecnosti sa LoRDAS útok snaží nechať otvorené relácie pri komunikácii so serverom. Príkladom najčastejšie použitého aplikačného serveru, ktorý používa súbežné spracovanie požiadaviek, je HTTP server [29]. LoRDAS mierený na HTTP server sa snaží zaplniť frontu týchto modulov pomocou keep-alive pripojenia, ktoré je od verzie 2.2 rovné piatim sekundám [15]. LoRDAS útok po nadviazaní pripojenia so serverom pomocou three-way hadshake-u odošle HTTP keep-alive požiadavku na server a počká približne 5 sekúnd a odošle ďalšiu, čím predĺži pripojenie na server a necháva toto pripojenie otvorené. Ak takýmto spôsobom vytvorí toľko relácií, aby zaplnil frontu serveru, a tak nebude schopný obsluhovať ďalších užívateľov, až pokým útočník nepreruší pripojenie [29].

V nasledujúcich kapitolách rozvrhneme, ako Apache server obstaráva požiadavky, a ďalej popíšeme správnu dimenzáciu LoRDAS útoku, ktorú v nasledujúcej kapitole využijeme v rámci implementácie útoku.

4.1 Obstarávanie požiadaviek Apache servera

Keďže v testovacej sieti budeme používať Apache server, tak sa zameriame na jeho použitie a možnosti obstarávania požiadaviek. Apache na obsluhu požiadaviek používa celkovo 7 multiporcesingových modulov (MPM) [39]. V rámci Linux-based

serverov sa používajú tri z nich, MPM worker, MPM prefork a MPM event. Tieto moduly sa používajú podľa jednotlivých potrieb daných serverov. Ako defaultný sa od verzie 2.4 používa MPM prefork [40].

4.1.1 Prefork

Modul MPM prefork nepoužíva v rámci multiprocessingu jednotlivé vlákna, ale separátne podradené procesy pre každé jedno pripojenie. Tento spôsob obstarávania požiadaviek je vhodný, ak v danom systéme nie sú podporované Threads alebo Non-thread-safe knižnice. U preforku je maximálny počet podradených procesov (angl. child proccess) regulovaných pomocou parametru MaxRequestsWorkers, ktorého defaultná hodnota je nastavená na 150 požiadaviek [41].

V praxi je MaxRequestWorkers použitý ako horná hranica pre maximálny počet klientov, keďže prefork používa len procesy na obstarávanie požiadaviek. To teda dovoľuje LoRDAS útoku odoprieť službu na servery posielaním 150 keep-alive požiadaviek každých 5 sekúnd, ak je mpm prefork modul v defaultných nastaveniach. Základné nastavenie modulu MPM prefork môžeme vidieť na výpise konfiguračného súboru `mpm_prefork.conf` 4.1 [41].

```
<IfModule mpm_prefork_module>
        StartServers                    5
        MinSpareServers                 5
        MaxSpareServers                 10
        MaxRequestWorkers               150
        MaxConnectionsPerChild          0
</IfModule>
```

Výpis 4.1: Nastavenie mpm prefork modulu.

4.1.2 Worker

Modul MPM worker zasa na druhú stranu používa vlákna aj procesy k obstarávaniu požadaviek. V základom nastavení je parameter ServerLimit stanovený na číslo 16. To znamená, že každý proces môže mať určitý počet vlákien, ktorými obstaráva pripojenia. Tento počet vlákien je definovaný parametrom ThreadsPerChild, ktorý je prednastavený na hodnotu 25 [42]. Avšak ak LoRDAS útočník použije všetkých 25 vlákien, tak zaplní celú frontu MPM workeru a ten nie je schopný obstarávať ostatné zariadenia, to znamená, že v základom nastavení mpm workeru serveru stačí LoRDAS útoku posielať periodicky 25 požiadaviek každých 5 sekúnd. Základné nastavenie modulu MPM worker môžeme vidieť na výpise konfiguračného súboru `mpm_worker.conf` 4.2.

```
<IfModule mpm_worker_module>
    ServerLimit               16
    StartServers              2
    MaxRequestWorkers         150
    MinSpareThreads           25
    MaxSpareThreads           75
    ThreadsPerChild           25
</IfModule>
```

Výpis 4.2: Nastavenie mpm worker modulu.

4.1.3 Event

Modul MPM event je veľmi podobný MPM worker modulu okrem toho, že MPM worker udržiava separátne vlákna pre každé pripojenie, ktoré je držané ako otvorené kvôli pretrvávajúcemu spojeniu a MPM event spojenia typu keep-alive drží pomocou vlákna, ktoré je určené na obstarávanie keep-alive požadaviek. Takéto riešenie uvoľňuje miesto pre ostatné pripojenia, a aj v prípade veľkého množstva keep-alive požadaviek zostane server dostupný [40].

To však rieši zraniteľnosť, ktorú zneužíva LoRDAS, ktorý je preto na modul typu MPM event neúčinný. Avšak v dnešných sieťach sa stále častejšie objavujú aj MPM worker a MPM prefork, ktoré sú stále zraniteľné voči LoRDAS útoku. Základné nastavenie modulu MPM event môžeme vidieť na výpise konfiguračného súboru `mpm_event.conf` 4.3 [40].

```
<IfModule mpm_event_module>
        StartServers              2
        MinSpareThreads           25
        MaxSpareThreads           75
        ThreadLimit               64
        ThreadsPerChild           25
        MaxRequestWorkers         150
        MaxConnectionsPerChild    0
</IfModule>
```

Výpis 4.3: Nastavenie mpm event modulu.

4.2 Nastavenie parametrov LoRDAS útoku

Pre správnu funkčnosť a efektívnosť sily LoRDAS útoku bude znova dôležitým parametrom nastavovanie časovania jeho periód. Ak predpokladáme znalosť útočníka

dĺžky keep-alive požiadavky, čo v našom prípade je 5 sekúnd, tak správna synchronizácia a nastavenie útoku odosielania paketov by mala byť v intervale každých 5 sekúnd. Avšak je dôležité rátať s RTT, ktoré definuje obojstranné oneskorenie odoslaných požiadaviek. Preto by perióda, medzi ktorou sa čaká na ďalšie odosielanie požadaviek, mala nastavovať podľa dĺžky keep-alive požiadavky mínus RTT a mínus dĺžky periódy, počas ktorej sa posielajú keep-alive požiadavky [37].

Vo všeobecnosti sa teda útok skladá z dvoch periód. Jednej nazývanej off-periódy, počas ktorej sa čaká na uplynutie času keep-alive požiadavky. A následne on-periódy, počas ktorej sa keep-alive požiadavky zasa posielajú na predĺženie plynutia keep-alive požiadaviek. Okrem toho treba rátať s tým, že keep-alive požiadavka by sa mala poslať tesne pred tým, než predošlej keep-alive požiadavke vyprší časovač, lebo by mohlo dôjsť k ukončeniu TCP spojenia a uvoľnilo by sa miesto vo fronte pre legitímneho používateľa [37]. Výpočet dĺžky off-periódy môžeme vidieť na rovnici 4.1, kde $t_{off-period}$ je parameter určujúci dĺžku off-periódy v sekundách, $t_{keep-alive}$ určuje dĺžku trvania jednej keep-alive požiadavky, od čoho musíme odrátať priemernú hodnotu RTT a taktiež dobu trvania odosielania požiadaviek, ktorá je označená ako $t_{on-period}$.

$$t_{off-period} = t_{keep-alive} - \overline{RTT} - t_{on-period} \tag{4.1}$$

Ďalším parametrom k nastaveniu je, koľko keep-alive požiadaviek útočník bude držať otvorených počas komunikácie, a to taktiež určuje, koľko požiadaviek útočník pošle počas každej on-periódy. Počet odosielaných keep-alive požiadaviek (N) by mal byť väčší alebo rovný ako počet vlákien alebo procesov (M), ktoré webový server dokáže obstarávať súčasne. Túto podmienku pre funkčnosť LoRDAS útoku môžeme vidieť na rovnici 4.2 [37].

$$N \geq M \tag{4.2}$$

Najefektívnejšie riešenie je, keď sa počet požiadaviek otvorených LoRDAS útokom rovná počtu požiadaviek, ktoré HTTP aplikačný server dokáže obstarať. To však ráta so skutočnosťou, že útočník pozná tento parameter servera. Okrem toho naznačuje, že práca s Low-rate útokmi je veľmi závislá od parametrov sietí a aplikačných serverov, od ktorých sa odvíja nastavenie Low-rate útokov [29].

4.3 Implementácia LoRDAS

Generátor LoRDAS útoku je implementovaný v programovacom jazyku Python3 vo verzii 3.8 a vyššie. Na správne fungovanie skriptu je potrebné nainštalovať Python moduly, ktorých zoznam a inštalácia je popísaná v prílohe B. V rámci útoku sme

mohli pristupovať viacerými možnosťami, ako otvárať spojenie so serverom. Prvou z nich je vytvoriť paket pomocou socket knižnice, kde by sme pridávali hlavičku HTTP, avšak riešenie týmto spôsobom by nám komplikovalo otváranie viacerých HTTP relácii súčasne. Preto sme pre nadväzovanie relácií s HTTP serverom použili knižnicu requests, ktorá ma predvolené možnosti HTTP hlavičiek, s ktorými je možné nadväzovať spojenie s HTTP serverom. V rámci tejto knižnice sme použili hlavičku keep-alive. Okrem toho útok LoRDAS potrebuje pre správnu funkčnosť súčasne nadväzovať niekoľko spojení naraz. Pre tento účel je vhodné použiť knižnicu threading, ktorá nám dovoľuje spúšťať viacero vlákien v rámci jedného programu. Tento prístup k programovaniu sa nazýva multitrheading, kde sa jeden proces skladá z viacerých vlákien, ktoré každé samostatne spúšťa určitú časť kódu. Každé vlákno v procese funguje nezávisle. Avšak vlákna, ktoré zdieľajú prostriedky, musia koordinovať svoju prácu pomocou semaforu alebo inej medziprocesovej komunikácie [43].

Pomocou threadingu teda spustime funkcie LoRDAS, ktorá otvára spojenie k webovému serveru a periodicky sa ich snaží držať otvorené pomocou ďalších keep-alive požiadaviek, ktoré sa posielajú v cykle spolu s čakaním na dobehnutie off-periódy. V rámci skriptu si dokáže užívateľ nastaviť, koľko relácií chce súčasne otvoriť, a na základe tohto parametru sa otvorí rovnaký počet vlákien, kde každé vlákno spúšťa jednu reláciu a drží ju otvorenú. Príklad otvorenia relácie, ktoré budeme držať otvorené, je možné vidieť na výpise kodú 4.4.

```
s = requests.Session()
s.verify = False  # self-signed cert
start_time = time.time()
while True:
    now = time.time()
    total_difference = now - start_time
    if total_difference > self.attack.lenght:
        break
    try:
        s.get(f'http://{self.attack.ip_address}:{self.attack.port}'
   )
        time.sleep(self.attack.sleep_time)
```

Výpis 4.4: Implemnetácia vytvorenia a odoslania HTTP požiadavky.

Takto použité otvorenie relácie sme následne definovali vo funkcii send_request(), ktorú sme následne niekoľkokrát spustili každú v inom vlákne, ako napríklad vo výpise kódu 4.5, kde sa otvorí v cykle niekoľko vlákien na základe definovaného množstva požiadaviek. Následne sa každé vlákno uloží do listu a jednotlivé vlákna sa spustia a zosynchronizujú pomocou funkcie join().

```
threads = []
requests = self.attack.requests
for i in range(requests):
    if i == 0:
        t = threading.Thread(target=self.send_first_request)
        t.start()
        threads.append(t)
    t = threading.Thread(target=self.send_request)
    t.start()
    threads.append(t)
for thread in threads:
    thread.join()
```

Výpis 4.5: Implementácia vytvorenia vlákien.

LoRDAS útok pozostáva z niekoľkých Python súborov. V rámci súboru nazvaného `LoRDAS.py` sa spúšťa celý skript, v ktorom sú pomocou knižnice argparse zadefinované argumenty, ktoré tento skript prijíma prostredníctvom terminálu. Argumenty, ktoré LoRDAS skript prijíma, sú zadefinované v návode v prílohe B. V rámci tohto súboru sa následne vytvára objekt LoRDAS, ktorý sa ďalej odovzdáva na ďalšie procesy ostatným triedam. V LoRDAS útoku bola implementovaná aj možnosť spúšťania útoku ako DDoS. Ako DoS sa útok spúšťa pomocou objektu single_thread_attack, kde spúšťa len na lokálnom hostovi a spúšťa sa na ňom niekoľko relácií každé v separátnom vlákne. Ak by sme útok chceli použiť ako DDoS útok, tak je v rámci skriptu potrebné definovať v konfiguračnom súbore `zombies.yaml` botov v rámci botnetu. Konfiguráciu tohto súboru je možné vidieť na výpise konfiguračného súboru 4.6.

```
botnet:
  zombie1:
    ip: '10.10.10.40'
    user: 'kali'
    password: 'kali'
  zombie2:
    ip: '10.10.10.50'
    user: 'kali'
    password: 'kali'
  zombie3:
    ip: 10.10.10.60
    user: 'kali'
    password: 'kali
```

Výpis 4.6: Nastavenie botnetu v konfiguračnom súbore zombies.yaml.

Z konfiguračného súboru 4.6 sa spracuje IP adresa, meno používateľa a heslo. Následné sa v rámci skriptu vytvorí list botnet, do ktorého sa uložia objekty Bot

s vyššie spomínanými informáciami. Realizácie botnetu je pomocou Secure Shell (SSH) pripojenia master útočníka, ktorý cez CLI spustí skript `zombie.py` uložený na vziadeľom zariadení, ktoré je použité ako zombie. Takto sa master snaží pripojiť na každý zadefinovaný zombie postupným pripájaním s časovým rozdielom 30 sekúnd. Slúži to na zložitejšiu detekciu DDoS útoku, kde by bola presnejšia detekcia v prípade pripojenia všetkých botov v jednom časovom okamihu. Pripojenie sa drží pomocou dvoch vlákien, kedy sa spustí v jednom vlákne útok z master hosta a v druhom vlákne sa spúšťa LoRDAS na jednotlivých botoch.

Ako je popísané v možných argumentoch skriptu B, tak je možné pomocou útoku vytvárať aj logovacie správy. Logovanie však neovplyvňuje priebeh útoku, preto ich nebudeme v tejto kapitole nejako významne popisovať. Vytvorili sme súbor `logger.py`, ktorej funkcie dodávajú logovacie správy do CLI za behu programu a zároveň ich odosiela do súboru, ktorý je zadefinovaný defaultnou cestou, alebo je možnosť ho meniť cez argument $-log$ zadávaný pomocou scriptu spusteného v termináli. Súbor `pars_checkers.py` sa používa na funkcie, ktoré kontrolujú tvar a formát vstupu, ktoré užívateľ zadáva do terminálu. Okrem toho sú tam zadefinované pomocné správy, ktoré sa zobrazia pri zadaní $-help$ argumentu.

4.3.1 Generovanie LoRDAS útoku

Útok LoRDAS sa spúšťa z terminálu, kde je potrebné zadefinovať len veľmi málo povinných argumentov, keďže veľa z nich má nastavenú defaultnú hodnotu kvôli úzkemu využitiu útoku. Čo sa týka povinných argumentov, tak útok požaduje nastavenie IP adresy a dĺžku trvania útoku. Ostatné parametre útoku sú prednastavené skriptom. V prípade použitia len povinných parametrov útoku by spúšťaný skript vyzeral ako vo výpise 4.7.

```
python3 LoRDAS.py -l 600 10.10.40.10
```

Výpis 4.7: Základné spustenie LoRDAS útoku.

Takto nastavený LoRDAS útok by útočil na server s adresou 10.10.40.10 a trval by po dobu 600 sekúnd. Preddefinovane by posielal 25 požadaviek každú on-periódu a off-perióda by sa rovnala 4.90 sekundy a útok by sa spúšťal ako DoS útok s logovaním do defaultného logovcieho súboru. Logovanie do terminálu môžeme vidieť na výpise terminálu 4.8.

```
2022-05-15 18:10:48.038760: The LoRDAS DoS attack from 10.10.10.10 has
   began to target: 10.10.40.10 with port 8080
                            Attack parameters:
                                Timeout:                        4.90
   seconds
                                Botnet:                         False
```

```
                                Number of requests:            25
2022-05-15 18:10:48.148508: 25 keep alive request is send to
   10.10.40.10 to application running on port:8080
2022-05-15 18:10:48.148508: and will keep open the request for 4.90
   seconds
2022-05-15 18:10:53.130121: 25 keep alive request is send to
   10.10.40.10 to application running on port:8080
2022-05-15 18:10:53.130121: and will keep open the request for 4.90
   seconds
2022-05-15 18:10:58.112726: 25 keep alive request is send to
   10.10.40.10 to application running on port:8080
2022-05-15 18:10:58.112726: and will keep open the request for 4.90
   seconds
2022-05-15 18:11:03.094270: 25 keep alive request is send to
   10.10.40.10 to application running on port:8080
2022-05-15 18:11:03.094270: and will keep open the request for 4.90
   seconds
2022-05-15 18:11:08.086639: 25 keep alive request is send to
   10.10.40.10 to application running on port:8080
2022-05-15 18:11:08.086639: and will keep open the request for 4.90
   seconds
```

Výpis 4.8: Logovanie LoRDAS útoku do terminálu.

V prípade, že by útočník chcel špecificky definovať všetky parametre, tak by spustenie skriptu s argumentmi vyzeralo ako na výpise terminálu 4.9, kde sa pomocou skriptu definuje dĺžka útoku, dĺžka periódy, počet požiadaviek, logovací súbor, port a IP adresa.

```
python3 LoRDAS.py -l 600 -p 8080 -s 4.95 -r 150 -log logfile.log
   10.10.40.10
```

Výpis 4.9: Spustenie LoRDAS útoku s definovanými parametrami.

5 Detekcia a ochrana proti Low-rate DoS útokom

Keďže sa táto práca zaoberá Low-rate útokmi, tak si predovšetkým predstavíme detekciu a obranné techniky pre Low-rate útoky. No niekoľko základých techník voči DoS útokom je funkčných aj voči ostatným typom útokov, ako napríklad flood DoS.

Neoddeliteľnou súčasťou ochrany voči DoS útokom je ich detekcia. Tieto metódy detekcie sú zvyčajne položené na analýze sieťovej prevádzky, keďže veľa útokov práve zneužívaním protokolov a zraniteľnosti spôsobuje neobvyklý priebeh sieťovej prevádzky. Prichádzajúce pakety sú analyzované väčšinou na základe dát každej hlavičky jednotlivej vrstvy protokolu, tie sa následne porovnávajú a analyzujú už známymi opísanými signatúrami a vlastnosťami jednotlivých DoS útokov. Keďže Low-rate útoky prichádzajú s novými taktikami na odopretie služby, tak ochrana voči týmto útokom nemusí byť dostačujúca. Napriek tomu je ale stále dôležité a potrebné správne nastaviť služby, ako sú napríklad Intrustion Decetion systémy (IDS) a Intrustion Prevention systémy (IPS) alebo všeobecné nastavenia firewallu [44].

Jedným zo základných stavebných bodov voči ochrane DoS útokov sú práve spomínané IDS a IPS. Táto dvojica funguje práve na schopnosti detekcie a prevencie voči podozrivým správaniam protokolov v sieti. Detekcia narušenia je proces monitorovania udalostí vyskytujúcich sa v sieti a ich analyzovaniu, ak dôjde k podozreniu na porušenie filtračných pravidiel nastavených v IDS. Prevencia narušenia je proces vykonávania detekcie narušenia siete a prípadné zastavenie tohto incidentu a zamedzenie prístupu do siete [45].

Tieto systémy však nemusia vykonávať ochranu len voči DoS útokom, ochranu vykonávajú voči hrozbám, ktoré je možné identifikovať na základe pravidiel nastavených v týchto systémoch. Dobrým nastavením môžeme práve chrániť sieť voči DoS útokom [45].

V prípade jednoduchého útoku môže mať firewall pridané jednoduché pravidlo na odmietnutie všetkej prichádzajúcej prevádzky od útočníkov na základe protokolov, portov alebo pôvodných IP adries. Zložitejšie útoky však bude ťažké blokovať jednoduchými pravidlami. Okrem toho môžu byť firewall príliš hlboko v hierarchii siete, pričom sú smerovače nepriaznivo ovplyvnené skôr, ako sa prevádzka dostane k firewallu. Mnohé bezpečnostné nástroje tiež stále nepodporujú IPv6 alebo nemusia byť správne nakonfigurované, takže brány firewall môžu byť počas útokov často obídené. Z tohto hľadiska môžu byť tieto ochrany voči Low-rate DoS útokom nedostatočné. Ako už bolo popísané, tak nastavenia firewallu alebo IPS a IDS môžu byť obídenie práve low-rate DoS útokmi, ktoré sa vyznačujú ich sofistikovanosťou [46].

Preto je v tomto smere dôležitý vývoj detekčných techník a obranných systémov práve voči Low-rate DoS útokom, ktoré sú popísané v nasledujúcich kapitolách 5.1 a 5.2.

5.1 Detekcia Low-rate útokov

Proti Low-rate útokom sa navrhlo už niekoľko metód detekcie. Hoci nie je ľahké pozorovať takéto útoky, existuje niekoľko metód na detekciu týchto útokov. No keďže sú stále pomerne nové, tak neexistuje dostatok špecializovaných obranných techník voči týmto útokom [47]. Existujúce metódy detekcie zhrnieme do troch kategórií:

- Metóda, ktorá deteguje vlastnosti DoS útokov
- Metóda detekcie frekvenčnej domény
- Metóda detekcie časovej domény

Tieto tri typy detekcie vo všeobecnosti obsahujú všetky metódy, ktoré v súčasnosti detegujú Low rate DoS útoky [47].

5.1.1 Detekcia na základe vlastností Low-rate DoS útoku

Metódu detekcie vlastností môžeme rozdeliť do troch kategórií, a to kategória detekcie založenej na vlastnostiach sieťovej prevádzky, detekčná metóda založená na charakteristike fronty sieťového smerovača a detekčná metóda založená na ostatných vlastnostiach [47].

Prvá metóda, ktorá deteguje sieťovú prevádzku Low-rate DoS útok, nastáva práve vtedy, ak cieľovému zariadeniu často kolísa využitá šírka pásma. V tomto čase bude prevádzka Low-rate DoS vykazovať abnormálne multi-fraktálne funkcie, ktoré sú základom pre určenie Low-rate DoS útoku. Takže legitímna sieťová prevádzka a prevádzka Low-rate DoS môžu byť správne klasifikované a rozoznateľné. Podľa [48] sa vyvinuli jednoduché matematické modely na odhalenie zložitých multifraktálnych štruktúr, aby bolo možné lepšie opísať a analyzovať sieťovú prevádzku. Hoci je návštevnosť útokov Low-rate DoS malá, stále existujú zmeny v multifraktálnych charakteristikách sieťovej prevádzky.

Metóda detekcie založená na charakteristikách toku sieťovej prevádzky dokáže presne rozlíšiť Low-rate DoS a legitímnu prevádzku, ale ak dôjde k zhlukom dát sieťovej prevádzky, tak miera falošne pozitívnych útokov bude vysoká, vhodným príkladom je streamovanie médií RTSP protokolom. Princípom Low-rate DoS útokov je použitie správneho TCP congestion control. Spôsob detekcie útokov Low-rate DoS založený na priemernej dĺžke vyrovnávacej pamäte analyzuje pomer obsadenosti paketov vo fronte a vzťah medzi veľkosťou a účinkom útoku. Následne je možné detegovať útoky low-rate DoS na základe dát zo smerovača [47].

Ostatné metódy zahrňujú veľa ďalších špecifických parametrov, ktorých je príliš veľa a často bývajú použité pri jednotlivých útokoch, preto ich nie je možné presne rozdeliť.

5.1.2 Metóda detekcie na základe frekvenčnej domény Low-rate DoS útokov

Detekcia Low-rate DoS vyžaduje jasné spôsoby zneužitia sieťovej prevádzky. Metóda môže zodpovedať správnej vzorke na detekciu útokov prostredníctvom možných zraniteľnosti Low-rate DoS útoku, ale pre nový typ zraniteľností zneužívaných Low-rate DoS útokmi, bude miera falošných detekcií útokov veľmi vysoká. Charakteristiky detekčnej metódy sa dajú ľahko ovplyvniť vonkajším rušením [47, 48].

Preto, aby sa tieto problémy vyriešili, niektoré detekčné algoritmy využívajú metódu frekvenčnej domény. Pri detekcii frekvenčnej metódy sa vo všeobecnosti hovorí o dvoch typoch detekcie, a to Wavelet analýze a Spektrálnej analýze [47].

Wavelet analýza môže súčasne zvýrazniť charakteristiky signálov vo frekvenčnej doméne. Takmer všetky signály môžu reprezentovať signál podľa niektorých charakteristík extrahovaných z pôvodných údajov. Ak sa deteguje Low-rate DoS, úspešnú detekciu možno dosiahnuť systematickým lokalizovaním pravidelných impulzov dát [47].

Počas Low-rate DoS útokov bežný tok TCP a abnormálny tok pravidelne posielajú dáta pri prenose, a periodické a neperiodické signály vo frekvenčnej doméne vykazujú rôzne charakteristiky, preto sa môžu použiť na detekciu týchto rozdielov vo Fourierovej transformácii v spektrálnej doméne [47].

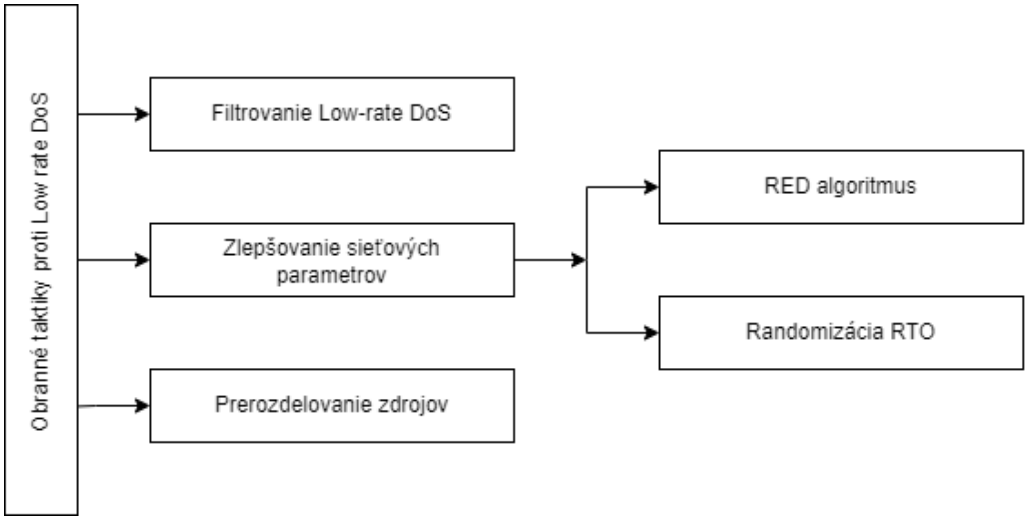
5.1.3 Metóda detekcie na základe časovej domény Low-rate DoS útoku

Metóda detekcie frekvenčnej domény zvyčajne využíva multifraktálne charakteristiky siete na transformáciu na frekvenčnú doménu detekcie Low-rate DoS útokov, napriek tomu, že fourierova transformácia je pomalá a zvýši časové náklady. Metóda detekcie časovej domény je oveľa účinnejšia a priamo využíva model nízkeho signálu pre detekciu Low-rate útokov. Zaoberá sa charakteristikami Low-rate DoS útoku v čase. Výhodou tejto metódy je zjednodušenie a skrátenie procesu detekcie [49].

Proces detekcie útokov Low-rate DoS na základe časovej domény je podobný detekcii vo frekvenčnej doméne, ale odstraňuje proces konverzie signálov na frekvenčnú doménu. Táto metóda má vo všeobecnosti tri podkategórie, a to štatistickú analýzu, koreláciu malých signálov a meranie toku informácií [48].

5.2 Obranné techniky voči Low-rate DoS útokom

Eliminácia Low-rate DoS cez bežné filtre a ďalšie nástroje ako IPS a IDS nie je 100 % funkčná, a preto je potrebné zmeniť ovládacie parametre filtrov alebo nájsť nové spôsoby obrany proti Low-rate DoS útokom [50]. Klasifikácia existujúcich obranných metód je znázornená na obr. 5.1.



Obr. 5.1: Rozdelenie obranných techník voči Low-rate DoS útokom.

5.2.1 Filtrácia Low-rate DoS útokov

Táto metóda filtruje návštevnosť útokov Low-rate DoS a prevádzku TCP pomocou časovej domény a frekvenčnej domény, čím filtruje tok útokov Low-rate DoS. Analýzou amplitúdového spektra sa dá zistiť, že čas RTT je hlavným bodom koncentrácie prevádzkovej energie TCP, takže je možné odhadnúť periodické parametre útoku Low-rate DoS na základe tohto parametru. Na základe toho je navrhnutý Dressing Filter [50]. Najväčšou vlastnosťou tohto filtra je nekonečná odozva Impulzu (IIR), ktorá dokáže dosiahnuť efekt filtrovania tokov Low-rate DoS vo frekvenčnej doméne. Je to preto, že väčšina bežného prevádzkového toku TCP prechádza "Comb filtrom"[50, 51].

5.2.2 Zlepšenie parametrov siete

Hoci "Comb filter"môže zabrániť väčšine toku Low-rate DoS, legitímna prevádzka TCP môže byť odstránená filtráciou a detegovaná ako falošný poplach. Pri útokoch Low-rate DoS je najbežnejšou obrannou stratégiou zlepšenie aktívneho riadenia vyrovnávacej pamäte smerovača. Jeho účelom je splniť charakteristiky dátovej sady

proti vyradenému paketu a pokúsiť sa ochrániť dátový tok TCP pred útokmi Low-rate DoS [52].

Prvou a najbežnejšou aplikáciou aktívnej správy vyrovnávacej pamäte (AQM) je algoritmus náhodnej skorej detekcie (RED) [53]. Technológia AQM je druh technológie správy vyrovnávacej pamäte smerovača na riešenie problémov s kontrolou preťaženia siete. Má vlastnosti aktívnej obrany. AQM vopred vypočíta určitú pravdepodobnosť straty paketov, aby sa znížilo preťaženie siete a zabránilo sa jej stratám a tým pádom sa aj zlepšila kvalita služieb. Priemerná dĺžka frontu vypočítava pravdepodobnosť straty paketov v algoritme RED, ktorá sa použije na reálnu stratu paketov. Útočník však môže zneužiť zraniteľnosť tohto algoritmu na spustenie Low-rate DoS útoku. Preto je možné brániť Low-rate DoS útok vylepšením AQM pomocou RED algoritmu [52].

Útok Low-rate DoS posiela periodicky dáta na základe intervalu RTO. Preto je možné implementovať randomizované RTO proti útokom Low-rate DoS, kde tým pádom pre Low-rate DoS nie je ľahké vypočítať RTO a dosiahnuť účinnosť útoku. Táto metóda dokáže znefunkčniť periodicitu RTO jeho randomizáciou, čo útočníkom znemožňuje presne predpovedať čas odosielania paketov TCP a odosielať dátový tok v presnom čase, aby sa mohol účinne brániť pred útokmi Low-rate DoS [54, 55].

5.2.3 Prerozdelenie zdrojov

Obranné mechanizmy dynamicky upravujú zdroje služieb podľa aktuálneho stavu systému, aby sa zabezpečila bežná rýchlosť odozvy požiadaviek. Systémy často používajú farebné Petriho siete v kombinácii s neurónovou sieťou spätnej propagácie (angl. back propagation) na obranu proti Low-rate DoS útokom [33]. Podľa [33] výsledky ukazujú, že metóda má dobrý výkon v reálnom čase a odozvu v reálnom čase, čo výrazne zlepšuje stupeň automatizácie obranného systému proti Low-rate DoS.

6 Testovanie NewShrew útoku

V tejto kapitole je popísaná funkčnosť NewShrew v sieťovej prevádzke na testovacej sieti. Na túto sieťovú infraštruktúru bol použitý počítačový program VMware verzie 16.1.0, v ktorej nakonfigurujeme jednotlivé sieťové zariadenia, ktoré budú v našej sieti figurovať. V ďalšej kapitole následne popíšeme výsledky testovania NewShrew útoku na nakonfigurovanej sieti.

V kapitole 3.4 sme sa venovali popisu a nastaveniu jednotlivých parametrov útoku a siete. Tieto parametre následne zosumarizujeme a uvedieme všetky predpoklady, ktoré dovoľujú NewShrew útoku zneužiť Server vytvárajúci TCP komunikáciu podľa [28]:

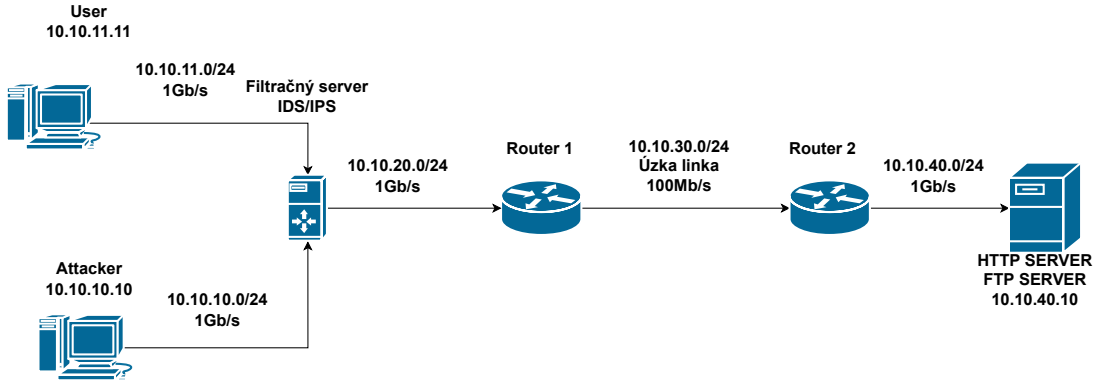
- Šírka pásma bottleneck linky je nastavená na 100 Mb/s, kde ostatné linky komunikujú na 1Gb/s linkách, aj keď v [28] je bottleneck linka nastavená na veľkosť 1,5 Mb/s, čo ale v dnešných sieťach je nevyužiteľná prenosová rýchlosť linky, a preto budeme pracovať s vyššie definovanými rýchlosťami liniek.
- Veľkosť vyrovnávacej pamäte prechodného uzla je nastavená na 1,25 MB, čo je niekoľkokrát vyššia hodnota ako základe nastavená.
- Predpokladaná veľkosť RTT je 20 ms až 600 ms ktorá je podľa [28] najčastejšie sa vyskytujúce RTT v sieti.
- Minimálna veľkosť RTO je 1000 ms s pripočítanou randomizáciou RTO.

Na základe parametrov by mal NewShrew útok pracovať najlepšie v týchto sieťových podmienkach. Od vyššie vymenovaných sieťových parametrov sa odvíja aj nastavenie NewShrew útoku, ktorého nastavenia sa budú v tejto kapitole porovnávať. Keďže linka úzkeho miesta je nastavená na limit 100 Mb/s, tak útok musí počas burst periódy vyvinúť silu útoku rovnú alebo väčšiu ako priepustnosť tejto linky, kde sme silu útoku nastavili na 110 Mb/s pre lepšiu účinnosť útoku. Táto hodnota sa nebude meniť počas celého testovania útoku. Veľmi dôležitým bude porovnávanie a vplyv na účinnosť útoku zmenou burst periódy a interburst periódy. Ako už je vyššie spomínané v kapitole 3, tak burst perióda by sa mala pohybovať v rozmedzí 20 ms až 600 ms, kde budeme sledovať zmenu priemerného oneskorenia spôsobenú RTO práve pri rôznych nastaveniach burst periódy pri odporúčanom nastavení interburst periódy.

Rovnakým spôsobom budeme sledovať zmenu interburst periódy a jej vplyv na efektivitu NewShrew útoku, kde sa hodnoty interburst periódy budú pohybovať vo väčších hodnotách ako minimálne RTO, čo budeme testovať od 1000 ms až po 4000 ms, a kde budeme sledovať, v akých hodnotách je útok najúčinnejší a zároveň najefektívnejší. V rámci testovania porovnáme aj rozdiel účinku útoku na pakety s veľkosťou väčšou ako 1000 B a taktiež veľkosťou menšou ako 100 B, kde sa užívateľ snaží zakaždým hneď obnoviť komunikáciu.

6.1 Konfigurácia siete pre NewShrew

Konfigurovaná sieť sa bude skladať z počítača legitímneho užívateľa, z počítača útočníka, z filtračného servera, webového servera a dvoch sieťových smerovačov. Všetky tieto zariadenia budú virtualizované a konfigurované s Linux-based operačnými systémami. Inštalácia týchto zariadení sa skladá z konfigurácie virtuálneho stroja v grafickom rozhraní VMware softvéru, kde po nakonfigurovaní všetkých virtualizovaných prostriedkov počítača a naimportovaní inštalačného súboru operačného systému sa môže virtuálny stroj spustiť a následne prejsť do konfigurácie samotného operačného systému, ktorá sa odlišuje v každej linuxovej distribúcii. Okrem inštalácie operačných systémov je vo VMware možnosť konfigurácie siete a rozdelenie siete do rôznych segmentov, čo budeme využívať na rozdelenie siete na vonkajšiu a vnútornú a zároveň tým zabezpečíme spolu s ďalšími nastaveniami smerovanie sieťovej prevádzky cez filtračný server. Nakonfigurovanú sieť môžeme vidieť na obrázku 6.1.



Obr. 6.1: Schéma použitej siete.

Konfigurácia sa začína nastavením filtračného serveru, ktorý je konfigurovaný s operačným systémom Ubuntu desktop 20.04. Napriek tomu, že tento server používa inštalačný súbor desktop verzie Ubuntu, tak bude plniť funkciu filtračného serveru a pre lepšiu vizualizáciu budeme používať grafické rozhranie, vďaka ktorému budeme môcť napríklad filtrovať komunikáciu cez IPS a IDS, ale aj možnosť odpočúvania a filtrovania sieťovej prevádzky pomocou softvéru Wireshark.

Hardvér filtračného servera bude pre kompatibilitu VMware nastavený na verziu Worskation 16.x, čo určuje hardvérové limitácie ako veľkosť pamäte, počet jadier procesoru, počet sieťových adaptérov, veľkosť disku a veľkosť zdieľanej pamäte grafickej karty. Ako už bolo spomínané, tak operačný systém filtračného serveru bude Ubuntu 20.04 desktop, preto na inštaláciu použijeme inštalačný súbor pre Ubuntu 20.04 desktop. Filtračný server bude používať dve virtuálne procesorové jadrá a bude

zdieľať RAM pamäť hostiteľského počítača o veľkosti 2048 MB. Operačný systém a súbory budu uložené na novom virtuálnom disku typu SCSI o veľkosti 20 GB. Ako vstupno-výstupný kontrolór bude použitý LSI logic. Sieťové adaptéry u filtračného serveru budú tri, keďže komunikuje so smerovačom, ktorého sieť smeruje k serveru a taktiež komunikuje so sieťou legitímneho užívateľa a sieťou útočníka. Po tomto nastavení je filtračný server nakonfigurovaný spolu aj s operačným systémom a je potrebné nastaviť niekoľko parametrov. Na to, aby takto fungoval a dokázal smerovať pakety z jedného segmentu siete do druhého, je potrebné nakonfigurovať smerovanie IPv4 adries na filtračnom serveri.

Okrem toho je pre správnu funkčnosť NewShrew útoku potrebné splniť niekoľko predpokladov. Podľa [28] je vhodné použiť congestion control systém s názvom reno. Taktiež popri úzkom mieste v sieti je potrebné definovať maximálnu veľkosť sieťovej vyrovnávacej pamäte na prechodnom uzle, kde sa v tomto prípade bude voliť na toto použitie filtračný server. Na filtračnom serveri zmeníme veľkosť sieťového zásobníku na 1,25 MB, čím zvýšime základné nastavenie takmer 6-násobne. Tieto nastavenie docielime zápisom týchto hodnôt do `/etc/sysctl.conf` 6.1.

```
net.ipv4.ip_forward = 1
net.core.wmem_max = 1310720
net.core.rmem_max = 1310720
net.ipv4.tcp_rmem = 102400 873800 1310720
net.ipv4.tcp_wmem = 102400 873800 1310720
net.core.wmem_default = 1310720
net.core.rmem_default = 1310720
net.core.optmem_max = 1310720
net.ipv4.tcp_no_metrics_save = 0
net.ipv4.tcp_congestion_control = reno
```

Výpis 6.1: Nastavenie parametrov filtračného serveru.

Taktiež útok NewShrew dokáže obmedziť priepustnosť linky v sieti minimálnym RTO nastaveným na 1 sekundu podľa [28]. Preto používateľ, webový server a filtračný server budú mať nastavený minimálne RTO na 1 sekundu. Toto nastavenie sa dá zadefinovať v ip route príkazoch, kde túto zmenu prevedieme pre každý zápis v ip route tabuľke, ako môžeme vidieť na výpise z konzoly 6.2.

```
sudo ip route change 10.10.10.0/24 dev ens33 proto kernel scope link
   src 10.10.10.1 metric 100 rto_min 1000ms
sudo ip route change 10.10.11.0/24 dev ens39 proto kernel scope link
   src 10.10.11.1 metric 102 rto_min 1000ms
sudo ip route change 10.10.20.0/24 dev ens38 proto kernel scope link
   src 10.10.20.1 metric 101 rto_min 1000ms
sudo ip route add 10.10.40.0/24 dev ens38 via 10.10.20.2 rto_min 1000ms
sudo ip route add 10.10.30.0/24 dev ens38 via 10.10.20.2 rto_min 1000ms
```

Výpis 6.2: Nastavenie minmálenho RTO filtračného serveru.

Ďalším krokom je konfigurácia samotného webového servera, pritom prebieha nastavenie v grafickom rozhraní softvéru VMware takmer totožne s tým rozdielom, že namiesto inštalaného súboru Ubuntu 20.04 Desktop sme použili inštalačný súbor Ubuntu 20.04 Server. Konfiguráciu Webového serveru je možné vidieť v tabuľke 6.1.

Workastation kompatibilita	16.1.0
Operačný systém	Ubuntu 20.04 Server
Počet virtuálnych jadier	1
Veľkosť pamäte RAM	2048 GB
Typ disku	SCSI
Vstupno-výstupný kontrolór	LSI logic
LAN segment rozhranie	Segemet2

Tab. 6.1: Hardvérové špecifikácie webového serveru.

Na webovom serveri po inštalácii Ubuntu Server 20.04 je potrebné nastaviť jednotlivé parametre systému. Ako účinné sa javí vypnutie parametrov `tcp_dsack` a `tcp_fack`, čo ma za následok, že pri chybách spôsobených pri komunikácii sa nepoužijú tieto techniky obnovenia, ale použije sa namiesto nich RTO. Nastavenie `tcp_fack` sa používa pri vyskytnutí fast retransmission timeout [56]. Pre duplikované ACK pakety sa používa `tcp_dsack` [57]. Taktiež sa javí potrebné vypnutie `tcp_sack`, ktoré informuje pomocou ACK o prijímaní určitej sekvencie bytov [58]. V rámci kapitoly venujúcej sa RTO 3.2 bolo popísané, ako sa vyvíja RTO pomocou `tcp_retries`. Preto pre lepšiu názornosť a efektivitu NewShrew útoku nastavíme maximálnu hodnotu opakovania odosielania paketu na hodnotu 17, čím sa zvýši maximálny počet sekúnd čakania na podanie informácií vyššej aplikačnej vrstvy o pravdepodobnej nefunkčnosti linky. Taktiež ako všetky prvky v sieti sa bude používať congestion control reno.

```
net.ipv4.tcp_sack = 0
net.ipv4.tcp_dsack = 0
net.ipv4.tcp_fack = 0
net.ipv4.tcp_retries2 = 17
net.ipv4.tcp_congestion_control = reno
```

Výpis 6.3: Nastavenie parametrov systému webového serveru.

```
sudo ip route change default via 10.10.40.1 dev ens33 proto static
   metric 20100 rto_min 1000ms
sudo ip route change 10.10.40.0/24 dev ens33 proto kernel scope link
   src 10.10.40.10 metric 100 rto_min 1000ms
```

Výpis 6.4: Nastavenie minimálneho RTO pre všetky ip route na webovom serveri.

Webový server je nakonfigurovaný s Apache verziou 2.4, ktorý sme nainštalovali pomocou **sudo apt-get install apache2**. A následne je na serveri nakonfigurované, aby pracoval na porte 8080. To sa prevedie pomocou úpravy konfiguračného súboru `ports.conf`.

Na konfiguráciu Virtuálneho stroja legitímneho užívateľa sme taktiež použili Ubuntu 20.04 operačný systém, ktorého konfigurácia neobsahovala špecifické požiadavky. Hlavným účelom legitímneho užívateľa je schopnosť otestovať funkčnosť webového serveru cez Webový prehliadač. Konfigurácia vo VMware je v tabuľke 6.2.

Workastation kompatibilita	16.1.0
Operačný systém	Ubuntu 20.04 Desktop
Počet virtuálnych jadier	2
Veľkosť pamäte RAM	2048 GB
Typ disku	SCSI
Vstupno-výstupný kontrolér	LSI logic

Tab. 6.2: Hardvérové špecifikácie virtuálneho stroja legitímneho užívateľa.

V prípade legitímneho užívateľa je taktiež potrebné zmeniť congestion control na reno 6.5 a nastaviť minimálnu hodnotu RTO na 1 sekundu 6.6.

```
net.ipv4.tcp_congestion_control = reno
```

Výpis 6.5: Nastavenie reno congestion control na užívateľovi.

```
sudo ip route change default via 10.10.11.1 dev ens33 proto static
   metric 20100 rto_min 1000ms
sudo ip route change 10.10.11.0/24 dev ens33 proto kernel scope link
   src 10.10.11.11 metric 100 rto_min 1000ms
```

Výpis 6.6: Nastavenie minimálnej hodnoty RTO na užívateľovi.

Pri konfigurácii virtuálneho stroja útočníka sme použili Linux Kali 2021.3 operačný systém, tak ako legitímny užívateľ, tak ani útočník nemá potrebu inštalácie špeciálnej konfigurácie stroja. Dôležitou súčasťou operačného systému je inštalácia Python verzie minimálne 3, ktorá je dôležitá pre možnosť spustenia skriptu NewShrew. Hardvérové špecifikácie môžeme vidieť v tabuľke 6.3.

Workastation komptabilita	16.1.0
Operačný systém	Kali 2021.3
Počet virtuálnych jadier	2
Veľkosť pamäte RAM	4096 GB
Typ disku	SCSI
Vstupno-výstupný kontrolér	LSI logic

Tab. 6.3: Hardvérové špecifikácie virtuálneho stroja útočníka.

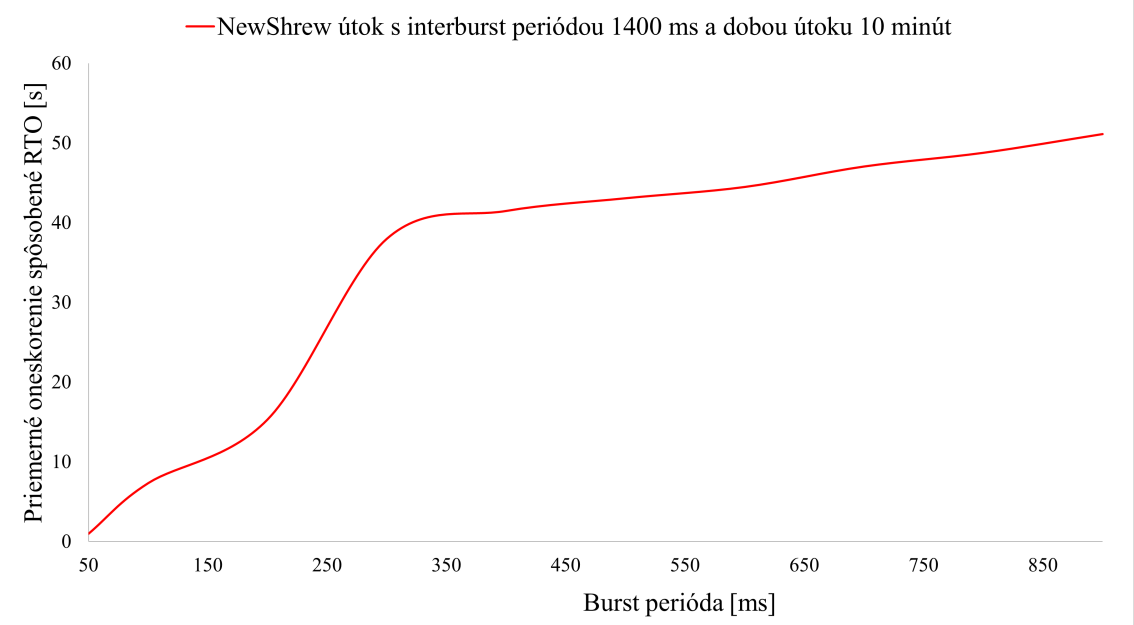
V rámci siete sú medzi filtračným serverom a webovým serverom zapojené dva virtualizované smerovače, ktoré používajú Vyos operačný systém, kde je základná konfigurácia sieťových rozhraní a statického smerovania. Linka medzi týmito smerovačmi je linka s nižšou priepustnosťou, a to 100 Mb/s. Podľa [28] je NewShrew mierený na servery, ku ktorým je sieťová prevádza vedená cez úzke miesto v sieti, čo nám zabezpečuje práve táto linka medzi smerovačmi. Hardvérové špecifikácie sieťových smerovačov sú v tabuľke 6.4.

Workastation komptabilita	16.1.0
Operačný systém	Vyos 1.4
Počet virtuálnych jadier	1
Veľkosť pamäte RAM	512 GB
Typ disku	SCSI
Vstupno-výstupný kontrolér	LSI logic

Tab. 6.4: Hardvérové špecifikácie virtuálnych smerovačov.

6.2 Testovanie vplyvu zmien interburst a burst periódy

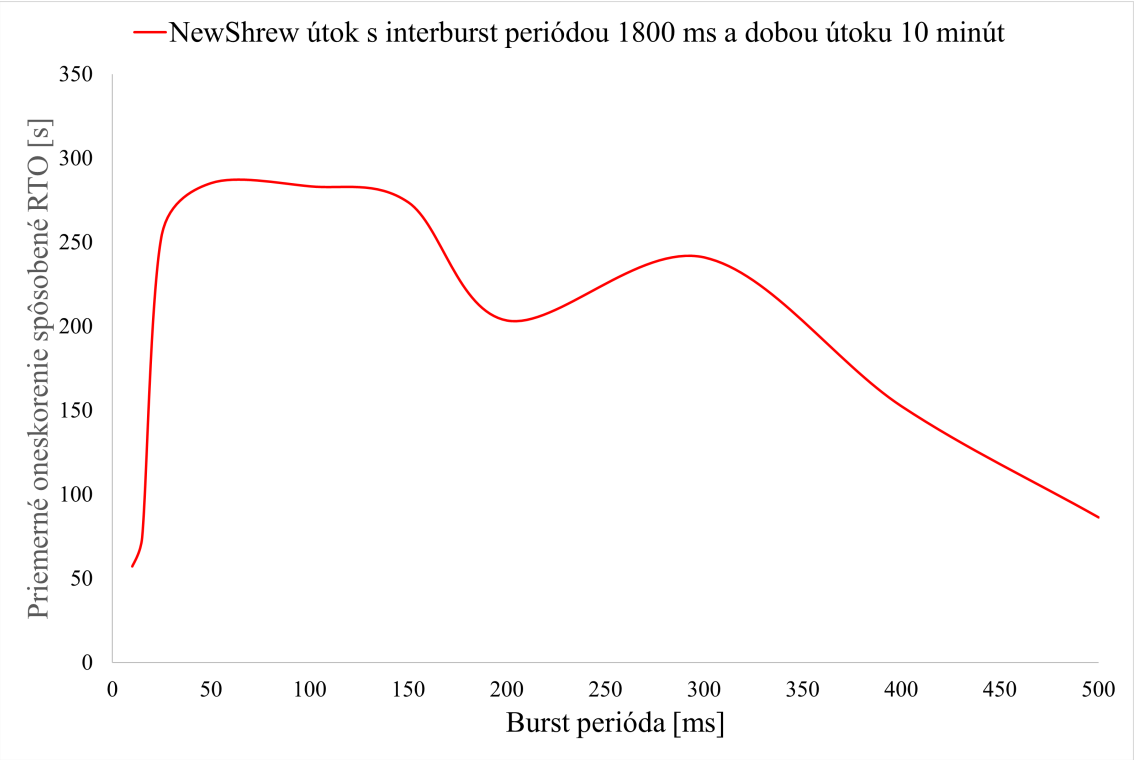
Ako prvé môžeme vidieť zmeny burst periódy a ich vplyv na zmenu priemerného oneskorenia spôsobenú RTO. V rámci grafov je vidieť výrazné rozdiely tohto oneskorenia vplyvom veľkosti paketov komunikácie medzi užívateľom a serverom. Na grafe 6.2 môžeme pozorovať, že NewShrew útok najlepšie obmedzuje služby serveru burst periódou väčšou ako 300 ms a toto oneskorenie sa zväčšuje zväčšovaním burst periódy. Avšak je potrebné tento parameter vyvážiť a nepoužívať burst periódu väčšiu ako 600 ms. Účinnosť útoku to síce zvýši, ale práve za cenu jeho efektivity a taktiež je možné, že zvyšovaním burst periódy by sa NewShrew útok degradoval z low-rate DoS útoku na klasický záplavový DoS útok, keďže by odosielal počas burst periódy oveľa väčšie množstvo dát.



Obr. 6.2: Zmena oneskorenia paketov menších ako 100 B vplyvom zmeny burst periódy.

Ak porovnáme graf 6.2 s 6.3, tak môžeme vidieť, že u paketov s veľkosťou dát väčšou ako 1000 B je oneskorenie vo výrazne vyšších hodnotách. Na tento fakt pôsobí najmä to, že pakety s väčším objemom dát dokážu ľahšie a rýchlejšie zaplniť vyrovnávaciu pamäť filtračného serveru, a tým pádom nie je potrebná rovnaká sila útoku pre obmedzenie komunikácie v oboch prípadoch komunikácie užívateľa so serverom. Preto však nevolíme možnosť zmeny sily útoku, ktorá je v podstate na hraničných hodnotách s priepustnosťou úzkej linky v sieti, ale práve naopak môžeme

znížiť efektívnu dobu trvania burst periódy. Z obrázku 6.3 môžeme vidieť, že priemerné oneskorenie dosahuje najvyššie hodnoty v rozmedzí 30 ms až 180 ms, čo sa dá považovať za najefektívnejší a zároveň najúčinnejší rozsah pre burst periódu v prípadoch, kde je NewShrew útok mierený proti službám odosielajúcim väčší objem dát pomocou TCP, ako je napríklad FTP alebo adaptívny web streaming. Z grafu nám vyplýva, že pri podmienkach v našej sieti je najúčinnejšie nastaviť NewShrew útok s burst periódou o veľkosti 50 ms proti aplikáciam s odosielaným väčším rozsahom dát.

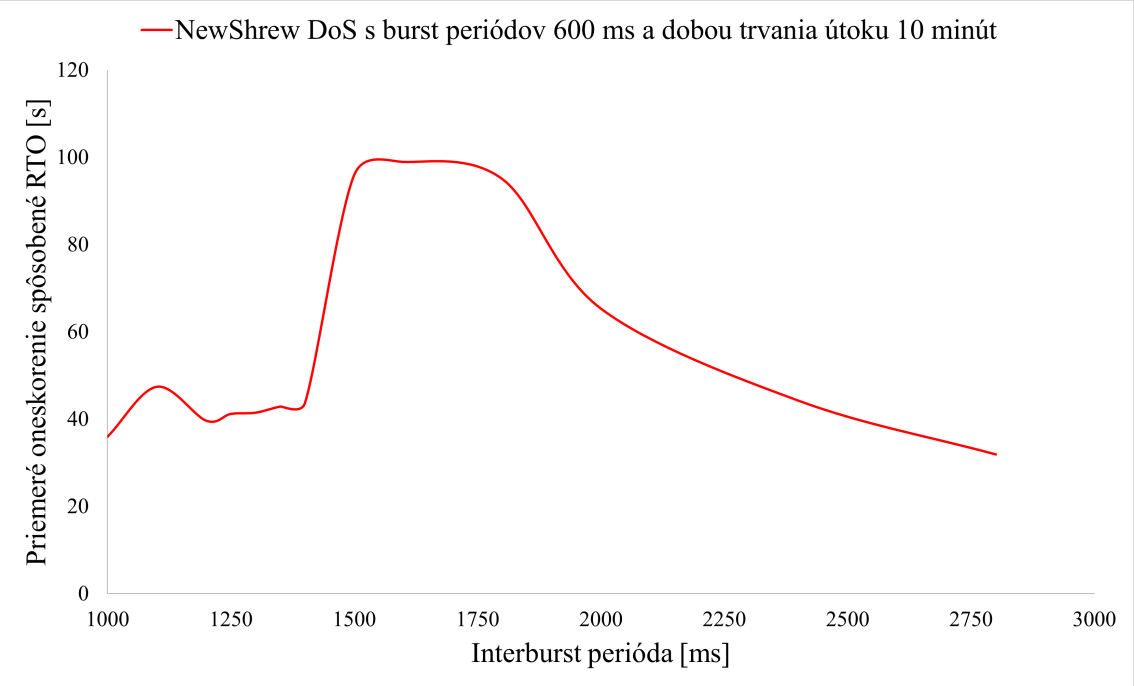


Obr. 6.3: Zmena oneskorenia paketov väčších ako 1000 B vplyvom zmeny burst periódy.

Ako ďalší dôležitý parameter v NewShrew útoku vystupuje interburst perióda, kde taktiež budeme porovnávať vplyv zmeny interburst periódy na zmenu oneskorenia pri rozdielnych veľkostiach paketov, kde ich dôvod je rovnaký, a to rozdiel v zaplnení sieťovej vyrovnávacej pamäte, ktorú jednoduchšie zaplní paket s väčším objemom dát. Na obrázkoch 6.4 a 6.5 môžeme vidieť grafy, na ktorých sa odvíja priemerná doba oneskorenia spôsobená RTO od zmeny interburst periódy. Na grafe, kde sa pracuje s oneskorením dát menších ako 100 B je vidieť v rozsahu 1000 ms až 1200 ms jemný nárast oneskorenia, ktorý je spôsobený vznikom RTO, avšak hodnoty

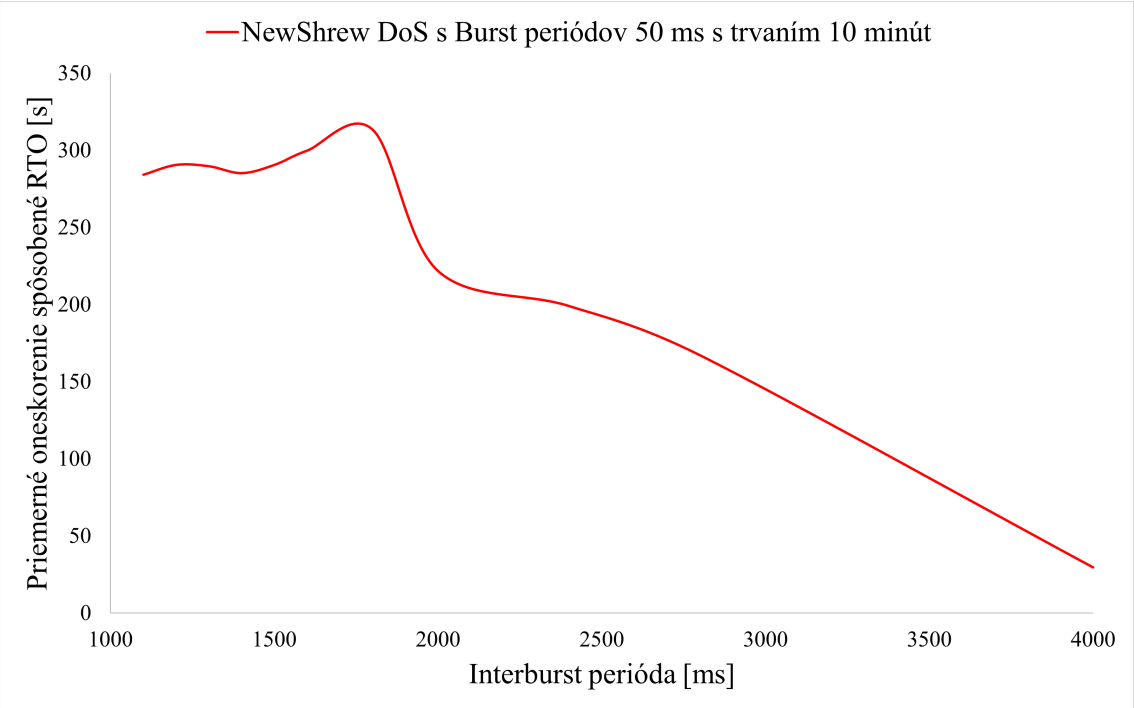
z tohto rozsahu degradujú útok na Shrew útok, keďže tento nárast oneskorenia je spôsobený len RTO.

V prípade NewShrew útoku sa okrem RTO zneužíva aj mechanizmus pomalého štartu, kde vplyv tejto zraniteľnosti môžeme vidieť na osi Interburst periódy od hodnoty 1400 ms až 2000 ms, kde je zrejmé, že NewShrew pracuje najúčinnejšie a dosahuje najvyššie oneskorenie služby. Z tohoto dôvodu je veľmi dôležité časovanie NewShrew útoku, kde môže dôjsť nesprávnym nastavením parametrov k degradácii na Shrew alebo dokonca na záplavový DoS útok.



Obr. 6.4: Zmena oneskorenia paketov menších ako 100 B vplyvom zmeny interburst periódy.

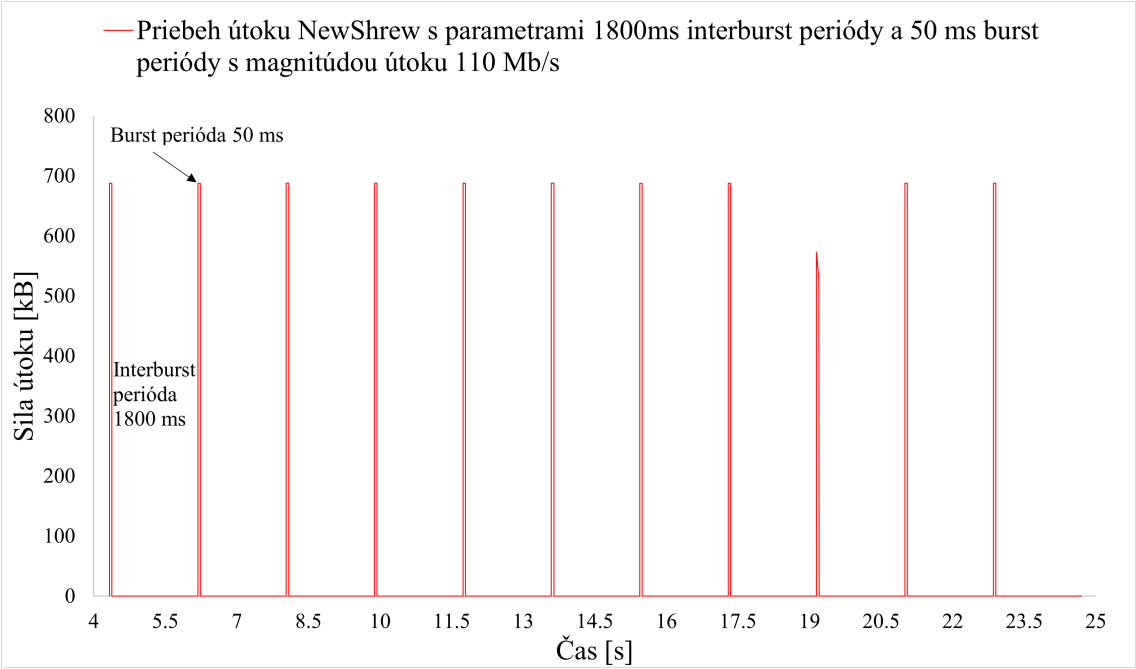
Rovnako ako u paketov menších ako 100 B toto pravidlo platí aj u služieb, ktoré prenášajú väčšie množstvo dát, a tým pádom aj pakety s väčším objemom dát. Na krivke 6.5 je taktiež vidieť medze, kde ide o Shrew DoS a kedy útok začína zneužívať aj mechanizmus pomalého štartu. Avšak v rámci mierenia NewShrew útoku voči aplikáciám s väčším množstvom prenášaných dát môžeme sledovať výrazne vyššie priemerné oneskorenie. Taktiež relatívne efektívnu funkčnosť aj pri interburst perióde vyššej ako 2000 ms. Avšak z výsledkov testovania vychádza, že NewShrew útok je najefektívnejší v rozmedzí 1500 ms až 2000 ms pre Interburst periódu mierený na služby odosielajúce veľký objem dát pomocou TCP.



Obr. 6.5: Zmena oneskorenia paketov väčších ako 1000 B vplyvom zmeny interburst periódy.

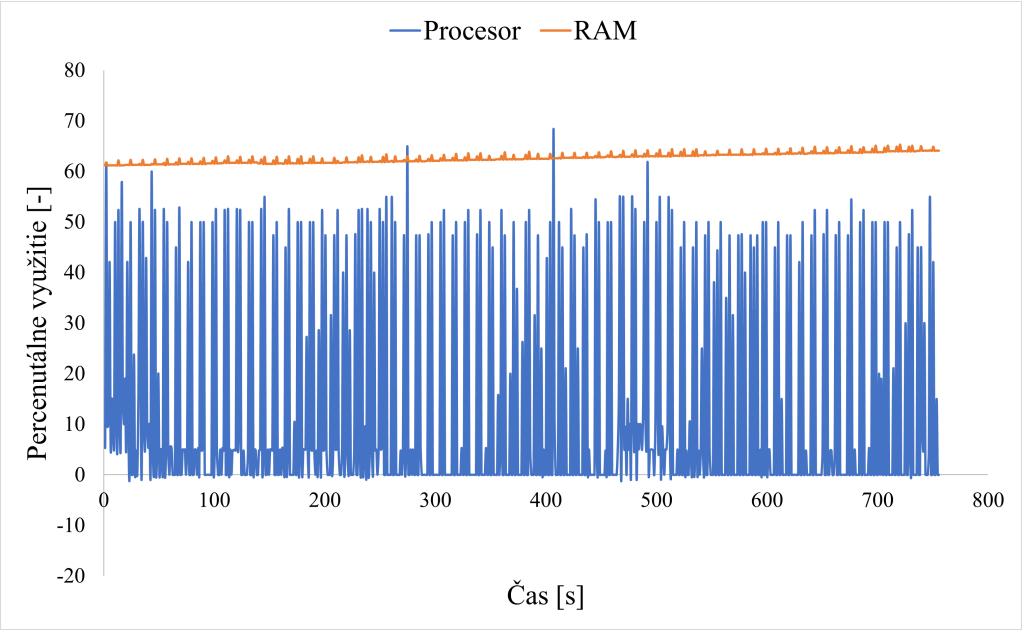
6.3 Testovanie priebehu útoku a účinnosti

V rámci testovania zmeny interburst a burst periódy sa na základe našich získaných dát došlo k záveru, že na testovaciu sieť má ideálne účinky s burst periódou rovnou 50 ms a interburst periódou rovnajúcou sa 1800 ms, ak berieme do úvahy účinnosť a efektívnosť v rámci sily a objemu dát, ktoré útokom vyvinieme. Na obrázku 6.6 môžeme vidieť krivku priebehu útoku, kde je viditeľný výrazný rozdiel v dĺžke burst periódy a interburst periódy. V podstate môžeme hovoriť o nezaznamenateľnej perióde posielania dát silou útoku definovanou ako burst magnitúda. V rámci tohto konkrétneho prípadu sa použilo posielanie UDP paketov s veľkosťou 1288 B, čo je prednastavená hodnota skriptu. S tuto veľkosťou paketov NewShrew útok posiela každú periódu presný počet paketov, a to 534, čo je rovné približne 690 KB. Ak túto hodnotu spriemerujeme, tak v tomto prípade NewShrew útok útočí silou 372 KB/s. Počas pôsobenia útoku na server prenesie útočník približne 670 MB za hodinu, kde už po niekoľkých sekundách dosiahneme úplné odopretie služby.

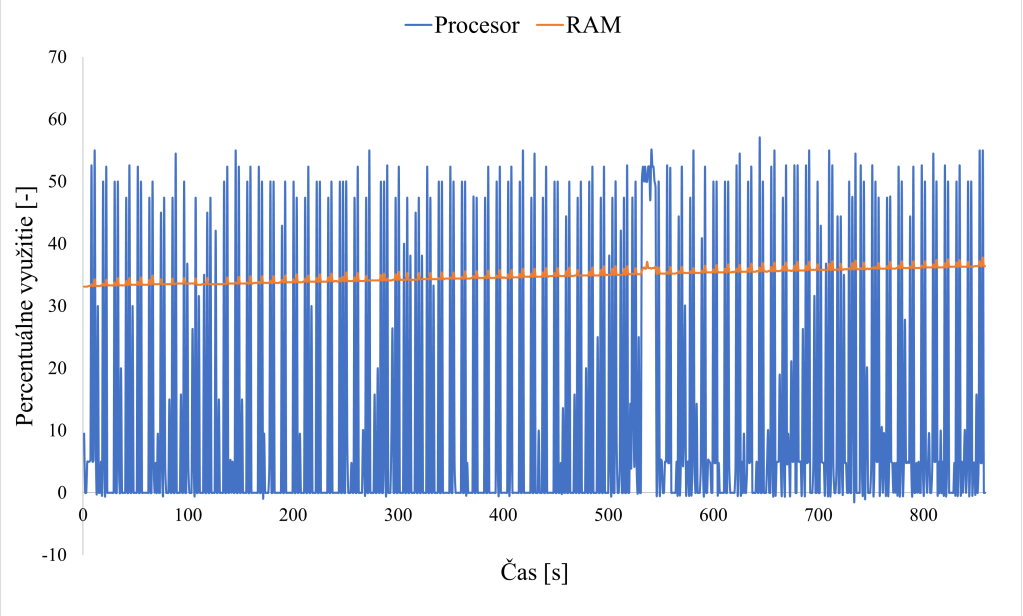


Obr. 6.6: Priebeh útoku NewShrew s parametrami 1800ms interburst periódy a 50 ms burst periódy s magnitúdou útoku 110 Mb/s.

Okrem toho môžeme vidieť na nasledujúcich obrázkoch grafy 6.7, 6.8 so zmenou zaťaženia procesoru a RAM pamäte serveru počas bežných podmienok a počas útočenia NewShrew útoku. Z priebehu je vidieť, že NewShrew nemá vplyv na zaťaženie procesora. Čo sa týka RAM pamäte, tak server spracováva jednotlive pakety a odosiela ich, čo zvýši zaťaženie približne o 20 %, avšak táto hodnota je približne väčšia o 400 MB, keďže server je virtualizovaný a ma dostupnú pamäť len o veľkosti 2048 MB.



Obr. 6.7: Zaťaženie procesora a RAM pamäte serveru počas NewShrew útoku.

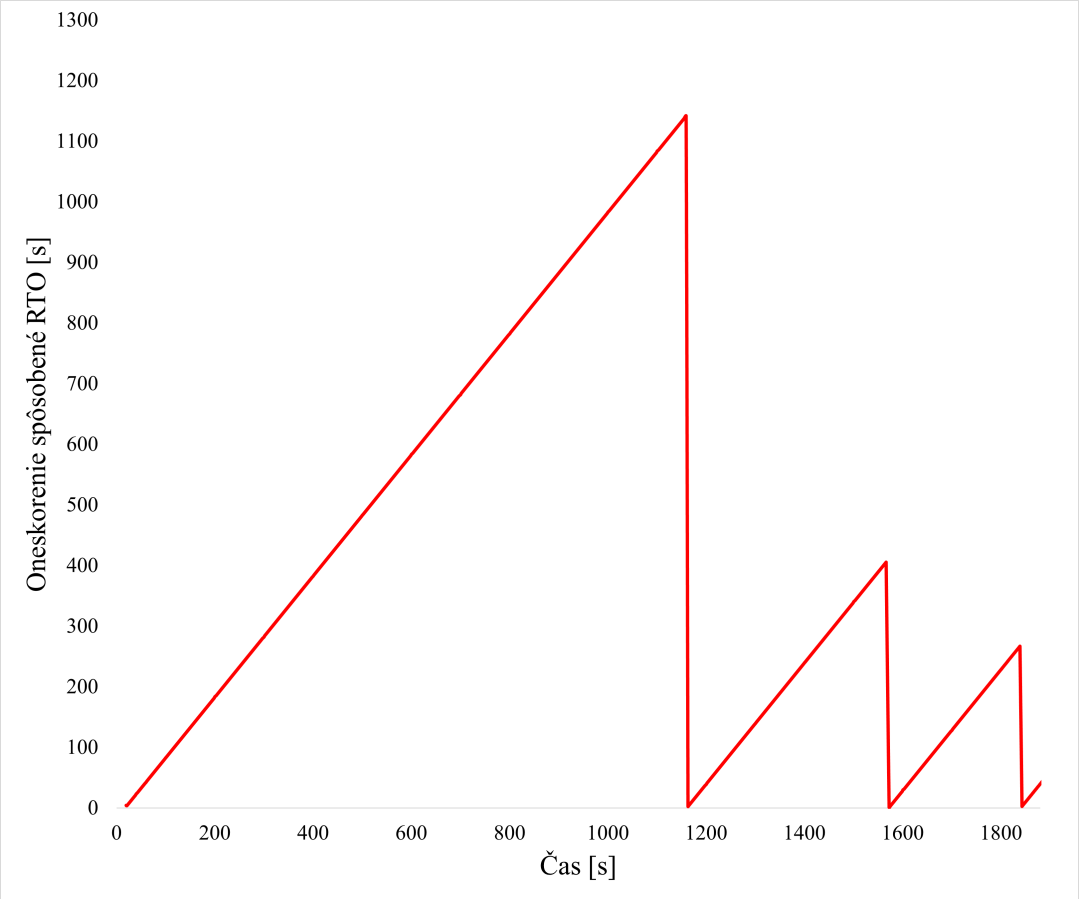


Obr. 6.8: Zaťaženie procesora a RAM pamäte serveru počas bežnej prevádzky.

Počas pôsobenia NewShrew útoku sa nami nastavenými parametrami dokáže sieťová vyrovnávacia pamäť smerovačov medzi linkou s nízkou priepustnosťou zaplniť počas prvej periódy útoku. V prípade, že sa legitímny užívateľ snaží nadviazať komunikáciu so serverom, tak počas NewShrew útoku sa len s ťažkosťou dokáže preniesť niekoľko paketov, avšak aj keď sa daný paket od užívateľa k serveru alebo naopak

prenesie, tak následne jedna z komunikujúcich strán nedostane potvrdzovací(angl. acknowledgement) paket, keďže sa nedokáže dostať cez prepojovací prvok v sieti, ktorý ma v ten moment zaplnenú sieťovú vyrovnávaciu pamäť. Ak po určitom čase serveru paket nedorazí, tak server prejde do fázy RTO a vypne komunikáciu s daným užívateľom. Po vypršaní RTO času sa server pokúsi o opätovné odoslanie paketu a znova čaká na doručenie potvrdzovacej správy o doručení paketu k užívateľovi, ten však znova nedorazí a kvôli funkcii `tcp_retries` spomenutej v kapitole 3.2, sa RTO exponenciálne navýši. Tento proces sa opakuje, až kým sa nezopakuje 17-krát, čo je definované ako maximálny počet pokusov o odoslanie paketov definovaných v `tcp_retrires2`.

Takýmto spôsobom dokáže užívateľ počas 30-minútovom priebehu nadviazania komunikácie poslať 4 pakety, kde prvý z paketov dosahuje oneskorenie až 1140 sekúnd, kvôli tomu, že sa server daný paket pokúšal 17-krát odoslať, kde počas každého pokusu o odoslanie zvýšil exponenciálne RTO. Tento priebeh komunikácie môžeme vidieť na grafe zobrazenom na obrázku 6.9.



Obr. 6.9: Priebeh oneskorenia spôsobeného NewShrew útokom s interburst periódou 1800 ms a burst periódou 50 ms počas tridsiatich minút.

6.4 NewShrew obranné a detekčné mechanizmy

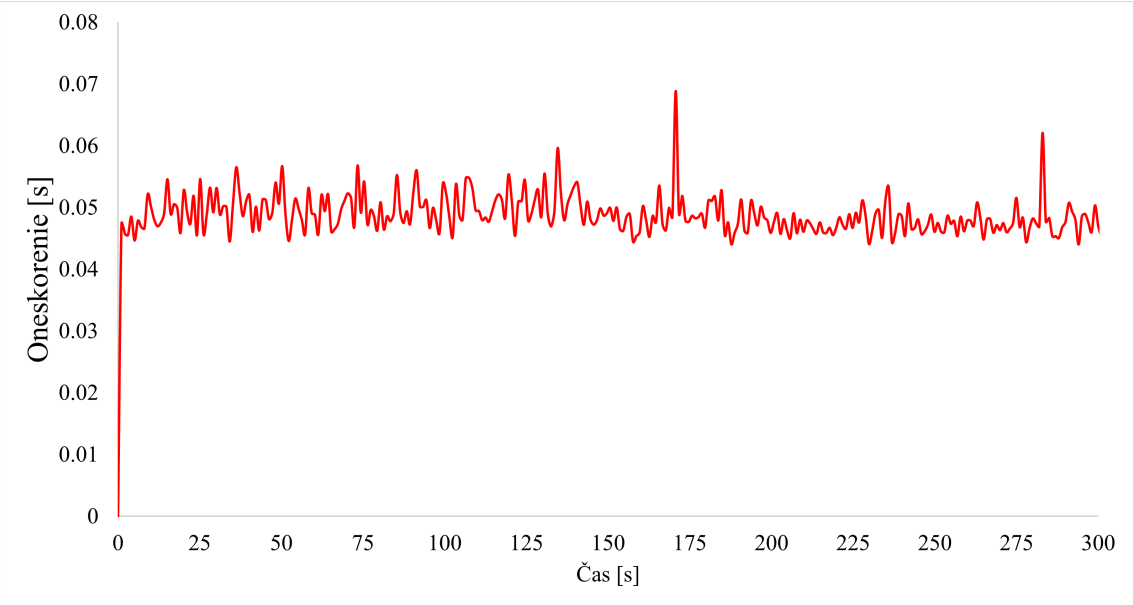
V rámci Low-rate útokov je filtrovanie prevádzky pomocou IDS/IPS veľmi nepresné a aj keď tieto systémy dokážu filtrovať sieťovú prevádzku a odhaliť Low-rate útok, tak spoločne aj s veľkým množstvom falošne vyvolaných detekcií [50]. Inak tomu nie je ani v prípade NewShrew útoku, kde jeho magnitúda útoku a krátka doba trvania burst periódy núti filtráciu na IDS/IPS nastaviť limit filtrácie podľa počtu paketov za časový interval na nízku hodnotu. V rámci testovacej siete sme na filtračnom serveri nainštalovali IDS/IPS systém Suricata 5.0.3. Na základe znalosti o NewShrew útoku a jeho možnosti nastavenia parametrov podľa parametrov siete, ktoré taktiež poznáme, dokážeme vytvoriť filtračné pravidlo, ktoré deteguje väčší počet paketov mierený na TCP služby v rozmedzí RTT, čo je medzi 20 ms až 600 ms. Taktiež je známe, že linka s nízkou priepustnosťou funguje na rýchlosti 100 Mb/s a v rozmedzí 20 ms dokáže preniesť približne viac ako 200 paketov, kde musíme rátať s minimom, ktoré by NewShrew útok dokázal použiť v rámci maximálneho časového rozmedzia. Takýmto spôsobom nastavené pravidlo môžeme vidieť vo výpise 6.7.

```
drop udp !$HOME_NET any -> $HOME_NET 8080 (msg:"Potential NewShrew
    attack detected"; threshold: type both, track by_src, count 200,
    seconds 0.6; classtype:successful-dos; sid:10001; rev:1;)
```

Výpis 6.7: Nastavenie pravidla na Suricata 5.0.3.

Dané pravidlo dokáže stopercentným spôsobom detegovať NewShrew útok a zahadzovať túto prevádzku, ako môžeme vidieť aj na obrázku 6.10, kde oneskorenie komunikácie užívateľa so serverom je bez nejakého výrazného oneskorenia. Avšak ak by s daným serverom komunikovala aplikácia pomocou UDP protokolu, tak by filtrovacie pravidlo hlásilo každú takúto komunikáciu a dosahovalo by takmer 100 % mieru falošných detekcií. Z týchto dôvodov nie je možné použiť bežne dostupné IDS/IPS ako Suricata proti Low-rate DoS útokom. Z toho vyplýva, že detekčné a prevenčné mechanizmy voči NewShrew útoku musia byť založené na metódach spomenutých v kapitole 5.1.

Ako už vieme, tak NewShrew útok je zložený zo striedajúcich sa periód, u ktorých je dôležitý presný čas ich striedania. Tieto periódy sú odvodené od parametrov siete ako RTT a RTO, kde sa toto časovanie periód dá detegovať práve pomocou vyššie uvedenej 5.1.3 metódy detekcie na základe časovej domény, kde sa dá vzápätí použiť na filtráciu spolu s predpokladanými parametrami siete v rámci Dressing Filtra [50], ktorý filtruje pomocou časovej domény odvodenej od parametrov siete ovplyvňujúcich funkčnosť NewShrew útoku (RTT, RTO).

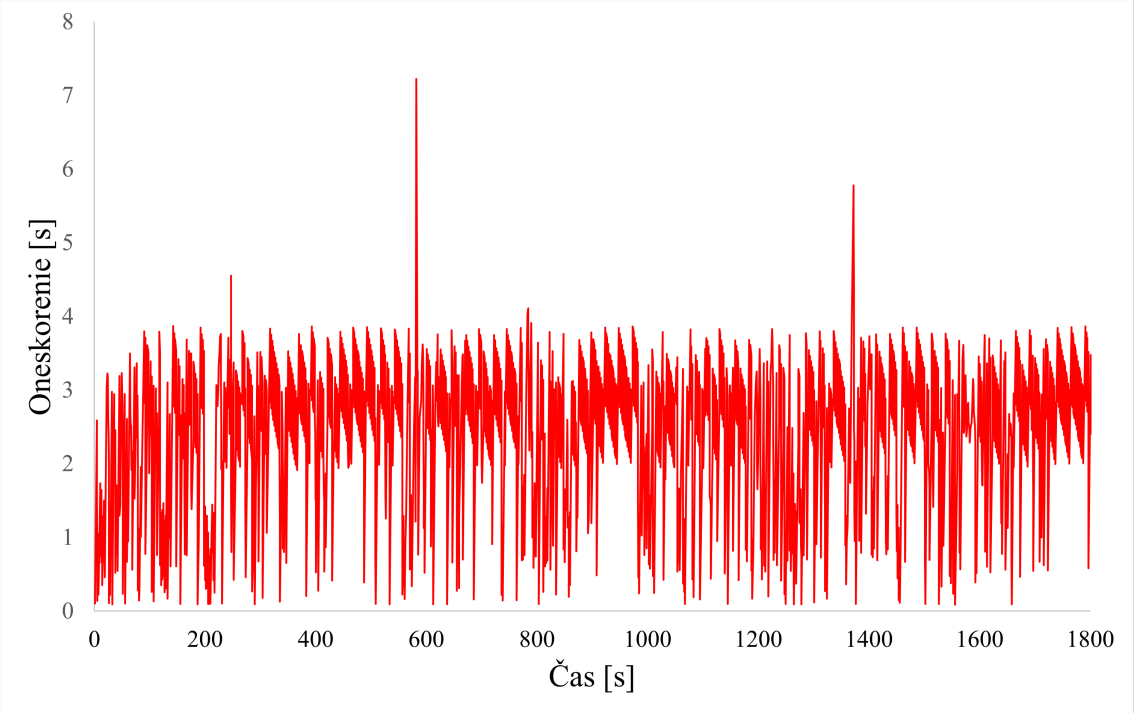


Obr. 6.10: Priebeh oneskorenia so spusteným NewShrew útokom s interburst periódou 1800 ms a burst periódou 50 ms počas tridsiatich minút s aplikovaným IDS/IPS na filtračnom serveri.

Okrem filtrácie je možné odvrátiť NewShrew útok pomocou zlepšenia parametrov siete. Jedným z týchto mechanizmov je randomizácia RTO, ktorá je už priamo zakomponovaná v jadre väčšiny operačných systémov, kde je táto randomizácia založená na prepočte z RTT z každého pripojenia a kombinovaná s minimálnou hodnotou RTO [55]. Pri vyšších hodnotách RTO, ako je napríklad 1 sekunda (Prípad v testovacej sieti 6.1), je táto randomizácia v radoch desiatok milisekúnd a nemá vysoký vplyv na veľkosť RTO.

Ďalším spôsobom, ako zlepšiť parametre siete pre lepšiu odolnosť voči NewShrew útoku, je minimalizácia RTO. V dnešných sieťach je minimálna hodnota RTO nastavená na hodnotu 200 ms [35]. Táto hodnota je v určitom smere stále zneužiteľná a NewShrew útok je možné zosynchronizovať s touto hodnotou, kde nepríde ešte stále k až tak výraznej degradácii útoku na záplavový útok. Administrátor má však možnosť RTO minimalizovať na ešte nižšiu hodnotu. V našom prípade sme RTO nastavili na hodnotu 50 ms, čo je približne rovné najčastejšie sa vyskytujúcemu RTT v komunikácii testovacej siete. Ak by chcel útočník zosynchronizovať interburst periódu s minimálnou hodnotou RTO, tak by sa musela rovnať alebo byť väčšia ako 50 ms, čo pri striedaní s burst periódou o veľkosti v rozsahu 20 ms až 200 ms by degradovalo NewShrew útok na záplavový útok a automaticky by sa stal jednoduchšie detegovateľnejší. Okrem toho má v tomto prípade väčší vplyv randomizácia RTO na jeho celkovú hodnotu. Na obrázku 6.11 je vidieť prípad, kedy randomizácia a mini-

malizácia výrazne znižuje oneskorenie spôsobené NewShrew útokom až o 98,85 %. Útok napriek tomu dokáže dosiahnuť určitú hodnotu oneskorenia v rádoch jednotiek sekúnd.



Obr. 6.11: Priebeh oneskorenia spôsobeného NewShrew útokom s interburst periódou 1800 ms a burst periódou 50 ms počas tridsiatich minút s implementovanou minimalizáciou RTO v sieti.

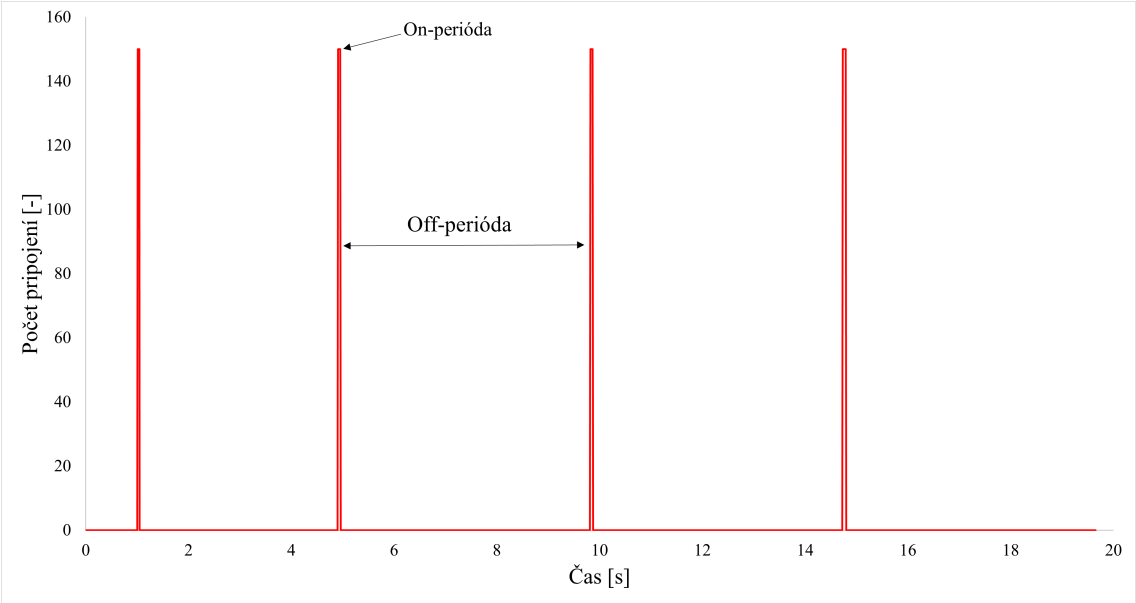
7 Testovanie LoRDAS útoku

V konfigurácii testovacej siete pre LoRDAS útok budeme postupovať podobne ako pri NewShrew útoku s tým rozdielom, že nebude potreba konfigurovať špecifické podmienky siete. Sieť bude jednoduchšia. Konfigurácia bude prevažne prebiehať na cielenom webovom serveri. V rámci testovania následne budeme testovať účinok LoRDAS útoku na jednotlivé možnosti obsluhovania požadaviek Apache serverom.

Testovanie LoRDAS útoku prebiehalo v sieti popísanej v kapitole 7.2, kde sa v rámci útočníka alebo botnetu spúšťal generátor LoRDAS útoku. V prípade Low-rate útokov ako NewShrew je potrebné testovanie prispôsobiť citlivosti vplyvu zmeny parametrov na beh a výsledky útoku. U LoRDAS útoku nie je nastavovanie parametrov až takou dôležitou vlastnosťou pre testovaniu. Aj keď parametre treba znova prispôsobiť sieťovým požiadavkám siete a jej vlastnostiam ako už bolo napísané v kapitole 4.2.

V testovacej sieti bola nameraná najčastejšia hodnota RTT rovná 55 milisekundám. Avšak tento parameter sa počas 6 hodín merania pohyboval v rozmedzí približne 20 až 100 milisekúnd. Taktiež sa ráta s možnosťou, že keep-alive časovač webového servera je nastavený na 5 sekúnd. Okrem toho je potrebné rátať aj s vplyvom dĺžky on-periódy, ktorá nie je LoRDAS útokom nastavovaná ani regulovaná. Preto jedna off-perióda LoRDAS útoku bude pri každom meraní rovná 4.85 sekundy, kedy rátame s najhorším prípadom RTT, ktorý v sieti môže nastať. Ak by sme danú periódu nastavili vyššie alebo napríklad rovnú KeepAliveRequest, tak by pôsobením RTT a dĺžky on-periódy mohli keep-alive požiadavky vypršať a spojenie by sa ukončilo a v danom momente by sa vo fronte serveru uvoľnilo miesto pre legitímneho užívateľa.

Taktiež nám to napovedá, že pri nastavovaní nebudeme riešiť dĺžku on-periódy, ale počet keep-alive požadaviek, ktoré je útok schopný poslať alebo predĺžiť počas jednej tejto periódy. Množstvo týchto požadaviek sa bude líšiť od modulu multiproccesingu, ktorý bude použitý na zneužitom webovom serveri, čo odvodíme pri každom testovaní jednotlivých modulov. V rámci celkovej dĺžky útoku záleží na útočníkovi, na akú dobu chce útok spustiť na cieľ útoku. V prípade zadefinovania tohto parametru ako 0 bude útok spustený, až pokým ho útočník nevypne ručne. Avšak účinnosť útoku by mala nastať už po prvej perióde útoku, preto budeme spúšťať útok pre testovacie účely po dobu desiatich minút, keďže účinky útoku sa prejavia už po niekoľkých sekundách a zostávajú nemenné po dobu celého behu útoku. Takto zostavený útok môžeme vidieť na obrázku 7.1, ktorý zobrazuje LoRDAS útok mierený na prefork modul. V prípade mierenia útoku na iné moduly sa bude líšiť iba množstvom otvorených pripojení na server.



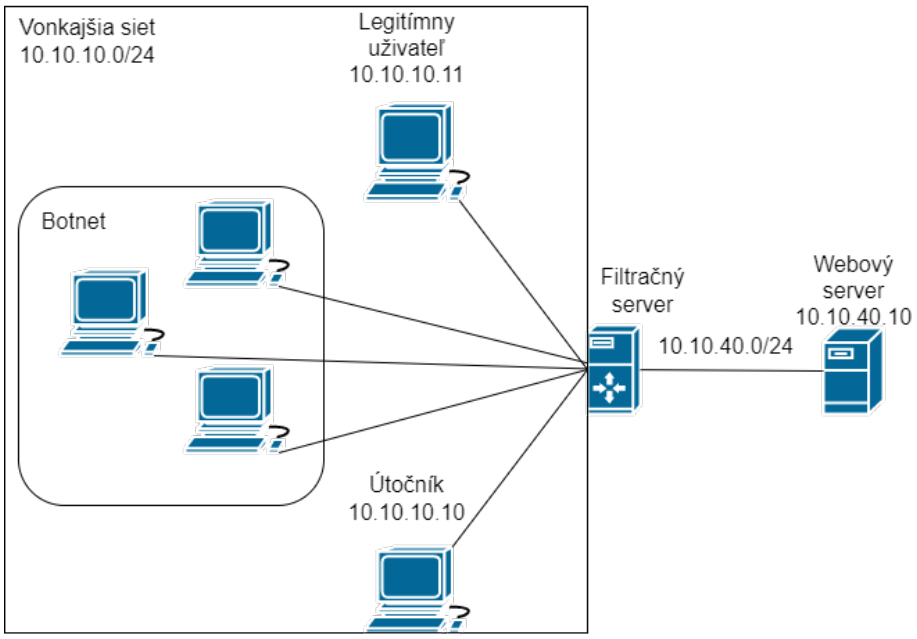
Obr. 7.1: Priebeh LoRDAS útoku s off periódou 4.85 sekundy a 150 keep-alive požiadaviek v on-perióde.

7.1 Konfigurácia siete pre LoRDAS

Pre testovanie loRDAS útoku sa sieť bude skladať z útočníka a zo skupiny počítačov v botnete ovládaných prostredníctvom počítača útočníka. K webovému serveru sa komunikácia bude smerovať prostredníctvom filtračného servera, ktorý bude rozdeľovať sieť na vonkajšiu a vnútornú. Okrem útočníka a botnetu bude vo vonkajšej sieti aj počítač legitímneho užívateľa. Všetky zariadenia budú virtualizované a spúšťané pomocou programu VMware verzie 16.1.0. Okrem konfigurácie virtualizovaných strojov je možné v rámci VMware rozdeliť sieť do logických podsietí. Filtračný server bude rozdeľovať sieť na dva segmenty. Schému navrhnutej siete pre testovanie LoRDAS útoku je možné vidieť na obrázku 7.2.

Inštalácia virtuálnych staníc v programe WMware bude prebiehať rovnakým spôsobom ako aj v kapitole 6.1. Preto sa v tejto kapitole zameriame na hardvérové špecifikácie jednotlivých virtuálnych strojov a prípadne aj ich konkrétnej konfigurácii. V prípade legitímneho používateľa sa použijú hardvérové špecifikácie podľa tabuľky 7.1, kde okrem použitia programu webserver tool na meranie zaťaženia servera nebude nič iné nastavené ani použité.

Filtračný server bude konfigurovaný s možnosťou preposielania (angl. forwarding) sieťovej komunikácie z vonkajšej siete do vnútornej a naopak pomocou príkazu z výpisu 7.1, ktorý je potrebné zapísať do konfiguračného súboru `sysctl.conf`. Ako



Obr. 7.2: Testovacia sieť.

Workastation kompatibilita	16.1.0
Operačný systém	Ubuntu 20.04 Desktop
Počet virtuálnych jadier	1
Veľkosť pamäte RAM	2048 GB
Typ disku	SCSI
Vstupno-výstupný kontrolér	LSI logic
LAN segment rozhranie	Segemet1

Tab. 7.1: Hardvérové špecifikácie virtuálneho stroja legitímneho užívateľa.

ďalší bude na filtračnom serveri nainštalovaný IDS/IPS systém Suricata 5.0.3 pre filtráciu paketov. Hardvérové špecifikácie sú definované v tabuľke 7.2.

```
net.ipv4.ip_forward = 1
```

Výpis 7.1: Nastavenie konfiguračného súboru filtračného servera.

Útočník a stanice z botnetu budú konfigurované ako Kali linix 2021.3. Rovnako aj hardvérové špecifikácie budú rovnaké ako v tabuľke 7.3. Taktiež všetky tieto stanice majú možnosť spustiť Python3 skript.

Počítač útočníka bude mať možnosť spustiť skript LoRDAS.py a definované informácie v súbore `zombie.yaml`. Na staniciach v botnete budú Python3 súbory spustiteľné pre botov z botnetu a to `zombie.py` a `pars_checkers.py`. Taktiež na staniciach obsiahnutých v botnete bude otvorený port pre možnosť pripojenia po-

Workastation kompatibilita	16.1.0
Operačný systém	Ubuntu 20.04 Server
Počet virtuálnych jadier	1
Veľkosť pamäte RAM	3072 GB
Typ disku	SCSI
Vstupno-výstupný kontrolór	LSI logic
LAN segment rozhranie 1	Segemet2
LAN segment rozhranie 2	Segemet1

Tab. 7.2: Hardvérové špecifikácie filtračného servera.

mocou SSH kvôli možnosti ovládania ich prostredníctvom Master stanice (stanica útočníka).

Workastation kompatibilita	16.1.0
Operačný systém	Kali 2021.3
Počet virtuálnych jadier	1
Veľkosť pamäte RAM	2048 GB
Typ disku	SCSI
Vstupno-výstupný kontrolér	LSI logic
LAN segment rozhranie	Segemet1

Tab. 7.3: Hardvérové špecifikácie virtuálneho stroja útočníka a počítačov z botnetu.

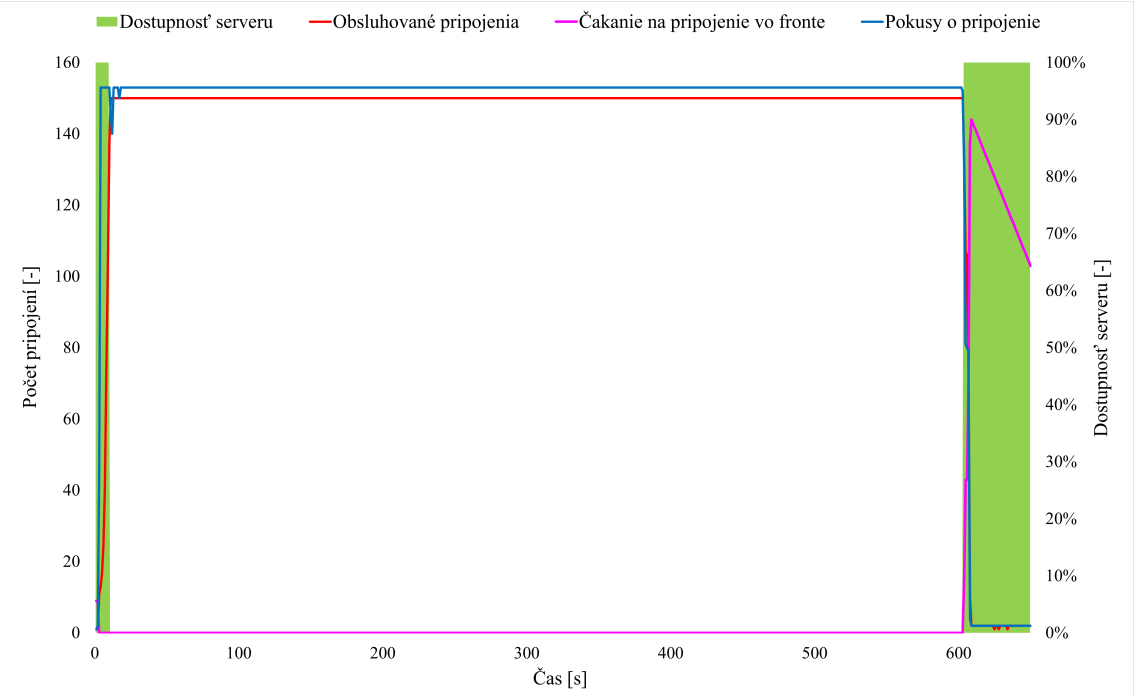
Virtuálna stanica webového servera bude inštalovaná s operačným systémom Ubuntu server 20.04, na ktorej bude nainštalovaný Apache 2.4 so službou bežiacou na porte 8080. V konfiguračnom súbore webového servera je definovaná maximálna dĺžka keep-alive požiadavky pomocou parametru KeepAliveTimeout a maximálny počet keep-alive požadaviek odoslaných jedným užívateľom pomocou parametru KeepAliveRequests. V prípade maximálnej možnosti odoslania keep-alive požadaviek sme túto hodnotu nastavili na 100. A parameter KeepAliveRequest je ponechaný na defaultnej možnosti 5 sekúnd. Tieto nastavenia sa nastavujú v súbore `/etc/apache2/apache2.conf` na Ubuntu Linux-based operačných systémoch [39]. Taktiež sa počas testovania budú meniť moduly multiproccesingu obsluhovania žiadostí podľa výpisov z konfiguračných súborov 4.1, 4.2, 4.3. Hardvérové špecifikácie webového serveru je možné vidieť v tabuľke 7.4.

Workastation kompatibilita	16.1.0
Operačný systém	Ubuntu 20.04 Server
Počet virtuálnych jadier	1
Veľkosť pamäte RAM	2048 GB
Typ disku	SCSI
Vstupno-výstupný kontrolór	LSI logic
LAN segment rozhranie	Segemet2

Tab. 7.4: Hardvérové špecifikácie webového serveru.

7.2 Testovanie vplyvu LoRDAS útoku na MPM prefork

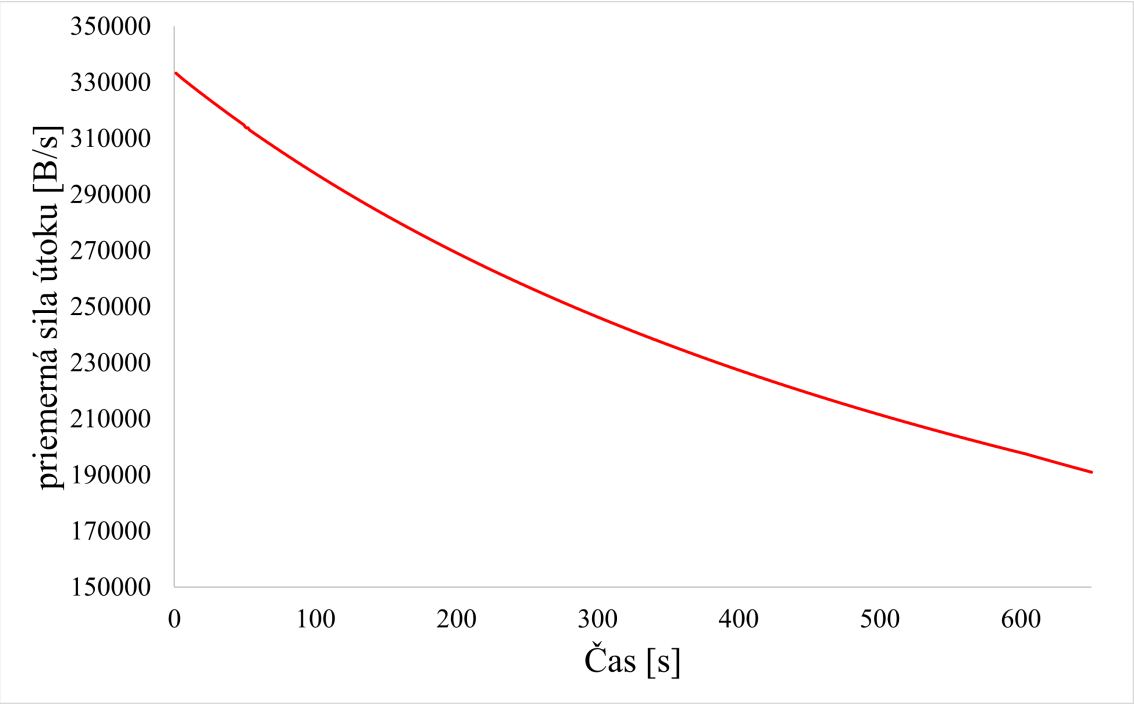
Pri testovaní mpm preforku je potrebné držať otvorených taký počet keep-alive požadaviek, ako je definovaných v maximálnom počte workerov pre požiadavky. V defualtných hodnotách je toto číslo rovné 150, ako môžeme vidieť na výpise konfiguračného súboru `mpm_prefork.conf` 4.1. V tomto druhu spracovania požiadaviek server požiadavky nesprácuváva vo vláknach, ale si ich tzv. vidličkuje (angl. forking). Každý proces serveru spracováva jednu požiadavku, pričom je potrebné, aby si udržiaval voľných 5 až 10 voľných procesov pre spracovanie nových požadaviek. Pre útok LoRDAS to znamená, že musí frontu serveru zaplniť úplne. Počet otvorených požadaviek útočníkom sa musí rovnať maximálnemu počtu možných požadaviek, ktoré server spracuje súčasne. V základom nastavení je tento parameter nastavený na hodnotu 150 požadaviek. Takto zaplnenú frontu otvorenými požiadavkami môžeme vidieť na obrázku 7.3, kde je znázornený počet požadaviek vo fronte serveru.



Obr. 7.3: Zobrazenie spracovávania požadaviek vo fronte pomocou modulu MPM prefork na servery.

Pomocou zelenej plochy na grafe 7.3 môžeme vidieť, že server je pred a po spustení 100% dostupný. Pri spustení útoku je naopak dostupnosť serveru nulová. Na server sa dohromady snažia užívatelia otvoriť 153 pripojení (modrá krivka), z čoho 150 je vytvorených pomocou LoRDAS útoku, ktoré sú znázornené aj ako obsluhované pripojenia (červená krivka). Tie postupne počas jednotiek sekúnd dokázali obsadiť všetky procesy na serveri, ktoré spracovávajú požiadavky a tieto požiadavky držia otvorené, čo znamená, že server nedokáže uvoľniť miesto žiadnemu inému užívateľovi, ktorý sa snaží o pripojenie počas celého behu LoRDAS útoku. Okrem toho sa užívatelia snažiaci sa o pripojenie nedostanú ani do stavu čakania na uvoľnenie fronty.

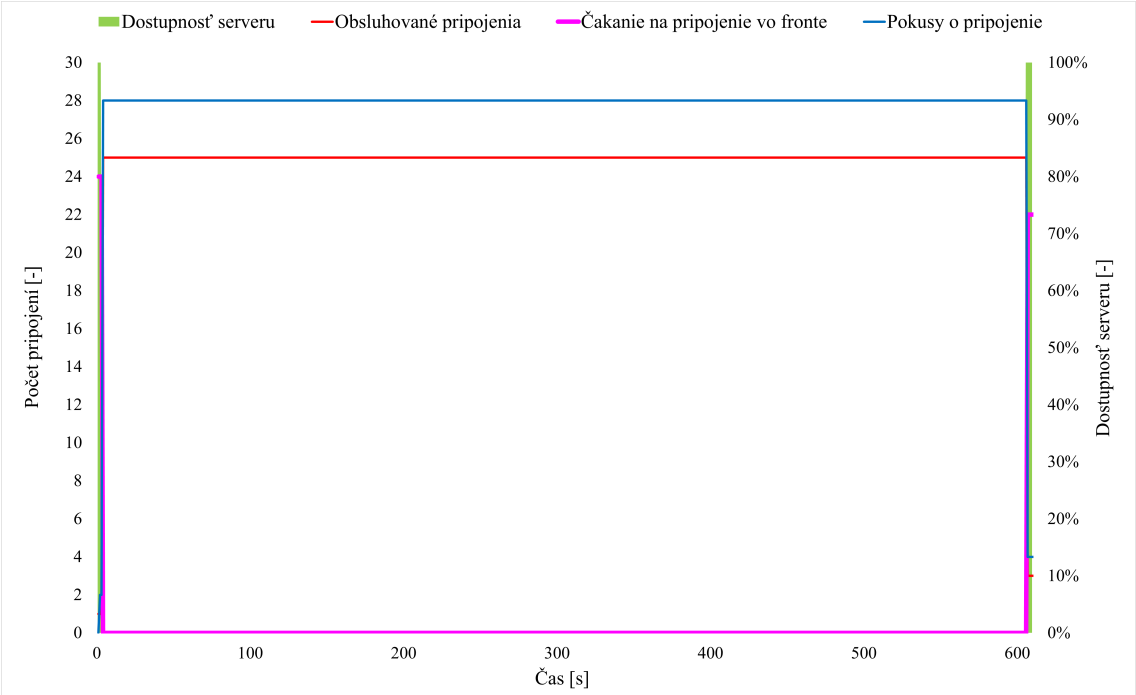
Z obrázku 7.4 môžeme vidieť koľko priemerne bajtov útok LoRDAS pošle v priebehu jednej sekundy, čo s porovnaním s bežnými záplavovými útokmi je pomerne nízke číslo. V prípade použitia distribuovaného LoRDAS útoku sú výsledky rovnaké s tým rozdielom, že počet odoslaných keep-alive požadaviek sa prerozdelí medzi botnet.



Obr. 7.4: Priemerná sila útoku pri útočení na mpm prefork.

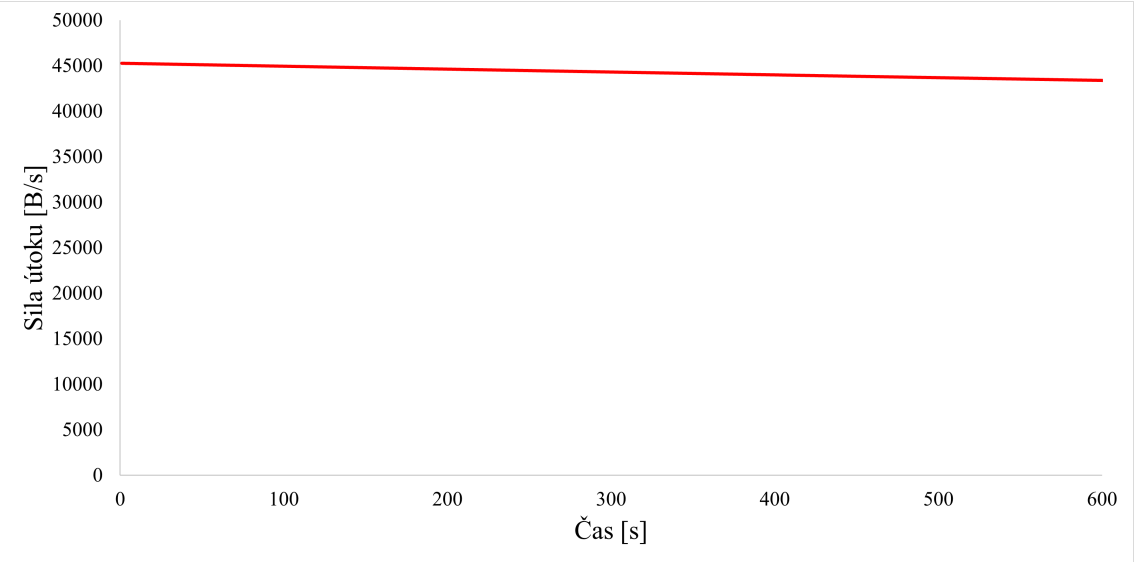
7.3 Testovanie vplyvu LoRDAS útoku na MPM worker

Modul mpm worker je zložený z možnosti hybridnej možnosti spracovávania požadaviek vo viacerých procesoch aj vláknach súčasne, čo dovoľuje spracovať väčšie množstvo požadaviek. V testovacej sieti sa použije modul worker nakonfigurovaný rovnako ako na výpise konfiguračného súboru `mpm_worker.conf` 4.2. Avšak maximálny počet týchto požadaviek sa taktiež odvíja od maximálneho množstva klientov, ktorí môžu byť obstarávaní súčasne. Toto číslo sa rovná podielu parametrov MaxRequestWorkers a parametru ThreadsPerChild. Server okrem toho dokáže spracovať len 25 požadaviek súčasne, čo v konečnom dôsledku znamená, že útočníkovi stačí otvoriť len 25 pripojení k serveru a držať ich stále otvorené. Tento prístup k obsluhovaniu požadaviek je najviac zraniteľný voči LoRDAS útoku. Priebeh otvárania spojení a obstarávania požadaviek vo fronte serveru je možné vidieť na obrázku 7.5. V prípade použitia distribuovaného LoRDAS útoku je použitie rovnaké aj počtom keep-alive požadaviek.



Obr. 7.5: Zobrazenie spracovávania požadaviek vo fronte pomocou modulu MPM worker na webovom servery.

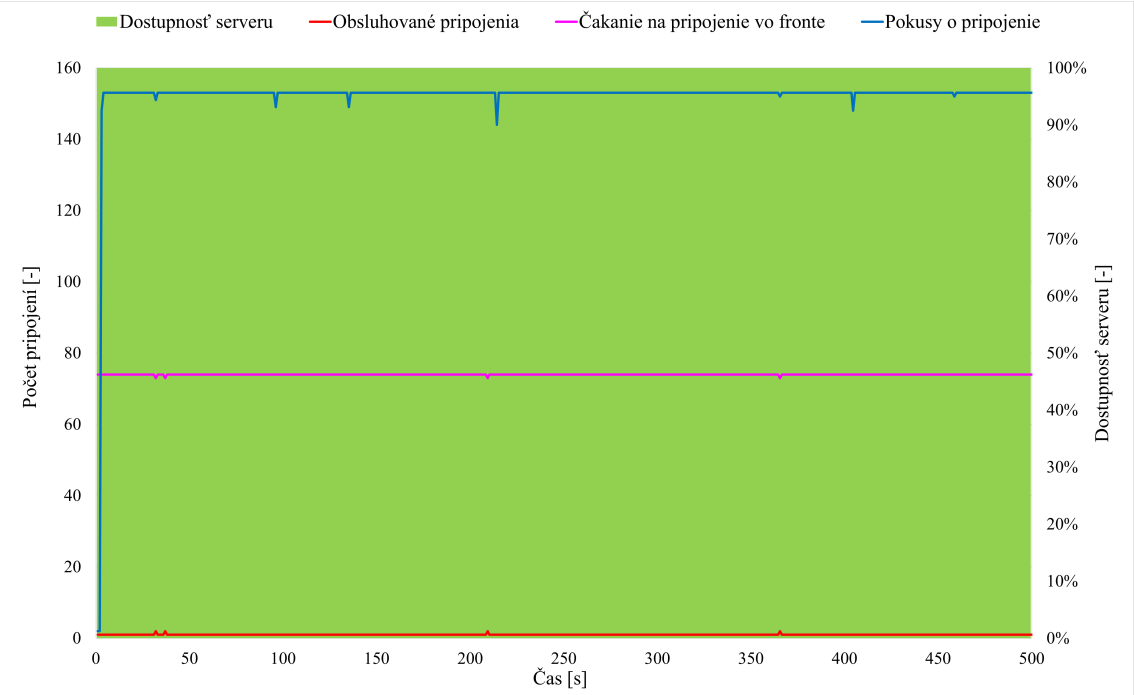
V grafe 7.5 môžeme vidieť, že na zaplnenie fronty je potreba 25 otvorených pripojení od útočníka, kde sa ďalší traja užívatelia snažia pripojiť na server, ale zasa ich nie možné posunúť ďalej do fronty, keďže fronta pre obsluhu požadaviek je počas celého trvania útoku plná a taktiež server nedostupný (zelená plošná krivka). V tomto prípade je sila útoku potrebná pre odstavenie služby serveru nižšia. Priebeh priemernej sily útoku môžeme vidieť na obrázku 7.6.



Obr. 7.6: Priemerná sila útoku pri útočení na worker.

7.4 Testovanie vplyvu LoRDAS útoku na MPM event

Ako už bolo spomínané, tak modul MPM event je od ostatných dvoch typov spracovania požadaviek rozdielny v tom, že nevytvára vlákno pre každé pripojenie zvlášť, ale prezistentné pripojenia, ako napríklad keep-alive pripojenia, spracováva všetky v jednom predurčenom vlákne, vďaka čomu mu ostávajú volné vlákna pre ostatné pripojenia, ktoré napríklad prebiehajú bez chýb alebo zlyhaní. Natavenie MPM event modulu v testovacej sieti bude rovnaké ako základné nastavenie zobrazené vo výpise konfiguračného súboru `mpm_event.conf` 4.3. Ak by sme sa odrážali od konfiguračného súboru, tak by na odopretie služby malo stačiť 150 požadaviek každých 4,85 sekúnd. Avšak priebeh obsadenia fronty pre spracovanie týchto požadaviek zobrazuje funkciu modulu event, ktorú je možné vidieť na obrázku 7.7.



Obr. 7.7: Zobrazenie spracovávania požadaviek vo fronte pomocou modulu MPM event na webovom servery.

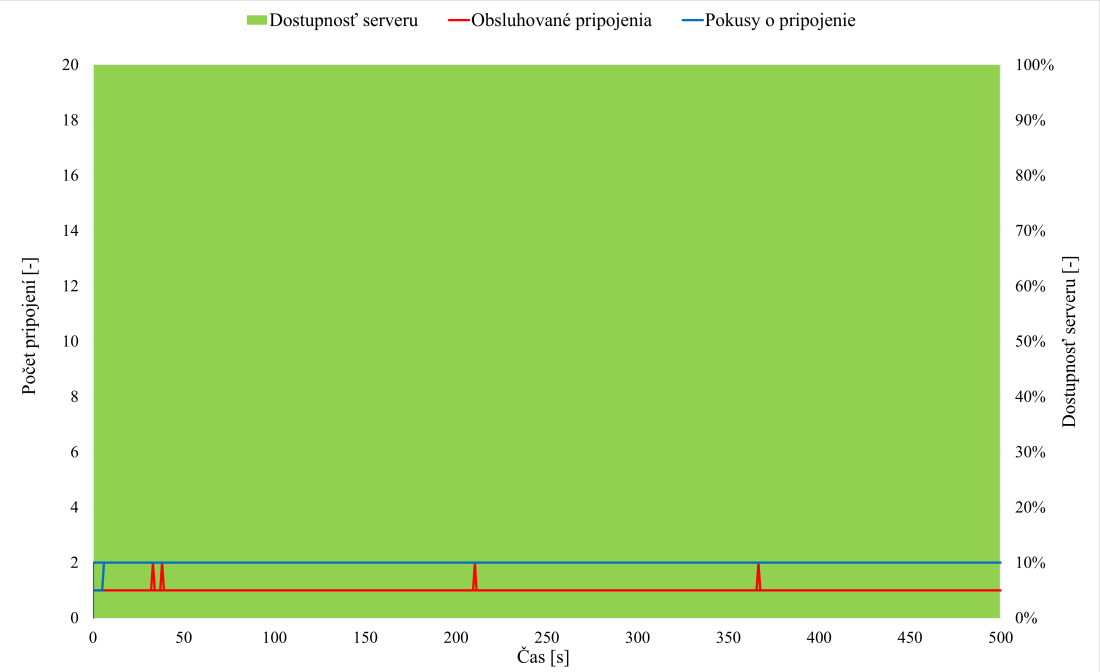
Z grafu 7.7 môžeme vidieť na základe plošného grafu dostupnosti serveru, že počas útočenia je server po celý čas 100 % prístupný. Je to z toho dôvodu, že modul MPM event všetky prichádzajúce keep-alive požiadavky obstaráva v samostatnom vlákne, ktoré na grafe môžeme vidieť ako krivku s požiadavkami vo fronte, ktoré čakajú na pripojenie. Tým pádom sú ostatné vlákna voľné pre používateľov, kde je vidieť, že event obsluhuje troch užívateľov. Okrem toho sa dokázala zaplniť aj fronta pre prezistentné pripojenia, kde pokusov o pripojenie na server je 153, z čoho 3 sú obsluhované, 75 čaká vo fronte pre perzistentné pripojenia a 75 keep-alive požadaviek nie je ani spracovaných. To však stále necháva možnosť pripojenia 147 ďalších užívateľov.

Aj v prípade použitia útoku ako DDoS útok so štyrmi útočiacimi počítačmi, vrátane Master stanice, server presunie všetky keep-alive pripojenia do vlákna perzistentných spojení aj v prípade viac ako 175 prezistentných pripojení, pretože v rámci jedného hosta je možné pripojiť sa maximálne s 75 keep-alive požiadavkami, ostatné perzistentné pripojenia nespracuje.

Toto riešenie spracovania požadaviek je vhodný obranný mechanizmus proti LoR-DAS útoku, kde sa dá zaradiť medzi zlepšenie parametrov a efektívnejšie prerozdelenie zdrojov servera, ako je aj spomenuté v 5.2.3.

7.5 Obrana a detekcia LoRDAS útoku

V prípade LoRDAS útoku je analógia k obranným a detekčným systémom podobná ako u NewShrew útoku. V prípade použitia filtračného pravidla na Surcita IDS/ISP systéme pre filtráciu keep-alive požadaviek s minimálnym počtom 24 požadaviek za 5 sekúnd dokáže toto pravidlo zachytiť LoRDAS útok, ako môžeme vidieť na priebehu zaplnenia front pre prefork a worker(pre ktoré je tento priebeh rovnaký) 7.8, kde je pre obidve pravidlá pripojený len jeden užívateľ, ktorý momentálne neposiela keep-alive požiadavky.



Obr. 7.8: Zobrazenie spracovávania požadaviek vo fronte pomocou modulu MPM prefork a MPM worker na webovom serveri s použitím IDS/IPS.

Ale taktiež v prípade inej komunikácie hlási túto komunikáciu ako LoRDAS útok, čo vo výsledku tvorí určitú mieru falošných detekcií vykonaných týmto pravidlom. Takéto pravidlo funguje v prípade LoRDAS útoku efektívnejšie ako u NewShrew útoku, ale je veľmi úzka hranica rozoznania útoku od bežnej komunikácie. Toto pravidlo je možné vidieť na výpise kódu 7.2.

```
drop tcp !$HOME_NET any -> $HOME_NET 8080 (msg:"Potential LoRDAS attack
    detected"; http.connection; content:"keep-alive"; threshold: type
   both, track by_src, count 24, seconds 5; classtype:successful-dos;
   sid:10002; rev:10;)
```

Výpis 7.2: Použité filtrovacie pravidlo v programe Suricata.

Takýto typ pravidla teda nie je efektívny, a preto klasické metódy filtrácie nie sú uspôsobené pre low-rate útoky. Preto je pre správnu detekciu LoRDAS útoku vhodné použiť metódy detekcie na základe časovej domény, kde obranný mechanizmus v určitom časovom rozmedzí bude na základe presného časového merania určovať anomáliu siete spôsobenú LoRDAS útokom, ktorá je založená na striedaní off-periódy po dobu dĺžky keep-alive požiadavky s odpočítaním RTT a on-periódy. Detekčný systém sa taktiež bude snažiť detegovať príliš dlhý a nemenný stav maximálne možných otvorených relácií jedného hosta alebo botnetu. Na základe detegovaných parametrov siete sa následne dá použiť špeciálny algoritmus na filtráciu týchto druhov útoku, a to Dressing Filter, ktorý je súčasťou kapitoly 5.2.1. Takýto typ filtrácie si určí možné hodnoty LoRDAS útoku na základe znalosti parametrov siete a taktiež so znalosťou ostatných priebehov komunikácie dokáže rozoznať LoRDAS útok od bežnej komunikácie prostredníctvom HTTP.

V prípade obranných mechanizmov sa ako najlepšie osvedčila možnosť obsluhovania požadaviek pomocou modulu MPM event, ktorý je popísaný v kapitole 7.4. Toto nastavenie je súčasťou obranného mechanizmu, ktorý sa sústreďuje na prerozdeľovanie požadaviek do jednotlivých vlákien napadnutého servera. Pre tento typ obranného mechanizmu bolo testovanie súčasťou kapitoly 7.4 a zaplnenie front servera je možné vidieť na priebehu grafu 7.7. Použitie obsluhovania požadaviek podľa modulu mpm_event je primárna metóda voči ich mitigácii, keďže sa dokáže vyhnúť zaplneniu front serveru v stopercentnej miere. Často sa však na serveroch ponecháva defaultné použite MPM modulov, a to MPM prefork, ktorý je spolu s MPM worker zraniteľný voči LoRDAS útoku.

Administrátor siete môže okrem toho znížiť účinnosť útoku znížením maximálneho počtu keep-alive požadaviek pre jedného užívateľa, ale nedosiahne úplne odvrátenie účinkov LoRDAS útoku.

Záver

V bakalárskej práci sme teoretickým úvodom získali potrebné informácie pre realizáciu generátorov NewShrew a LoRDAS útoku, ich detekciu a obranné techniky. Kvôli nefunkčnosti bežných filtračných metód bolo potrebné navrhnúť použitie nových filtračných metódy, ako Dressing Filter alebo obranné techniky ako randomizáciu RTO, ktorú NewShrew útok dokáže obísť.

Pri popísaní NewShrew útoku nás to viedlo k opísaniu troch hlavných parametrov útoku, ktorými sú burst perióda, interburst perióda a burst magnitúda, kde správnym nastavením týchto útokov docielime výrazné spomalenie TCP protokolov na serveroch, v našom prípade na webových serveroch. Tomuto nastaveniu útoku sa venovala aj implementácia skriptu NewShrew, kde sme pravidelným opakovaním striedania burst periódy a interburst periódy dosiahli realizáciu tohto útoku.

Na otestovanie sme zostavili sieť, ktorá pozostávala z niekoľkých parametrov, ktoré tvorili prostredie na realizovanie NewShrew DoS. Tieto parametre sú linka s nízkou šírkou pásma, predpoklad FIFO vyrovnávacej pamäte filtračného serveru s veľkosťou B a minimálnym RTO rovným jednej sekunde. Po nakonfigurovaní siete bol útok spustený, kde sme napriek randomizácii RTO dosiahli odstavenie služieb serveru s oneskorením v radoch stovkách sekúnd po správnej konfigurácií parametrov útoku. V rámci testovania sme zistili, aké je správne nastavenie parametrov podľa použitej siete, kde práve parametre siete v značnej miere ovplyvňujú funkčnosť NewShrew útoku. Tieto parametre boli nastavované v rozmedzí 20 ms až 900 ms pre burst periódu a 1000 ms až 4000 ms pre interburst periódu, kde sa ich zmenou zisťovali hodnoty, v ktorých má útok najvyššiu účinnosť. Pomocou správneho nastavenia parametrov NewShrew útoku sme dosiahli takmer stopercentné odstavenie serveru. Taktiež sme zistili, že bežné dostupné IDS/IPS systémy ako Suricata nie sú účinné voči NewShrew útoku a je potreba použiť filtráciu založenú na Dressing Filtrácii. Testovanie prebehlo aj na sieti s aplikovaným obranným mechanizmom minimalizácie RTO, čo znížilo oneskorenie spôsobené útokom o 98,85 %. Taktiež sme zistili, že randomizácia RTO, ktorá je obranným mechanizmom proti Shrew DoS, má účinnosť proti NewShrew útoku, len ak je zároveň aj implementovaná minimalizácia RTO v sieti serveru. NewShrew je teda DoS útok, ktorého účinnosť je vysoká v prípade presných parametrov a podmienok siete, ktoré sa v dnešnom Internete stále vyskytujú.

To predstavuje priestor pre ďalší vývoj NewShrew útoku. A taktiež ďalší vývoj obranných mechanizmov na úplne odstavenie NewShrew útoku.

V prípade útoku LoRDAS sme sa venovali spôsobu, akým dokáže zaplniť frontu servera, v ktorom sú súčasne obsluhované niekoľké požiadavky. Pomocou implementácie skriptu v programovacom jazyku Python sme dokázali súčasne otvoriť a držať

otvorené dostatočné množstvo keep-alive požadaviek, ktoré malým objemom dát dokázali odstaviť webový server. Po popise útoku a jeho implementácii sme zostavili testovaciu sieť, na ktorej sme otestovali funkčnosť LoRDAS útoku. V rámci testovania sme sa venovali testovaniu účinku LoRDAS útoku na moduly MPM prefork, worker a event. Testovaním sme zistili, že moduly prefork a worker sú zraniteľné a LoRDAS útok dokáže 100 % zneprístupniť webový server. Naopak, modul event má v sebe zabudovaný mechanizmus, ktorý bráni zahlteniu fronty obsluhovaných požadaviek a preto bol vrámci testovania tento modul označený ako vhodný obranný mechanizmus proti LoRDAS útoku.

Okrem toho sa potvrdilo, že bežná filtrácia pomocou IDS/IPS systémov ako Suricata má vysokú mieru falošných poplachov a je potreba použiť špecializovanú filtráciu Dressing Filter. Taktiež bol v rámci nadobudnutých výsledkov testovania teoreticky navrhnutý spôsob detekcie LoRDAS útokov.

LoRDAS útok teda predstavuje nebezpečenstvo a dokáže veľmi ľahko odoprieť službu, avšak na špecifických aplikačných serveroch, ktoré nepoužívajú mechanizmus obsluhovania požadaviek ako modul event. Útok je teda správnou konfiguráciou webových serverov možné odvrátiť. Avšak nie u všetkých serverov je možné daný typ modulu použiť, preto je stále vhodné vyvíjať a používať obranné a detekčné mechanizmy proti tomuto útoku.

Literatúra

[1] Blakowski, G.; Steinmetz, R.: A media synchronization survey: Reference model, specification, and case studies. *IEEE journal on selected areas in communications*, ročník 14, č. 1, 1996: s. 5–35.

[2] Alani, M.: *TCP/IP model*, kapitola OSI Model. SpringerBriefs in Computer Science, 03 2014, ISBN 978-3-319-05151-2, s. 19–50, DOI: 10.1007/978-3-319-05152-9_3.

[3] Pužmanová, R.: *Moderní komunikační sítě od A do Z*. Brno: Computer Press, druhé vydání, 2006, ISBN 80-251-1278-0.

[4] Held, G.: *Understanding Data Communications: From Fundamentals to Networking*. New York: Wiley, 2000, ISBN 0471627453.

[5] Kumar Ahuja, D. G.: Denial of service attacks – an updated perspective. *Systems Science & Control Engineering*, ročník 4, 01 2016: s. 285–294, DOI: 10.1080/21642583.2016.1241193.

[6] Yeung, A.; Dereck, F.; Wong, K. Y.: Tools for Attacking Layer 2 Network Infrastructure. *Lecture Notes in Engineering and Computer Science*, ročník 2169, 03 2008.

[7] Kumar, G.; Kumar, K.: Network security – an updated perspective. *Systems Science & Control Engineering*, ročník 2, č. 1, 2014: s. 325–334, DOI: 10.1080/21642583.2014.895969.
URL `https://doi.org/10.1080/21642583.2014.895969`

[8] Hussain, A.; Heidemann, J.; Papadopoulos, C.: A Framework for Classifying Denial of Service Attacks. In: *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, New York, NY, USA: Association for Computing Machinery, 2003, ISBN 1581137354, s. 99–110, DOI: 10.1145/863955.863968.
URL `https://doi.org/10.1145/863955.863968`

[9] Eddy, W. M.: Defenses against TCP SYN flooding attacks. *The Internet Protocol Journal*, ročník 9, č. 4, 2006: s. 2–16.

[10] Lopez, J. A.; Sun, Y.; Blair, P. B.; aj.: TCP three-way handshake: linking developmental processes with plant immunity. *Trends in Plant Science*, ročník 20, č. 4, 2015: s. 238–245, ISSN 1360-1385, DOI: https://doi.org/10.1016/j.tplants.2015.01.005.

URL https://www.sciencedirect.com/science/article/pii/S1360138515000060

[11] Obaid, H. S.; Abeed, E. H.: Dos and DDoS attacks at OSI layers. *International Journal of Multidisciplinary Research and Publications*, 2020: s. 9.

[12] Durcekova, V.; Schwartz, L.; Shahmehri, N.: Sophisticated Denial of Service Attacks Aimed at Application Layer. ISBN 978-1-4673-1178-6. In: *Sophisticated Denial of Service Attacks Aimed at Application Layer. ISBN 978-1-4673-1178-6*, 05 2012, ISBN 978-1-4673-1180-9, s. 55–60, DOI: 10.1109/ELEKTRO.2012.6225571.

[13] The Internet Society: RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. [online], 1999 [cit. 2022-05-21].
URL https://datatracker.ietf.org/doc/html/rfc2616

[14] Gamage, T. A.: Evolution of HTTP — HTTP/0.9, HTTP/1.0, HTTP/1.1, Keep-Alive, Upgrade, and HTTPS. [online], 2017, [cit. 2022-5-27].
URL https://medium.com/platform-engineer/evolution-of-http-69cfe6531ba0

[15] The Apache Software Foundation: Apache Core Features - keep alive timeout. [online], 2022 [cit. 2022-05-21].
URL https://httpd.apache.org/docs/2.4/mod/core.html#keepalivetimeout

[16] The Apache Software Foundation: Apache Core Features - Maixaml keep alive requests. [online], 2022 [cit. 2022-05-21].
URL https://httpd.apache.org/docs/2.4/mod/core.html#maxkeepaliverequests

[17] Bogdanoski, M.; Suminoski, T.; Risteski, A.: Analysis of the SYN flood DoS attack. *International Journal of Computer Network and Information Security (IJCNIS)*, ročník 5, č. 8, 2013: s. 1–11.

[18] Luo, X.; Chan, E.; Chang, R.: Detecting Pulsing Denial-of-Service Attacks with Nondeterministic Attack Intervals. *EURASIP J. Adv. Sig. Proc.*, ročník 2009, 12 2009, DOI: 10.1155/2009/256821.

[19] Douligeris, C.; Mitrokotsa, A.: DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, ročník 44, č. 5, 2004: s. 643–666, ISSN 1389-1286, DOI: https://doi.org/10.1016/j.comnet.2003.10.003.

URL https://www.sciencedirect.com/science/article/pii/S1389128603004250

[20] Yu, S.: *Distributed Denial of Service Attack and Defense*. Springer Briefs in Computer Science, New York, NY: Springer New York, 2013, ISBN 9781461494904.

[21] Sharafaldin, I.; Lashkari, A. H.; Hakak, S.; aj.: Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In: *2019 International Carnahan Conference on Security Technology (ICCST)*, IEEE, 2019, s. 1–8.

[22] Joncheray, L.: A Simple Active Attack Against TCP. In: *5th USENIX UNIX Security Symposium (USENIX Security 95)*, Salt Lake City, UT: USENIX Association, Červen 1995, s. 14.
URL https://www.usenix.org/conference/5th-usenix-unix-security-symposium/simple-active-attack-against-tcp

[23] Juniper Networks, Inc.: Attack Detection and Prevention User Guide for Security Devices. [online], 2022 [cit. 2022-05-20].
URL https://www.juniper.net/documentation/us/en/software/junos/denial-of-service/topics/topic-map/security-network-dos-attack.html

[24] Xiang, Y.; Li, K.; Zhou, W.: Low-rate DDoS attacks detection and traceback by using new information metrics. *IEEE transactions on information forensics and security*, ročník 6, č. 2, 2011: s. 426–437.

[25] Cambiaso, E.; Papaleo, G.; Aiello, M.: Taxonomy of slow DoS attacks to web applications. In: *International Conference on Security in Computer Networks and Distributed Systems*, Springer, 2012, s. 195–204.

[26] Wen, K.; Yang, J.; Zhang, B.; aj.: Survey on research and progress of low-rate denial of service attacks. *Journal of Software*, ročník 25, č. 3, 2014: s. 591–605.

[27] Zhijun, W.; Qingbo, P.; Meng, Y.: Detection method of LDoS attack based on ACK serial number step-length. *Journal on Communications*, ročník 39, č. 7, 2018: s. 139.

[28] Luo, J.; Yang, X.: The NewShrew attack: A new type of low-rate TCP-Targeted DoS attack. In: *2014 IEEE International Conference on Communications (ICC)*, 2014, s. 713–718, DOI: 10.1109/ICC.2014.6883403.

[29] Maciá-Fernández, G.; Díaz-Verdejo, J. E.; García-Teodoro, P.; aj.: LoRDAS: A low-rate DoS attack against application servers. In: *International Workshop on Critical Information Infrastructures Security*, Springer, 2007, s. 197–209.

[30] Kuzmanovic, A.; Knightly, E. W.: Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, s. 75–86.

[31] Allman, M.; Paxson, V.: On estimating end-to-end network path properties. *ACM SIGCOMM Computer Communication Review*, ročník 29, č. 4, 1999: s. 263–274.

[32] Zhang, Y.; Mao, Z. M.; Wang, J.: Low-Rate TCP-Targeted DoS Attack Disrupts Internet Routing. In: *NDSS*, Citeseer, 2007, s. 1–15.

[33] Yang, G.; Gerla, M.; Sanadidi, M.: Defense against low-rate TCP-targeted denial-of-service attacks. In: *Proceedings. ISCC 2004. Ninth International Symposium on Computers And Communications (IEEE Cat. No. 04TH8769)*, ročník 1, IEEE, 2004, s. 345–350.

[34] Cloudflare, Inc.: Cloudflare. Cloudflare - The Web Performance & Security Company. [online], 2021 [cit. 2022-05-21].
URL `https://www.cloudflare.com/learning/cdn/glossary/round-trip-time-rtt/`

[35] The Linux Fondation: ip-sysctl.txt. [online], 2021, [cit. 2022-5-21].
URL `https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt`

[36] Hung, W.-C.; Law, K. E.: Simple slow-start and a fair congestion avoidance for TCP communications. In: *2008 Canadian Conference on Electrical and Computer Engineering*, 2008, s. 001771–001774, DOI: 10.1109/CCECE.2008.4564849.

[37] Maciá-Fernández, G.; Díaz-Verdejo, J. E.; García-Teodoro, P.: Mathematical model for low-rate DoS attacks against application servers. *IEEE Transactions on Information Forensics and Security*, ročník 4, č. 3, 2009: s. 519–529.

[38] Liu, Z.; Niclausse, N.; Jalpa-Villanueva, C.: Traffic model and performance evaluation of web servers. *Performance Evaluation*, ročník 46, č. 2-3, 2001: s. 77–100.

[39] The Apache Software Foundation: Multi-processing modules (mpms). [online], 2022 [cit. 2022-05-21].
URL https://httpd.apache.org/docs/2.4/mpm.html

[40] The Apache Software Foundation: Apache MPM event. [online], 2022 [cit. 2022-05-21].
URL https://httpd.apache.org/docs/2.4/mod/event.html

[41] The Apache Software Foundation: Apache MPM prefork. [online], 2022 [cit. 2022-05-21].
URL https://httpd.apache.org/docs/2.4/mod/prefork.html

[42] The Apache Software Foundation: Apache MPM worker. [online], 2022 [cit. 2022-05-21].
URL https://httpd.apache.org/docs/2.4/mod/worker.html

[43] Microsoft Corporation: Multithreading S použitím jazyka C a prostředí win32. [online], 2022 [cit. 2022-05-21].
URL https://docs.microsoft.com/cs-cz/cpp/parallel/multithreading-with-c-and-win32?view=msvc-170

[44] Wu, Z.; Zhang, L.; Yue, M.: Low-rate DoS attacks detection based on network multifractal. *IEEE Transactions on Dependable and Secure Computing*, ročník 13, č. 5, 2015: s. 559–567.

[45] Nam, K.; Kim, K.: A study on sdn security enhancement using open source ids/ips suricata. In: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, 2018, s. 1124–1126.

[46] Zhang, J.; Hu, H.-P.; Bo, L.; aj.: Detecting LDoS attack based on ASPQ. *Journal on Communications*, ročník 33, č. 5, 2012: s. 79.

[47] Chen, K.; Liu, H.-Y.; Chen, X.-S.: Detecting LDoS attacks based on abnormal network traffic. *KSII Transactions on Internet and Information Systems (TIIS)*, ročník 6, č. 7, 2012: s. 1831–1853.

[48] Cheng, C.-M.; Kung, H.; Tan, K.-S.: Use of spectral analysis in defense against DoS attacks. In: *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, ročník 3, IEEE, 2002, s. 2143–2148.

[49] Gu, Y. H.; Wu, W. M.: DDoS Detection and Prevention Based on Joint Entropy and Conditional Entropy. In: *Key Engineering Materials*, ročník 474, Trans Tech Publ, 2011, s. 2129–2133.

[50] Jin, C.; Wang, H.; Shin, K. G.: Hop-count filtering: an effective defense against spoofed DDoS traffic. In: *Proceedings of the 10th ACM conference on Computer and communications security*, 2003, s. 30–41.

[51] Wu, Z.; Wang, M.; Yan, C.; aj.: Low-rate DoS attack flows filtering based on frequency spectral analysis. *China Communications*, ročník 14, č. 6, 2017: s. 98–112.

[52] Zhang, C.; Yin, J.; Cai, Z.; aj.: RRED: robust RED algorithm to counter low-rate denial-of-service attacks. *IEEE Communications Letters*, ročník 14, č. 5, 2010: s. 489–491.

[53] Mohan, L.; Bijesh, M.; John, J. K.: Survey of low rate denial of service (LDoS) attack on RED and its counter strategies. In: *2012 IEEE International Conference on Computational Intelligence and Computing Research*, IEEE, 2012, s. 1–7.

[54] Chang, C.-W.; Lee, S.; Lin, B.; aj.: The taming of the shrew: Mitigating low-rate TCP-targeted attack. *IEEE Transactions on Network and Service Management*, ročník 7, č. 1, 2010: s. 1–13.

[55] The Internet Society: Computing TCP's Retransmission Timer. [online], 2011 [cit. 2022-05-20].
URL `https://www.rfc-editor.org/rfc/rfc6298`

[56] Mathis, M.; Mahdavi, J.: Forward Acknowledgement: Refining TCP Congestion Control. *SIGCOMM Comput. Commun. Rev.*, ročník 26, č. 4, aug 1996: s. 281–291, ISSN 0146-4833, DOI: 10.1145/248157.248181.
URL `https://doi.org/10.1145/248157.248181`

[57] Abed, D.; Ismail, M.; Jumari, K.: A Survey on Performance of Congestion Control Mechanisms for Standard TCP Versions. *Australian Journal of Basic and Applied Sciences*, ročník 5, 12 2011: s. 1345–1352.

[58] McManus, P.: Performance tradeoffs of TCP Selective Acknowledgment. [online], 2008, [cit. 2022-5-26].
URL `http://web.archive.org/web/20111223115255/http://www.ibm.com/developerworks/linux/library/l-tcp-sack/index.html`

Zoznam symbolov a skratiek

A	Average Attack rate – Priemerná rýchlosť útoku
ACK	Acknowledgement – Potvrdzovací príznak
AQM	Active queue management – Aktívna správa front
ARP	Address resolution protocol – Protokol zisťovania MAC adresy
B	Bottleneck Buffer – Buffer filtračného serveru
C	Bottleneck link bandwidth – linka s nízkou šírkou pásma
CAM	Channel Access Method – Metóda pristupovania ku kanálu
CLI	Command line interface – Príkazový riadok
DDoS	Distributed Denial of Service – Distribuované odopretie služby
DHCP	Dynamic Host Configuration Protocol – Protokol dynamického prideľovania zariadeniam
DoS	Denial of Service – Odopretie služby
FIFO	First in first out – Postup ukladania do pamäte alebo spracovania požadaviek prvý prijatý, prvý odoslaný
FIN	Finish – Príznak ukončenia komunikácie
FTP	File transfer protocol – protokol pre prenos dát
HTTP	Hypertext transfer protocol – Hypertextový prenosový protokol
ICMP	Internet Control Message Protocol – Protokol internetových kontrolných správ
IDS	Intrastion detection system – Systém pre detekciu prienikov
IIR	Infinite impulse response – Nekonečná odpoveď impulzu
IP	Internet procol – Internetový protokol
IPS	Intrustion protection system – Sýstém pre ochranu proti prienikom
L	Burst period lenght – dĺžka burst periódy
LAN	Local area network – Lokálna sieť

L_{t0}	Dĺžka burst periódy rovná RTT
M	Počet požadaviek, ktoré server dokáže súčasne obsluhovať
MAC	Media Access Control – Kontrola prístupu k rozhraniu
minRTO	Minimal Retransmission timeout – Minimálný časový limit opätovného prenosu
MPM	Multiprocessing module – Multiprocesingový modul
N	Počet relácií otvorených LoRDAS útokom
P	Priepustnosť úzkej linky vedúcej k webovému serveru
PDU	Protocol Data Unit – Protokolová dátová jednotka
Q	occupied buffer – Obsadená vyrovnávacia pamäť filtračného serveru počas burst periódy
QoS	Quality of Service – Kvalita služieb
R	Magnitude of burst period – Veľkosť toku dát počas burst periódy
RAM	Random access memory - Pamäť s náhodným prístupom
RED	Random early detection – náhodná skorá detekcia
RTO	Retransmission timeout – Časový limit opätovného prenosu
RTSP	Real-time streaming protocol
RTT	Round trip time – Obojsmerné oneskorenie
SSH	Secure shell
SSL	Secure socket layer
SYN	Synchronization – Synchronizačný príznak
T	Interburst period — dĺžka interburst periódy
T_0	Dĺžka interburst periódy, ktorá je nižšia ako minRTO
TCP	Transmission Control Protocol – Protokol riadenia prenosu
$t_keep-alive$	Doba časovača keep-alive požiadavky
$t_off-period$	Doba trvania off-periódy

$t_on - period$	Doba trvania on-periódy
VLAN	Virtual local area network - Virtuálna lokálna sieť
UDP	USer datagram protocol – Používateľský datagramový protokol
V	Average transmission rate of victim – Priemerná prenosová rýchlosť počas burst periódy a pretečenia vyrovnávacej pamäte filtračného serveru
VLAN	Virtual local area network – Virtuálna lokálna sieť
WWW	World wide web

Zoznam príloh

A Použitie skriptu NewShrew

Požiadavky pre testovanie

V prípade ideálnych podmienok pre testovanie NewShrew útoku je v sieti potrebné nastaviť:

- Minimálna hodnota RTO rovná 1 sekunde.
- Použitie congestion control reno.
- Sieť s úzkym miestom v prechode od útočníka k webovému serveru. Napríklad využitím dvoch smerovačov, medzi ktorými je bottleneck linka.

Cielený server:

- Akýkoľvek server s HTTP službou alebo inou službou, ktorá komunikuje prostredníctvom TCP a pomocou ktorej je prenášaný väčší objem dát, ako napríklad adaptívny streaming.

Útočník:

- Zariadenie pre spustenie skriptu NewShrew, ktorý generuje škodlivú sieťovú prevádzku s odporúčaním použitia Linux-based operačného systému.
- Python interpreter verzie 3.8.X pre spustenie skriptu NewShrew, kde nie je potreba nainštalovať dodatočné moduly.
- Spojenie s cieleným serverom cez prechodný bod v sieti, ako napríklad filtračný server alebo smerovač.

Spustenie skriptu

Pre spustenie skriptu je potrebné v termináli pracovať zo zložky `newshrew`, v ktorej je uložený spustiteľný Python súbor `NewShrew.py`. V prípade zobrazenia možnosti použitia argumentov je možné v rámci skriptu použiť argument `-h` alebo `--help`, ktorý zobrazí pomocnú správu v angličtine pre ich použitie.

```
python3 NewShrew.py –h
```

NewShrew skript dokáže spracovať nasledovné argumenty (parametre) útoku:

- **IP adresa** – V rámci tohto argumentu útočník zadefinuje IP adresu serveru, na ktorý je NewShrew útok cielený. IP adresa je povinný parameter, ktorý je potreba zadefinovať pri každom spustení útoku. Argument sa definuje ako pozičný argument len definovaním IP adresy.
- **Port** – Taktiež ako IP adresu je nutné zadefinovať aj port, na ktorom beží aplikácia, ktorá je obmedzená. Číslo portu ma zadefinovanú defaultnú hodnotu portu, a to port číslo 80. Vo všeobecnosti je účinnejšie mieriť na aplikácie, ktoré prenášajú väčší objem dát, ako napríklad FTP server alebo adaptívny

web streaming, kde pomocou argumentu `-p` *číslo portu* alebo `--port` *číslo portu* sa zadefinuje bežiaca služba, na ktorú sa bude útočiť.

- **Dĺžka útoku** – V rámci útoku sa ako integer definuje celková dĺžka útoku, ktorá sa zadáva v sekundách. Je to taktiež povinný argument, ktorý sa definuje ako `-l` *dĺžka útoku* alebo `--lenght` *dĺžka útoku*.
- **Dĺžka burst periódy** – Je jeden z parametrov definujúci NewShrew útok, konkrétne dĺžku burst periódy, počas ktorej sa posiela veľké množstvo dát. Tento argument sa definuje ako float v jednotkách sekundy. Útočník môže burst periódu zadefinovať pomocou argumentu `-bl` *dĺžka burst periódy* alebo `--burst_lenght` *dĺžka periódy*.
- **Burst magnitúda** – V rámci útoku sa počas posielania burst periódy definuje aj to, akou silou sa budú posielať dáta. Burst magnitúda sa zadáva prostredníctvom argumentu `-bm` *sila útoku v* Mb/s alebo pomocou argumentu `--burst_magnitude` *sila útoku v* Mb/s. Tento argument je povinný.
- **Interburst perióda** – Medzi burst periódami je interburst perióda, počas ktorej útočník čaká na vypršanie RTO a prejdenie komunikácie do mechanizmu pomalého štartu. Táto hodnota sa definuje ako float v jednotkách sekúnd pomocou argumentu `-ib` *dĺžka interburst periódy* alebo `--interburst_period` *dĺžka interburst periódy*.
- **Logovanie** – V rámci logovania skript dokáže logovať do konzoly, ale taktiež do súboru s príponou `.log`, kde je nastavená cesta k defaultnému logovaciemu súboru, ale taktiež si môže útočník sám zvoliť niekoľko súborov, do ktorých chce taktiež zapisovať logovacie správy. V rámci súboru sa kontroluje prípona súboru, ktorá musí byť `.log`. Útočník tento súbor zadefinuje pomocou argumentu `-log` *názov logovacieho súboru*.
- **Veľkosť paketu** – Okrem týchto argumentov si môže útočník zvoliť aj možnosť, aké veľké pakety bude posielať. Pomocou parametru `-s` *veľkosť paketov v bajtoch* alebo `--size` *veľkosť paketov v bajtoch*.

Použitím týchto argumentov je možné skript spustiť s rôznymi možnosťami parametrov nasledovnými príkladmi spustenia:

```
python3 NewShrew.py –l 2000 –ib 1.4 –bm 110 –bl 0.2 10.10.40.10
```

V prvom príklade použitia je útok spustený s interburst periódou 1,4 sekundy, burst magnitúdou 110 Mb/s a burst periódou 0,2 sekundy s cieľovou IP adresou 10.10.40.10 s použitím defaultnej hodnoty portu 80 s dĺžkou útočenia 2000 sekúnd.

```
python3 NewShrew.py ––lenght 600 ––iterburst_period 1.8 ––
   burt_magnitude 40 ––burst_lenght 0.05 ––size 1100 ––port 8080
   10.10.40.10 –log log_file.log –log log_file2.log
```

V druhom príklade použitia je útok spustený pomocou dlhšej verzie použitia argumentov s dĺžkou útoku 600 sekúnd, s interburst periódou 1,8 sekundy, burst magnitúdou 40 Mb/s, burst periódou 0,05 sekundy, veľkosťou odosielaných paketov 1100 B, definovaným portom 8080 mierený na IP adresu 10.10.40.10 s logovaním do súborov `log_file.log` a `log_file2.log`.

B Použitie skriptu LoRDAS

Požiadavky pre testovanie

Cielený server:

- Aplikačný server s možnosťou spracovania viacerých požiadaviek súčasne (napr. Webový server Apache 2.4 s modulmi MPM prefork a MPM worker).
- Operačný systém serveru je odporúčané použiť ako Linux-based.

Útočník:

- Zariadenie pre spustenie skriptu LoRDAS, ktorý generuje škodlivú sieťovú prevádzku s odporúčaním použitia Linux-based operačného systému.
- Python interpreter verzie 3.8.X pre spustenie skriptu LoRDAS, kde je potreba nainštalovať dodatočné moduly.
- Spojenie s cieleným serverom priamo cez client-server spojenie alebo vedený cez prechodný bod v sieti ako napríklad filtračný server.

Inštalácia dodatočných modulov

Keďže skript spúšťaný útočník nepracuje len so základnými modulmi programovacieho jazyka Python, tak je potreba pomocou programu `pip3` nainštalovať rozširujúce balíčky. V prípade, ak dané zariadenie neobsahuje program pip3, je vo verzii 3.4 a vyššie možné pip nainštalovať pomocou príkazu nižšie, kde túto možnosť je možné spustiť aj na zariadeniach s operačným systémom Windows:

```
python3 –m ensurepip ––upgrade
```

V opačnom prípade je možné na Linux-based zariadeniach Ubuntu nainštalovať pomocou:

```
sudo apt install python3–pip
```

Následne je v prílohe C v adresári LoRDAS textový súbor `requirements.txt`, ktorý obsahuje nasledovné moduly:

- requests
- yaml
- pexpect

Tento súbor použijeme pri inštalácii dodatočných modulov pomocou príkazu:

```
pip3 install –r requirements.txt
```

Spustenie skriptu

Pre spustenie skriptu je potrebné v termináli pracovať so zložky `LoRDAS`, v ktorej je uložený spustiteľný Python súbor `LoRDAS.py`. V prípade zobrazenia možnosti

použitia argumentov je možné v rámci skriptu použiť argument `-h` alebo `--help`, ktorý zobrazí pomocnú správu v angličtine pre ich použitie. Argumenty, ktoré je možné použiť v skirpte LoRDAS sú:

- **IP adresa** – V rámci tohto argumentu útočník zadefinuje IP adresu serveru na ktorý je LoRDAS útok cielený. IP adresa je povinný parameter, ktorý je potreba zadefinovať pri každom spustení útoku. Argument sa definuje ako pozičný argument len definovaním IP adresy.
- **Port** – Taktiež ako IP adresu je nutné zadefinovať aj port, na ktorom beží služba webového serveru, ktorá je obmedzená na minimálne a maximálne číslo portu (0 - 65535). Číslo portu ma zadefinovanú defaultnú hodnotu portu, a to port číslo 80. Pomocou argumentov `-p` *číslo portu* alebo `--port` *číslo portu* sa zadefinuje port bežiacej služby, na ktorú sa bude útočiť.
- **Dĺžka útoku** – V rámci útoku sa ako integer definuje celková dĺžka útoku, ktorá sa zadáva v sekundách. Je to taktiež povinný argument, ktorý sa definuje `-l` *dĺžka útoku* alebo `--lenght` *dĺžka útoku*.
- **Off-perióda** – Je parameter, ktorý definuje dobu off-periódy v sekundách ako typ float, ktorú bude LoRDAS útok čakať medzi každou on-periódou. Argument sa definuje kľúčovým slovom `-s` *dĺžka off-periódy* alebo `--sleep` *dĺžka off-periódy*, kde v prípade nezadania parametru sa zvolí defaultná hodnota 4,9 sekundy.
- **Botnet** - Zadefinovaním tohto argumentu skript spustí útok ako DDoS s tým, že sa pripojí k botom definovaných v súbore `zombies.yaml`. Argument v sebe ukladá buď hodnotu true, kedy spustí útok ako DDoS, alebo false, kedy útok spustí ako DoS. Kľúčovým slovom je `-b` alebo `--botnet`, kde pri nepoužití tohto kľúčového slova je v základom nastavení útok spúšťaný ako DoS.
- **Počet požiadaviek** - Na odopretie služby musí útočník posielať viacero požadaviek súčasne. Túto hodnotu je v rámci LoRDAS útoku možné nastaviť pomocou argumentu requests, kde ako integer nastavíme koľko požadaviek sa otvorí počas jednej on-periódy. Tento parameter by sam mal odvíjať od rovnice 4.2. Kľúčové slovo pre tento argument je `-r` *počet požadaviek* alebo `--requests` *počet požadaviek*. V prípade nezadefinovania tohto argumentu je defaultná hodnota nastavená na číslo 25.
- **Logovanie** – Skript dokáže logovať do konzoly, ale taktiež do súboru s príponou `.log`, kde je nastavená cesta k defaultnému logovaciemu súboru, ale taktiež si môže útočník sám zvoliť niekoľko súborov, do ktorých chce taktiež zapisovať logovacie správy. V rámci súboru sa kontroluje prípona súboru, ktorá musí byť `.log`. Útočník tento súbor zadefinuje pomocou argumentu `-log` *názov logovacieho súboru*.

Použitím týchto argumentov je možné skript spustiť s rôznymi možnosťami pa-

rametrov nasledovnými príkladmi spustenia:

```
python3 LoRDAS.py –l 1800 10.10.40.10
```

V príkladom spustení vyššie je prípad, kedy je LoRDAS útok spustený s minimálnym možným množstvom zadaných argumentov, kde je útočníkom zadefinované len dĺžka útoku (1800 sekúnd) a IP adresa. Ostatné parametre sú definované defaultnými hodnotami. Off-perióda je rovná 4,9 sekundám, číslo portu je 80, počet otvorených požadaviek je 25 a logovanie je do defaultného logovacieho súboru.

```
python3 LoRDAS.py ––lenght 1800 ––port 8080 ––sleep 4.85 ––requests 150
    ––botnet –log log_file.log –log log_file2.log 10.10.40.10
```

V príklade na spustenie vyššie sme použili všetky parametre v dlhšej forme použitých kľúčových slov. Útok teda trvá 1800 sekúnd, off-perióda je nastavená na 4,85 sekundy, počas on-periódy sa drží 150 otvorených pripojení a útok je spustený ako DDoS pomocou parametru –botnet, kde je možné logovať do dvoch logovacích súborov a je mierený na IP adresu 10.10.40.10 s číslom portu 8080.

V konfiguračnom súbore zombies.yaml je potrebné definovať kľúčové slova `botnet`, ktorým zadefinujeme začiatok botnetu, kde sú následne definované zombie stanice pomocou `zombie` a čísla stanice začínajúce od 1, ako napríklad zombie1, zombie2,... Pre každý zombie je potrebné definovať jeho IP adresu, a prístupové údaje k SSH ako napríklad v prípade nižšie:

```
botnet:
  zombie1:
    ip: '10.10.10.2'
    user: 'kali'
    password: 'kali'
  zombie2:
    ip: '10.10.10.3'
    user: 'kali'
    password: 'kali'
```

Nastavenie zombie stanice

V prípade spustenia zombie stanice je potrebné mať na tejto stanici uložené Python súbory obsiahnuté v adresári `/LoRDAS/zombie/` pre vzdialené spustenie súboru `zombie.py`.

C Priložené súbory

- `/` ..... koreňový adresár priloženého archívu
  - `LoRDAS` ..... Adresár pre súbory na spustenie útoku LoRDAS
    - `logs` ..... Adresár pre defaultný logovací súbor
      - `lordas_default.log`
    - `zombie` ..... Adresár s Python súbormi pre zombie stanice
      - `pars_checkers.py`
      - `zombie.py`
    - `attack.py`
    - `bot.py`
    - `botnet.py`
    - `HTTP_requests.py`
    - `logger.py`
    - `LoRDAS.py`
    - `pars_checkers.py`
    - `requirements.txt`
    - `single_thread_attack.py`
    - `ssh_client.py`
    - `zombies.yaml`
  - `NewShrew` ..... Adresár pre súbory na spustenie útoku NewShrew
    - `logs` ..... Adresár s defaultným logovacím súborom
      - `NewShrewlogging.log`
    - `attack.py`
    - `logger.py`
    - `newshrew_attack.py`
    - `NewShrew.py`
    - `pars_checkers.py`