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

SYSTÉM PREVENCE PRŮNIKU PRO POMALÉ DOS ÚTOKY

INTRUSION PREVENTION SYSTEM FOR SLOW DOS ATTACKS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Karel Lanžhotský

AUTHOR

VEDOUCÍ PRÁCE — Ing. Marek Sikora

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Karel Lanžhotský ***ID:*** 211801

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

NÁZEV TÉMATU:

Systém prevence průniku pro pomalé DoS útoky

POKYNY PRO VYPRACOVÁNÍ:

V dnešním světě DoS útoků patří pomalé DoS útoky zcela jistě k těm nejsofistikovanějším. Díky napodobování legitimních uživatelů s pomalým Internetovým připojením jsou tyto útoky obtížně detekovatelné a také velmi účinné. Jejich rozdílný charakter datového toku od běžných DDoS útoků představuje problém pro konvenční open-source detekční systémy jako Suricata a Snort, které neumožňují definovat dostatečně podrobnou signaturu útoku pro detekování pomalých útoku v síťovém provozu.

Úkolem této diplomové práce je vytvořit vlastní software pro detekci DoS útoků pomocí detekce signatur. Software bude umožňovat přidávat sady signatur jednoduchým způsobem. Signatury budou umožňovat dostatečně podrobnou definici útoků pro pokrytí klasických záplavových a také pomalých DoS útoků. Software bude detekovat nejméně pět pomalých a pět záplavových (D)DoS útoků. Student dále vytvoří podrobný manuál a systém doplní o schopnost útoky filtrovat – vytvoří systém prevence průniku.

DOPORUČENÁ LITERATURA:

[1] TRIPATHI, Nikhil a Neminath HUBBALLI. Application Layer Denial-of-Service Attacks and Defense Mechanisms. ACM Computing Surveys. 2021, 54(4), 1-33. ISSN 0360-0300. DOI:10.1145/3448291

[2] JUREK, Michael. Detekce moderních Slow DoS útoků. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Marek Sikora

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

Vedoucí práce: Ing. Marek Sikora

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

ABSTRAKT

Tato diplomová práce se zabývá problematikou detekce a prevence záplavových a pomalých útoků na odepření služeb. Výstupem práce je software, který je schopen detekovat pět záplavových a pět pomalých útoků na základě signatur. V úvodu práce je seznámení s protokoly využívaných v síťové komunikaci, zejména těch pracujících na transportní a aplikační vrstvě. Součástí práce je úvod do problematiky útoků na odepření služeb, jejich popis a rozdíly mezi zmíněnými typy útoků. Dále je obsažen popis přístupů k detekci zmíněných útoků a způsob prevence. Nakonec se práce věnuje vytvoření, popisu a testování navrženého software, určeného k detekci těchto typů útoků.

KLÍČOVÁ SLOVA

prevence, detekce, pomalý DoS, záplavový DoS, systém prevence průniku

ABSTRACT

This master's thesis deals with the matter of detection and prevention of flood and slow denial of service attacks. Output of this thesis is software capable of detection five attacks and five slow attack based on their signatures. At the beginning of the thesis there is an introduction to protocols used in network communication, especially these working on transport and application layer. Included in this thesis in an introduction into the matters of denial of service attacks, their description and differences between mentioned attack types. After that there is description of approaches to detection and prevention techniques to the mentioned attacks. At the end thesis is dedicated to the process of making and testing of the devised software.

KEYWORDS

prevention, detection, slow DoS, flood DoS, intrusion prevention system

LANŽHOTSKÝ, Karel. *Systém prevence průniku pro pomalé DoS útoky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024, 61 s. Diplomová práce. Vedoucí práce: Ing. Marek Sikora

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Karel Lanžhotský
VUT ID autora:	211801
Typ práce:	Diplomová práce
Akademický rok:	2023/24
Téma závěrečné práce:	Systém prevence průniku pro pomalé DoS útoky

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

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Marku Sikorovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále bych rád poděkoval své rodině a přátelům za podporu ve studiu.

Obsah

Úvod **11**

1 Protokoly v síťové komunikaci **12**
1.1 Protokoly transportní vrstvy 12
1.1.1 Transmission Control Protocol – TCP 13
1.1.2 User Datagram Protocol – UDP 14
1.2 Protokoly aplikační vrstvy 15
1.2.1 Hypertext Transfer Protocol – HTTP 15

2 Útoky DoS **19**
2.1 Záplavové DoS útoky 20
2.1.1 SYN Flood 20
2.1.2 RST Flood 20
2.1.3 ICMP Flood 21
2.1.4 DNS Flood 21
2.1.5 UDP Flood 21
2.2 Pomalé DoS útoky 21
2.2.1 Slow GET - Slowloris 22
2.2.2 Slow POST - R.U.D.Y. 22
2.2.3 Slow READ 23
2.2.4 Slow NEXT 23
2.2.5 Slow DROP 23

3 Detekce a mitigace DoS útoků **24**
3.1 Detekce pomocí signatur 24
3.2 Detekce na základě anomálií 25
3.3 Příklady detekčního software 25

4 Implementace systému prevence průniku **26**
4.1 Využité nástroje a knihovny 26
4.2 Signatury pro jednotlivé typy útoků 26
4.2.1 Záplavové útoky 26
4.2.2 Pomalé útoky 27
4.3 Popis funkcionality aplikace 28
4.3.1 Vyhodnocování záplavových DoS útoků 30
4.3.2 Vyhodnocování pomalých DoS útoků 31

5 Testování systému prevence průniku **34**
5.1 Testovací prostředí 34
5.2 Příprava a spuštění aplikace na serveru 35
5.3 Testování detekce záplavových útoků 36
5.3.1 Testování SYN Flood 37
5.3.2 Testování RST Flood 38
5.3.3 Testování UDP Flood 40
5.3.4 Testování DNS Flood 41
5.3.5 Testování ICMP Flood 43
5.4 Testování detekce pomalých DoS útoků 44
5.4.1 Slowloris 46
5.4.2 Slow POST 48
5.4.3 Slow READ 50
5.4.4 Slow NEXT 52
5.4.5 Slow DROP 54

Závěr **56**

Literatura **57**

Seznam symbolů a zkratek **60**

A Obsah elektronické přílohy **61**

Seznam obrázků

1.1 Modely ISO/OSI a TCP/IP 12
1.2 Záhlaví protokolu TCP 13
1.3 Záhlaví protokolu UDP 14
2.1 Rozdíl mezi DoS a DDoS 19
2.2 Rozdíl mezi legitimním spojením a SYN Flood útokem 20
2.3 Princip segmentace paketů 22
5.1 Topologie testovacího prostředí 34
5.2 Výsledek testu SYN Flood útoku bez použití IPS 37
5.3 Výsledek testu SYN Flood útoku s použitím IPS 38
5.4 Výsledek testu RST Flood útoku bez použití IPS 39
5.5 Výsledek testu RST Flood útoku s použitím IPS 39
5.6 Výsledek testu UDP Flood útoku bez použití IPS 40
5.7 Výsledek testu UDP Flood útoku s použitím IPS 41
5.8 Výsledek testu DNS Flood útoku bez použití IPS 42
5.9 Výsledek testu DNS Flood útoku s použitím IPS 42
5.10 Výsledek testu ICMP Flood útoku bez použití IPS 43
5.11 Výsledek testu ICMP Flood útoku s použitím IPS 44
5.12 Výsledek testu Slowloris útoku bez použití IPS 46
5.13 Výsledek testu Slowloris útoku s použitím IPS 47
5.14 Výsledek testu Slow POST útoku bez použití IPS 48
5.15 Výsledek testu Slow POST útoku s použitím IPS 49
5.16 Výsledek testu Slow READ útoku bez použití IPS 50
5.17 Výsledek testu Slow READ útoku s použitím IPS 51
5.18 Výsledek testu Slow NEXT útoku bez použití IPS 52
5.19 Výsledek testu Slow NEXT útoku s použitím IPS 53
5.20 Výsledek testu Slow DROP útoku bez použití IPS 54
5.21 Výsledek testu Slow DROP útoku s použitím IPS 55

Seznam výpisů

1.1 Hlavička požadavku HTTP 17
1.2 Hlavička odpovědi HTTP 17
4.1 Obecná signatura pro záplavový útok 27
4.2 Obecná signatura pro pomalý útok 28
4.3 Funkce `sniff_packets` 29
4.4 Pravidlo pro blokování adresy v IPTables 30
4.5 Funkce pro spuštění Tcpkill 33
5.1 Instalace nástrojů a spuštění IPS 36
5.2 Test pomocí trafgen 36
5.3 Test pomocí **SlowHttpTest** 44
5.4 Test pomocí **SlowDosGen** 45
5.5 Spuštění generátoru **PySlowDos** 45
5.6 Příkaz ke spuštění testu Slow POST 48
5.7 Příkaz ke spuštění testu Slow READ 50

Úvod

Diplomová práce pojednává o problematice prevence proti pomalým útokům na odepření služeb (Denial of Service – DoS). Pomalé DoS útoky představují v moderním světě velkou hrozbu, jelikož je obtížné je rozeznat od legitimních uživatelů s pomalým připojením. Cílem práce je vytvoření vlastního software, který umožňuje detekci tohoto typu útoků na základě signatur. Signatury mají, mimo pomalé DoS útoky, pokrýt také záplavové útoky.

V první kapitole se nachází úvod do principu fungování a významnosti protokolů využívaných v síťové komunikaci v internetovém prostředí. Kapitola popisuje vysvětlení základního principu fungování vrstvových modelů. Dále se kapitola zaměřuje na bližší seznámení s protokoly transportní vrstvy a rozdílu mezi dvěma nejdůležitějšími protokoly, pracujícími na této vrstvě. Závěr kapitoly se zaměřuje na protokol HTTP (Hypertext Transfer Protocol) a popisuje strukturu požadavků a odpovědí.

Ve druhé kapitole je obsaženo seznámení s útoky na odepření služeb, včetně jejich porovnání s distribuovanou formou. Dále kapitola obsahuje popis rozdílu mezi klasickým záplavovým útokem a pomalým DoS útokem a uvádí ke každému typu jejich příklady.

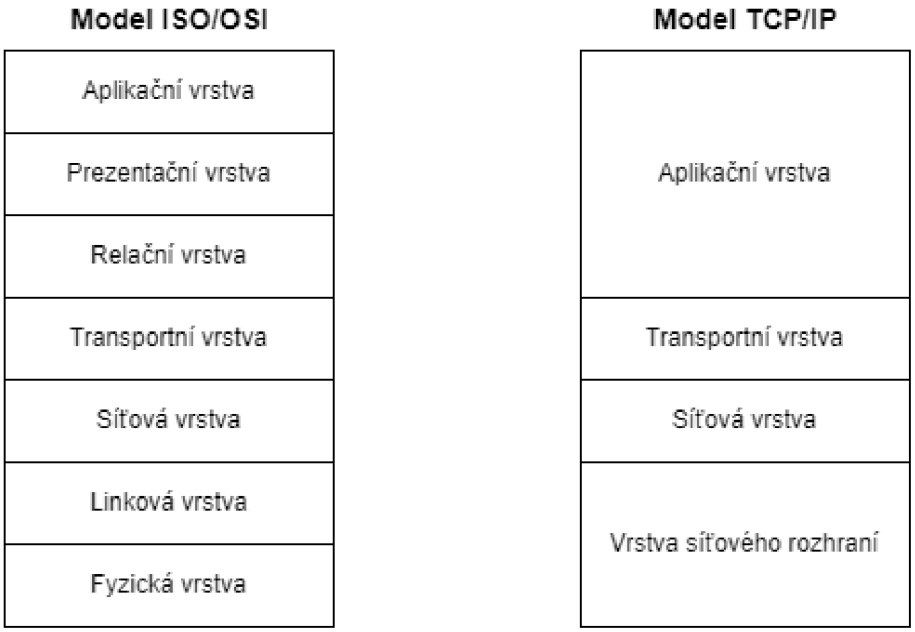
Třetí kapitola se věnuje problematice detekce a mitigace útoků na odepření služeb. Uvádí příklady možné prevence proti tomuto typu útoku. Dále popisuje principy detekce na základě signatur a anomálií v síťovém provozu a uvádí příklady detekčního software.

Ve čtvrté kapitole je popsán samotný návrh detekčního software, popisu využitých nástrojů, návrhu sledovaných parametrů a vytvoření šablon pro signatury, které jsou schopny zachytit oba typy útoků. Na závěr kapitoly je obsaženo vyhodnocování obou typů útoků a popis funkcionality tohoto navrženého software.

Pátá a poslední kapitola obsahuje testování odolnosti software proti konkrétním útokům. Součástí této kapitoly je popis a příprava testovacího prostředí a seznámení s použitými testovacími nástroji. Poté kapitola popisuje samotné nasazení navrženého software proti pěti záplavovým a pěti pomalým útokům. Součástí testování jsou také popsané testovací nástroje včetně jejich použitého nastavení. Pro účely porovnání provozu jsou v kapitole obsaženy graficky znázorněné průběhy útoků před a při použití navrženého systému prevence průniku.

1 Protokoly v síťové komunikaci

V současné době probíhá komunikace mezi zařízeními v počítačových sítích na základě definovaných modelů. Tyto modely dělí síťovou komunikaci do vrstev, kde každá vrstva zastřešuje specifickou sadu protokolů, kde každý protokol plní v síťové komunikaci určitou roli. Mezi nejdůležitější modely pro síťovou komunikaci patří model ISO/OSI (International Organization for Standardization/Open Systems Interconnection) nebo v internetovém prostředí častěji užívanější TCP/IP (Transmission Control Protocol/Internet Protocol). Porovnání těchto modelů je na obrázku 1.1. Při komunikaci dvou účastníků prochází data jednotlivými vrstvami a to při odesílání od nejvyšší k nejnižší a při přijímání naopak. To v praxi, například u modelu TCP/IP, znamená, že při odesílání bude datům aplikační vrstvy bude přidána nejdříve hlavička transportní vrstvy, poté hlavička síťové vrstvy a nakonec vrstvy síťového rozhraní.[1]



Obr. 1.1: Modely ISO/OSI a TCP/IP

Tato práce se zaměřuje zejména na analýzu transportní a aplikační vrstvy a protokolů v nich obsažených.

1.1 Protokoly transportní vrstvy

Hlavní úlohou protokolů transportní vrstvy je zajištění přenosu mezi dvěma koncovými účastníky. Přenos mezi účastníky může být dvojího charakteru: spojově ori-

entovaný (spolehlivý) přenos zajištěný protokolem TCP a spojově neorientovaný (nespolehlivý) přenos zajištěný protokolem UDP.[2]

1.1.1 Transmission Control Protocol – TCP

Jak již bylo zmíněno spolehlivost protokolu TCP je zajištěna jeho spojovou orientací. To znamená, že před samotným odesíláním dat musí být s protější účastnickou stranou navázáno spojení. Po navázání spojení je na každý odeslaný paket je ze strany příjemce vyžadováno potvrzení o úspěšném přijetí. Po dokončení veškeré výměny dat mezi účastníky komunikace musí být spojení uzavřeno. Záhlaví protokolu TCP je možné vidět na obrázku 1.2.[3]

Záhlaví protokolu TCP

Zdrojový port (16 bitů)			Cílový port (16 bitů)
Pořadové číslo (32 bitů)			
Potvrzovací číslo (32 bitů)			
Délka záhlaví (4 bity)	Rezervováno (6 bitů)	Příznaky (6 bitů)	Velikost okna (16 bitů)
Kontrolní součet (16 bitů)			Ukazatel naléhavých dat (16 bitů)
Volitelné parametry (0 - 32 bitů)			

Obr. 1.2: Záhlaví protokolu TCP

K určení konkrétního typu TCP zprávy slouží 6 příznakových bitů:

- **URG** – data jsou naléhavá
- **ACK** – úspěšné přijetí předchozí zprávy
- **PSH** – data jsou určena pro aplikační vrstvu
- **RST** – resetování spojení
- **SYN** – jedná se o synchronizační paket (žádost o navázání spojení)
- **FIN** – žádost o ukončení spojení

Příklad průběhu TCP komunikace

Protokol TCP využívá k navázání spojení takzvaného „třícestného podání ruky“ (3-way handshake). Klient při žádosti o navázání spojení odesílá synchronizační zprávu s příznakem SYN na příslušný cílový port (například port 80 využívaný webovým HTTP serverem). Server odpovídá zprávou s příznakem SYN–ACK, odeslaným z tohoto portu směrem zpět ke klientovi. Klient na tuto zprávu odpovídá potvrzením ACK, čímž je navázání spojení úspěšně dokončeno.[3]

Po sestavení TCP spojení je možné začít s odesíláním klientských žádostí o data. Klient odesílá TCP zprávu s příznakem PSH, která již navíc obsahuje i samotná data protokolu vyšší (aplikační) vrstvy. Strana serveru po úspěšném přijetí potvrzuje odesláním zprávy ACK a následně odesílá svá data aplikační vrstvy, taktéž zprávou s příznakem PSH, na kterou opět očekává potvrzení přijetí. Spolehlivost protokolu a návaznost zpráv zajišťují pořadová a potvrzovací čísla obsažená v hlavičce každé TCP zprávy. Ke každému pořadovému číslu je přiděleno číslo potvrzovací. Potvrzovací číslo předchozí přijaté TCP zprávy je využito jako pořadové číslo zprávy následující. Pokud by nastala situace, kdy nedojde k potvrzení předchozí zprávy, je tato zpráva považována za nedoručenou a je odeslána opětovně.[3]

Po dokončení odeslání všech vyžádaných dat je spojení řádně ukončeno. K tomu slouží příznakový bit FIN, odeslaný jednou z účastnických stran. Mějme situaci, kdy se klient rozhodne ukončit navázané spojení. Klient odešle zprávu s příznakem FIN, na kterou server odpovídá pomocí ACK. Server následně také odesílá zprávu FIN a v případě ACK potvrzení ze strany klienta je spojení ukončeno.[3]

Pokud však v průběhu navázaného spojení dojde k neočekávané chybě, například pokud zdrojový port již dané komunikaci nenaslouchá, může být spojení předčasně ukončeno. K předčasnému ukončení je využit příznak RST, který je obdobou příznaku FIN, avšak na rozdíl od tohoto příznaku nevyžaduje potvrzení druhé komunikující strany a spojení ukončuje okamžitě.[3]

1.1.2 User Datagram Protocol – UDP

Protokol UDP poskytuje minimalistický přístup k přenosu dat mezi účastníky komunikace. Hlavička tohoto protokolu je podstatně jednodušší než u protokolu TCP (viz. obrázek 1.3). Na rozdíl od protokolu TCP protokol UDP nenavazuje s protější stranou spojení. Tato skutečnost přenáší zodpovědnost za úspěšný přenos na konkrétní aplikace, které tohoto protokolu využívají a to včetně důsledků v případě nedoručení. Nespojový charakter tohoto protokolu se uplatňuje především v aplikacích, kde je nutné co nejrychlejší doručení dat protější účastnické straně. Mezi tyto služby patří například služby hovoru, přenosu videa nebo služby serverů DNS (Domain Name Server), kde je důležité doručení v co nejkratším možném čase.[4]

Záhlaví protokolu UDP

Zdrojový port (16 bitů)	Cílový port (16 bitů)
Délka záhlaví (16 bitů)	Kontrolní součet (16 bitů)

Obr. 1.3: Záhlaví protokolu UDP

1.2 Protokoly aplikační vrstvy

Protokoly aplikační vrstvy představují konkrétní aplikace běžící na zařízení, které slouží k přenosu konkrétních dat. Mezi služby těchto aplikací patří například služby webových, souborových či DNS serverů. V rámci této diplomové práce je stěžejní protokol HTTP.

1.2.1 Hypertext Transfer Protocol – HTTP

HTTP protokol je určen pro přenos dat mezi stranou klienta (například internetového prohlížeče) a stranou serveru (například webové stránky). Protokol je postaven na principu dotaz/odpověď a k přenosu využívá transportního protokolu TCP.[5]

Každá zpráva v komunikaci pomocí protokolu HTTP je buďto požadavkem nebo odpovědí. Úlohou serveru je naslouchat požadavkům, udělat jejich rozbor, zjistit jejich význam a odeslat zpět jednu čí více odpovědí. Klient sestavuje specifické požadavky, analyzuje přijaté odpovědi a vyhodnocuje, zda se v odpovědi nachází požadované výsledky.[5]

Požadavek HTTP

Začátek požadavku protokolu HTTP má následující podobu

```
GET / HTTP/1.1,
```

kde `GET` je metoda díky které může server rozlišit, jakou odpověď si klient žádá, `/` označuje cestu ke zdrojům serveru (URI) a `HTTP/1.1` označuje použitou verzi protokolu HTTP. Žádost může obsahovat jednu z níže popsaných metod:[5]

- GET
- HEAD
- POST
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE

Metoda GET je základní metodou protokolu HTTP. Požadavek obsahující metodu GET značí, že klient žádá si od webového serveru jeho data. Mezi tato data patří například webové stránky ve formátu HTML či obrázky.[5]

Metoda HEAD slouží ke stejnému účelu jako metoda GET s tím rozdílem, že vyžaduje pouze hlavičku odpovědi, nikoliv samotná data. Server by měl na tuto metodu

odpovědět stejnou hlavičkou jako v případě použití metody GET. Metoda je ve většinou využívána k testování validity, dostupnosti nebo modifikací hypertextových odkazů.[5]

Metoda POST značí, že klient odesílá serveru data. Mezi tato data patří vyplněný HTML formulář, příspěvek na webový blog, přidání dat k již existujícím zdrojům či nahrání souborů na server.[5]

Metoda PUT slouží k vytvoření či nahrazení zdrojů na serveru. Na rozdíl od metody POST umožňuje vytvoření nového URL na serveru. Přímým opakem je metoda DELETE, která slouží k vymazání zdrojů ze serveru.[5]

Metoda CONNECT žádá o vytvoření tunelovaného spojení s cílovým serverem. Pokud je tomuto požadavku vyhověno, data jsou obousměrně slepě odesílána dokud není tento tunel uzavřen. Metoda je určena k použití v komunikaci s proxy serverem.[5]

Metoda OPTIONS je užívána ke zjištění informací o komunikačních možnostech konkrétních zdrojů serveru. Metoda umožňuje klientovi zjistit možnosti, požadavky či schopnosti bez předem specifikované konkrétní akce. Speciální žádostí je odeslání metody OPTIONS s hvězdičkou (OPTIONS *), kde hvězdička nahrazuje konkrétní cestu ke zdrojům na serveru a vyžádá metody dostupné pro celý server. [5]

Metoda TRACE slouží k diagnostickým účelům a umožňuje klientovi zjistit validitu cesty ke zdrojům umístěným na serveru. V odpovědi na tento dotaz se nachází kopie původní žádosti a případné proxy servery, přes které musel požadavek projít.[5]

Po deklaraci typu požadavku je v hlavičce přítomno pole obsahující adresu dotazovaného cílového webového serveru

```
Host: www.priklad.com.
```

Mezi další volitelné parametry patří například pole

```
Connection: Keep-Alive,
```

identifikující o jaký typ spojení se jedná. Tento konkrétní případ by znamenal, že klient v rámci jednoho navázaného spojení plánuje odeslání více HTTP požadavků. Další variantou je

```
Connection: close,
```

značící, že v rámci jednoho spojení bude odeslán pouze jeden HTTP požadavek.[5]

Pokud by nastala situace, že jeden odesílaný HTTP požadavek (případně odpověď) musí být rozdělen na více požadavků, například při odesílání dat použitím metody POST, je součástí hlavičky také pole

```
Content-Length: 4096,
```

kde číselná hodnota uvádí celkový počet odesílaných bajtů. Dalšími poli, které se můžou nacházet v hlavičce, jsou pole `User-Agent` označující klientskou aplikaci či `Accept` specifikující přijímaný typ dat.[5]

Každé pole v hlavičce protokolu HTTP je nutné oddělit (odřádkovat) pomocí znaků `\r\n` a samotná hlavička je ukončena pomocí dvojitého odřádkování, tedy znaků `\r\n\r\n`.

Výsledný požadavek by mohl vypadat jako na výpisu 1.1.

Výpis 1.1: Hlavička požadavku HTTP

```
GET / HTTP/1.1\r\n
Host: www.priklad.com\r\n
Connection: keep-alive\r\n
User-Agent: Mozilla/5.0\r\n
Accept: text/html,application/xhtml+xml\r\n
\r\n
```

Odpověď HTTP

Odpověď serveru je strukturovaná obdobně jako požadavek. Odpověď začíná vždy následovně

`HTTP/1.1 200 OK,`

kde `HTTP/1.1` je verze protokolu a `200 OK` zobrazuje stavový kód. Stavový kód je trojmístné číslo, informující o vyřízení žádosti. Tyto kódy se dělí do pěti skupin:[5]

- 1xx – informační
- 2xx – potvrzení
- 3xx – přesměrování
- 4xx – chyba klienta
- 5xx – chyba serveru

V případě, že se jedná o potvrzovací kód `200`, je součástí odpovědi i klientem požadovaný obsah (kupříkladu HTML soubor webové stránky či obrázek). Ostatní pole hlavičky HTTP odpovědi jsou téměř shodná, jako ta obsažená v požadavku. Může se zde nacházet například pole `Date`, zobrazující datum a čas vytvoření odpovědi či pole `Server` informující o typu a verzi využívaného webového serveru.[5] Obdržená HTTP odpověď serveru může mít podobu jako na výpisu 1.2.

Výpis 1.2: Hlavička odpovědi HTTP

```
HTTP/1.1 200 OK\r\n
Date: Mon, 30 Oct 2023 11:28:11 GMT\r\n
```

```
Server: Apache/2.4.41 (Ubuntu)\r\n
Last-Modified: Mon, 16 Oct 2023 20:44:48 GMT\r\n
Content-Length: 3138\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html\r\n
\r\n
```

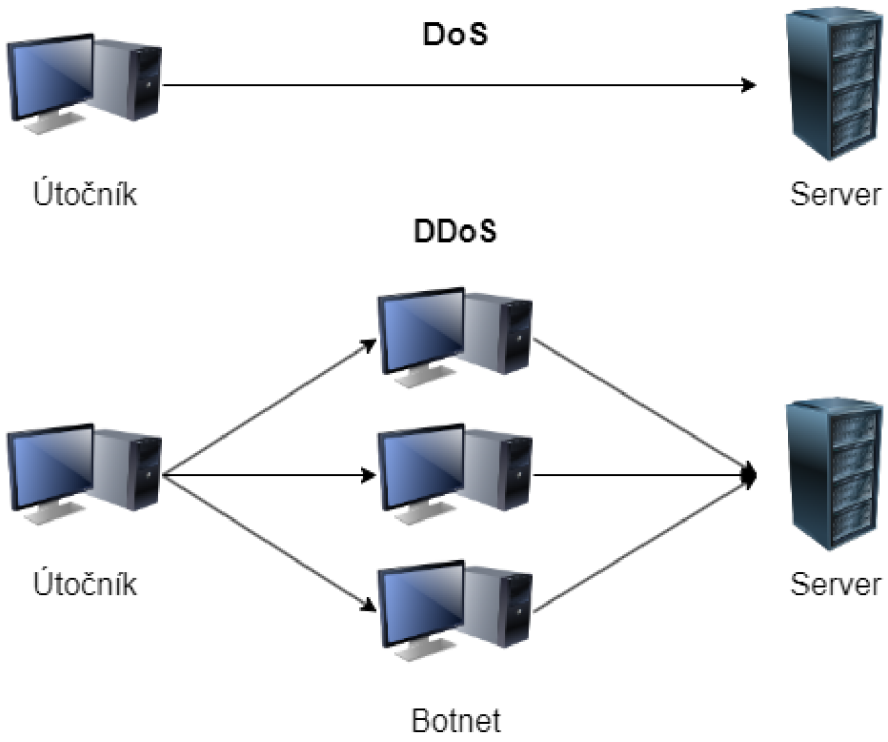
---

2 Útoky DoS

DoS je označení pro útoky zaměřující se na odepření přístupu legitimním uživatelům k určitému počítačovému systému, síti, službě nebo jinému informačnímu zdroji. Útočník tohoto jevu nejčastěji dosáhne zaplavením webového serveru, systému nebo sítě provozem, který vyčerpá systémové zdroje oběti. K tomu často dochází zneužitím slabin komunikačních protokolů.[6]

Útoky jsou nejčastěji cíleny na webové servery velkých organizací, zaměřujících se například na bankovnictví, obchodování, mediálních služeb nebo vládní organizace. Jejich cílem povětšinou není samotná ztráta nebo krádež dat, ale často slouží pro odlákání pozornosti od většího útoku. [7]

Útoky DoS nejčastěji napadají jednu nebo více vrstev referenčního síťového modelu OSI/ISO, nejčastěji je to síťová, transportní nebo aplikační vrstva. Variantou těchto útoků je DDoS (Distributed Denial of Service), což je koordinovaný DoS útok pocházející z více zařízení, které má útočník pod kontrolou bez vědomí jejich uživatelů (takzvaný botnet). Díky tomu je obtížné přesně zaměřit původ útoku.[8] Rozdíl mezi těmito útoky je možné vidět na obr. 2.1.



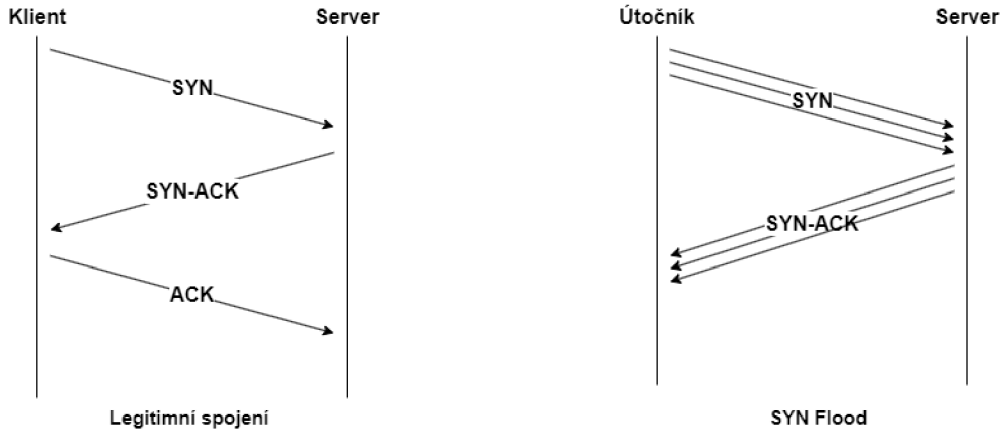
Obr. 2.1: Rozdíl mezi DoS a DDoS

2.1 Záplavové DoS útoky

Záplavové útoky patří mezi nejznámější a nejběžnější. Jejich princip, jak již z názvu vyplývá, spočívá ve vyčerpání systémových prostředků oběti odesíláním velkého množství paketů v co nejkratším časovém okamžiku. Tento typ útoků však není náročný pouze oběť, ale klade i velké nároky na útočníka. Útočník potřebuje k úspěšnému vyřazení strany oběti velkou výpočetní kapacitu k vygenerování dostatečně velkého provozu.[9]

2.1.1 SYN Flood

K provedení útoku je zneužíváno transportního protokolu TCP (Transmission Control Protocol), zejména třícestného handshake využívaného k navázání spojení. Navázání spojení by v případě legitimního požadavku probíhalo mezi stranami ve třech krocích. Klient odesílá na server zprávu s příznakem SYN. Server na tento požadavek odpoví zprávou SYN-ACK a od klienta očekává potvrzení formou zprávy ACK. V případě útoku odesílá útočník oběti velké množství paketů s příznakem SYN, na které server reaguje jako v běžném případě zprávou SYN-ACK, ale útočník již handshake nedokončí, čímž spojení zůstává polootevřené a dochází k vyčerpání systémových prostředků.[10]



Obr. 2.2: Rozdíl mezi legitimním spojením a SYN Flood útokem

2.1.2 RST Flood

Útok taktéž zneužívá transportního protokolu TCP, ale zaplavuje stanici oběti požadavky s příznakem RST. TCP paket s příznakem RST se využívá v legitimním provozu k předčasnému ukončení spojení, například v situaci, kdy jedna z komunikujících stran dlouho neodeslala potvrzující paket. Ve své podstatě se princip útoku

RST Flood dá přirovnat k útoku SYN Flood, kde útok SYN Flood útočí na navázání spojení a RST na již spojení navázaná. Při velkém počtu odeslaných RST paketů může dojít k buď k přerušení legitimních TCP spojení nebo k zahlcení komunikačního média.[11]

2.1.3 ICMP Flood

Zneužívá protokolu ICMP (Internet Control Message Protocol), který slouží ke komunikaci mezi zařízeními v síti, kde poskytuje například informace o dosažitelnosti zařízení. Útok spočívá v zasílání velkého množství „echo request“ dotazů v rámci protokolu ICMP. Na tyto žádosti oběť odpovídá zprávou „echo reply“ čímž dojde k obousměrnému zahlcením komunikačního kanálu.[12]

2.1.4 DNS Flood

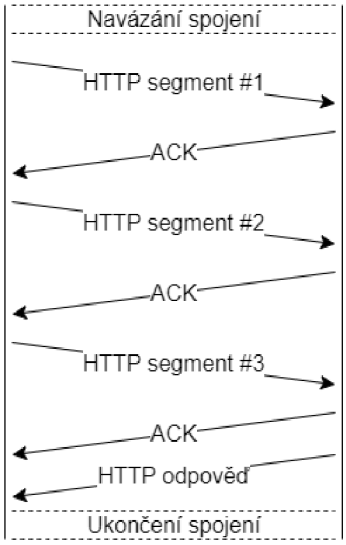
Protokol DNS slouží k překladu webových adres z jednoduše zapamatovatelné podoby (například *www.google.com*) na odpovídající IP adresu webového serveru. Tyto informace jsou uloženy na DNS serverech, ke kterým může uživatel přistupovat právě pomocí protokolu DNS. Útok DNS Flood spočívá v tom, že na DNS server je odesíláno velké množství žádostí o překlad adres, čímž může dojít k jeho zahlcení a odepření služeb legitimním uživatelům.[13]

2.1.5 UDP Flood

Útok, při kterém je oběti zasíláno velké množství paketů protokolu UDP na určené porty. Oběť se poté snaží zjistit, zda na těchto portech naslouchá nějaká aplikace. Pokud zde žádná aplikace nenaslouchá, odešle zprávu „Destination Unreachable“ pomocí protokolu ICMP, které stanici oběti zahltí a tím znemožní přistup legitimním požadavkům.[14]

2.2 Pomalé DoS útoky

Pomalé DoS útoky se na rozdíl od těch záplavových soustředí na udržování velkého počtu otevřených spojení po co nejdelší dobu tím, že odesílá data co nejpomaleji, čímž dochází k tomu, že legitimním uživatelům je odepřen přístup ke službám serveru. Toho může být dosaženo například pomocí odesílání nekompletních hlaviček paketů, kdy strana oběti udržuje spojení otevřené a čeká na dokud nedorazí zbývající data (princip na obr.2.3). Další možností je záměrně oběti odpovídat v co nejdelších časových intervalech. Toto chování simuluje uživatele s pomalým připojením a může být jednoduše zaměněno za legitimní požadavky.[15]



Obr. 2.3: Princip segmentace paketů

Tento typ útoků je v porovnání s jeho záplavovým protějškem pro útočníka poměrně nenáročný na provedení a vyžaduje podstatně menší výpočetní kapacitu.

2.2.1 Slow GET - Slowloris

Slow GET využívá k napadení serveru nekompletní GET dotaz protokolu HTTP (Hypertext Transfer Protocol), který není řádně ukončen znaky `\r\n\r\n`.

Server po obdržení takového požadavku spustí časovač a čeká na další část dotazu, která bude tyto ukončovací znaky obsahovat. Po vypršení doby čekání by bylo spojení ukončeno, avšak útočník těsně před vypršením odešle paket s takzvanou *keep-alive* zprávou, která tento časovač resetuje.

Tento proces je opakován dokud nejsou na straně serveru vyčerpány všechny prostředky a dojde k odepření služeb.[17]

2.2.2 Slow POST - R.U.D.Y.

Slow POST, jak již z názvu vyplývá, využívá k útoku požadavek HTTP POST, který slouží k odesílání dat do formulářových oken na webových stránkách.

K útoku využívá pole *Content-Length*, které je součástí hlavičky protokolu HTTP jenž určuje velikost odesílaných dat směrem k serveru. Útočník tedy nastaví vysokou hodnotu tohoto pole a server bude očekávat velké množství příchozích dat, avšak ze strany útočníka přijde pouze malé množství dat.

Strana útočníka pravidelně resetuje časovač ukončení spojení a udržuje spojení otevřené odesíláním dalších dat pomocí HTTP POST požadavků.[17]

2.2.3 Slow READ

Tento útok nepracuje pouze na aplikační vrstvě, ale využívá i protokolu TCP z vrstvy transportní. V hlavičce protokolu TCP pracuje s polem *Window-size*, které určuje kolik bajtů dat je možné odeslat bez nutnosti potvrzení přijetí. Pole slouží k regulaci toku dat v případě, že jedna strana nemá dostatečnou kapacitu na zpracování dat.

Spočívá v navázání TCP spojení se serverem, kde je v poli *window-size* velmi malá hodnota, čímž lze zaměnit útočníka za legitimního uživatele se zařízením s malou výpočetní kapacitou nebo s pomalým připojením. V praxi by to znamenalo, že i malý soubor (například obrázek) by byl potenciálně odesílán i několik dní.[16]

2.2.4 Slow NEXT

Tento typ také spadá mezi aplikačně nezávislé útoky, jelikož zneužívá časovače mezi odpovědí serveru a následujícím požadavkem od klienta. Útočník odesílá validní požadavek na server, který na tento požadavek validně odpovídá. Útočník se poté odmlčí a server čeká na žádost o další data. Tato žádost je odeslána těsně před vypršením časovače, který by ukončil spojení, čímž dochází k udržení neustále otevřeného spojení. Pokud je tímto způsobem otevřeno větší množství spojení může dojít k odepření služeb legitimním uživatelům. Tento typ útoku lze použít i na vícero protokolů aplikační vrstvy, například FTP (File Transfer Protocol), SSH (Secure Shell) čí SMTP (Simple Mail Transfer Protocol).[18]

2.2.5 Slow DROP

Při tomto útoku se útočník snaží napodobit uživatele s nespolehlivým internetovým připojením. Na server jsou odesílány validní požadavky, kde jsou následně zpracovány a server na ně odpovídá. Na straně útočníka jsou tyto odpovědi uměle zahazovány a to buď periodicky v určitých intervalech nebo jsou intervaly náhodné.[16]

Za normálních okolností by tato situace vypadala z pohledu serveru jako uživatel se špatným připojením a snažil by se ztracené pakety znovu odeslat. Tímto chováním v případě útoku dochází k udržení otevřeného spojení, což může mít při větším počtu za následek vyčerpání systémových zdrojů a odepření služeb.[16]

3 Detekce a mitigace DoS útoků

Kapitola popisuje možnosti detekce, mitigace a prevence proti útokům DoS. Její obsah vychází z článku [19].

Bránit se proti DoS útokům lze několika způsoby. Základní a nejjednodušší způsob obrany je preventivní opatření proti útokům samotným. Toho lze docílit pomocí dodržení několika zásad:

- udržovat aplikace a systémy aktualizované,
- limitace počtu příchozích spojení,
- filtrace provozu,
- využití maximální výpočetní kapacity serveru,
- využití *Honeypotů* – klamných serverů sloužících k odlákání útočníka,
- vyvažování zátěže mezi více serverů.

Další možností je použití externích programů a bezpečnostních modulů, což nemusí být dostačující, protože v některých případech dokáží pouze zmírnit sílu probíhajícího útoku. Jednou z dalších možností je použití specializovaného systému detekce průniku (Intrusion Detection System – IDS) nebo systému prevence průniku (Intrusion Prevention System – IPS). Rozdíl mezi těmito systémy je, že IDS pouze zaznamenává a upozorňuje na podezřelou aktivitu, kdežto IPS dokáže mimo tyto činnosti i aktivně zasahovat proti potenciálnímu útoku podle předefinovaných pravidel.[20]

Tyto systémy se dále dělí na *network-based* a *host-based*. První zmíněné se implementují na strategická místa v počítačové síti, typicky hned za firewall, a monitorují příchozí i odchozí provoz v dané síti. Druhý typ je implementován na jednotlivých koncových bodech sítě jako je například stanice nebo server.[20]

K detekci lze využít dva přístupy, první je pomocí signatur a druhý na základě anomálií v provozu. Každý z těchto přístupů přináší své výhody a nevýhody, proto je v ideálním případě nejvhodnější použít jejich kombinaci.[20]

3.1 Detekce pomocí signatur

Signatury popisují podobu nebezpečného provozu. Mohou se skládat například ze specifických dat obsažených v paketu, opakujících se vzorů nebo řetězce příkazů. Systém využívající tento způsob detekce zachytává příchozí data, porovnává jej s těmito předefinovanými signaturami a hledá mezi nimi shody.[20]

Výhodou je účinné použití na známé útoky a slabiny, popsané například ve veřejných databázích. Z tohoto důvodu je jejich použití neefektivní vůči novým a dosud neznámým útokům.[20]

3.2 Detekce na základě anomálií

Detekce na základě anomálií pracuje na základě vzorů a chování, které jsou předem definovány jako legitimní. Získání takových dat může být dosaženo pomocí statistik, strojového učení nebo obdobných algoritmů k určených k analýze. Po vytvoření modelu je využit k porovnávání s aktuálními daty a detekci odchylek. V případě zaznamenání odlišností od legitimní aktivity je chování považováno za anomálii.[20]

Výhodou tohoto přístupu detekce je schopnost reagovat na neznámé hrozby. Nicméně pokud je legitimní model složitý nebo se často mění, může být náchylný na generování falešných poplachů.[20]

3.3 Příklady detekčního software

Mezi detekční software patří například open-source systémy Suricata nebo Snort. Tento software je vhodný pro ochranu proti záplavovým DoS útokům, zejména kvůli tomu, že tento typ útoku lze jednoduše definovat pomocí určitých signatur. Tyto systémy je obtížné aplikovat na detekci pomalých DoS útoků, protože neposkytují možnost podrobné definice jejich signatur.[21]

4 Implementace systému prevence průniku

Kapitola popisuje vytvoření systému prevence průniku, který je napsán v programovacím jazyce Python. Tento systém je určen pro použití na serveru využívající Linuxovou distribuci. Ke své funkci využívá detekci na základě předem definovaných signatur.

4.1 Využité nástroje a knihovny

IPTables

IPTables je nástroj určený ke správě síťové komunikace na unixových operačních systémech. Umožňuje filtraci příchozích, průchozích a odchozích paketů pomocí příslušných tabulek.[22]

Scapy

Scapy je knihovna vytvořená pro programovací jazyk Python, která umožňuje odesílání, odposlouchávání, rozebírání a padělání paketů v počítačové síti. Odposlouchávání paketů znamená, že sleduje konkrétní rozhraní a monitoruje komunikaci, která na něm probíhá. Dále poskytuje možnost rozložení paketu podle jednotlivých protokolů.[23]

Tcpkill

Tcpkill je nástroj určený k přerušení TCP spojení pomocí vytváření a odesílání paketu s příznakem RST. Lze jej použít na uzavírání spojení z určité adresy, portu nebo celého rozhraní. Nástroj je součástí balíčku nástrojů Dsniff.[24]

4.2 Signatury pro jednotlivé typy útoků

Nejdříve bylo důležité připravit sledované parametry pro jednotlivé útoky. Toho bylo dosaženo vytvořením souboru JSON (JavaScript Object Notation), obsahujícím signatury pro oba typy útoků. V souboru se nachází dvě pole, jedno s názvem *FloodDoS* a druhé *SlowDoS*. Jak již z názvů vyplývá, první zmíněné obsahuje signatury pro záplavové útoky a druhé pro pomalé DoS útoky.

4.2.1 Záplavové útoky

Pro záplavové útoky byla vytvořena signatura zobrazená na výpisu 4.1. Parametr `attack` obsahuje název pro detekovaný útok. Parametr `request` definuje hlavičku

protokolu užívaného záplavovým útokem. Obsahuje podparametry pro možnost definice názvu protokolu (`protocol`) a případného typu příznaku (`request-type`). Jelikož jsou záplavové útoky založeny na odesílání co největšího počtu paketů za co pokud nejkratší čas, je další skupina parametrů `volume` zaměřena na definici maximálního počtu povolených otevřených spojení (`connections`) za daný časový interval v sekundách (`per-sec`).

Výpis 4.1: Obecná signatura pro záplavový útok

```
"FloodDoS":
    [{
      "attack": string,
      "request": {
        "protocol": "{PROTOKOL}",
        "request-type": "{PRIZNAK}"
      },
      "volume": {
        "connections": integer,
        "per-address": integer,
        "per-sec": integer
      }
    }]
```

4.2.2 Pomalé útoky

Na výpisu 4.2 je možné vidět specifikovanou signaturu pro pomalý DoS útok. Parametr `attack` je určen pro název útoku. Následuje parametr `request` definující hlavičku přijímané HTTP žádosti a obsahuje několik podparametrů. Prvním z nich je `method`, který slouží pro definici metody HTTP dotazu, jako je například metoda GET. Podparametr `connection` určuje zda má být součástí paketu instrukce *keep-alive*, která umožňuje jednomu TCP spojení sestavit perzistentní spojení. Tím může odesílat více HTTP požadavků v rámci jednoho spojení. Předposledním podparametrem je `content-length`, díky kterému je možné specifikovat, zda má být součástí hlavičky stejnojmenné pole, například pro použití s metodou POST.

Parametr `next-request` slouží k definici pravidel pro přijímaní neúplných HTTP požadavků nebo jiných dodatečných paketů pro jednu IP adresu. Tento parametr se skládá ze tří podparametrů. Prvním je `window-size` vyjadřující velikost okna následujícího paketu. Dalším z nich je `max-time-between` definující dovolený časový interval mezi přijatými dodatečnými pakety v rámci spojení. Posledním je `max-number`, který definuje maximální počet dodatečných paketů z jedné IP adresy.

Na závěr je definován parametr `properties` zastřešující pravidla pro parametry spojení. Mezi tyto parametry patří například omezení na maximálního počtu otevřených podezřelých spojení pro jednu IP adresu `opened-connections` za určitý počet sekund (`per-sec`) nebo zda má být sledováno zahazování paketů v průběhu navázaného spojení (`drops`).

Výpis 4.2: Obecná signatura pro pomalý útok

```
"SlowDoS":
    [{
      "attack": string,
      "request": {
          "method": "{METODA},
          "connection": string,
          "content-length": boolean,
          "terminator": boolean
      },
      "next-request":{
          "window-size": integer,
          "max-time-between": integer,
          "max-number": integer
      },
      "properties": {
          "opened-connections": integer,
          "per-sec": integer,
          "drops": boolean
      }
    }]
```

4.3 Popis funkcionality aplikace

Po určení signatur následovalo vytvoření samotné aplikace. Jak již bylo zmíněno, aplikace pracuje na základě signatur, které načítá ze souboru typu JSON. Po spuštění aplikace je zavolána funkce `sniff_packets` (viz výpis 4.3). Funkce přijímá volitelný parametr `iface`, který umožňuje definici konkrétního síťového rozhraní. Toto rozhraní je možné definovat při spouštění aplikace použitím argumentu `-i {název rozhraní}`. Dále je umožněno uživateli specifikovat vlastní soubor se signaturami útoků. Toho lze dosáhnout použitím argumentu `-f {název souboru}` a to taktéž při spouštění aplikace v terminálu. Pokud není tento soubor definován, je využit výchozí soubor s názvem `signatures.json`.

V tomto souboru jsou ve výchozím stavu definovány signatury pro deset útoků:

- Záplavové útoky:
  - DNS Flood
  - ICMP Flood
  - UDP Flood
  - SYN Flood
  - RST Flood
- Pomalé útoky:
  - Slowloris
  - Slow POST
  - Slow READ
  - Slow NEXT
  - Slow DROP

Funkce `sniff_packets` zastřešuje volání funkce pro vymazání záznamů v IPTables (`reset_iptables()`). Dále jsou načteny signatury ze souboru JSON pomocí funkce `load_signatures`. Při volání této funkce je pro každý definovaný útok zvlášť vytvořen objekt příslušné třídy, do kterého jsou vloženy parametry ze signatur. Tento objekt je poté vložen do pole objektů podle typu útoku pod jeho názvem.

Po rozdělení signatur pro útoky je spuštěna funkce z knihovny Scapy pro odposlech paketů `sniff` s využitím buď předem definovaného rozhraní nebo za použití výchozího rozhraní pro síťovou komunikaci. Tato funkce bude při každém zachycení paketu volat funkci `process_packet`.

Výpis 4.3: Funkce `sniff_packets`

```
def sniff_packets(iface=None, filename=None):
    """
    Sniff packets on specified interface
    and load signatures from specific file.
    If any parameter is None, default values are used
    """
    reset_iptables()

    if filename:
        load_signatures(filename)
    else:
        load_signatures('signatures.json')

    if iface:
```

```
        logging.info(f"Listening␣on␣interface␣{iface}")
        sniff(prn=process_packet, iface=iface,
        store=False)
    else:
        logging.info(f"Listening␣on␣default␣interface")
        sniff(prn=process_packet, store=False)
```

Funkce `process_packet` poté analyzuje obsah každého přijatého paketu. Tato funkce poté zaznamenává pakety obsahující IP vrstvu, ze které zjistí zdrojovou IP adresu a transportní protokol.

4.3.1 Vyhodnocování záplavových DoS útoků

Vyhodnocování záplavových DoS útoků je poměrně jednoduché a přímočaré. Aplikace se při vyhodnocování zaměřuje na pakety obsahující hlavičku transportního protokolu, jsou sledovány následující parametry:

- název protokolu
- příznak paketu protokolu (v případě protokolu TCP)

Aplikace dále rozhoduje zda se jedná o protokoly se spojově orientovaným či neorientovaným charakterem. V případě spojově neorientovaného přenosu (tudíž jakéhokoli protokolu mimo protokolu TCP) je přihlíženo pouze na název protokolu. Pokud se název protokolu shoduje s názvem protokolu definovaným v signatuře je zaznamenán čas prvního příchozího paketu a zdrojová IP adresa je dále sledována. Při naplnění podmínky o maximálním počtu přijatých žádostí je vypočítán čas dosažení této skutečnosti a v případě naplnění definice signatury (parametr `volume`) je zdrojová IP adresa zablokována. Pokud se jedná o spojově orientovaný přenos využívající protokolu TCP je zjištěn typ příchozí žádosti. U TCP komunikace obsahující příznak RST probíhá vyhodnocování podobně jako u výše zmíněného spojově neorientovaného provozu. V případě, že se jedná o žádost s příznakem SYN je jako podezřelé označeno každé nově otevřené spojení, na které ze strany odesílatele nepřijde paket s příznakem ACK, která by uzavřela *TCP handshake*. Při zaznamenání takového chování ze strany odesílatele je zaznamenán čas příchozí zprávy. Pokud odesílatel otevře definovaný počet spojení v signatuře bez toho, aniž by dokončil zmíněný *handshake*, je vypočítán čas za kterého bylo této skutečnosti dosaženo. Jestliže byl časový úsek kratší než čas definovaný v parametru `volume`, je zavolána funkce na blokování IP adresy, která vytvoří záznam *DROP* v IPTables a veškerá komunikace z této IP adresy je zahazována. Toto pravidlo je zobrazeno na výpisu 4.4, kde parametr **-A** určuje, do které filtrační tabulky bude záznam přidán, parametr **-s** označuje zdrojovou adresu a **-j** akci, která má být provedena.

Výpis 4.4: Pravidlo pro blokování adresy v IPTables

```
iptables -A FORWARD -s {IP ADRESA} -j DROP
```

Při dokončení *TCP handshake* je vyloučena možnost záplavového útoku a aplikace pokračuje v kontrole, zda se nejedná o pomalý DoS útok.

4.3.2 Vyhodnocování pomalých DoS útoků

Vyhodnocování pomalých DoS útoků je na rozdíl od jejich záplavového protějšku komplexnější a v rámci implementace probíhá ve třech fázích. První fáze je zaměřená na identifikaci a klasifikaci potenciálních útoků podle údajů v hlavičce HTTP požadavku nebo pomocí specifického chování navázaného TCP spojení. Druhá fáze je zaměřena na bližší sledování spojení, která byla v první fázi označena jako podezřelá. Pokud by se stalo, že spojení vykazuje známky útoku (respektive naplní podmínky, které byly nastaveny v signatuře), přechází aplikace do třetí fáze. V této fázi probíhá blokace zdroje útoku a přerušení veškerých navázaných spojení. Bližší popis jednotlivých fází se nachází níže.

Fáze 1 – Klasifikace útoků

Aplikace po přijetí paketu obsahující HTTP požadavek nejdříve vyhodnocuje podobnosti hlavičky samotné HTTP žádosti s hlavičkou definovanou v parametru `request` signatury. Tento přístup je vhodný pokud jde o útok vyznačující se specifickými daty v hlavičce. Mezi tyto útoky se řadí například Slowloris nebo Slow POST. Útok Slowloris je specifický tím, že jeho požadavek postrádá ukončovací znak. Útok Slow POST je na druhou stranu jako jediný obsahuje ve své hlavičce pole `Content-Length`, značící odesílání dat směrem k serveru. Pro tento útok je navíc zaznamenána tato avizovaná velikost.

Pokud jsou zaznamenané informace ze zachycené žádosti shodné s těmi, které jsou definovány v signatuře, je spojení vyhodnoceno jako podezřelé a zaznamenány jsou následující parametry:

- zdrojová IP adresa
- zdrojový port
- cílový port
- velikost odesílaných dat (v případě podezření na Slow POST)
- název podezřelého útoku

Jestliže nastane situace, že k detekci a klasifikaci útoku nestačí specifikace hlavičky HTTP požadavku, což může platit například pro útoky Slow READ a Slow DROP, aplikace dále kontroluje chování probíhajících spojení na transportní vrstvě.

Pokud by nastala situace, že se v síťovém provozu objeví TCP paket, který obsahuje nulovou velikost okna (pole window-size je rovno 0) a příchozí paket je směřován na port 80 (tudíž na port využívaný webovým serverem) je spojení označeno jako podezřelé na útok Slow READ.

Aplikace dále kontroluje, zda v provozu nedochází k zahazování ze strany odesílatele či k duplicitnímu odesílání paketů ze strany serveru. Takovéto chování by potenciálně mohlo znamenat, že navázané spojení je ve skutečnosti útok Slow DROP.

Fáze 2 – Sledování spojení

Po klasifikaci potenciálních útoků přichází na řadu bližší sledování podezřelých spojení. Tato fáze se zaměřuje na rozbor především paketů transportní vrstvy. Pakety jsou rozlišovány podle jejich příznaků, zejména PUSH–ACK, ACK a případně FIN–ACK či RST–ACK.

V případě útoku Slowloris je kontrolována každý PUSH-ACK paket, odeslaný ze sledovaného zdrojového portu, který ve svých datech nenese již zmiňovaný ukončovací znak (tedy dvojité odřádkování `\r\n\r\n`). Pro každý paket, který nenese tento ukončovací znak, je navýšeno interní počítadlo. Zároveň je zaznamenáván čas mezi těmito pakety. Pokud spojení dosáhne či překročí čas (`max-time-between`) nebo počet paketů bez ukončovacího znaku(`max-number`) definované v `next_request` signatury, je spojení vyhodnoceno jako nebezpečné a je zablokováno.

Postup pro útok Slow POST je obdobný jako v případě Slowloris. Opět je kontrolován každý příchozí paket s příznakem PUSH-ACK. S přijatým paketem je přičtena aktuální velikost odeslaných dat k té v databázi a inkrementováno počítadlo a zaznamenán čas od předchozího přijatého paketu. Pokud spojení nedoručí celá data a neuzavře spojení před dosažením maximálního počtu povolených doplňujících paketů nebo je překročen maximální povolený čas mezi pakety, je spojení vyhodnoceno nebezpečným a IP adresa je blokována.

Při podezření na útok Slow NEXT je sledována primárně prodleva mezi aktuálním a předchozím přijatým paketem směřovaným na aplikační vrstvu. S každým dalším paketem je navýšeno počítadlo a zaznamenána již zmíněná prodleva. V případě, že je čas mezi požadavky delší než ten definovaný nebo počet paketů je vyšší než definovaný, je spojení označeno za nebezpečné a zdrojová adresa je blokována.

Pro útok Slow READ jsou sledovány pakety s příznakem ACK, jelikož těmi dává serveru najevo, že délka jeho TCP okna je nulová. Pro vyhodnocení útoku je, jako v předešlém případě u útoku Slow NEXT, důležitá prodleva mezi aktuálním a předešlým přijatým paketem. Pro tento případ nejsou však sledována samotná data pro aplikační vrstvu, ale změny velikosti zmíněného TCP okna. Pokud počet těchto změn překročí povolený počet `max-number`, nebo prodleva mezi příchozími

pakety překročí povolený čas `max-time-between`, je IP adresa blokována.

U útoku Slow DROP probíhá sledování jiným způsobem než v předešlých případech, jelikož jako jediný udržuje spojení otevřené tím, že data nepřijímá a záměrně zahazuje. Pro tento útok je tedy spojení sledováno spíše z pohledu serveru. Každý unikátní serverem odeslaný paket je zaznamenán do databáze pomocí jeho pořadového a potvrzovacího čísla. V případě kdy je zaznamenán duplicitní paket, tedy paket jehož pořadové a potvrzovací číslo je již obsaženo v databázi, je navýšeno počítadlo. Spolu s každým pokusem o doručení paketu je vypočítán čas od poslední reakce strany klienta. V případě, že počet neúspěšných pokusů o doručení nebo čas od poslední reakce klienta dosáhne stejné nebo vyšší hodnoty, než je definováno v polích `max-number` nebo `max-time-between` signatury, je spojení vyhodnoceno jako nebezpečné a zablokováno.

Jako poslední situace může nastat, že jakékoliv sledované podezřelé spojení je ukončeno, a to buď řádně (pomocí kombinace příznaků FIN–ACK) či předčasně (kombinací RST–ACK). V takovémto případě je sledování ukončeno a je odebráno z databáze.

Fáze 3 – Blokování a uzavření spojení

Po vyhodnocení spojení jako nebezpečného přichází na řadu blokace jeho zdrojové IP adresy. Blokace je, jako v případě záplavových útoků, přidáním pravidla do IPTables (viz pravidlo z výpisu 4.4).

Samotné zablokování adresy by nemuselo k úplnému zastavení útoku stačit. Proto je na případné zbylé otevřené spojení odeslán TCP paket s příznakem RST, kterým toto spojení uzavře. Tato akce probíhá pomocí spuštění nástroje **Tcpkill**. Nástroj při spuštění sleduje TCP pakety obsahující tuto adresu a na zachycené pakety odesílá RST zprávu. Funkce pro spuštění je zobrazena na výpisu 4.5.

Výpis 4.5: Funkce pro spuštění Tcpkill

```
def tcp_kill(dst):
    proc = subprocess.Popen(["tcpkill", "host", dst],
                            stderr=subprocess.PIPE)
    p = proc.pid
    time.sleep(10)
    os.kill(p,signal.SIGINT)
```

Funkce přijímá jako parametr cílovou adresu, která zastává zdrojovou adresu potenciálního útočníka.

5 Testování systému prevence průniku

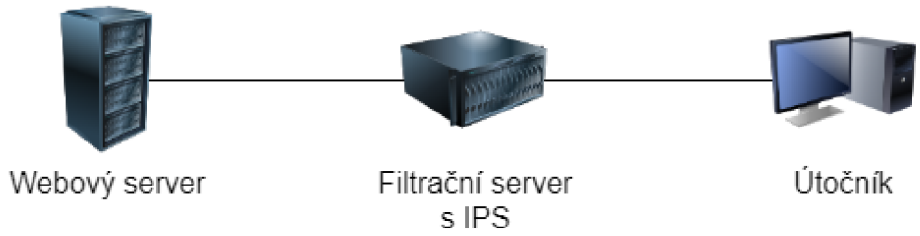
Kapitola popisuje testování vytvořeného systému prevence průniku proti záplavovým útokům SYN Flood, RST Flood, UDP Flood, DNS Flood a ICMP Flood a pomalým útokům na odepření služeb Slowloris, Slow POST, Slow READ, Slow NEXT a Slow DROP.

5.1 Testovací prostředí

Pro otestování funkčnosti bylo vytvořeno testovací prostředí skládající se ze tří virtuálních strojů v hostitelském stroji s následujícími parametry:

- OS Windows 10 64-bit
- CPU AMD Ryzen 5 5600
- RAM 16 GB
- virtualizační software VMware Workstation 17 Player

Topologie testovacího prostředí je zobrazena na obrázku 5.1.



Obr. 5.1: Topologie testovacího prostředí

Prvním z nich byl stroj běžící nad operačním systémem Ubuntu Server ve verzi 20.04. Na tomto stroji byl spuštěn webový server Apache ve verzi 2.4. Tento server obsahoval výchozí nastavení a testovací webovou stránku včetně jednoduchého formuláře. Pro sledování síťové komunikace byl na tomto stroji využit nástroj Wireshark. Specifikace stroje s webovým serverem:

- 2 jádra CPU
- 2 GB RAM
- IP 192.168.2.129.

Druhý virtuální stroj, běžící na operačním systému Ubuntu ve verzi 22.04, zastával funkci filtračního serveru a obsahoval implementaci navrženého systému prevence průniku. Tento stroj má dvě síťová rozhraní. Jedno rozhraní v síti **192.168.3.0/24**, ve které se nacházel stroj útočníka. Druhé rozhraní je určeno pro síť **192.168.2.0/24**, ve které se nacházel webový server. Směrování v síti bylo pro účely testování uzpůsobeno tak, že veškerý provoz směřovaný k webovému serveru musel projít přes tento virtuální stroj. Parametry stroje s filtračním serverem:

- 2 jádra CPU
- 4 GB RAM
- IP 192.168.2.128, 192.168.3.128.

Třetí virtuální stroj měl nainstalovaný operační systém Kali Linux ve verzi 2023.3 a představoval roli útočníka. Jeho hardwarové specifikace byly následující:

- 4 jádra CPU
- 2 GB RAM
- IP 192.168.3.129 (192.168.3.0 až 192.168.3.20 pro záplavové útoky)

Ke generování útoků bylo využito nástrojů dvou nástrojů již předinstalovaných na zmíněném operačním systému Kali Linux. Tyto nástroje jsou:

- **trafgen** pro generování záplavových útoků,
- **SlowHttpTest** pro generování pomalého DoS útoku.

Nástroj **trafgen** je určen pro generování síťového provozu pro zátěžové a výkonnostní testování či ladění navržený pro Linuxové operační systémy. K vytváření paketů využívá paketového rozhraní operačního systému, čímž umožňuje plnou kontrolu nad vytvářením dat a hlaviček protokolů k odeslání. Pracuje na bázi vlastního nízkoúrovňového konfiguračního jazyka a není limitován žádným komunikačním protokolem. Jeho jedinou slabinou je pouze generování provozu a nemožnost navazování spojení. [25] V rámci této práce je využit ke generování záplavových útoků SYN Flood, RST Flood, UDP Flood, DNS Flood a ICMP Flood. Konfigurační soubory k těmto útokům byly pro účely testování poskytnuty vedoucím práce.

SlowHttpTest je nástroj, jak již z názvu vyplývá, určený k testování odolnosti serverů proti pomalým DoS útokům. Dokáže napodobit útoky Slow GET (Slowloris), Slow POST (R.U.D.Y), Apache Range Header a Slow Read. Umožňuje široké možnosti nastavení útoku, jako je například počet otevřených spojení za sekundu, intervaly mezi segmenty HTTP požadavků nebo typ odesílaného HTTP požadavku. V neposlední řade umožňuje po ukončení testu vygenerování statistik útoku ve formátu CSV nebo HTML.[26] Tento generátor je při testování využíván ke generování pomalých DoS útoků Slowloris, Slow POST a Slow READ.

Pro generování útoku Slow NEXT je využíván generátor **SlowDosGen** navržený v rámci bakalářské práce Dominikem Richterem.[18] K vytvoření útoku Slow DROP je využíván skript **PySlowDoS**, vytvořený v diplomové práci Michaelem Jurkem.[16]

5.2 Příprava a spuštění aplikace na serveru

Jak již bylo zmíněno v předešlé kapitole, systém využívá k monitorování provozu na síťovém rozhraní knihovnu Scapy pro jazyk Python. Knihovna není základní

součástí operačního systému a je nutné ji doinstalovat. Nejjednodušší způsob instalace je pomocí instalátoru balíčků `pip` (viz výpis 5.1). Dalším potřebným nástrojem je Tcpkill, který je součástí balíčku Dsniff. Instalační příkaz k tomuto balíčku je také na stejném výpisu.

Po doinstalování knihovny Scapy lze samotné spuštění aplikace provést v terminálu po přesunu do domovské složky `/ips` pomocí příkazu třetího příkazu u výpisu 5.1. Z důvodu manipulace s IPTables je nutné spustit aplikaci s právy administrátora.

Výpis 5.1: Instalace nástrojů a spuštění IPS

```
$ pip3 install scapy
$ sudo apt install dsniff
$ sudo python3 ips_main.py
```

Při úspěšném spuštění by mělo být do konzole vypsáno potvrzení o úspěšném resetování IPTables, načtení signatur ze souboru a na kterém rozhraní aplikace naslouchá.

5.3 Testování detekce záplavových útoků

Pro testování všech záplavových útoků byl využit již zmíněný nástroj **trafgen** pomocí příkazu na výpisu 5.2.

Výpis 5.2: Test pomocí trafgen

```
$ trafgen --dev eth0 --conf ./{CONFIG} -p
-t 9000ns
```

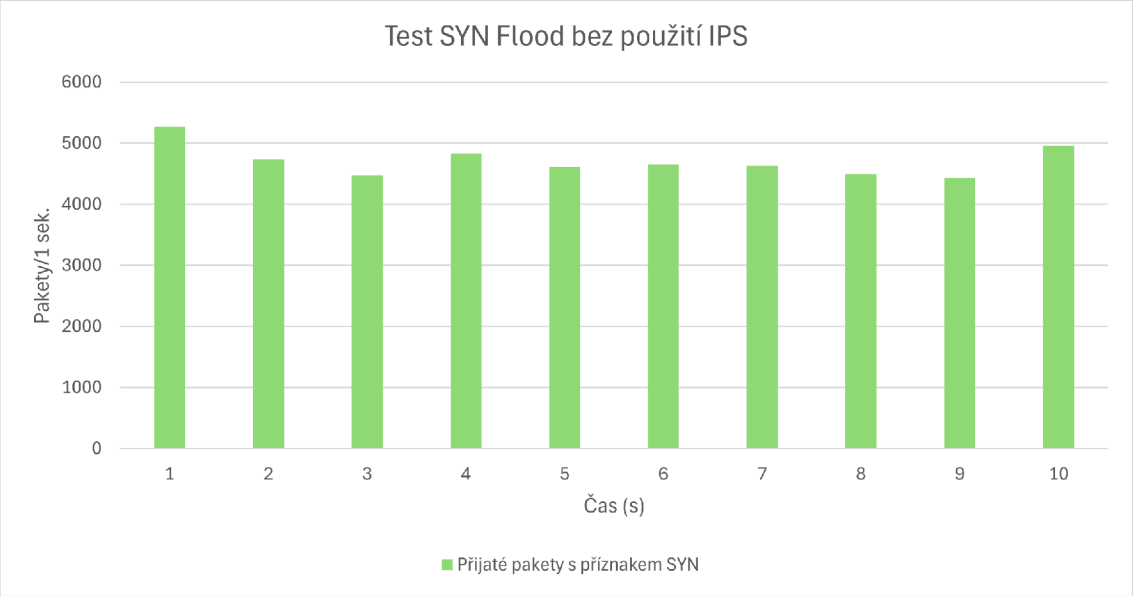
Specifikované parametry značí následující:

- **–dev** – rozhraní použité k odesílání
- **–conf** – konfigurační soubor specifikující odesílaný paket,
- **-p** – využití „push“ módu
- **-t** – interval mezi odesíláním paketů

Příkaz zůstává neměnný pro všechny případy, s výjimkou použitého konfiguračního souboru. Konfigurační soubory byly upraveny tak, aby bylo možné směrovat pakety až na cílový server. Pakety byly odesílány z rozhraní `eth0` stroje útočníka. Mód „push“ zajišťuje odesílání paketů ve vysoké intenzitě. Interval mezi odesíláním paketů byl nastaven na 9000 nanosekund. Pro každý útok bylo simulována distribuována forma útoku (DDoS) ze zdrojových adres z rozsahu **192.168.3.0** až **192.168.3.20**. Tento test byl pro každý útok spuštěn dvakrát a každý test trval 10 sekund. V prvním případě byl útok směřován přímo na server a v druhém procházel přes filtrační server se spuštěnou implementací systému prevence průniku. Pro zachycení provozu byl využit analyzátor Wireshark, který běžel na stroji serveru.

5.3.1 Testování SYN Flood

V případě útoku SYN Flood byly na server odesílány požadavky o navázání spojení s příznakem SYN protokolu TCP. Výsledek testu bez použití filtračního serveru je možné vidět na grafu 5.2. Bez použití jakéhokoliv zabezpečení bylo na server doručováno průměrně 4700 požadavků za sekundu. Server na tyto požadavky standardně odpovídal, jako v případě běžného navazování spojení, avšak tyto odpovědi nejsou v grafu zahrnuty.



Obr. 5.2: Výsledek testu SYN Flood útoku bez použití IPS

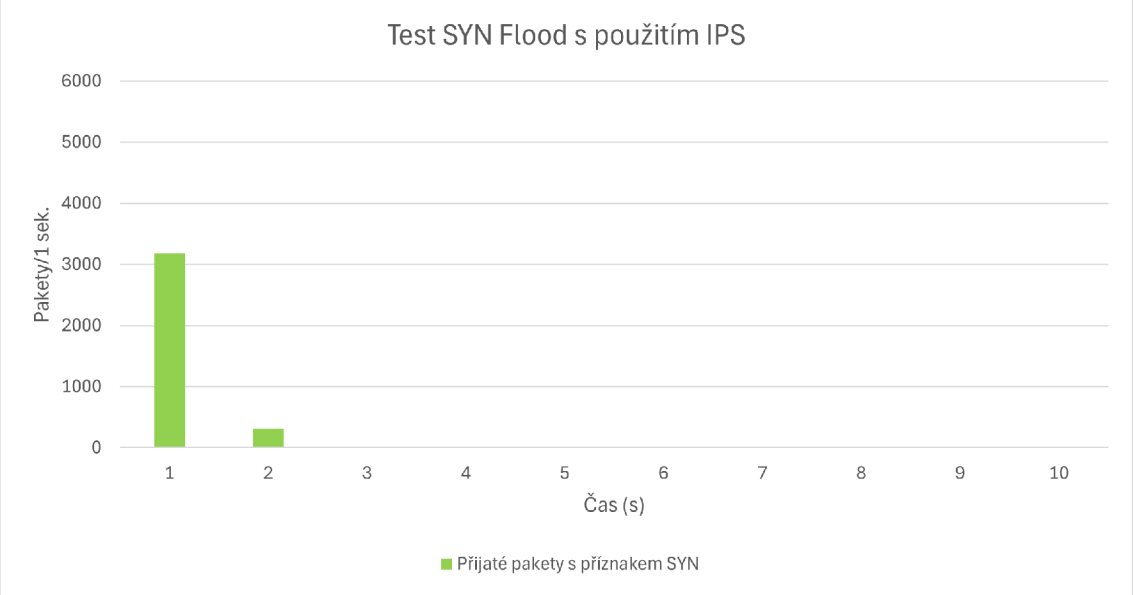
Pro druhý test byla využita signatura, kterou je možné vidět na výpisu 4.1, do které byly vyplněny následující údaje:

- attack: "SYN Flood",
- protocol: "TCP",
- request-type: "SYN",
- connections: 1000,
- per-address: 100,
- per-sec: 10.

Z takto vyplněných parametrů bylo systému zadáno, že má sledovat pakety transportního protokolu TCP s příznakem SYN. Celkový povolený počet otevřených spojení je 1000 za 10 sekund, případně 100 požadavků pro jednu IP adresu. To znamená, že pokud na server přijde celkem 1000 spojení za kratší dobu než je 10 sekund (případně 100 požadavků z unikátních IP adres) bude zdroj tohoto provozu zablokován.

Po opětovném spuštění testu bylo na základě předešlého grafu očekáváno, že s použitím IPS bude spojení zablokováno hned v prvním sekundě po spuštění útoku,

jelikož rychlost odesílání paketů převyšuje povolené hodnoty a tím pádem naplňuje podmínky zadané v signatuře útoku. Tuto skutečnost lze ověřit na grafu 5.3



Obr. 5.3: Výsledek testu SYN Flood útoku s použitím IPS

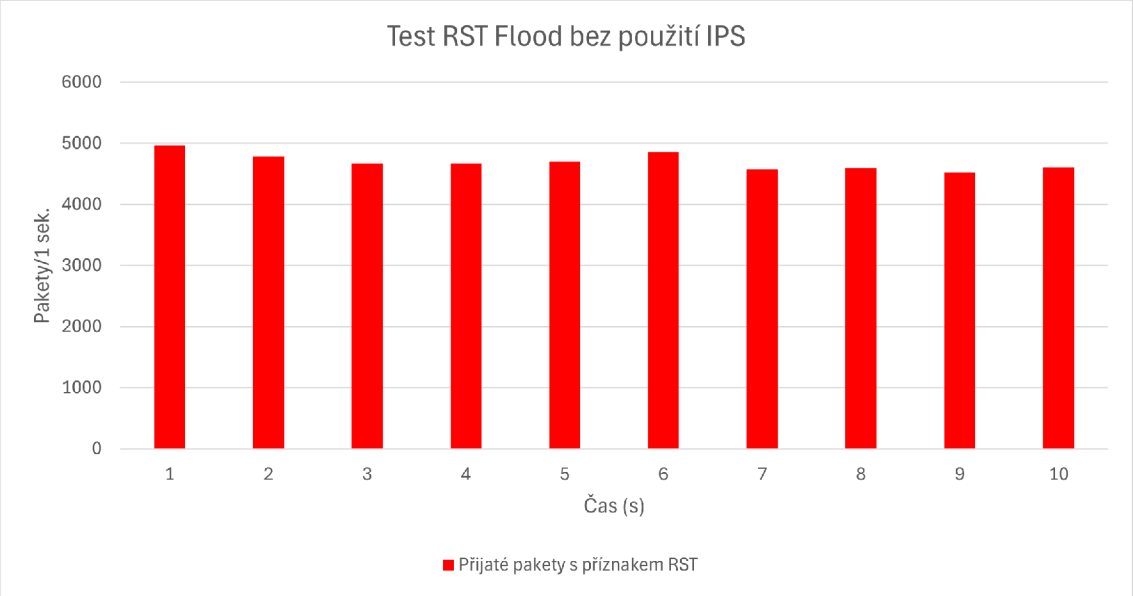
Z grafu vyplývá, že filtrační server skutečně zaznamenal útok v první sekundě a zahájil jeho blokování. Toto blokování probíhalo až do druhé sekundy a na webový server bylo propuštěno celkem 3497 požadavků.

5.3.2 Testování RST Flood

Při simulaci útoku RST Flood byly na server odesílány TCP požadavky s příznakem RST. Jelikož na server nebyly v průběhu útoku navázána žádná TCP spojení, jednalo se pouze o snahu zahltit přenosovou cestu směrem k serveru. Bez použití filtračního serveru je průběh testu zobrazen na grafu 5.4. Na server bylo průměrně odesláno 4692 paketů za sekundu. Následně byl zahájen test druhý. Pro ten byla využita opět signatura pro záplavové útoky (viz 4.1) vyplněná následovně:

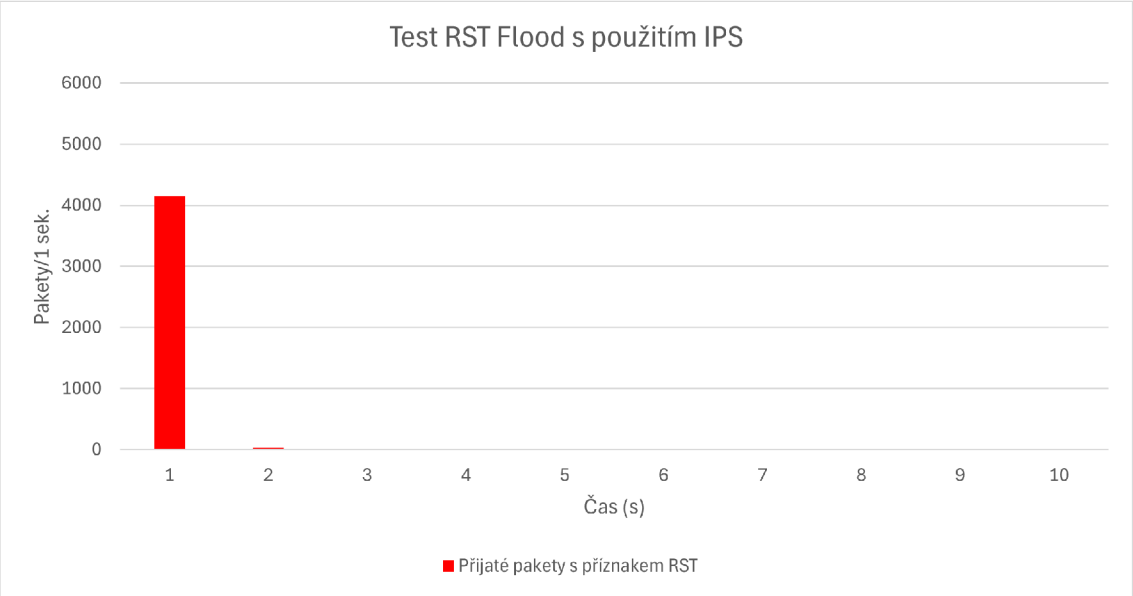
- attack: "RST Flood",
- protocol: "TCP",
- request-type: "RST",
- connections: 1000,
- per-address: 100,
- per-sec: 10.

Jako v případě předešlého útoky byl opět sledován transportní protokol TCP, v tomto případě každý TCP paket s příznakem RST. Podmínky pro vyhodnocení útoku zůstaly stejné jako v předešlém případě.



Obr. 5.4: Výsledek testu RST Flood útoku bez použití IPS

Při stejném přenosu přes filtrační server bylo předpokládáno, že útok bude detekován v první sekundě. Skutečný průběh testu je vyobrazen na grafu 5.5.

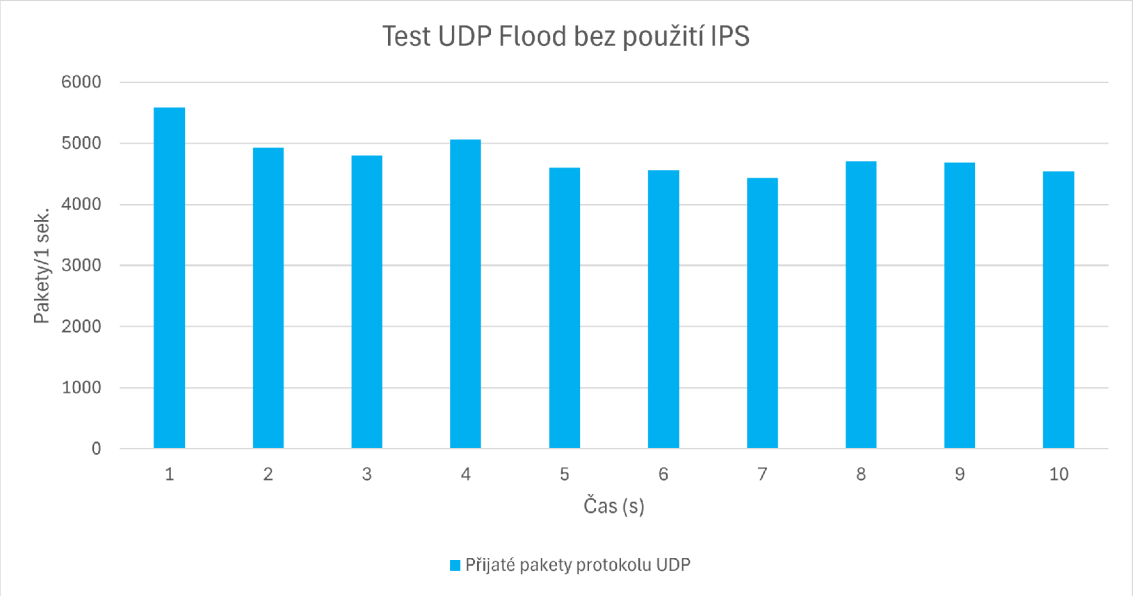


Obr. 5.5: Výsledek testu RST Flood útoku s použitím IPS

Hodnoty na grafu ukazují, že útok byl zachycen ihned po jeho začátku a to v první sekundě po jeho zahájení. Následně probíhala blokace zdrojových IP adres příchozích požadavků. Na server bylo propuštěno 4177 těchto požadavků.

5.3.3 Testování UDP Flood

V případě útoku UDP Flood byly na server odesílány UDP pakety v tomto konkrétním případě směřované na port 50 s cílem zahlcení komunikačního média. Na tomto portu neběžela na straně serveru žádná aplikace, tudíž server odpovídal na tyto pakety ICMP zprávou „Destination Unreachable“. Průběh testu bez použití filtračního serveru je vyobrazen v grafu 5.6. V tomto grafu jsou znázorněny přijaté pakety, jejichž průměrná rychlost příchodu byla 4790 paketů za sekundu.



Obr. 5.6: Výsledek testu UDP Flood útoku bez použití IPS

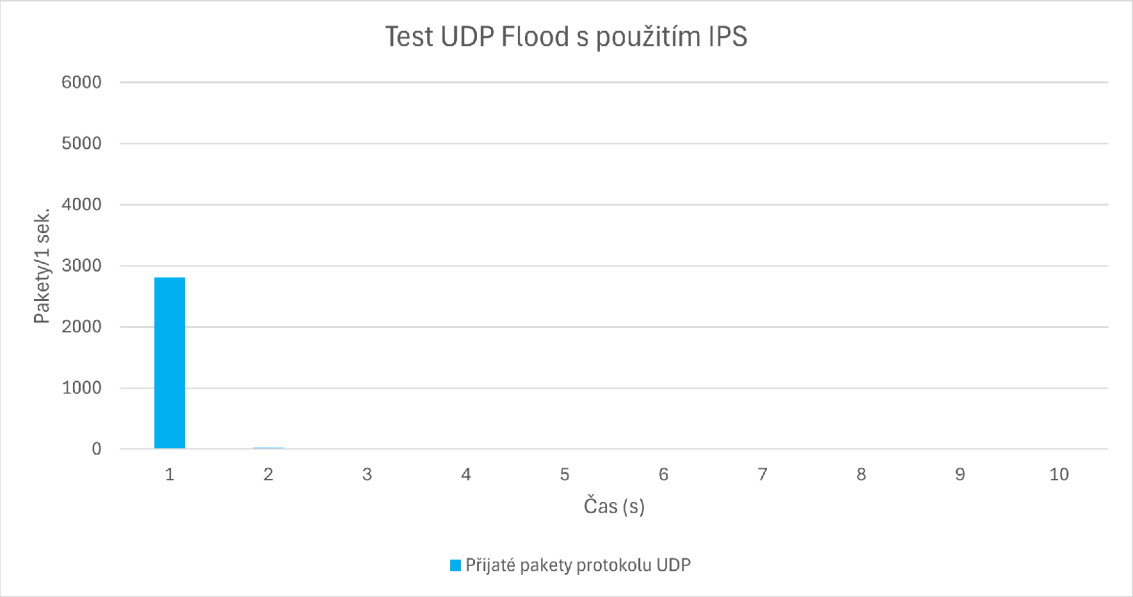
Pro druhý test byla použita signatura pro záplavové útoky (viz výpis 4.1) s těmito parametry:

- attack: "UDP Flood",
- protocol: "UDP",
- request-type: "",
- connections: 1000,
- per-address: 100,
- per-sec: 10.

Do parametru `protocol` byl doplněn název příslušného protokolu, tedy protokolu UDP. Parametr `request-type` byl ponechán prázdný, protože protokol UDP nevyužívá při komunikaci žádných příznaků jako protokol TCP. Parametry na povolené množství paketů za počet sekund byly použity stejné jako v případě předešlých testů.

Následně byl spuštěn test přes filtrační server. U tohoto testu byla očekávána blokace v první sekundě, jelikož průměr přijatých požadavků za deset sekund převyšuje

povolený limit. Skutečný vývoj testu je možné vidět na grafu 5.7.



Obr. 5.7: Výsledek testu UDP Flood útoku s použitím IPS

Z grafu je možné vidět předpokládaný výsledek, tedy že došlo k odchycení a zablokování hned v první sekundě.

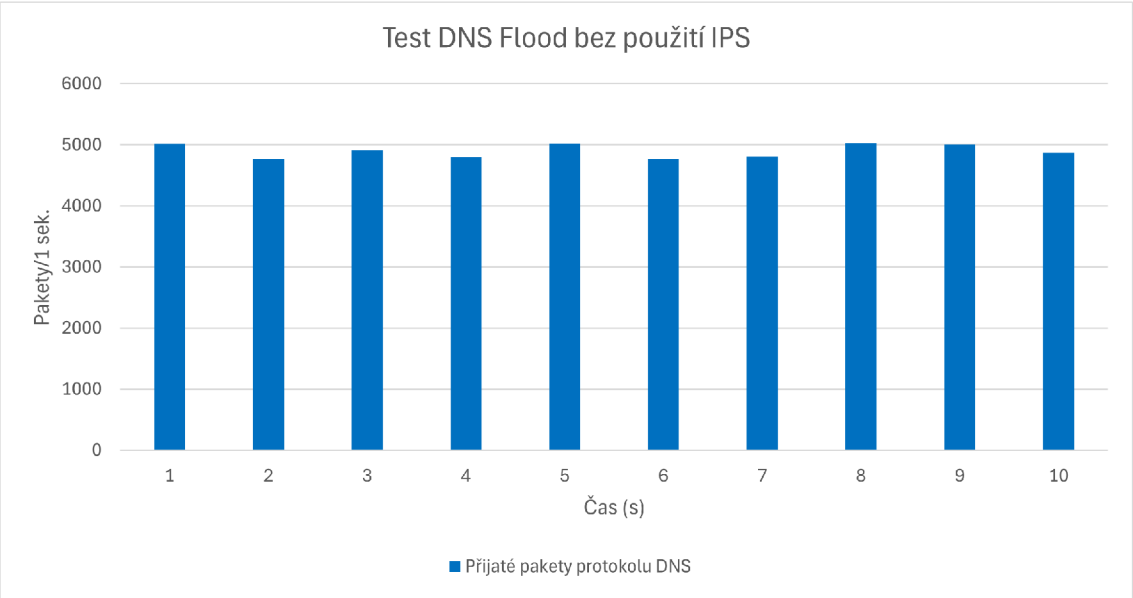
5.3.4 Testování DNS Flood

Útok DNS Flood je ve své podstatě shodný s útokem UDP Flood, jelikož protokol DNS využívá ke svému fungování taktéž protokolu UDP. Princip se liší v tom, že aby byl útok klasifikován jako DNS Flood, musí být pakety směřovaný na port využívaný protokolem DNS, kterým je port 53. Test bez použití systému prevence průniku je vyobrazen na grafu 5.8. Na grafu je možné vidět přijaté pakety protokolu DNS. Průměrný počet paketů v průběhu testu čítal 4900 paketů za sekundu.

K detekci byla použita následují signatura:

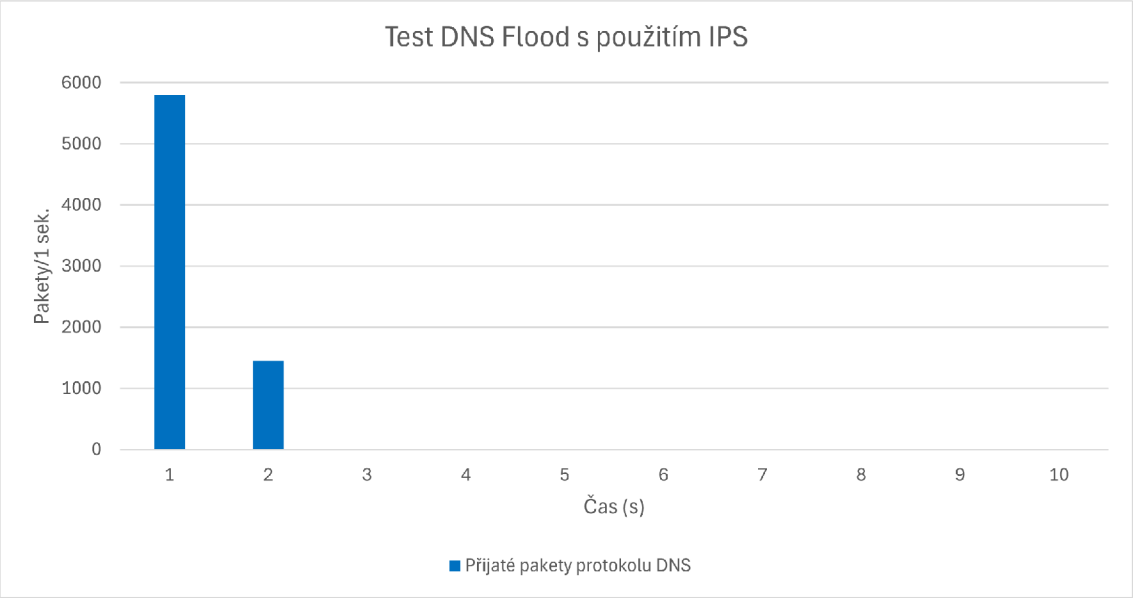
- attack: "DNS Flood",
- protocol: "DNS",
- request-type: "",
- connections: 1000,
- per-address: 100,
- per-sec: 10.

Jako u signatury předešlého útoku byl ponechán prázdný parametr `request-type`, jelikož protokol DNS taktéž nevyužívá žádného specifického příznaku.



Obr. 5.8: Výsledek testu DNS Flood útoku bez použití IPS

Následně byl spuštěn test, při kterém byl provoz sledován implementací systému prevence průniku. Jelikož se princip útoku podobá průběhu UDP Flood, byly očekávány i podobné výsledky, tedy že bude útok zachycen a zablokován ihned v první sekundě. Výsledek testu je vyobrazen na grafu 5.9.



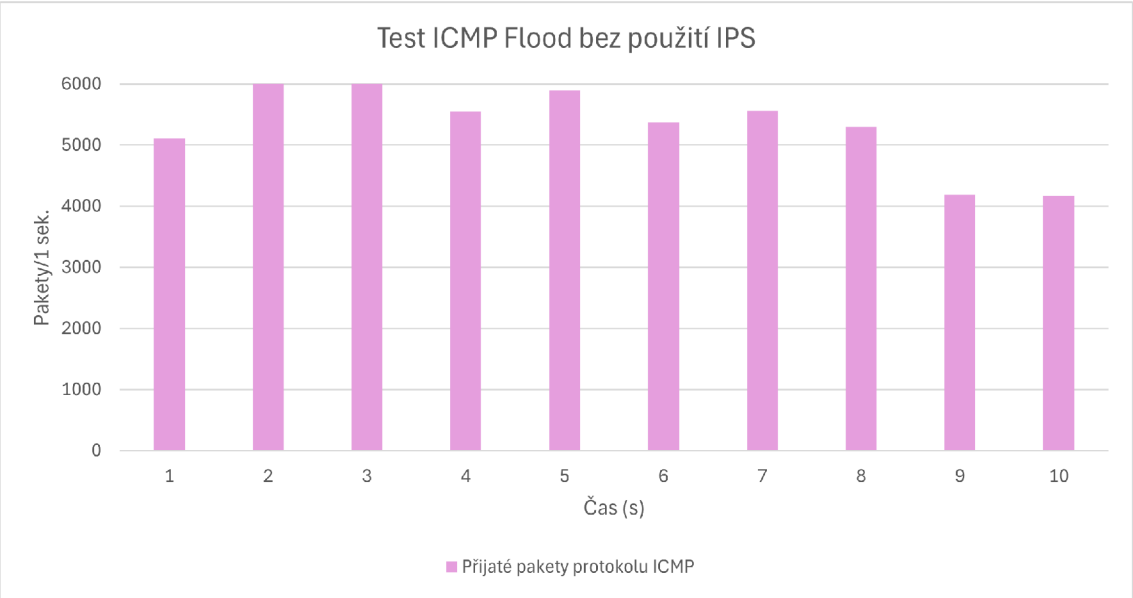
Obr. 5.9: Výsledek testu DNS Flood útoku s použitím IPS

I přesto, že útok DNS Flood je principiálně stejný jako útok UDP Flood, trvalo jeho zastavení déle. Útok byl detekován až v druhé sekundě testu. Tento jev mohl

být zapříčiněn dodatečným rozlišováním, o který útok využívající protokolu UDP se ve skutečnosti jedná. Z důvodu tohoto rozhodování bylo na stroj serveru vpuštěno 7251 paketů.

5.3.5 Testování ICMP Flood

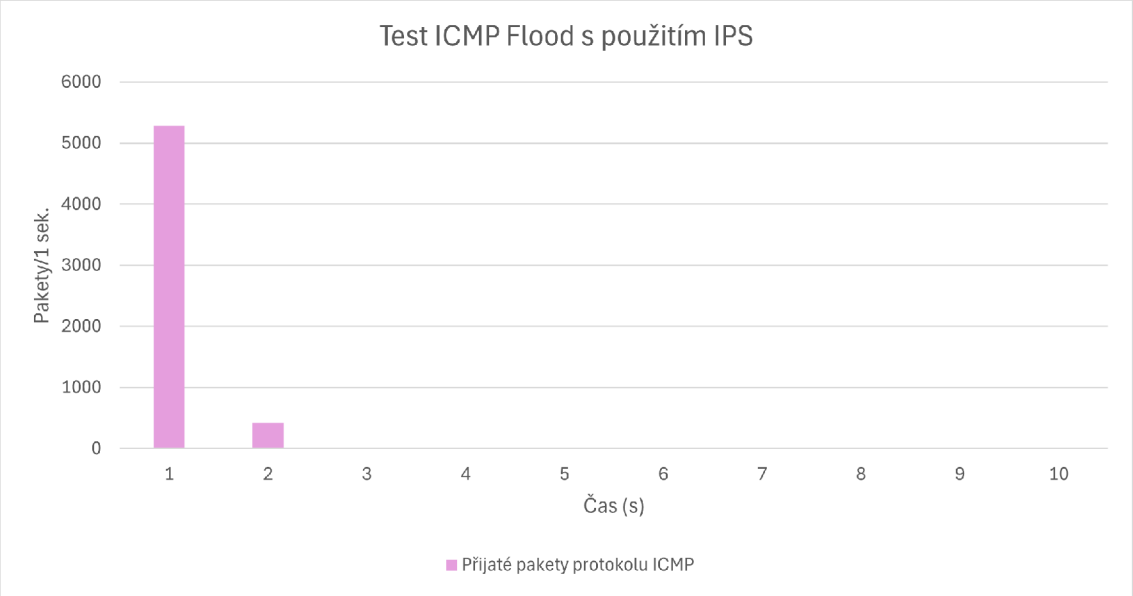
Při útoku ICMP Flood je server zaplavován zprávami „echo request“. Ve své podstatě se jedná o intenzivní snahu útočníka zjistit, zda je server aktivní. Útok opět cílí na zahlcení komunikačního kanálu tímto typem paketů, respektive i odpovědmi serveru na tuto zprávu. Jako v předešlých případech bylo nejprve testováno jak vypadá situace bez filtračního serveru. Graf z tohoto testu je možné vidět na obrázku 5.10. Bez filtračního serveru bylo serveru doručeno průměrně 5334 paketů za sekundu. Pro druhý test byla použita tato signatura:



Obr. 5.10: Výsledek testu ICMP Flood útoku bez použití IPS

- attack: "ICMP Flood",
- protocol: "ICMP",
- request-type: "",
- connections: 1000,
- per-address: 100,
- per-sec: 10.

Po tomto nastavení systém vyhledává pakety protokolu ICMP, opět bez konkrétního doplňujícího parametru `request-type`. Po spuštění testu bylo opět očekávaným výsledkem podchycení útoku hned v první sekundě. Skutečný výsledek testu je možné vidět na grafu 5.11



Obr. 5.11: Výsledek testu ICMP Flood útoku s použitím IPS

Z grafu testu je však vidět, že útok byl plně zachycen a zastaven až v druhé sekundě testu. Tento fakt mohl být zapříčiněn tím, že útočník byl schopen vygenerovat větší množství ICMP paketů a filtrační server je nebyl vzhledem k jeho výpočetní kapacitě dostatečně rychle zastavit. Nicméně i přes tuto skutečnost došlo ve výsledku k zastavení přenosu.

5.4 Testování detekce pomalých DoS útoků

Jako již bylo zmíněno k simulaci většiny útoků bylo využito nástroje **SlowHttpTest**. Příklad příkazu se nachází na výpisu 5.3,

Výpis 5.3: Test pomocí **SlowHttpTest**

```
$ slowhttptest -c 400 -H -g -o slowhttp -r 100 -i 5
-u http://192.168.2.129 -l 60 -p 3
```

kde parametry v příkazu udávají následující [26]:

- **-c** – počet otevřených spojení,
- **-H** – typ útoku (v tomto případě Slowloris),
- **-g** – vygenerování statistik,
- **-o** – název souboru se statistikami,
- **-r** – počet spojení za sekundu,
- **-i** – interval mezi dodatečnými daty,
- **-u** – adresa cíle,

- **-l** – délka testu v sekundách,
- **-p** – interval čekání odezvy od serveru.

Pro toto konkretní nastavení by bylo otevřeno celkem 400 spojení rychlostí 20 spojení za sekundu. Prodleva mezi dodatečnými daty je nastavena na 5 sekund. Tato prodleva je klíčová pro simulaci útoků Slowloris a Slow POST. Interval čekání odezvy od serveru je využit ke sledování, zda je server dostupný. Trvání celého testu je v tomto případě stanovena na 1 minutu a výsledné statistiky testu budou po skončení testu uloženy do souborů s názvem *slowhttp*. Pro testování útoků Slowloris, Slow POST a Slow Read je použita většina těchto parametrů. Případné další parametry jsou v textu popsány.

Pro útok Slow NEXT byl využit nástroj **SlowDosGen**, jehož příkaz ke spuštění se nachází na výpisu 5.4.

Výpis 5.4: Test pomocí **SlowDosGen**

```
$ python3 SlowDoSGen.py -a N -ip 192.168.2.129 -c 320
-p 80
```

kde použité parametry nastavují tyto hodnoty[18]:

- **-a** – typ použitého útoku
- **-ip** – cílovou IP adresu
- **-c** – počet otevřených spojení
- **-p** – cílový port

Písmeno „N“ za parametrem `-a` udává, že má být vytvářen útok Slow NEXT na adresu a port webového serveru. Počet otevřených spojení je nastaven na 320, ale jejich reálný počet je pětkrát vyšší, protože generátor ve výchozím stavu využívá pět vláken, kde každé navazuje definovaný počet spojení.

Pro útok Slow DROP byl použit již zmíněný generátor **PySlowDos**. Ke spuštění tohoto generátoru byl použit příkaz z výpisu 5.5.

Výpis 5.5: Spuštění generátoru **PySlowDos**

```
$ python3 pyslowdos.py -c 100 -d 60 192.168.2.129
slow_drop -D 0.9
```

Jednotlivé parametry znamenají následující:[16]

- **-p** – specifikace cílového portu
- **-c** – počet navázaných spojení
- **-d** – délka testu
- **-D** – míra zahazování

S použitím parametrů generátor vytvoří 100 spojení pod dobu jedné minuty na cílovou adresu **192.168.2.129**, tedy adresu webového serveru. Doplňující parametr

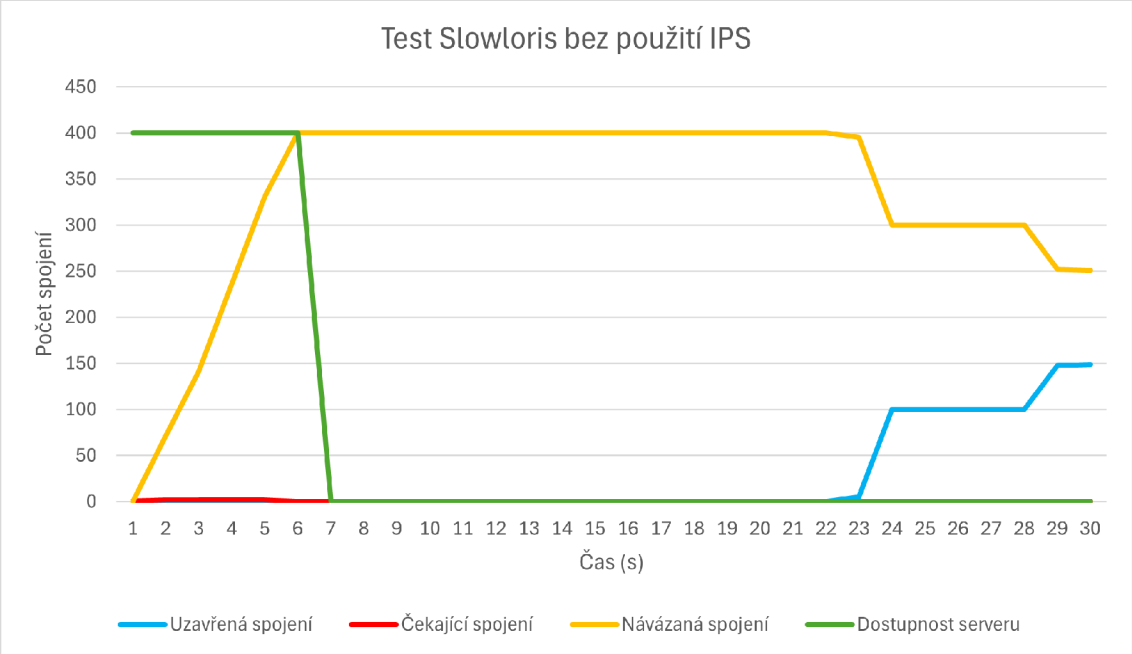
-D specifický pro útok Slow DROP udává, že útočník bude zahazovat 90% příchozí komunikace ze strany serveru.

Pro všechny následující scénáře byly spuštěny dva testy, kde pro oba testovací scénáře bylo použito téže nastavení generátoru. První test byl vždy čistě na server bez použití serveru filtračního. Druhý test byl poté spuštěn přes filtrační server za použití implementace navrženého systému prevence průniku.

Ke zjištění výsledků testování útoků obsažených v generátoru **SlowHttpTest** bylo využito statistik vytvářených samotným generátorem v kombinaci s analyzátorem **Wireshark**. Pro zbylé dva útoky bylo využito pouze nástroje **Wireshark**.

5.4.1 Slowloris

Test odolnosti serveru vůči útoku Slowloris byl vykonán pomocí příkazu shodného s tím na výpisu 5.3. To znamená, že na server bylo otevíráno 20 spojení za sekundu dokud tento počet nedosáhl celkem 400 spojení. U neúplně odeslaných HTTP požadavků byl nastaven interval dodatečného odesílání nastaven na 5 sekund. Graf s průběhem testu bez použití implementace systému prevence průniku je možné vidět na obrázku 5.12. Žlutá stoupající křivka ukazuje jak byla v čase navazována



Obr. 5.12: Výsledek testu Slowloris útoku bez použití IPS

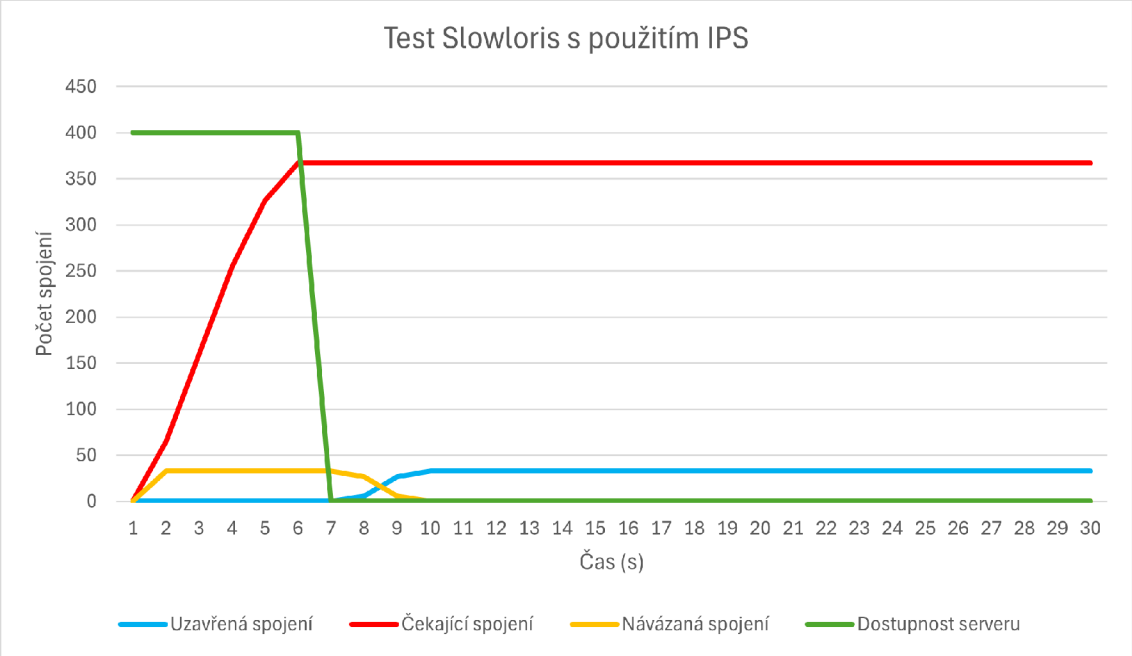
spojení se serverem. Z grafu je možné vidět, že k odepření služeb došlo v 7. sekundě, kdy bylo se serverem navázáno přibližně všech 400 spojení. Tento fakt znázorňuje zelená křivka, která symbolizuje dostupnost webového serveru.

Poté byl spuštěn totožný test využívající k detekci signaturu pro pomalé DoS útoky (viz 4.2 s těmito vyplněným údaji:

- attack: "Slowloris",
- method: "",
- connection: "",
- content-length: false,
- terminator: false,
- window-size: "",
- max-time-between: 5,
- max-number: 5,
- opened-connections: 20,
- per-sec: 1,
- drops: false.

Aby bylo spojení vyhodnoceno jako potenciální útok, musí být otevřeno více než 20 spojení za sekundu (parametry `opened-connections` a `per-sec`), které ve svých HTTP požadavcích neobsahuje ukončovací dvojité odřádkování (znaků `\r\n\r\n`). Tento fakt udává parametry `terminator`, který je nastaven na hodnotu `false`. Poté je spojení dále sledováno a to tak, že doplňující data musí být doručena v následujících pěti požadavcích (`max-number`), kde nesmí být přesaženo časového limitu `max-time-between` mezi daty.

Po spuštění druhého testu byly zjištěny výsledky vyobrazené na grafu 5.13 Z



Obr. 5.13: Výsledek testu Slowloris útoku s použitím IPS

tohoto grafu je možné vidět díky narůstající červené křivce, která značí nárůst čekajících spojení, že bylo zaznamenáno navázání 20 podezřelých spojení hned v první sekundě testu. Filtrační server se poté zaměřil na sledování provozu. Na žluté křivce je možné vidět, že v rozmezí mezi 2. a 7. sekundou byla naplněna podmínka pro časový limit mezi doplňujícími daty a v 8. sekundě začalo ukončování provozu (zobrazeno na modré křivce). Pokles zelené křivky v tomto případě znamená, kdy byl útočníkovi odepřen přístup k webovému serveru filtračním serverem.

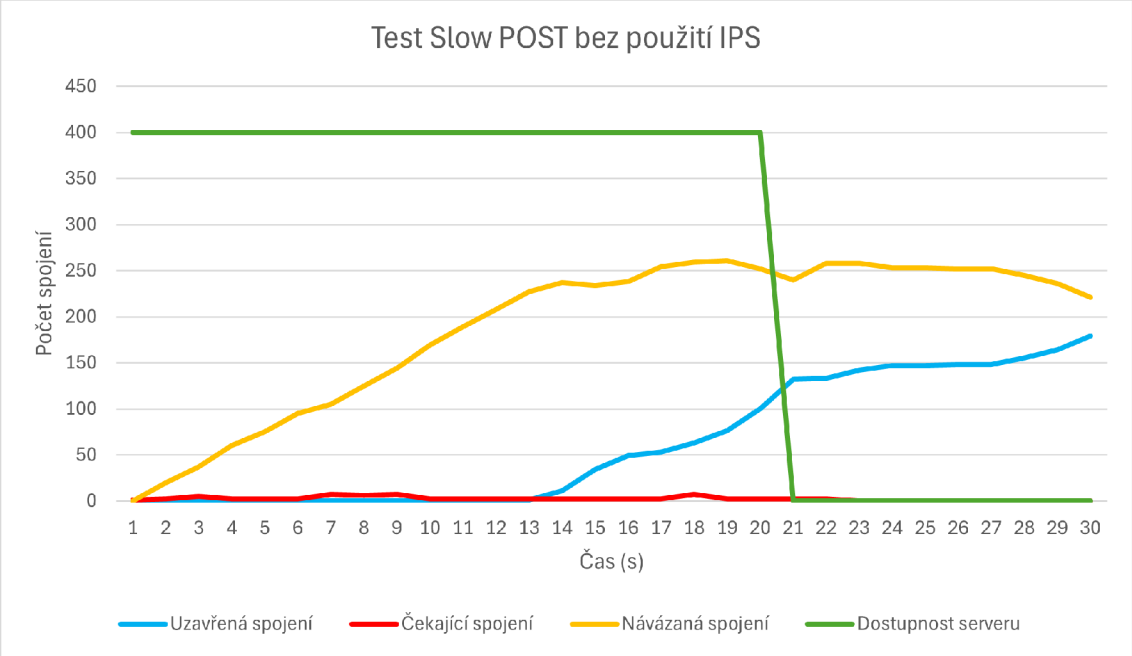
5.4.2 Slow POST

Pro testování útoku Slow POST byl využit příkaz na výpisu 5.6.

Výpis 5.6: Příkaz ke spuštění testu Slow POST

```
$ slowhttptest -c 400 -B -g -o slowpost -r 20 -i 5
-u http://192.168.2.129 -l 60 -x 600 -p 3
```

Na webový server bylo otevřeno 400 spojení s rychlostí 20 spojení za sekundu. Interval mezi dodatečnými daty byl nastaven na 5 sekund. Dodatečná data byla odesílána v blocích o maximální velikosti 600 bajtů, což je v příkazu nastaveno parametrem **-x**. Průběh testu bez filtračního serveru je vyobrazen v grafu na obrázku 5.14. Z po-



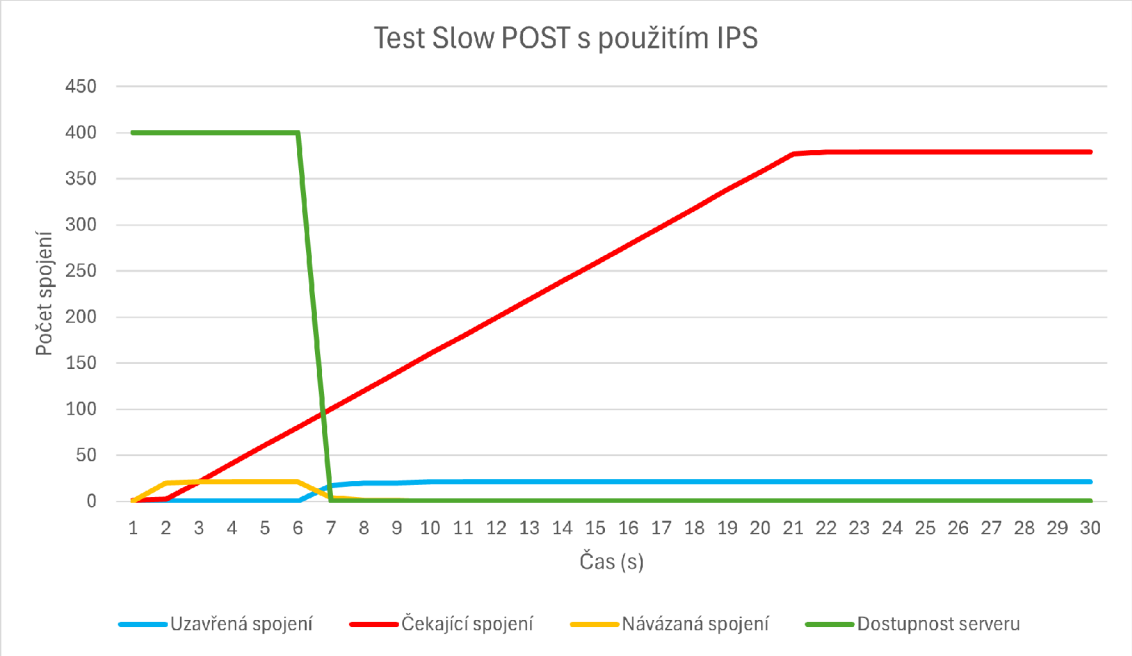
Obr. 5.14: Výsledek testu Slow POST útoku bez použití IPS

klesu zelené křivky je možné vidět, že bez jakékoliv ochrany došlo k odepření služeb zahlcením serveru došlo v 21. sekundě.

K detekci byla využita takto vyplněná signatura:

- attack: "Slow POST",
- method: "POST",
- connection: "close",
- content-length: true,
- terminator: true,
- window-size: "",
- max-time-between: 5,
- max-number: 5,
- opened-connections: 20,
- per-sec: 1,
- drops: false.

Filtrační server díky této signatuře vyhledával spojení, jejichž metodou HTTP požadavku byl POST. Vyplněním tohoto parametru se pouze ulehčil způsob detekce. Pokud by nebyl tento parametr definován, spojení by bylo přesto zachyceno pomocí nastaveného pole `content_length`, které je pro tento typ útoku specifické. Paramter `terminator` je v tomto případě nastaven na hodnotu `true`, jelikož požadavky tohoto útoku jsou řádně uzavřeny. Parametr `connection` je v tomto případě nastaven na `close`, jelikož se od spojení je uzavřeno ihned po vyřízení transakce se serverem. Dále signatura specifikuje, že všechna dodatečná data odesílaná na server musí být obdržena do 5 sekund a maximálně 5 doplňujícími požadavky. Podmínky pro otevření spojení zůstávají i pro tento útok na 20 spojení za sekundu.



Obr. 5.15: Výsledek testu Slow POST útoku s použitím IPS

Po druhém testu se zapnutým filtračním serverem bylo dosaženo výsledků zobrazených na grafu 5.15. Jelikož bylo v tomto případě otevíráno přesně 20 spojení za sekundu, bylo útočníkovi zabráněno hned v první sekundě v otevírání dalších spojení. Tuto skutečnost ukazuje červená křivka, která narůstá od druhé sekundy testu. Již v 1. sekundě filtrační server začal zpracovávat přijaté požadavky. V 6. sekundě bylo vyhodnoceno, že se jedná o škodlivý provoz a spojení byla ukončena. Jak je možné vidět na zelené křivce, útočník plně ztratil přístup k serveru po uzavření všech spojení značený (modrou křivkou).

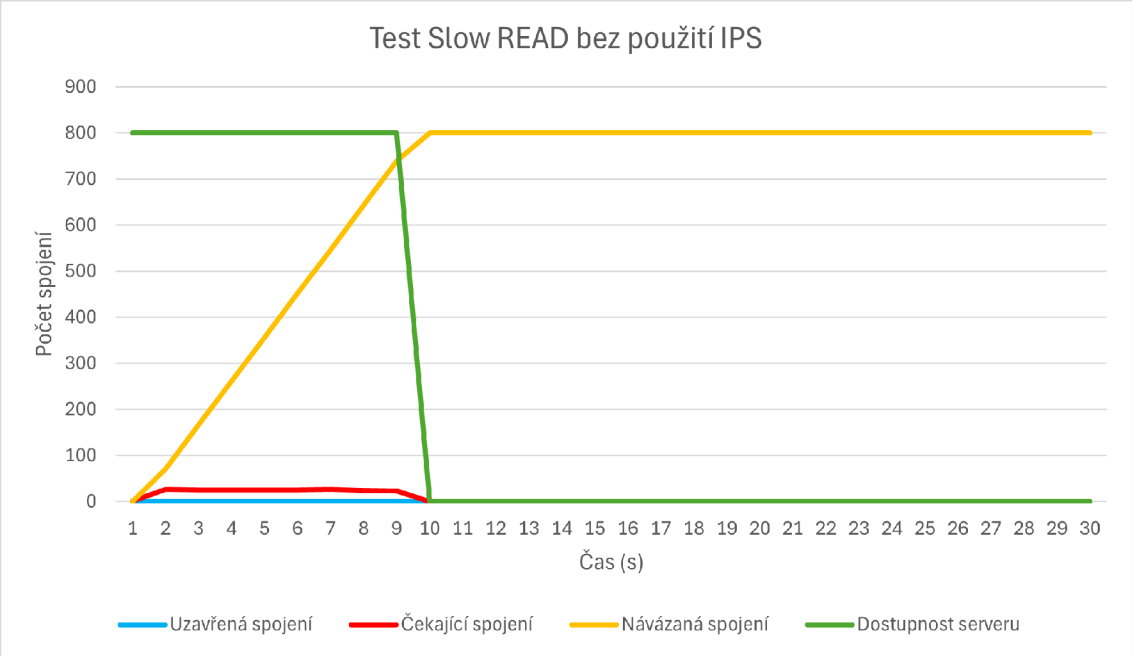
5.4.3 Slow READ

Ke spuštění testu Slow READ bylo využito příkazu z výpisu 5.7.

Výpis 5.7: Příkaz ke spuštění testu Slow READ

```
$ slowhttptest -c 800 -X -g -o slowread -r 100
-u http://192.168.2.129 -l 60 -p 3 -k 10
```

Aby došlo k odepření služeb na serveru, musel být pro tyto testy zvednut počet navazovaných spojení. Po spuštění příkazu bylo navázáno celkem 800 spojení v intervalu 100 spojení za sekundu. Parametr **-k** v tomto příkazu značí tunelované spojení, což znamená, že součástí jednoho spojení bude požadavek 10 krát opakován. Výsledný graf z testu bez použití filtrace je vyobrazen na obrázku 5.16. Z poklesu zelené křivky je možné vidět, že k odepření služeb bylo dosaženo v 10. sekundě.



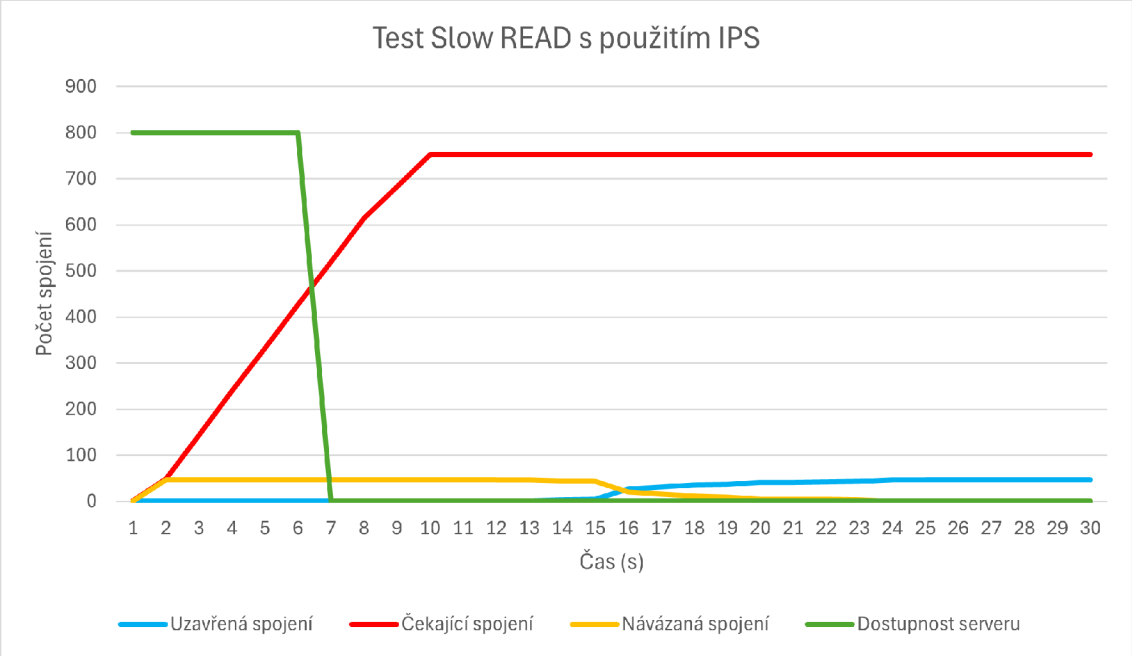
Obr. 5.16: Výsledek testu Slow READ útoku bez použití IPS

Pro druhý test byla pro detekování využita tato signatura:

- attack: "Slow POST",
- method: "GET",
- connection: "keep-alive",
- content-length: false,
- terminator: true,
- window-size: "0",
- max-time-between: 5,
- max-number: 5,
- opened-connections: 20,
- per-sec: 1,
- drops: false.

V případě detekce Slow READ jsou dostupné dva způsoby detekce. Jedním z nich je ten, že hlavička požadavku v případě využití tunelovacího režimu obsahuje pole s parametrem `keep-alive`. Druhým způsobem, vycházejícího z principu útoku, je detekce zmenšujícího se TCP okna v průběhu navázaného spojení. Obě možnosti jsou definovány v signatuře v polích `connection` a `window-size` (v tomto pořadí). Parametr `max-time-between` zde slouží k nastavení maximální povolené prodlevy mezi serverem přijatými daty ze strany útočníka. Po překročení tohoto časového limitu je spojení určeno k ukončení.

Výsledek druhého testu je možné vidět na obrázku 5.17. S rostoucí červenou



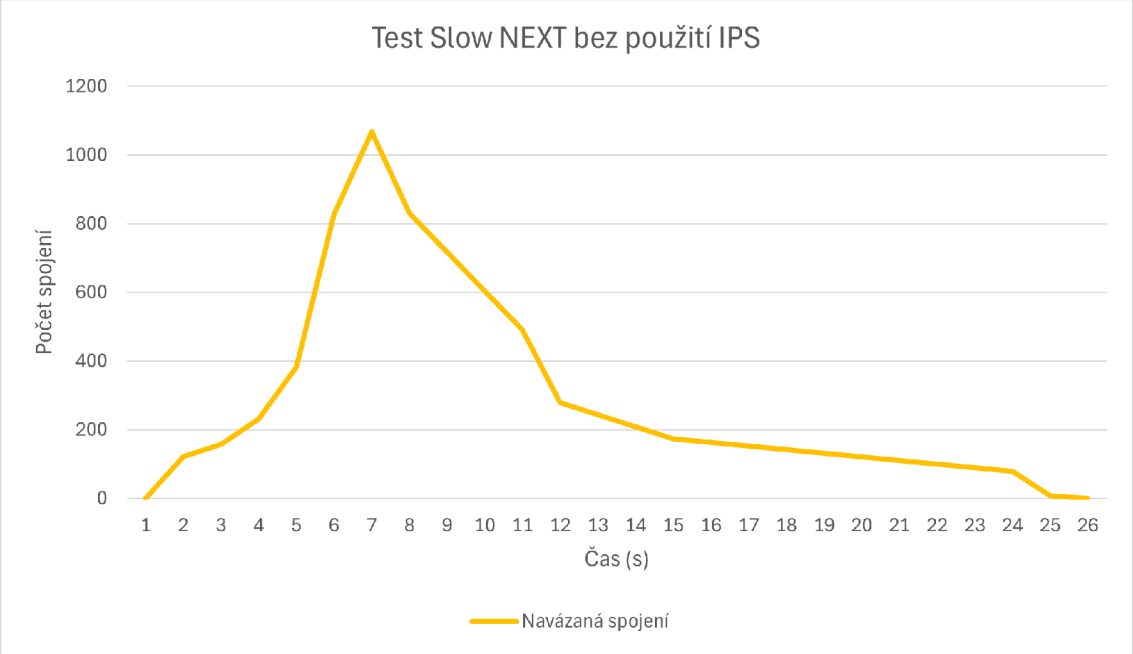
Obr. 5.17: Výsledek testu Slow READ útoku s použitím IPS

křivkou lze vidět, podezřelý provoz byl detekován v 2. sekundě od začátku testu, kdy byla straně útočníka odebrána možnost navazovat další spojení. Filtrační server poté analyzoval provoz a ve 14. sekundě lze podle klesající žluté a rostoucí modré křivky poznat, že začalo ukončování navázaných spojení.

5.4.4 Slow NEXT

K provedení útoku Slow NEXT by využit stejný příkaz uveden na výpisu 5.4. Nástroj k testování bohužel nenabízí možnost vlastního generování výstupu či výsledných grafů, proto byl pro tento účel využito síťového analyzátoru **Wireshark** v kombinaci s nástrojem **tshark**. I bez použití filtračního serveru se v průběhu testování nepodařilo na serveru dosáhnout bodu, kdy by došlo k úplnému odepření služeb a to ani při vyvinutí velké zátěže na jeho výpočetní kapacitu. Tato skutečnost však nebránila porovnání vlivu filtračního serveru na síťový provoz.

Výsledky testu bez použití ochrany je vyobrazeno v grafu na obrázku 5.18. Jak



Obr. 5.18: Výsledek testu Slow NEXT útoku bez použití IPS

vyplývá z křivky vyobrazené na grafu, se serverem byly soustavně navazovány spojení, dokud nedošlo k dovršení maxima spojení, které byl server schopen udržet. Maximum perzistentních spojení bylo dosaženo v 7. sekundě testu, kdy bylo navázáno celkem asi 1100 spojení z původně odhadovaných 1600. Po dosažení tohoto počtu spojení začal server spojení uzavírat.

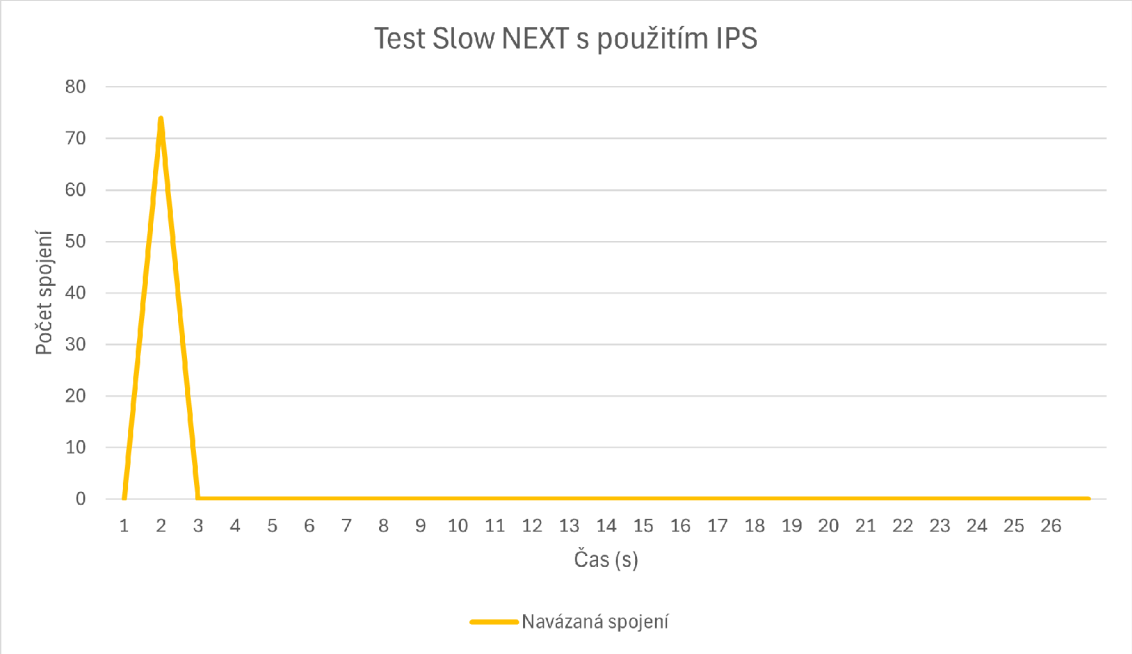
Následný druhý test využíval níže popsanou signaturu:

- attack: "Slow NEXT",

- method: "HEAD",
- connection: "",
- content-length: false,
- terminator: true,
- window-size: "",
- max-time-between: 4,
- max-number: 10,
- opened-connections: 20,
- per-sec: 1,
- drops: false.

Jelikož je útok Slow NEXT zaměřen na navázání a udržení co největšího počtu spojení, využívá k tomuto účelu metody HEAD. Pro jeho detekci je tedy tato skutečnost zaznamenána ve výše definované signatuře. Pokud je na server otevřeno více než 20 spojení za sekundu, bude spojení zablokováno a sledováno. Protože útok využívá timeoutu webového serveru, který je defaultně nastaven na 5 sekund je zde `max-time-between` nastaven na 4 sekundy s tím, že v případě tohoto typu útoku představuje maximální interval mezi přijatými HTTP požadavky.

Výsledný graf testu je k vidění na obrázku 5.19 Z hodnot v grafu je možné



Obr. 5.19: Výsledek testu Slow NEXT útoku s použitím IPS

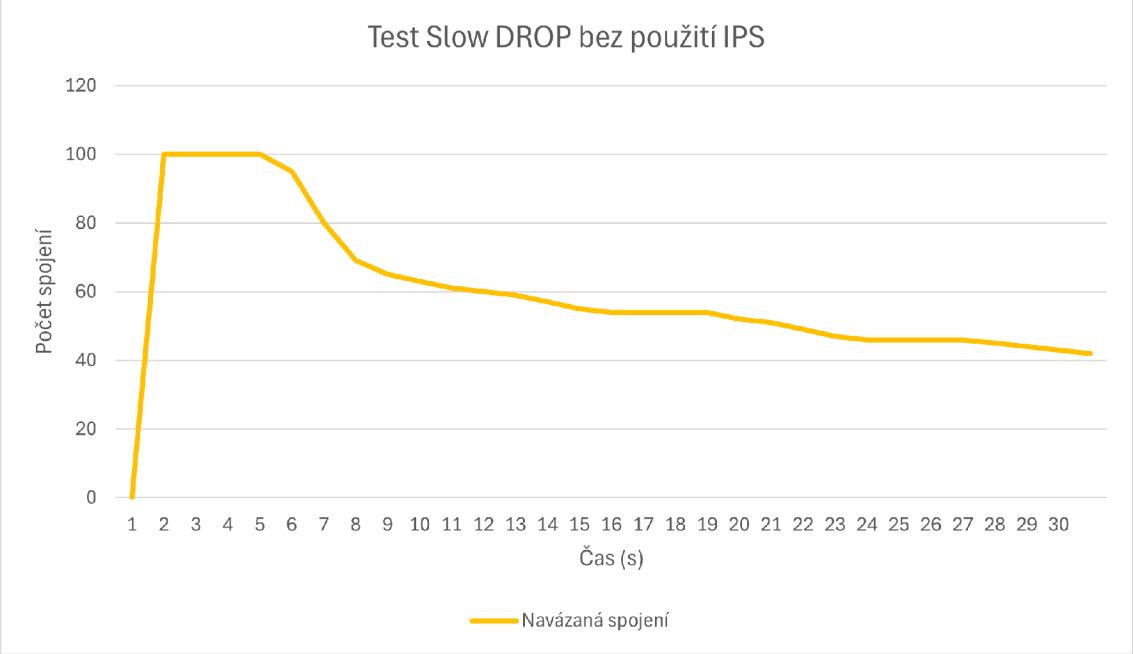
vidět skutečnost, že i přes limit spojení 20 za sekundu bylo na server navázáno 74 spojení, tudíž došlo k blokaci zdroje komunikace. Po zablokování spojení ztratil stroj útočníka možnost odesílání `keep-alive` zpráv, tudíž byla serverem všechna spojení

sama ukončena.

5.4.5 Slow DROP

Pro útok Slow DROP byl použit příkaz identický tomu uvedenému na výpisu 5.5. Při testovacím scénáři nevyužívajícím služeb filtračního serveru se i přes testování veškerých možných nastavení útoků nepodařilo na stroji serveru vyvolat stav odepření služeb. Nicméně tato skutečnost neměla na princip detekce a blokování útoku žádný vliv. Pro zpracování výsledků útoku bylo i v tomto případě využito kombinace nástrojů **Wireshark** a **tshark**.

Graf útoku bez využití filtračního serveru je k vidění na obrázku 5.20. V průběhu



Obr. 5.20: Výsledek testu Slow DROP útoku bez použití IPS

grafu je možné zaznamenat úspěšné navázání všech 100 požadavků. Z principu útoku bylo po navázání spojení zahájeno zahazování 90% příchozích paketů, což jak křivka značí, výrazně zpomalilo přirozené uzavírání spojení. Uzavírání spojení lze v těchto podmínkách provést pouze v malých časových okamžicích, jak je možné vidět mezi 5. a 8. sekundou.

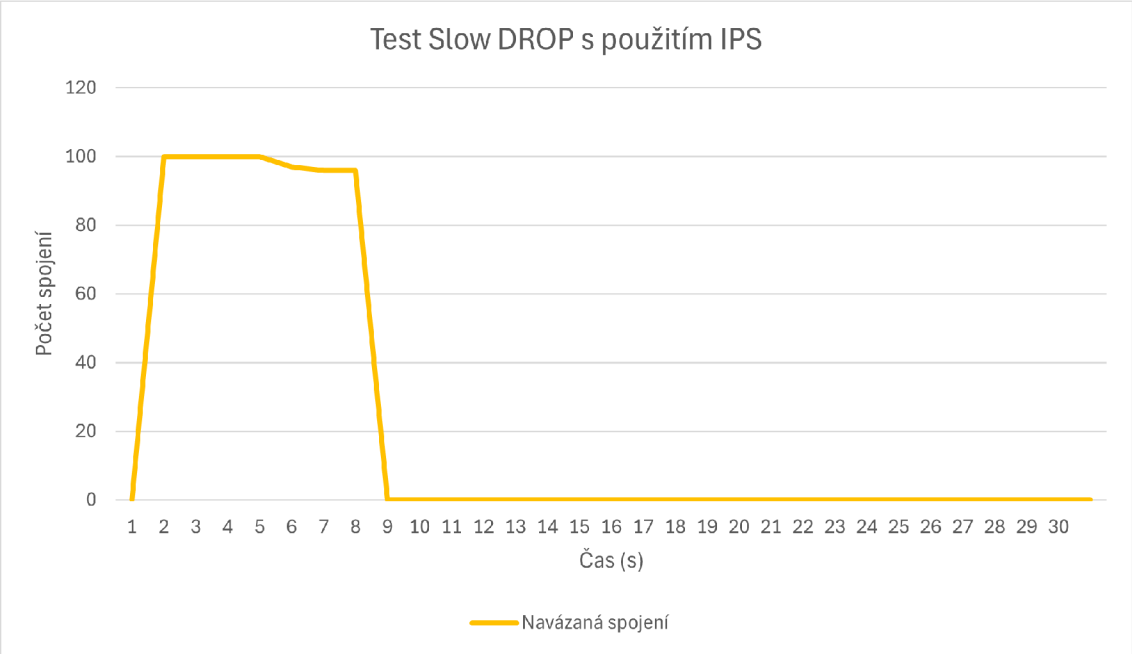
Pro detekci útoku Slow DROP byla takto vyplněna signatura:

- attack: "Slow DROP",
- method: "GET",
- connection: "",
- content-length: false,
- terminator: true,

- window-size: "",
- max-time-between: 5,
- max-number: 5,
- opened-connections: 20,
- per-sec: 1,
- drops: true.

Nejdůležitějším parametrem pro detekci tohoto typu útoku je parametr `drops`, který systému oznamuje, že má být dáván důraz na monitorování zahazování paketů směrovaných k webovému serveru. Parametr `max-time-between` zde slouží jako interval pro udání reakčního času mezi jednotlivými pakety. Podobně je využito i parametru `max-number`, který v tomto případě sleduje počet paketů, na které ze strany útočníka nepřišla odpověď.

Druhý test přinesl výsledky, které je možné vidět na grafu 5.21 Křivka grafu



Obr. 5.21: Výsledek testu Slow DROP útoku s použitím IPS

značí, že na začátku testu bylo vytvořeno všech 100 spojení. Filtrační server v tento moment ještě nezaznamenal žádný útok. K zaznamenání útoku došlo mezi 2. až 4. sekundou, protože server v tomto časovém intervalu držel otevřených všech 100 spojení. Když došlo k přirozenému uzavření spojení v 6. sekundě, filtrační server zaznamenal čas reakce strany útočníka. V 9. sekundě došlo k naplnění časového limitu mezi odesílanými pakety a bylo zahájeno uzavírání navázaných spojení.

Závěr

Cílem této diplomové práce bylo vytvoření software, který umožňuje detekci záplavových a pomalých útoků odepření služeb na základě signatur. K tomuto účelu byla navrhnuta šablona signatury pro každý typ útoku, uložená v souboru ve formátu JSON. Tento soubor umožňuje jednoduchou editaci a vyplnění zvolených parametrů, které jsou vytvořeným softwarem sledovány. Celý software je napsán v programovacím jazyce Python a ke sledování a rozboru síťové komunikace využívá jeho knihovnu Scapy.

V úvodu práce proběhlo seznámení s principy vybraných protokolů využívaných v síťové komunikaci. V kapitole bylo popsáno fungování nejdůležitějších protokolů transportní vrstvy, mezi které patří spojově orientovaný protokol TCP a jeho nespojovaný protějšek, protokol UDP. V neposlední řadě bylo v kapitole popsáno fungování protokolu aplikační vrstvy HTTP, ketý představoval stěžejní část celé práce.

V následující kapitole práce bylo obsaženo seznámení s útoky na odepření služeb (DoS). Součástí kapitoly bylo porovnání nedistribuované a distribuované formy útoků, dále rozlišení záplavových a pomalých útoků a na závěr příklady obou typů útoků.

V další kapitole byly obsaženy techniky detekce, mitigace a prevence proti DoS útokům. Byly popsány přístupy k detekci založené na signaturách nebo anomáliích v síťového provozu a uvedeny příklady již existujícího detekčního software.

Následující kapitola se zabývala samotným vytvořením signatur pro detekci záplavových a pomalých DoS útoků a návrhem software, který na základě těchto signatur detekoval tyto typy útoků v síťovém provozu. Součástí kapitoly byl popis základní funkcionality software včetně principu vyhodnocování, zda je síťový provoz škodlivý či nikoliv.

Závěrečná kapitola popisovala testování vytvořeného software a jeho schopnost detekce záplavových útoků SYN Flood, RST Flood, UDP Flood, DNS Flood a ICMP Flood a pomalých útoků Slowloris, Slow POST, Slow READ, Slow NEXT a Slow DROP. Součástí kapitoly byl popis testovacího prostředí, nastavení jednotlivých testovacích nástrojů i konkrétní vyplnění parametrů signatur použitých pro detekci zmíněných útoků. Součástí každého testu byly výsledky z testování zobrazené v grafech, které zobrazovaly průběhy a vlivy útoků před a po použitím navržené implementace software.

Literatura

[1] YASAR, K. *TCP/IP* [online]. 2024, [cit. 10. 05. 2024]. Dostupné z URL: <https://www.techtarget.com/searchnetworking/definition/TCP-IP>

[2] WRIGHT, G. *transport layer* [online]. [cit. 13. 05. 2024]. Dostupné z URL: <https://www.techtarget.com/searchnetworking/definition/Transport-layer>

[3] EDDY, W. *Transmission Control Protocol (TCP)* [online]. RFC 9293, August 2022, [cit. 10. 05. 2024]. Dostupné z URL: <https://datatracker.ietf.org/doc/html/rfc9293>

[4] POSTEL, J. *User Datagram Protocol (UDP)* [online]. RFC 768, August 1980, [cit. 10. 05. 2024]. Dostupné z URL: <https://datatracker.ietf.org/doc/html/rfc768>

[5] FIELDING, R.; NOTTINGHAM, M.; RESCHKE, J. *HTTP Semantics* [online]. RFC 9110, June 2022, [cit. 10. 05. 2024]. Dostupné z URL: <https://datatracker.ietf.org/doc/html/rfc9110>

[6] FERGUSSON, K. *denial-of-service attack* [online]. 2019, [cit. 6. 12. 2023]. Dostupné z URL: <https://www.techtarget.com/searchsecurity/definition/denial-of-service>

[7] Palo Alto Networks *What is a denial of service attack (DoS)?* [online]. [cit. 6. 12. 2023]. Dostupné z URL: <https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos>

[8] FRANKENFIELD, J. *Denial-of-Service (DoS) Attack* [online]. 2020, poslední aktualizace 24. 5. 2023 [cit. 6. 12. 2023]. Dostupné z URL: <https://www.investopedia.com/terms/d/denial-service-attack-dos.asp>

[9] *Denial-of-Service Attack* [online]. [cit. 6. 12. 2023]. Dostupné z URL: <https://en.wikipedia.org/wiki/Denial-of-service_attack>

[10] Cloudflare *SYN flood attack* [online]. [cit. 6. 12. 2023]. Dostupné z URL: <https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/>

[11] BESCHOKOV, M. *What Is TCP Reset Attack?* [online]. [cit. 13. 05. 2024]. Dostupné z URL: <https://www.wallarm.com/what/what-is-syn-spoofing-or-tcp-reset-attack>

[12] Cloudflare *Ping (ICMP) flood attack* [online]. [cit. 13. 05. 2024]. Dostupné z URL: <https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/>

[13] Cloudflare *DNS flood attack* [online]. [cit. 10. 05. 2024]. Dostupné z URL: <https://www.cloudflare.com/learning/ddos/dns-flood-ddos-attack/>

[14] Cloudflare *UDP flood attack* [online]. [cit. 13. 05. 2024]. Dostupné z URL: <https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/>

[15] TRIPATHI, N.; HUBBALLI, N. *Application Layer Denial-of-Service Attacks and Defense Mechanisms* [online]. ACM Computing Surveys. 2022, roč. 54, č. 4, s. 1-33. ISSN 0360-0300. [cit. 6. 12. 2023]. Dostupné z URL: <https://dl.acm.org/doi/10.1145/3448291>

[16] JUREK, M. *Detekce moderních Slow DoS útoků* [online]. 2022, [cit. 6. 12. 2023]. Dostupné z URL: <http://hdl.handle.net/11012/204760>

[17] MAZÁNEK, P. *Modelování a detekce útoku SlowDrop* [online]. 2020, [cit. 6. 12. 2023]. Dostupné z URL: <http://hdl.handle.net/11012/189192>

[18] RICHTER, D. *Slow rate DoS útoky nezávislé na protokolu aplikační vrstvy* [online]. 2020, [cit. 10. 05. 2024]. Dostupné z URL: <http://hdl.handle.net/11012/190239>

[19] MAHJABIN, T; XIAO, Y; SUN, G; JIANG, W. *A survey of distributed denial-of-service attack, prevention, and mitigation techniques* [online]. International Journal of Distributed Sensor Networks. 2017, roč. 13, č. 12. ISSN 1550-1477. [cit. 6. 12. 2023]. Dostupné z URL: <https://doi.org/10.1177/1550147717741463>

[20] IBM *What is an IPS?* [online]. [cit. 10. 05. 2024]. Dostupné z URL: <https://www.ibm.com/topics/intrusion-prevention-system>

[21] SAMSON, R. *Top 10 Intrusion Detection And Prevention Systems* [online]. [cit. 10. 05. 2024]. Dostupné z URL: <https://www.clearnetwork.com/top-intrusion-detection-and-prevention-systems/>

[22] *iptables(8) - Linux man page* [online]. [cit. 6. 12. 2023]. Dostupné z URL: <https://linux.die.net/man/8/iptables>

[23] *Welcome to Scapy's documentation!* [online]. [cit. 6. 12. 2023]. Dostupné z URL: <https://scapy.readthedocs.io/en/latest/#>

[24] *tcpkill(8) - Linux man page* [online]. [cit. 6. 12. 2023]. Dostupné z URL: <https://linux.die.net/man/8/tcpkill>

[25] BORKMANN, D. *trafgen(8) — Linux manual page* [online]. 2013, [cit. 10. 05. 2024]. Dostupné z URL: <https://man7.org/linux/man-pages/man8/trafgen.8.html>

[26] *slowhttptest(1) - Linux man page* [online]. [cit. 6. 12. 2023]. Dostupné z URL: <https://linux.die.net/man/1/slowhttptest>

Seznam symbolů a zkratek

DoS	Denial of Service
DDoS	Distributed Denial of Service
DNS	Domain Name Server
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
JSON	JavaScript Object Notation
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

A Obsah elektronické přílohy

```
/ ..... kořenový adresář přiloženého archivu
└── ips ..... složka obsahující software
    ├── ips_main.py ..... spustitelný Python skript
    ├── manual.pdf ..... manuál k použití
    └── signatures.json ..... soubor definující signatury
```