VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

PROSTŘEDÍ PRO TESTOVÁNÍ ALGORITMŮ POTLAČENÍ ÚTOKŮ DDOS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PAVLÍNA PATOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAN KUČERA**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

154734

Ústav:	Ústav počítačových systémů (UPSY)
Studentka:	**Patová Pavlína, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Inteligentní systémy
Název:	**Prostředí pro testování algoritmů potlačení útoků DDoS**
Kategorie:	Počítačové sítě
Akademický rok:	2023/24

Zadání:

1. Nastudujte dostupnou literaturu o DDoS útocích a způsobech jejich potlačení.
2. Seznamte se s obecnými generátory síťového provozu (Spirent TestCenter, Cisco TRex) a se specifickými nástroji pro generování vybraných typů útoků.
3. Navrhněte rozšiřitelné prostředí umožňující vygenerování útočného vektoru a simulující rovněž provoz legitimních uživatelů pro potřeby testování vybraných způsobů potlačení DDoS útoků.
4. Navržené prostředí implementujte jako sadu softwarových nástrojů. V rámci implementace vytvořte rovněž sadu testů demonstrujících a ověřujících vlastnosti vytvořeného prostředí.
5. Diskutujte dosažené výsledky a možnosti dalšího pokračování práce.

Literatura:

- Dle pokynů vedoucího a konzultanta.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Kučera Jan, Ing.**
Konzultant:	Vojanec Kamil, Ing. (CESNET)
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1.11.2023
Termín pro odevzdání:	17.5.2024
Datum schválení:	30.10.2023

Abstrakt

Práce je zaměřena na vytvoření testovacího prostředí pro simulaci DoS útoků. Toto prostředí je užitečné pro ověřování zranitelnosti sítí a služeb proti fenoménu DDoS útoků. Bude tak sloužit pro testování vyvíjených nástrojů a technik pro obranu proti DDoS útokům. Výstupy těchto experimentů pak mohou být použity jako doporučení pro správce sítě, která pomohou zlepšit obranyschopnost jejich systémů. Výsledný nástroj je implementován v jazyce Python pomocí modulu PyTest. Podařilo se naimplementovat řadu v dnešní době nejběžnějších DoS útoků, například záplavové a amplifikační útoky. Je tak možné provádět testování služeb proti těmto nejzávaznějším hrozbám v oblasti DDoS. Výsledný nástroj je jednoduše rozšiřitelný o další metodiky nebo typy útoků. Lze ho tak využít i v budoucnu pro nově vzniklé DoS útoky.

Abstract

The work is focused on creating a test environment for simulating DoS attacks. This environment is useful for verifying the vulnerability of networks and services against the phenomenon of DDoS attacks, therefore it will be used to test the tools and techniques developed to defend against DDoS attacks. The outputs of these experiments can then be used as recommendations for network administrators to help improve the defensibility of their systems. The resulting tool is implemented in Python using the PyTest module. We were able to implement many of today's most common DoS attacks, such as floods and amplification attacks. It is thus possible to test services against these most serious DDoS threats. The resulting tool is easily extensible to other methodologies or attack types, so it can also be used for emerging DoS attacks in the future.

Klíčová slova

DoS, DDoS, nástroj pro DDoS, pomalé útoky, záplavové útoky, amplifikační útoky

Keywords

DoS, DDoS, DDoS tool, low and slow attacks, flood attacks, amplification

Citace

PATOVÁ, Pavlína. *Prostředí pro testování algoritmů potlačení útoků DDoS*. Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Kučera

Prostředí pro testování algoritmů potlačení útoků DDoS

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením pana Ing. Jana Kučery. Další informace mi poskytl Ing. Kamil Vojanec. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

. . . . . .
Pavlína Patová
13. května 2024

Poděkování

Chtěla bych poděkovat vedoucímu práce Ing. Janu Kučerovi za odborné vedení, ochotu a trpělivost při vytváření písemné části práce. Dále bych chtěla poděkovat Ing. Kamilu Vojancovi za pomoc a připomínky k implementaci. V neposlední řadě také všem dalším čtenářům, kteří dopomohli ke zlepšení výsledného textu práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Společnosti a jednotlivci v dnešní digitální éře stále více závisí na internetových službách a digitální infrastruktuře. Bezpečnost online prostoru se stává klíčovým aspektem. Jednou z nejzávažnějších bezpečnostních hrozeb, která může ohrozit webové stránky, servery a síťovou infrastrukturu, jsou útoky na dostupnost známé jako Denial of Service (DoS). Tato forma kybernetického útoku si klade za cíl vyřadit cílový systém nebo službu z provozu prostřednictvím přetížení jeho kapacit nebo využitím zranitelností v jeho architektuře.

Cílem práce je vytvoření testovacího prostředí, které bude implementovat popsané DoS útoky. Hlavním účelem vytvářeného prostředí je testování odolnosti koncových systémů proti DDoS útokům a testování metod pro obranu proti těmto hrozbám. Výstup z takového testování může správcům systému poskytnout cenné informace a určitá doporučení ohledně zabezpečení a odolnosti vůči konkrétním typům útoků.

Nejprve jsou rozebrány obecné principy DoS a DDoS útoků společně s informacemi o takzvaných botnetech a významných útocích za poslední dva roky. Následně se zaměřujeme na popis význačných DoS útoků, buď kvůli jejich rozšířenosti, nebo kvůli specifickým odlišnostem od ostatních používaných metod. Všechny popsané DoS útoky jsou reálnými příklady útoků, které byly v minulosti použity. Třetí kapitola představuje generátory provozu, které jsou klíčové pro vytvoření realistického testovacího prostředí. Potřebujeme být totiž schopni simulovat reálné útoky, abychom mohli efektivně vyhodnotit, jak jsou systémy v tomto ohledu proti útokům odolné. Nástroje pro generování tak musí být schopny generovat pakety určitou rychlostí, aby dokázaly simulovat realistický útok. Prostředí musí být schopné nejen vygenerovat daný typ útoku, ale také vygenerovat daný objem provozu.

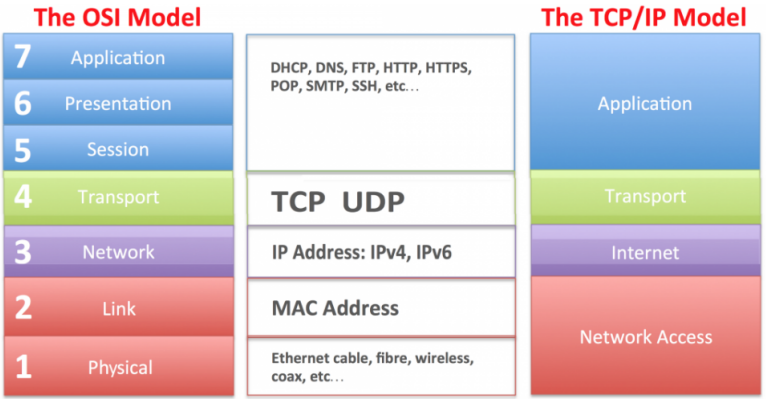
Čtvrtá kapitola se věnuje návrhu testovacího prostředí, zejména s ohledem na možné případy užití. Pátá kapitola pak popisuje samotnou implementaci nástroje a jednotlivých útoků. Poslední kapitola se věnuje testování a vyhodnocení výsledků.

Kapitola 2

Útok typu DoS a DDoS

Útok typu odepření přístupu ke službě, anglicky Denial of Service (zkráceně DoS), je typ útoku, při kterém útočník záměrně generuje síťový provoz, který nějakým způsobem vyřadí z provozu cílovou službu. Často se tak děje pomocí blokování přístupu ke zdrojům, které služba potřebuje ke správnému fungování (šířka pásma, výpočetní zdroje na serveru, ...). Nejčastěji k tomu dochází generováním provozu, který nutně nedává smysl, čímž dojde k vytížení zdrojů a legitimní uživatelé nebudou mít k dané službě přístup, protože všechny zdroje budou vyčerpané útočným provozem (tzv. flood). Kapitola čerpá z [54], [13], [55].

DoS útoky mohou cílit na různé vrstvy TCP/IP modelu. TCP/IP je implementovaná verze ISO/OSI modelu, která popisuje postup zpracování síťového provozu. Oba modely jsou zobrazeny na obrázku 2.1. Vrstvy jsou někdy označovány pouze písmenem L (zkratka z anglického Layer) a číslem vrstvy ISO/OSI. Například aplikační vrstva se označuje jako L7. Tato práce se ovšem zabývá pouze útoky, které jsou proveditelné napříč sítěmi, tzn. L3 a výše. Útoky na lokálních (L2) sítích, např. ARP spoofing, se práce nezabývá.



Obrázek 2.1: ISO/OSI a TCP/IP model [46].

2.1 Popis DoS útoku

První útok tohoto typu byl zaznamenán již v devadesátých letech minulého století [2]. I přesto jsou DoS útoky v dnešní době poměrně běžné a závažné. Příkladem je relativně nedávný útok na české banky [21]. Jedním z důvodů, proč jsou tyto útoky neustále používané, je i to, že není vždy jednoduché rozpoznat útočný a legitimní provoz. Úspěšný

útok může službu vyřadit z provozu na delší dobu a způsobit tak majiteli finanční ztráty, ale i újmu na reputaci. Například pokud dojde k výpadku internetového obchodu, zákazníci přejdou ke konkurenci, kde si zvyknou nakupovat a k původnímu obchodníkovi už se nevrátí, v důsledku čehož první obchodník přijde o odběratele [11].

Motivace

Agresor může mít mnoho důvodů, proč zahájit DoS útok. Nejčastějšími důvody jsou [35]:

- Politicky motivované útoky – Útoky, které útočník provádí se záměrem posílit vliv vybraných politických/státních subjektů. Případně mohou být použity v rámci takzvané kybernetické války. V poslední době tvoří převážnou část nahlášených útoků. Pravděpodobným důvodem nedávného nárůstu počtu politicky motivovaných útoků je i eskalace válečných operací na Ukrajině. Dále byly zaznamenány politicky motivované excesy na Izrael a Taiwan [22].
- Peníze a konkurenční boj – Motivací k DoS útoku mohou být mimo jiné i peníze. Například v přímé podobě, kdy agresor požádá o výkupné za to, že útok ukončí nebo že vůbec nezačne útočit. K obohacení útočné strany může dojít i nepřímo, například vyřazením z provozu stránky, nebo služby konkurenta kvůli čemuž v něj jeho klienti ztratí důvěru a rozhodnou se přejít k jinému poskytovateli. Nutno podotknout, že konkurence nemusí provádět útok přímo, ale může si ho zaplatit externě.
- Osobní důvody – V některých případech může být útok proveden jenom proto, že si člověk chce dokázat, že je toho schopen. K této motivaci můžeme zařadit i případy, kdy útočník zaútočí na server hostující online hru s cílem uškodit svému protivníkovi. Příklady v tomto odstavci by v některých případech mohly být zařazeny i ke konkurenčnímu boji, ale na rozdíl od předchozího se v tomto případě aktér nemusí vždy k útoku uchýlit z peněžních důvodu, ale i například z tužby dosažení lepšího skóre ve své oblíbené hře.

Reálné DoS útoky v posledních dvou letech

DoS útoky jsou i v dnešní době stále běžné. V této podkapitole budou zmíněny reálné případy významných útoků na státní i privátní organizace za poslední dva roky. Chceme tak demonstrovat aktuálnost této hrozby společně s nutností vyvíjet nové nástroje pro obranu proti stále se měnícím DoS útokům. Některé DoS útoky jsou činností organizovaných skupin.

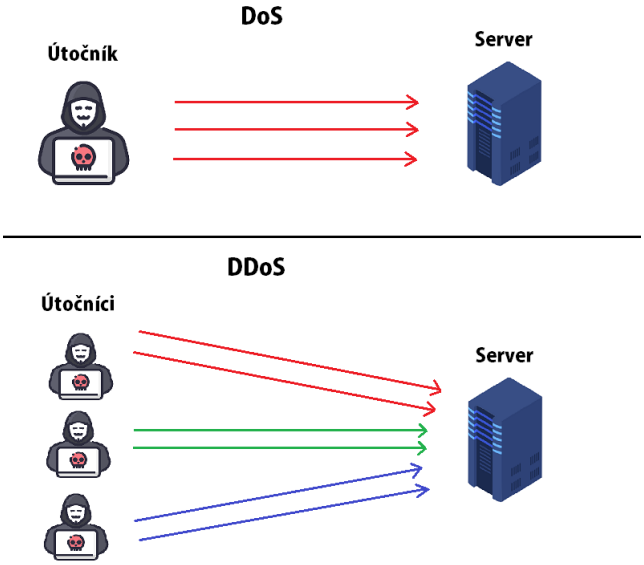
Proruská skupina KillNet je aktivní od ledna 2022 a má na svědomí několik nedávných kybernetických útoků. Sami útočníci tvrdí, že napadli více než dvě stě estonských webových stránek včetně estonského platebního systému ESTO AS. Skupina provedla i několik útoků na Litvu, kde zaútočila na online služby energetické společnosti Ignitis Group, která uvedla, že se jednalo o největší útok, kterému čelila za posledních patnáct let [22]. Kromě útoků na pobaltské státy, vyřadili z provozu stránky amerického kongresu [52] a federální systém placení daní v USA [19].

NoName057(16) je další proruskou skupinou. Svoji činnost začala v březnu 2022 a od té doby provedla několik kybernetických útoků na Ukrajinu, Spojené státy, Dánsko nebo i Českou republiku. Tato skupina je mimo jiné také zodpovědná za nedávné DoS útoky na české banky [21] nebo na stránky kandidátů na prezidenta republiky při volbách v roce 2023. Skupina se celkově zaměřuje na umlčení protiruských skupin [26].

Rozdíl mezi DDoS a DoS

V dnešní době mají síťové služby a linky k nim vedené celkem vysokou kapacitu, takže vydrží poměrně vysokou zátěž. Útok vedený pouze z jednoho zdroje tedy nemusí mít požadovaný účinek. Dnes se proto používá vylepšená verze útoku, distribuovaný útok odepření služby (Distributed Denial of Service) zkráceně DDoS. K provedení útoku se často využívá velké množství cizích zařízení, jak je zobrazeno na obrázku 2.2 dole. Na obrázku jsou rozdílné zdroje paketů naznačeny rozdílnými barvami.

Zařízení spolu při DDoS útoku spolupracují většinou nedobrovolně nebo bez vědomí jejich majitele. Kromě cizích strojů může agresor zkusit do iniciativy zapojit další podobně smýšlející jedince, kteří pak svá zařízení dobrovolně použijí k distribuovanému útoku. Další princip je v podstatě stejný jako u tradičního DoS útoku. Distribuovaná varianta navíc umožňuje koordinované zapojení všech zařízení ve stejnou chvíli. Pokud se útoku účastní více útočníků, mohou se domluvit na přesný čas zahájení. Jeden útočník může rovněž ovládat více zařízení. V takovém případě by všem strojům v pravou chvíli zaslal příkaz o začátku útočení. V obou případech dojde k zefektivnění útoku. Na oběť tak přichází větší množství provozu, než které by útočník byl schopen generovat sám. Z pohledu útočníka je tak útok efektivnější. Další výhodou DDoS oproti Dos útoku je náročná identifikace původu, jelikož útok přichází z různých zdrojů.

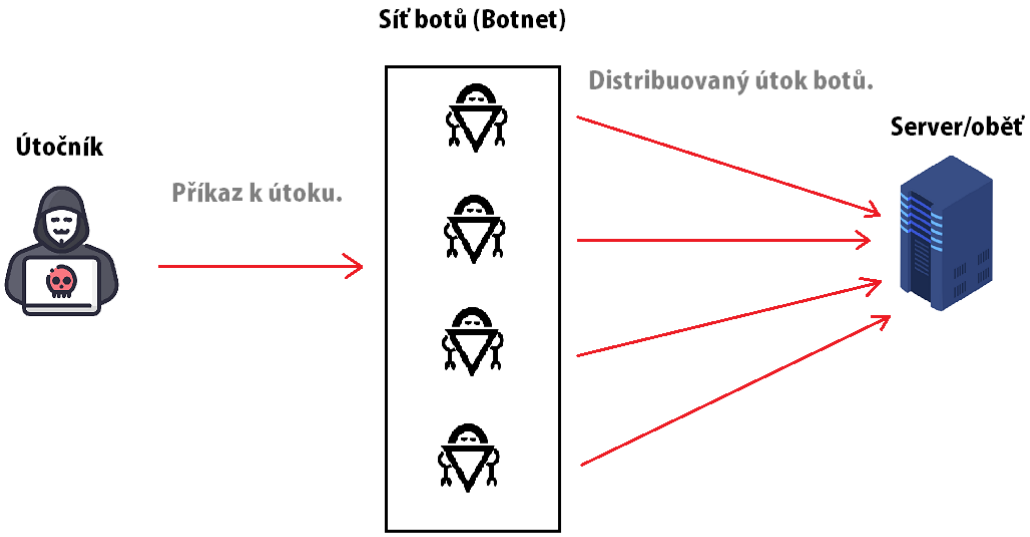


Obrázek 2.2: Vizualizace DoS útoku (horní část) a DDoS útoku (spodní část).

Botnet

Výrazem botnet se označuje síť tvořená zařízeními, která jsou řízena z hlavního počítače a koordinovaně provádějí útok na vybranou oběť, jak je zobrazeno na obrázku 2.3. Obrázek ukazuje zjednodušený průběh útoků pomocí botů, kdy útočník vydá všem nakaženým zařízením příkaz o zahájení útoku. Každý bot pak začne generovat provoz na oběť, čímž dojde k DDoS.

V botnetu jako takovém jsou zařízení většinou nedobrovolně. K připojení zařízení do sítě botů dochází většinou po napadění zařízení škodlivým software, který útočník distribuuje za



Obrázek 2.3: Útok pomocí sítě botů.

účelem vytvoření botnetu. Bezprostředně po nakažení nutně nedochází k aktivaci malware. Vlastník zařízení tak nemusí být schopen nakažené zařízení ihned odhalit, protože do doby než útočník vydal povel k útoku, nezpůsoboval škodlivý program žádné problémy.

Jakmile chce útočník na vybranou oběť zaútočit větší silou, využije k tomu vytvořený botnet z předem infikovaných zařízení. Součástí botnetu mohou být klasické počítače, malá zařízení pro internet věcí (IoT) nebo virtuální stroje na vzdálených serverech (cloud) [18].

V poslední době se začaly do botnetů, kromě klasických počítačů, nedobrovolně zapojovat i zařízení IoT. Jedním z důvodů časté účasti v botnetech je jejich velký počet a horší zabezpečení. Zařízení pro chytré domácnosti jsou v dnešní době hojně využívány, ale nejsou vždy správně nakonfigurované, protože jejich vlastníci nemusí mít vždy povědomí o správě sítě a všech nebezpečích, která mohou ohrozit nejen jejich zařízení.

Vzdálené servery (cloud), mají velkou výpočetní sílu, díky které na nich lze provozovat velké množství menších virtuálních strojů. Ty mohou být útočníkem využity namísto klasického botnetu nebo sítě z IoT zařízení. Botnety, které využívají možnosti cloudu a používají HTTP/2 dokážou generovat až pět tisíckrát větší sílu na jeden uzel, což umožňuje spustit silný útok i s menším počtem zařízení v síti (jen přibližně 5-20 tisíc uzlů). Pro srovnání botnety využívající IoT zařízení, složené z milionů botů jsou sotva schopny dosáhnout nízkých jednotek požadavků (requestů) za sekundu. Oproti tomu útoky, které využívaly virtuální stroje na vzdálených serverech, dokázaly generovat až 201 milionů požadavků za sekundu [64].

Příklady botnetů

Botnety jsou důležitou součástí DDoS útoků. Pro potřeby práce je užitečné znát parametry reálných útoků (počet využitých zdrojů, počet paketů za sekundu, ...), abychom je následně dokázali simulovat ve vlastním prostředí. Jelikož jsou botnety často využívány k útokům, je důležité znát i jejich vlastnosti. Všechny tyto informace nám přispějí k lepšímu modelování

reálné hrozby pro testovanou síť a pomohou tak k vytvoření realistického generátoru DDoS útoků.

V následující části boudou krátce představeny některé známé botnety, jejichž pojmenovávání většinou pochází z malware, který je použit pro jejich ustavení. Tabulka 2.1 shrnuje popisované informace v několika klíčových bodech.

Storm botnet

První útok za použití této sítě byl zaznamenán v lednu 2007 kdy bylo pod útokem několik anti-spam sledovačů. Během stejného roku botnet zaútočil na několik dalších webových serverů. K roku 2007 se velikost botnetu odhadovala přibližně na padesáti milionů zařízení. Oproti ostatním botnetům té doby Storm implementoval celou řadu novinek, například šifrování nebo detekci virtuálních strojů. Storm cílil na stroje s operačním systémem Windows. Byl distribuován pomocí trojského koně nejčastěji v příloze nevyžádaného e-mailu. Jakmile došlo k nakažení, začalo zařízení posílat nevyžádanou poštu, stahovat nástroje potřebné k útočení a následně útočit [35].

Mirai

Původním záměrem tvůrců bylo zaútočit na servery hry Minecraft, jejímž správcům pak mělo být nabídnuto řešení pro ochranu proti DDoS. První útok byl zaznamenán v roce 2016 společností Cloudflare. Mirai malware pracuje tak, že na internetu vyhledává IoT zařízení běžící na specifickém typu procesoru ARC. Procesor využívá odlehčenou verzi operačního systému Linux. Pokud správce nezměnil výchozí přihlašovací jméno a heslo, Mirai se byl schopen připojit k zařízení a napadnout ho. Z originálního malware vzniklo několik mutací. Stále se tak i po odhalení původního botnetu využívají jeho alternace [59] [60].

Meris

Jméno botnetu vychází z lotyšského slova pro mor. První útok sítě Meris se objevil v červnu 2021. Prvotní měření ukazovaly, že se skládá z třiceti až padesáti tisíc zařízení. Reálný počet se ale odhadoval až na 250 000. Síť Meris se skládala z nakažených směrovačů a dalšího síťového hardware od lotyšské firmy MikroTik, které byly napadeny přes zranitelnost v operačním systému [9].

Mantis

První útok byl zaznamenán v červnu 2022. Mantis se skládal přibližně z pěti tisíc zařízení. Dokázal vygenerovat 26 milionů požadavků za sekundu, což je v průměru 5 200 HTTPS požadavků za sekundu na jednoho bota. Takové množství provozu je těžké vygenerovat i za použití nezabezpečeného HTTP. Bylo tak překvapením, že se takové síly povedlo dosáhnout i s režií TLS (Transport Layer Service), neboli režií šifrovaného spojení. Mantis místo IoT používá virtuální stroje, i proto dokázal vytvořit nečekaně velké množství provozu. Mantis se vyvinul z Meris botnetu [63].

Název	Rok vzniku	Zranitelnost	Počet zařízení
Storm botnet	2007	Windows	50 milionů
Mirai	2016	ARC procesor	145 tisíc
Meris	2021	MikroTik směrovače	250 tisíc
Mantis	2022	Využití virtuálních strojů	5 tisíc

Tabulka 2.1: Shrnutí informací o botnetech.

2.2 Vektory útoků

Pojem vektor útoku (angl. attack vector) se používá pro označení metody, kterou útočník použil k DDoS útoku. Může například specifikovat, které atributy paketu byly nastaveny. Případně popisuje další parametry specifické pro konkrétní typy útoku [62]. V dnešní době se často využívají i útoky, které používají více různých vektorů, takzvaný multi-vector útok. DDoS útoky se také začaly používat k odvedení pozornosti, ať už od jiného typu útoku, kterým například útočník získá přístup do systému, nebo i k zastínění jiných nelegálních činností.

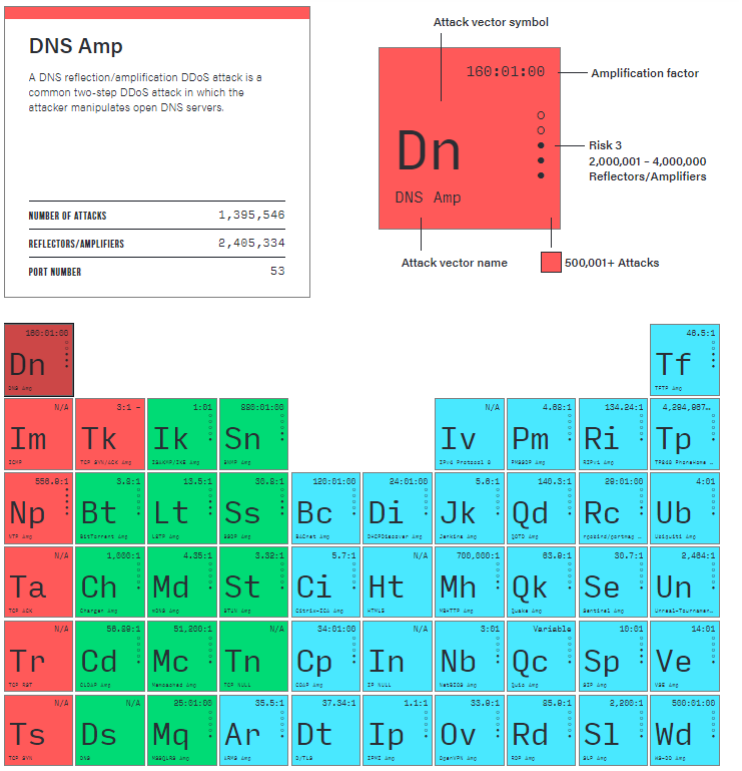
Existuje nepřeberné množství různých variant a typů DDoS útoků. Společnost Netscout pravidelně vydává analýzu trendů v útocích za uplynulé období [4]. Součástí jedné analýzy je i „periodická tabulka“ nejpoužívanějších vektorů útoků, obrázek 2.4. V tabulce je barevně odlišeno, jak moc jsou útoky daného typu časté. Při kliknutí na jednotlivé vektory, je pak možné se dozvědět doplňující informace, princip útoku, nebezpečí, a podobné. Na obrázku je společně s tabulkou vidět i detail tzv. DNS amplifikace. I tento útok je detailněji popsán v následující podkapitole.

V této části práce se snažíme důkladně pochopit a popsat významné DDoS útoky používané v dnešní době. Věnujeme se tak útokům, které představují v nynějších sítích reálnou hrozbu. Cílem práce je vytvořit rozšiřitelné prostředí, které dokáže simulovat realistické DDoS útoky. Z hlediska praktického použití vytvářeného nástroje je tak rozumné se věnovat právě těmto reálně používaným útokům. Nástroj pak umožní testovat odolnost síťových prvků a reálné síťové infrastruktury vůči takovým útokům.

2.2.1 Útok odrazem

Útok odrazem (angl. reflection attack) je jedním z nejčastějších DoS útoků. Jednodušší a více používaná varianta útoku je realizována pomocí UDP paketů. Útok lze provést i pomocí TCP rámců [30].

Průběh útoku je následující a je demonstrován i na obrázku 2.5. Útočníkem je vytvořen falešný paket, ve kterém změní zdrojovou adresu. Ta je ve výchozím stavu nastavena na útočníkovo zařízení. Svoji zdrojovou adresu nahradí adresou zařízení, na které probíhá útok. Zmíněné technice se v literatuře říká spoofing (na obrázku je naznačeno oddělenými barvami IP adres, útočníkova červená, modrá pro službu a zelená adresa oběti). Takto vytvořený paket je následně odeslán na veřejně dostupnou službu, například DNS server. Služba paket zpracuje a odešle odpověď. Protože byla přenastavena zdrojová adresa, není paket odeslán zpět k útočníkovi, ale oběti, která o data nežádala. Pokud budeme generovat dostatečný provoz, docílíme tím zahlcení oběti. Navíc bude provoz nerozpoznatelný od legitimního provozu služby. Toto znesnadňuje detekci původu útoku.



Obrázek 2.4: Periodická tabulka vektorů útoku [4].

Amplifikace

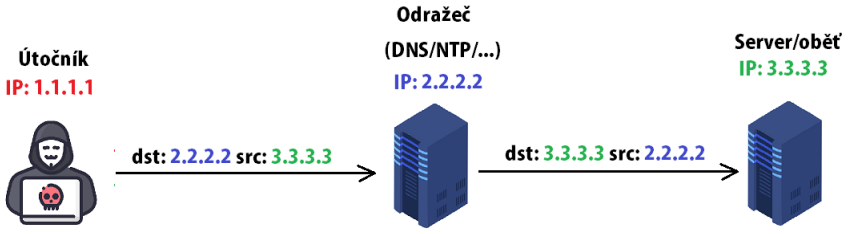
Amplifikační útok je vylepšený/silnější druh útoku odrazem, obrázek 2.6. I zde si útočník nejprve vytvoří podvržený paket se změněnou zdrojovou IP adresou na adresu oběti. Rámec je opět poslán na vybraný odražeč. Hlavní rozdíl je v odeslaném paketu a chování veřejného serveru. Paket útočníka je připraven tak, aby odražeč donutil poslat odpověď, která je ale řádově větší, tj. obsahuje více dat, než kolik bylo přijato. Provoz je následně odeslán na oběť. Útočník tak i s menším výkonem dokáže vygenerovat poměrně silný útok. Na obrázku je demonstrováno nepoměrem šipek od útočníka k serveru a z odražeče k oběti. Šipky v obrázku demonstrují směr toku provozu.

Amplifikační faktor

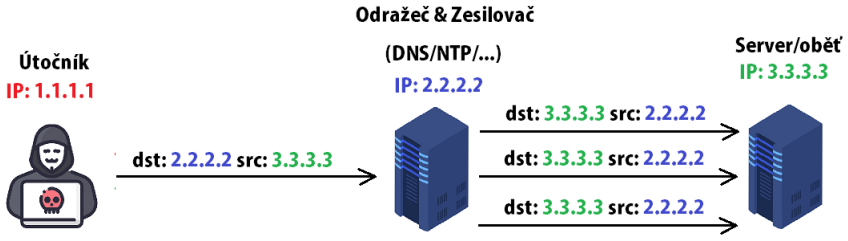
Důležitou metrikou pro amplifikační útoky je takzvaný amplifikační faktor. Reprezentuje poměr mezi velikostí požadavku a velikostí odpovědi. Tedy čím větší amplifikační faktor, tím méně zdrojů bylo útočníkem vynaloženo k dosažení stejně silného útoku. Máme-li například systémy s amplifikačními faktory 50 a 100, útočník využívající druhý systém může generovat dvakrát slabší provoz za dosažení stejného výsledku, jako při použití prvního systému [5].

DNS amplifikace

Domain Name System (DNS) je jednou ze základních komponent internetu. Zajišťuje především překlad IP adres na doménová jména, čímž usnadňuje práci běžným uživatelům internetu. Bez této služby by všechny webové služby musely být hledány pomocí těžko



Obrázek 2.5: Útok odrazem.



Obrázek 2.6: Zesílený útok.

zapamatovatelné IP adresy. Komunikace probíhá pomocí bezstavového protokolu UDP na portu 53. Jelikož se jedná o službu, která fungovala už v počátcích internetu, protokol není navržen tak, aby zvládal zlomyslné uživatele. I proto není těžké DNS zneužít. Útočníkům pomáhá i to, že služba je potřebná a běžná. Není tak jednoduché odlišit legitimní komunikaci od škodlivé.

Útok většinou probíhá tak, že si útočník nejprve najde vhodný DNS server, který umožňuje odpovídat na rekurzivní požadavky. Špatně konfigurované DNS servery často dovolují tyto typy dotazů z jakéhokoli zdroje. Jakmile je takový server nalezen, útočník pošle požadavek o přesun zón, který obvykle generuje hodně dat v odpovědi. Jelikož byl požadavek poslán s podvrženou zdrojovou adresou, tj. útočníkova adresa je nahrazena adresou oběti, začne k oběti přicházet velké množství dat, které mohou způsobit výpadek služby. Velikost odpovědi může být až sedmdesátkrát větší než velikost původního požadavku. Při zneužití DNSSEC lze dosáhnout až padesátkrát většího amplifikačního faktoru [37]. Nahrazením své zdrojové IP adresy adresou oběti si navíc útočník zajišťuje jistou míru anonymity [5].

NTP amplifikace

Network Time Protocol (NTP) je síťový protokol, který slouží k synchronizaci času mezi zařízeními. Synchronizace času není důležitá pouze pro uživatele, aby věděl kolik je přesně hodin, ale také pro správné fungování některých aplikací. V minulosti byl zaznamenán útok, který zneplatňoval certifikáty pouze posunem času. Útočník byl schopen zahájit nezabezpečenou komunikaci a napadnout službu. Samotný DoS pomocí NTP amplifikace je prováděn za pomoci zneužití příkazu monlist. Tímto dotazem jsou od serveru požadovány IP adresy posledních 600 synchronizovaných zařízení. Příkaz monlist je povolen zejména na starších

NTP serverech. Pokud útočník takový server zneužije, může dosáhnout amplifikačního faktoru až 200 [27].

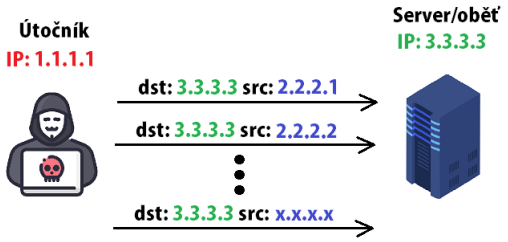
Memcached amplifikace

Memcached je výkonný systém pro ukládání dat do mezipaměti. Většinou se používá pro zvýšení výkonu aplikace pomocí přednačítání dat. Může ukládat data jako mezivýsledky databázových dotazů nebo i celé přednačtené HTML stránky [24]. Memcached amplifikace funguje na stejném principu jako ostatní amplifikace. Útočník pošle paket s upravenou zdrojovou IP adresou. Zároveň se jedná o typ zprávy, která donutí server poslat v odpovědi co nejvíce dat. Ty jsou potom, v důsledku podvržení IP adresy, odeslány oběti. Dojde tak k výraznému zvětšení poměru odeslaných dat od útočníka a dat, které na oběť přijdou. Amplifikační faktor této metody může být až 51 200 [23]. Memcached amplifikace je proto zodpovědná za největší útoky poslední doby [2].

2.2.2 Záplavové útoky

Dalším poměrně hojně využívaným typem útoků jsou záplavové útoky, flood attacks. Principem útoku je zaplavit uživatele kvantem paketů, které způsobí přetížení linky nebo jiného zdroje. Obrázku 2.7 ukazuje jak útočník posílá velké množství paketů na server, které ho zahltí. Útočník většinou navíc používá techniky spoofing, která znesnadňuje administrátorům útok odstínit, protože přichází z většího množství adres. Na obrázku je naznačeno různými zdrojovými IP adresami (v obrázku modrou barvou).

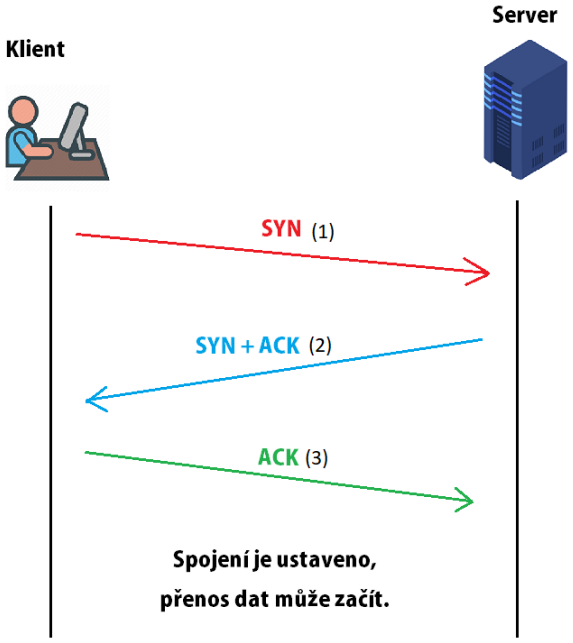
Častým cílem, z pohledu provedení, jsou vlastnosti protokolů, kdy útočník zneužije předepsané fungování určitého síťového standardu.



Obrázek 2.7: Průběh obecného flood útoku. Útočník zaplaví server pakety a ten pak není schopen odpovídat legitimním klientům.

Jeden z principů zneužívá 3-way-handshake, který se používá při ustavování TCP spojení. 3-way-handshake je vizualizován na obrázku 2.8 a zjednodušeně funguje následovně:

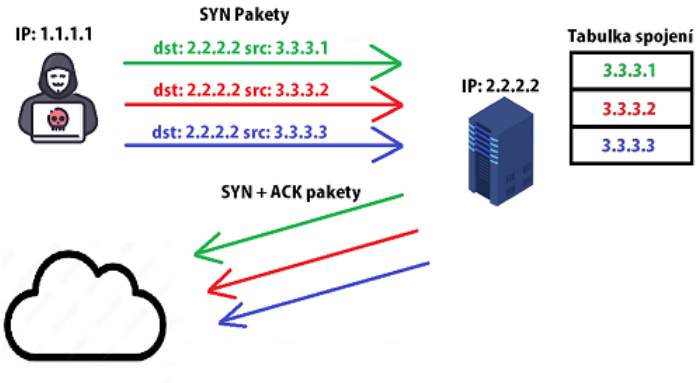
1. Klient, který chce zahájit spojení, pošle serveru SYN paket (červená šipka).
2. Server odpoví paketem se SYN-ACK příznakem (modrá šipka).
3. Jakmile přijde odpověď zpět k žadateli o spojení, odešle ACK paket, po jehož přijetí je spojení považováno za ustavené (zelená šipka).



Obrázek 2.8: Ustavení TCP spojení mezi klientem a serverem.

Záplava synchronizačními pakety

Záplava synchronizačními pakety (angl. SYN Flood) je DoS útok, který se snaží vytížit všechny zdroje služby a tím jí zabránit ve vytváření dalších spojení. Noví klienti tak nebudou moci službu využívat (stará spojení zůstanou dále otevřena). Postup útoku je zobrazen i na obrázku 2.9. Útok využívá vlastností TCP protokolu při navazování spojení. Posílá podvržené pakety s příznakem SYN. Serveru se provoz jeví jako legitimní uživatel, který chce zahájit komunikaci. Alokuje potřebné zdroje a odešle příslušnou odpověď. Následně čeká až přijde potvrzení o přijetí nebo do vypršení nastaveného časového limitu. V případě útoku však potvrzení nikdy nedorazí a server je nucen spojení ukončovat až po uplynutí čekací lhůty. Když útočník začne posílat velké množství falešných paketů, zahltí tím zdroje serveru, který pak není schopen odpovídat legitimním uživatelům. Někdy se spojením, která ještě nejsou zcela ustavena, tj. jsou jen napůl vytvořena, anglicky říká half-open, tudíž se v literatuře můžeme setkat s názvem half-open attack pro zde popisovaný SYN Flood. Při použití SYN Flood dokáže útočník dosáhnout zablokování služby s menším množstvím vygenerovaného provozu než u jiných, například volumetrických, DDoS útoků. Na rozdíl od nich se totiž nesnaží přímo zahltit kapacitu přípojené síťové linky, ale jen vyčerpat zdroje služby. Pokud získá informace o nastavení serveru, může svůj útok lehce přizpůsobit a dosáhnout tak stejného výsledku za méně práce [43].



Obrázek 2.9: Průběh SYN flood útoku. Útočník zaplaví server synchronizačními pakety a ten pak není schopen odpovídat legitimním klientům.

Útočník může k útoku přistoupit třemi různými způsoby:

- Přímý útok: Adresa není podvržena, ale použije jen svoji vlastní. Útok je prováděn pouze z jednoho zařízení. Útočníkem musí být zajištěno, aby odpovědi serveru nebyly zpracované nebo vůbec nepřišly. Ustavilo by se tím spojení, což není v případě útoku žádoucí. Další nevýhodou je poměrně jednoduchá obrana proti takto vedenému útoku. Na straně serveru lze zablokovat pouze útočníkovu adresu. Proto se většinou útok tímto způsobem nepoužívá.
- Útok s podvrženými pakety: Útočník změní zdrojovou IP adresu, čímž může poměrně jednoduše vytvořit větší množství polootevřených spojení.
- DDoS: Posledním a asi nejhůře identifikovatelným způsobem užití je distribuovaný útok, kdy je k provedení útoku použito větší množství zařízení. Pakety mohou a nemusí být podvržené, neboť každé zařízení má jinou IP adresu. Útočník neposílá útočné pakety přímo ze své IP adresy, ale přes prostředníky v podobě botů. Přímo z útoku tak nedokážeme získat přesné informace o útočníkovi.

Záplava ICMP pakety

DDoS útok pomocí ICMP (Intenet Control Message Protocol), kterému se také někdy říká Ping Flood, je typ útoku, kdy se útočník snaží zaplavit oběť ICMP echo-request pakety a znemožnit komunikaci s legitimními uživateli. ICMP je protokol síťové vrstvy. Používá se především k diagnostice sítě, například pomocí nástrojů traceroute a ping. Při standardním fungování sítě, jsou echo-request a echo-reply zprávy použity k diagnostice spojení mezi zařízeními.

Záplava ICMP se řadí mezi volumetrické útoky, protože existuje přímá úměra mezi počtem požadavků útočníka a paketů příchozích na oběť. Proto se většinou používá pouze se sítí botů, kde je výsledná síla útoku rovna součtu provozu vygenerovaného všemi zařízeními v botnetu. Záplava pomocí ICMP byla útočníky dříve hojně využívána. Dnes se útok sám o sobě tolik nepoužívá, většinou však doprovází jiné útoky a společně s nimi vytváří komplikovanější útoky, které je náročnější odstínit. V takovém případě se pak jedná o multi-vector útok.

Samotný DDoS pomocí ICMP záplavy nepatří mezi nejnebezpečnější DDoS útoky, neboť není tak složité ho odhalit. Může ovšem způsobit problémy pokud k němu dojde ve spojení s dalšími DDoS útoky [58] [29].

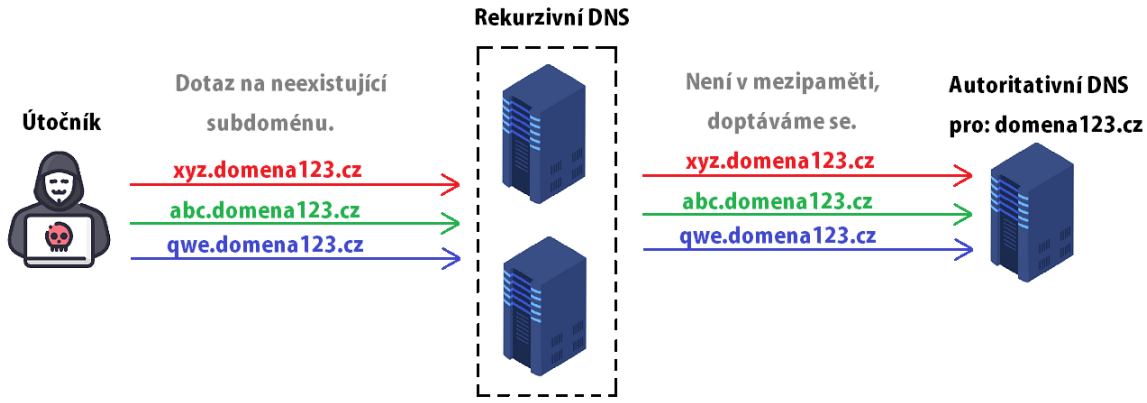
DNS záplava dotazů

Jedná se o speciální typ záplavového útoku. V literatuře se můžeme setkat s označením DNS Water Torture nebo také NXDomain útok. Poslední je odvozeno od odpovědi DNS serveru (NXDomain), která značí, že se požadovanou doménu nepodařilo najít.

Název útoku vychází ze středověké mučící techniky, kdy trestanému člověku na obličej nepravidelně padaly kapky vody do obličeje, čímž byl připraven o příčetnost. V případě síťového útoku jsou kapky vody nahrazeny za subdomény. Útočník před skutečnou adresu cíle vloží nové znaky, čímž vytvoří neexistující subdoménu, například v případě útoku na doménu vutbr.cz vytvoří asdfg.vutbr.cz. Podobných náhodných domén vytvoří větší množství a následně se na ně bude ptát DNS serveru. Protože domény neexistují, DNS server poskytovatele pro ni nedokáže najít odpovídající překlad. Začne se proto rekurzivně doptávat dále. Útok nemusí být v první chvíli patrný, neboť DNS servery si po určitou dobu udržují již jednou hledané adresy. To znamená, že když se uživatel připojí na stránku vutbr.cz před útokem a následně dojde k útoku, nemusí okamžitě dojít k výpadku přístupu k webu [6].

DNS Water Torture je v podstatě další formou volumetrického útoku, který může poškodit nejen cílovou doménu, ale i poskytovatele internetu, který zajišťuje mimo jiné i přístup k překladu adres. Útok samotný se poměrně špatně odráží, neboť DNS dotazy jsou důležitou součástí internetového provozu [53].

Obrázek 2.10 vizualizuje průběh DNS Water Torture útoku. Útočník chce napadnout doménu domena123.cz, bude tedy generovat nové náhodné řetězce (na obrázku naznačeny názvy xyz, abc a qwe), které pak použije jako subdoménu, například xyz.domena123.cz. Rekurzivní DNS tyto nové domény nezná a pošle dotaz na autoritativní server pro doménu domena123.cz.

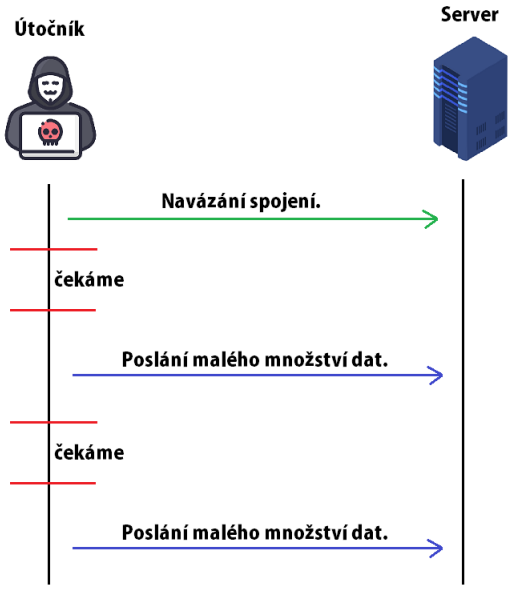


Obrázek 2.10: Ukázka útoku DNS Water Torture.

2.2.3 Pomalé útoky

Pomalé útoky, angl. low and slow attacks, jsou útoky, které na rozdíl od ostatních způsobují výpadek serveru tím, že posílají pakety velice pomalu. V podstatě dojde k zahájení

velkého počtu spojení se serverem, ale provoz je posílán jen takovou rychlostí, aby nedošlo k přerušení spojení. Útočná strana tak na delší dobu vyřadí všechny zdroje serveru na nesmyslnou činnost, tj. přijímání útočných paketů, místo obsluhování legitimních klientů, kteří kvůli tomu nemohou přistoupit ke službě. Na rozdíl od volumetrických útoků, nepotřebují pomalé útoky velkou šířku pásma. Díky tomu není nutné vytvářet velké sítě botů, které by útočily společně. Low and slow útoky se špatně odhalují, neboť většina nástrojů při DDoS útoku počítá s velkým přívalem paketů. Proto nedokáží pomalý provoz odlišit od klasické síťové komunikace [56]. Vizualizaci k fungování pomalých útoků poskytuje obrázek 2.11. Zelenou šipkou je naznačeno vytvoření spojení (pomocí 3-way-handshake). Následně dochází k čekání před odesláním další malé části dat (modré šipky). Čekání je naznačeno červeným ohraničením na ose útočníka.



Obrázek 2.11: Pomalý útok ukázán na jednom spojení. Pří reálném útoku by útočník vytvořil takovýchto spojení více.

Nejpopulárnějšími nástroji pro low and slow útoky jsou Slowloris a R.U.D.Y. Oba jsou popsány v následujících odstavcích.

Slowloris

Nástroj získal název po asijském primátovi slow loris, česky outloň. Útok pomocí Slowloris nástroje cílí na aplikační vrstvu. Snaží se na vybraném serveru vytvořit a udržet co nejvíce souběžných HTTP spojení. Spojení se snaží udržet co možná nejdéle aktivní, čehož je dosaženo tak, že jsou pro každé ustavené spojení posílány pomalé HTTP požadavky na server. Požadavky jsou ideálně generovány s co nejdelším odstupem, který musí být přiměřeně dlouhý, aby server samovolně spojení neukončil. Útočník se tedy snaží poslat paket těsně před tím, než časovač předčasně ukončí spojení. Útok využívá vlastnosti HTTP protokolu rozdělit GET a POST požadavky do více paketů. Server dokáže souběžně otevřít jen určitý počet spojení. Slowloris se snaží ustavit tolik spojení, kolik server dovoluje a vyčerpat tím kapacity pro legitimní uživatele. Server se ke spojením, která založil útočník chová stejně jako k legitimním, protože nemá jak rozeznat, jestli se jedná o útok, nebo jestli uživatel

nemá dostatečně rychlé spojení. Bude tedy čekat, až budou odeslána všechna data a spojení se řádně ukončí, k čemuž však v případě útoku nemusí dojít. Slowloris může být použit i pokud má útočník k dispozici jen malou šířku pásma, protože i v takovém případě se jedná o velice efektivní způsob útoku na webové servery. Provoz nevypadá příliš podezřele a dokáže tak obejít některé obrané mechanismy. I proto může útok probíhat delší dobu, než dojde k jeho odhalení [57] [41].

R.U.D.Y.

Jedná se o další nástroj pro low and slow DoS. Název nástroje R.U.D.Y. je zkratkou „R-U-Dead-Yet?“ (česky: „Už jsi zemřel?“). Vychází z alba finské death metalové skupiny. Ve výsledku pracuje na podobném principu jako Slowloris. Ustaví velké množství spojení, které velmi pomalu obsluhuje. R.U.D.Y. navíc postupuje tak, že na cílovém webu nejprve vyhledá formuláře, které následně zneužije ke svému úkonu. Jakmile se nástroji podaří lokalizovat vhodné formuláře, pošle speciálně vyrobený paket. Hlavní odlišností od klasického provozu je, že u hlavičky pro určení délky přenášené zprávy (Content-Length) nastaví velkou hodnotu. Příjemci tak oznamuje, že zpráva, kterou bude útočník posílat obsahuje hodně dat. Útok opět využije vlastnosti HTTP protokolu, rozdělit POST požadavek na více paketů. Data budou odeslány po co nejmenších částech v největších možných rozestupech tak, aby nebyl přenos ukončen od časovače serveru. V některých případech proto mohou být data zasílána po jednom byte s rozestupy v řádu desítek sekund mezi každým paketem. Útočník udržuje co největší množství souběžných spojení a na každém z nich pomalu posílá data. Pokud si útočník správně pohlídá rozestupy mezi pakety, velice znepříjemní práci jakýmkoli zařízením, které mají DDoS útokům bránit. Na chráněnou síť tedy nejde extrémně velký provoz, ale pouze požadavky od velmi pomalých klientů, což se v praxi může běžné stát.

Existují i mutace původního nástroje R.U.D.Y., které přidávají další vylepšení, zejména pro ztížení jejich detekce. Používá se například přidání náhodnosti do velikosti intervalu mezi odesílanými pakety nebo znovupoužití metadat sezení (cookies), pro zvýšení zdání legitimity provozu [39] [38].

Srovnání Slowloris a R.U.D.Y.

Webové servery alokují zdroje jakmile je uživatelem deklarováno, že potřebuje danou službu využívat, tedy v době, kdy je odeslán požadavek na server. Zdroje zůstanou alokované dokud je spojení otevřeno, což je zranitelnost, kterou oba nástroje (Slowloris a R.U.D.Y.) využívají. Výhodou obou je navíc, že jim pro úspěšný útok stačí pouze malá šířka pásma a nepotřebuji tudíž vytvářet velké botnety. V některých případech stačí pouze jedno zařízení.

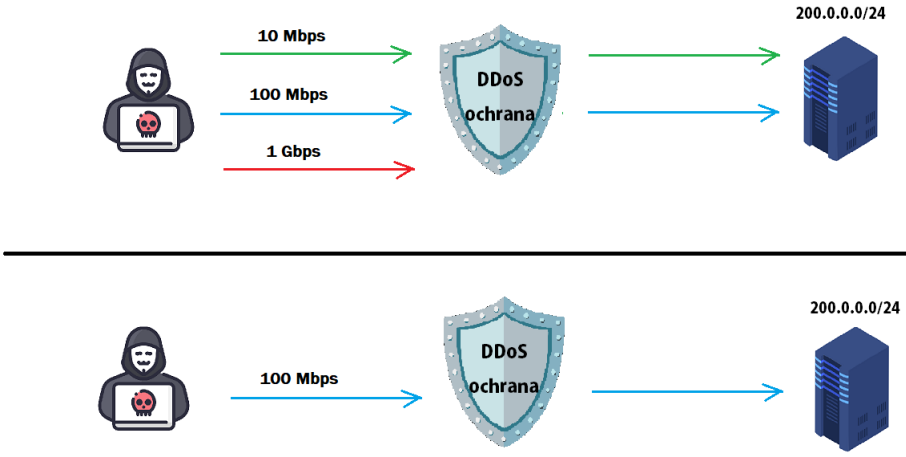
Oproti záplavovým útokům musí útočník zadat správnou zdrojovou adresu, aby došlo k navázání spojení. Neposkytuje mu tak v tomto ohledu žádnou anonymitu. Hlavní rozdíl mezi samotnými nástroji je pouze v tom, že Slowloris využívá požadavku GET, tedy naváže spojení, ale nikdy ho nedokončí. R.U.D.Y. na druhou stranu používá požadavek POST, kdy naváže spojení a nedokončí posílání zprávy. Jedním z rozdílů mezi GET a POST požadavkem je, že při GET požadavek obsahuje pouze hlavičku, zatímco POST nese i data [3].

2.2.4 Ostatní útoky

V této části se budeme zabývat útoky, které přímo nezapadají do žádné z předešlých kategorií. Neznamená to ovšem že by byly méně používané. Například dále popisovaný útok Carpet Bombing zaznamenal podle společnosti Netscout nárůst v použití [4].

Carpet Bombing

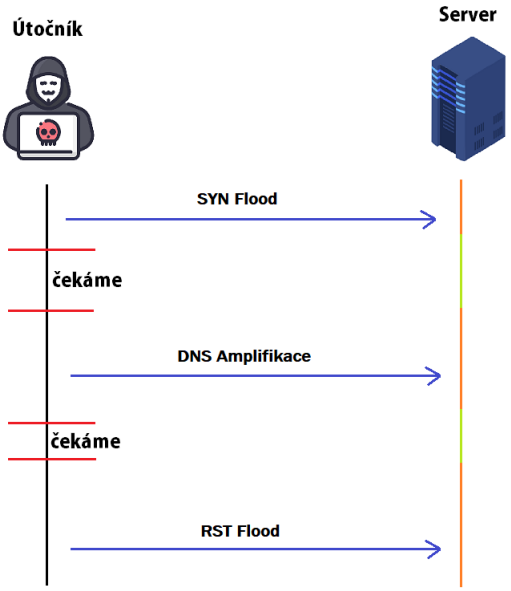
Carpet Bombing je útok, který cílí na celou podsíť. Od ostatních útoků se liší i tím, že počítá s ochranou sítě. V první fázi proto pouze testuje nastavené prahové hodnoty použité DDoS ochrany. Následně se na přiměřenou dobu odmlčí a se získanými informacemi zaútočí na celou podsíť. Generuje se několik toků paketů. Rychlost všech se pohybuje těsně pod zjištěným prahem z první fáze. Popsaným mechanismem dokáže efektivně obejít nastavenou DDoS ochranu [36]. Na obrázku 2.12 je naznačeno, jak se útočník snaží odhadnout nastavený práh DoS ochrany. Poté, co nalezne vhodnou rychlost, zaútočí na síť přiměřenou silou. Vhodná rychlost je taková, která je dost velká, aby způsobila DoS, ale zároveň musí být i dostatečně malá, aby ji DoS ochrana nezaznamenala a nezablokovala. V příkladu na obrázku útočník zvolí rychlost 100 megabitů za sekundu.



Obrázek 2.12: Carpet Bombing útok.

Pulse Wave

Pulse Wave útok na rozdíl od ostatních útoků, které posílají provoz nepřetržitě, funguje tak, že generování provozu na chvíli zastaví a následně opět spustí. V grafech poté vytížení vypadá jako pulsy. Útok může navíc mezi pauzami změnit vektor útoku, čímž se stává ještě nepředvídatelnějším [12]. Na obrázku 2.13 je naznačeno jak tento útok funguje. Útočník posílá různé útoky (modré šipky). Mezi jednotlivými útoky ale čeká vždy jinou dobu, což je naznačeno různou délkou červených intervalů. Útok a pauza mezi útoky jsou také reprezentovány na časové ose pro server, oranžovou při probíhajícím útoku a zelenou při pauze.



Obrázek 2.13: Pulse Wave útok.

Kapitola 3

Generování síťového provozu

Kapitola krátce představuje dostupné generátory síťového provozu. Schopnost generování síťového provozu je pro tvorbu testovacího prostředí klíčová. Potřebujeme simulovat reálné útoky, abychom mohli vyhodnotit, odolnost cílových systémů proti jednotlivým typům útoků. Pro vytvoření testovacího prostředí je nezbytné použít generátor, který dokáže posílat pakety dostatečnou rychlostí, aby bylo možné realisticky simulovat záplavové útoky. Pro specifické typy útoků, například pro pomalé útoky, potřebujeme nejprve navázat TCP spojení se serverem. Generátor tak musí nutně umožňovat stavové generování síťového provozu a do jisté míry implementovat TCP/IP stack. Stavové generování nám umožňuje vytvářet provoz, který dokáže udržet kontext/stav. Generátor tak v podstatě zároveň generuje provoz i simuluje odpověď na zprávy, například při navazování TCP spojení.

Generátor síťového provozu je nezbytný nejen ke generování samotného útoku, ale také k simulování legitimního provozu. Pro otestování obrany proti DoS je totiž vhodné začlenit do testovacího scénáře i legitimní síťovou komunikaci. Simulováním legitimního provozu můžeme sledovat další statistiky, jako je například počet úspěšně navázaných spojení legitimním klientem. V případě útoku pak můžeme sledovat snižování úspěšnosti navazování spojení nebo také nárůst latence v průběhu navazování spojení.

V rámci této kapitoly jsou popsány tři open-source generátory síťového provozu a dva komerční nástroje. Dále jsou také popsány dostupné otevřené softwarové nástroje umožňující generování konkrétních DDoS útoků.

3.1 Pktgen

Pktgen je software postavený nad knihovnou DPDK [7] sloužící pro obecné generování síťového provozu. DPDK je otevřený framework, poskytující prostředí pro rychlé zpracování paketů. Vysoká výkonnost je dosažena díky možnosti přímé komunikace uživatelské aplikace se síťovou kartou, čím se do jisté míry obchází jádro operačního systému. Takto je výrazně snížena režie přepínání kontextu mezi userspace (kontext uživatelské aplikace) a kernelspace (kontext jádra operačního systému).

Podle oficiální dokumentace [47] dokáže Pktgen generovat provoz rychlostí 10 gigabitů za sekundu i na nejmenších 64bajtových paketech. Může figurovat buď jako generátor provozu, nebo jako analyzátor příchozích paketů. Pktgen pracuje v bezstavovém režimu, poskytuje metriky ze vstupních a výstupních portů v reálném čase a dokáže pracovat s nejpoužívanějšími síťovými protokoly, jako UDP, TCP, ICMP, ARP nebo MPLS. Pktgen aplikaci je

možno konfigurovat pomocí skriptu v jazyce Lua, díky čemuž je možné vytvořit reprodukovatelné testovací případy. Software je dostupný pod BSD licencí.

3.2 Scapy

Scapy je Python knihovna, která umožňuje uživatelům vytvářet a zachytávat pakety. Dokáže sestavit téměř jakýkoliv paket a poslat jej přes síťové rozhraní. Pakety je možné vytvářet intuitivně prostým skládáním protokolů za sebe, přičemž většina hlaviček je upravitelná. Tímto způsobem lze nastavit například IP adresy, nebo i příznaky v TCP paketech. Nástroj je postavený nad knihovnou libpcap [45] a nedosahuje takové výkonnosti jako Pktgen. Knihovna libpcap je totiž založena na nízkoúrovňové práci se síťovými sockety. Využívá systémových volání jádra operačního systému, což představuje značnou režii v podobě přepínání kontextu mezi samotnou aplikací a operačním systémem. Největší výhodou knihovny Scapy je zmíněné jednoduché vytváření paketů a úprava existujících hlaviček. Možné také je poměrně jednoduše definovat nové vlastní hlavičky [1]. Knihovnu Scapy lze i jednoduše integrovat do libovolného Python programu.

3.3 Cisco TRex

TRex je otevřený software pro obecné generování realistického síťového provozu. Podobně jako Pktgen běží nad frameworkem DPDK a dokáže generovat síťový provoz na vrstvách L3 až L7. Oproti generátoru Pktgen dokáže Cisco TRex generovat i stavový provoz. Požadovanou komunikaci (tzn. profil) je možno nadefinovat buď pomocí šablon, nebo ručně pomocí jazyka Python, knihovny Scapy a připraveného TRex API. TRex sice integruje knihovnu Scapy, ale používá ji pouze pro definici profilu provozu. Pro samotné generování provozu ale nepoužívá knihovnu libpcap. Implementace je založena právě na výkonnějším frameworku DPDK.

Kromě samotného generování provozu TRex poskytuje i doplňující informace, jako měření latence paketů nebo statistiky o generovaném i příchozím provozu. Cisco uvádí, že dokáže škálovat až na propustnost 200 Gb/s. TRex může fungovat v bezstavovém (stateless) nebo stavovém (stateful) režimu. Bezstavový režim zahrnuje i podporu více souběžných toků a možnost změny jakéhokoli pole paketu. Zároveň pro každý vytvořený tok poskytuje jeho statistiky, latenci a jitter. Pro stavový režim pak zahrnuje podporu emulace provozu L7 s plně funkční škálovatelnou podporou TCP/UDP [50] [28] [48]. TRex je tedy vhodný i pro implementaci legitimního klientského provozu nebo útoků, které vyžadují nejprve plnohodnotné ustavení TCP spojení mezi klientem a serverem (například Slowloris).

Bezstavový režim

Bezstavový režim disponuje následujícími vlastnostmi [51]:

- Dokáže posílat provoz rychlostí 10-22 milionů paketů za sekundu (mpps) za použití jednoho jádra. Pokud použijeme jader více, dostaneme ještě větší výkon, protože výkon škáluje s počtem jader.
- Pro každé síťové rozhraní je možné vytvořit větší množství profilů.
- Profil podporuje použití několika streamů současně. Paralelně lze aplikovat až deset tisíc toků.

- Každý stream se skládá z paketů. Pomocí knihovny Scapy může být vytvořen jakýkoliv paket (včetně neplatných). Díky komponentě Field Engine můžeme také modifikovat pole v paketu, například měnit IP adresy nebo upravovat velikost paketu. Rychlost posílání toku je možné definovat několika způsoby:
  - pakety za sekundu,
  - šířka pásma L1,
  - šířka pásma L2,
  - využití linky v procentech.

Stavový režim

TRex podporuje hned dva typy stavových režimů. Klasický Stateful mód (STF) a novější Advanced Stateful (ASTF), který implementuje plnohodnotnou podporu TCP vrstvy. ASTF režim podporuje výrazně více funkcí. Narozdíl od režimu STF je v současné době také stále aktivně vyvíjen, proto je v rámci této práce dále uvažován jen ASTF.

Pokud zvolíme stavový režim, můžeme nakonfigurovat realistickou komunikaci klienta a serveru, kdy obě entity mohou navíc běžet na odlišných zařízeních.

Stejně jako bezstavový režim i ASTF mód má množství významných vlastností. Při použití ASTF módu může být dosaženo propustnosti až 200Gb/s, přičemž dokáže udržet řádově miliony ustavených spojení. Dokáže také emulovat L7 aplikace, například HTTP komunikaci. Služby na aplikační vrstvě může ASTF simulovat i za použití TLS přes knihovnu OpenSSL [49].

Některé užitečné propagované funkce nejsou ještě dostupné, například možnost úpravy paketů v toku (tj. podpora komponenty Field Engine). ASTF v současné době také nepodporuje simulování jitteru, latence a zahazování.

3.4 Komerční generátory

Existuje řada komerčních nástrojů pro generování síťového provozu, například IXIA IxNetwork [17] nebo Sprirent TestCenter [42].

Software IxNetwork primárně řeší testování L2 a L3 vrstev a dokáže generovat provoz na rychlostech od 1Gb/s až po 800Gb/s. Podle dostupných informací by měl systém umět velmi dobře modelovat realistický provoz, který se běžně na síti vyskytuje, a přizpůsobit tomu samotné generování paketů. IxNetwork podporuje velkou škálu různých protokolů i softwarově definované sítě (SDN) [17]. Stejná společnost nabízí také odlehčenou verzi svého software jako open-source pod názvem Ixia-C [16].

Spirent TestCenter je konkurenční generátor provozu, který nabízí flexibilní a přenositelné řešení jak pro funkční tak i výkonnostní testování síťových zařízení. Dokáže testovat síťové technologie na rychlostech od 100Mb/s až do 800Gb/s, v závislosti na použitém hardware. Testy lze nastavovat a vytvářet pomocí grafického uživatelského rozhraní nebo přes REST API [48] [42].

Komerční hardwarová řešení nabízí velmi vysokou kapacitu a umožňují velmi přesné generování síťového provozu. Pro realizaci diplomové práce se tak nabízí využití i takového typu zařízení. Řešení diplomové práce probíhá ve spolupráci se sdružením CESNET, kde lze k takovému generátoru získat přístup. Celková agregovaná kapacita, která je pro generování síťového provozu v tomto případě k dispozici, je až 1,2 Tb/s. Ačkoli komerční generátory nabízí velmi vysokou kapacitu a spoustu možností co se konfigurace a přesnosti generování

provozu týče, nejsou pro naše potřeby zcela vhodnou volbou. Preferujeme totiž, aby byl implementační výstup práce co nejvíce univerzální, rozšiřitelný a uživatelsky dostupný. Komerční nástroje nabízí sice řadu technických možností, jsou ale obecně velmi drahé. Cena takového hardwarového zařízení s podporou rychlostí 100-400 Gb/s se pohybuje řádově v jednotkách až desítkách milionů korun, což není pro menší komunity uživatelů a poskytovatele síťových služeb jednoduše finančně dostupné. V tomto případě není tedy žádoucí, aby bylo vytvářené testovací prostředí přímo závislé na takovém specializovaném zařízení.

Pro implementaci praktické části práce jsme se proto rozhodli pro primární využití open-source generátoru Cisco TRex, který je vhodným kompromisem a lze jej provozovat nad komoditním a běžně dostupným hardware s výrazně nižšími náklady. Chceme tím umožnit používání nástroje i menším subjektům, které nemusí mít nutně finance na velkou investici do drahého komerčního generátoru provozu.

3.5 Generátory útoků

Kromě obecných generátorů síťového provozu je k dispozici také řada otevřených implementací některých konkrétních a známých DDoS útoků. Kromě použití generátoru Cisco TRex a implementace jednotlivých útoků ve vlastní režii se tak nabízí i možnost začlenění existujících externích nástrojů. V tomto směru se podařilo nalézt vhodné implementace útoků Slowloris a DNS Water Torture. Díky dostupnosti jejich zdrojových kódů je lze navíc relativně snadno upravit a přidat další potřebné funkcionality, které by mohly chybět v jinak dobře funkčním řešení.

Slowloris generátor

První dostupná implementace simple slowloris [61] je vytvořena v jazyce Python a je i součástí distribuce Kali Linux. Jedná se o jednoduchou implementaci Slowloris útoku. Podporuje několik nastavení, například výběr náhodného agenta při každém novém požadavku nebo nastavitelný interval mezi posílanými požadavky. Při komunikaci se serverem se v rámci hlaviček posílá i údaj o používaném prohlížeči společně s dalšími informacemi o systému. Každý prohlížeč tyto informace formátuje trochu jiným způsobem, ale vždy se vyskytují v hlavičce User-Agent. Data pro tuto hlavičku vyplňujeme při výběru náhodného agenta při útoku Slowloris.

Kromě výpisu průběhu útoku do konzole, negeneruje nástroj žádné statistiky. Také nepodporuje přepínač na nastavení času běhu. V případě přímého využití tohoto nástroje by bylo potřebné některé tyto funkcionality doimplementovat. Jelikož se jedná o poměrně jednoduchý skript v jazyce Python, případný zásah do kódu by proto neměl být složitý.

Nástroj slowhttptest [40] implementuje útok v jazyce C++. V porovnání s předchozí implementací, poskytuje slowhttptest velké množství nastavení, jako například nastavení času běhu nebo nastavení velikosti zprávy. Pro generátor slowhttptest existuje i stránka s instrukcemi k používání aplikace. Ta mimo jiné také prezentuje jak spustit útoky typu Slowloris a Slowread (druhý typ low and slow útoku, tj. R.U.D.Y.). Aplikace dokáže sbírat statistiky o průběhu útoku, které následně uloží do souboru. Současně vygeneruje jednoduchou HTML stránku, která získané statistiky zobrazí v přehledné formě. Jedná se o komplexní implementaci pomalých útoků. Zdrojové kódy jsou velmi rozsáhlé a případné úpravy programu by mohly být složitější.

Nástroj slowloris-ng [8] je opět implementován v jazyce Python. Vychází pravděpodobně z první zmiňované varianty, neboť v některých částech je kód velice podobný. V porovnání

s původní verzí však podporuje více nastavení, například posílání paketů v dávkách nebo nastavení maximální odchylky při čekání mezi odesláním paketů. Stejně jako u první představované varianty nepodporuje přepínač na nastavení času běhu. Program je tak potřebné ukončit signálem SIGINT (tj. CTRL + C).

Z pohledu funkčnosti a variací nastavení se jeví jako nejlepší druhá zmíněná implementace Slowloris generátoru, tedy slowhttptest. V tabulce 3.1 je vidět detailnější a přehledné srovnání všech prezentovaných implementací.

Rys/Nástroj	**simple slowloris**	**slowhttptest**	**slowloris-ng**
Jazyk	Python	C++	Python
Výběr agenta	Náhodný pro každé spojení	Jeden náhodně zvolený na začátku programu	Náhodný pro každé spojení
Čas běhu	Ukončení pomocí CTRL + C	Možnost nastavit čas běhu	Ukončení pomocí CTRL + C
Statistiky	Statistiky o počtu aktivních spojení v konzoli	Průběžné statistiky	Statistiky o počtu aktivních spojení v konzoli
Nastavení zdrojové IP	NE	NE	ANO
Čas mezi odeslanými pakety	Nastavení při spuštění	Nastavení při spuštění	Nastavení při spuštění, navíc možnost nastavení odchylky

Tabulka 3.1: Srovnání Slowloris implementací

DNS Water Torture generátor

Nástroj DNSWaterTorture [14] pojmenovaný přímo podle samotného útoku je implementovaný v jazyce Go. Generátor opět dovoluje několik nastavení, mezi která ovšem nepatří nastavení času běhu. Máme možnost nastavit pouze počet dotazů, které budou vygenerovány, nebo můžeme program nechat běžet nepřetržitě. Variantu, kdy bychom běh aplikace zastavili po určitém čase, by bylo možné nejspíš doimplementovat. Problém může v takovém případě představovat forma implementace, v níž je složitější se zorientovat.

Nástroj DNS Flood [20] je implementován v jazyce C a podporuje řadu nastavení, například nastavení náhodné zdrojové IP adresy. Dovoluje tak vybrat typ DNS dotazu, který se bude používat. Implementace není nikterak rozsáhlá, není proto složité se v ní poměrně rychle zorientovat a v případě potřeby provádět úpravy.

Pro DNS Water Torture útok jsme nakonec zvolili druhý zmíněný generátor. Obě varianty jsou z hlediska funkcionality a dostupných nastavení velice podobné. Druhou variantu jsme tak zvolili hlavně z toho důvodu, že je napsána v jazyce C. Lépe se v ní tak dokážeme orientovat a v případě potřeby provádět úpravy v kódu.

3.6 Shrnutí

V předchozí kapitole byly popsány nejčastější nebo pro implementaci v testovacím prostředí jinak zajímavé a vhodné vektory DDoS útoků. V této kapitole jsme se dále podrobně věnovali obecným generátorům síťového provozu a vybraným generátorům konkrétních DDoS útoků, které by bylo možné pro implementaci testovacího prostředí využít. V tabulkách níže uvádíme shrnutí a srovnání parametrů vybraných útoků, na které při implementaci testovacího prostředí cílíme. Zaměřujeme se především na skupinu záplavových a amplifikačních útoků. Jednotlivé útoky a jejich parametry porovnáváme pouze v rámci kategorií, protože například u amplifikačních útoků nás více zajímá amplifikační faktor než celkový počet spojení, který je spíše užitečný pro low and slow útoky. Všechny tabulky vychází ze zprávy společnosti Netscout [4].

V tabulce 3.2 vidíme srovnání záplavových útoků z hlediska četnosti jejich výskytu za první polovinu roku 2023. Data byla převedena z celkového počtu výskytů na průměrný denní výskyt.

Síla těchto útoků se ve většině případů uvádí v paketech nebo bitech za sekundu. Některé záplavové útoky v minulosti dokázaly vygenerovat až stovky gigabitů za sekundu [25].

Název	**Četnost [útoků/den]**	**Síla**
ACK Flood	12 961	až N krát 100 Gb/s
ICMP Flood	6 544	
SYN Flood	7 988	
RST Flood	6 787	

Tabulka 3.2: Srovnání záplavových útoků.

Tabulka 3.3 prezentuje parametry amplifikačních útoků. U těchto útoků srovnáváme následující vlastnosti:

- Počet odražečů – Parametr udávající přibližný počet dostupných nesprávně nakonfigurovaných zařízení provozujících danou službu. Taková zařízení pak mohou být zneužita pro provedení korespondujících amplifikačních útoků.
- Četnost – Počet výskytů útoku za první polovinu roku 2023. Převedena z celkové četnosti na denní výskyt.
- Amplifikační faktor – Udává amplifikační faktor pro daný typ amplifikace.

Název	**Počet odražečů**	**Četnost [útoků/den]**	**Amplifikační faktor**
DNS	2 405 334	7 753	160 : 1
NTP	7 098 600	2 813	557 : 1
Memcached	4 017	411	51 200 : 1

Tabulka 3.3: Srovnání amplifikačních útoků.

V poslední době byl také zaznamenán zvýšený nárůst použití DNS Water Torture útoku. Nejsilnější útok tohoto typu generoval až 89,4 milionů dotazů za sekundu. Přičemž průměrně bylo zaznamenáno přibližné šest set útoků tohoto typu za den.

Frekvence použití novějšího DDoS útoku Carpet Bombing byla podle zprávy společnosti Netscout [4] až tisíc za den. Pro tento typ útoku byla zaznamenána rychlost posílaného provozu až 300 Gb/s [36].

Z popisovaných obecných generátorů paketů jsme se pro výslednou implementaci nástroje pro simulování realistických DDoS útoků rozhodli použít open-source Cisco TRex generátor. Převážně z toho důvodu, že pro jeho provozování není potřebné využít specializovaného hardware, přičemž stále dokážeme generovat provoz uspokojivou rychlostí. Také nastavování specifického provozu pro generování by mělo být dostačující pro potřeby implementace testovacího prostředí. Kromě TRex generátoru jsme se rozhodli použít také již existující implementace útoků Slowloris (ve stejném nástroji bude možné realizovat i útok R.U.D.Y) a nástroj pro DNS Water Torture útok. Nástroj pro Slowloris bychom měli být schopni použít bez zásahů do zdrojového kódu. Pro použití nástroje pro generování DNS Water Torture útoku bude minimálně potřebné změnit délku generovaného prefixu. Ta v některých případech způsobí, že dojde k vytvoření neplatného paketu. O tomto problému se zmiňuje jeden z uživatelů této implementace [44].

Samotným návrhem vytvářeného testovacího prostředí se zabývá následující kapitola.

Kapitola 4

Návrh řešení

Hlavním cílem této práce je vytvořit prostředí, které dovolí testovat aplikace proti DDoS útokům. V této kapitole se budeme soustředit na zhodnocení požadavků, které jsou od aplikace očekávány a které zohledníme při návrhu a implementaci.

Nejprve definujeme některé možné případy užití aplikace pro simulování DDoS útoků. Z této analýzy nám následně vyplynou požadavky a vlastnosti pro vytvářenou aplikaci. Funkcionality, které by aplikace měla podporovat, budou v této části představeny společně s návrhem jejich řešení. Následně budou probrány jednotlivé kategorie DDoS útoků a způsob jejich začlenění do vytvářeného nástroje.

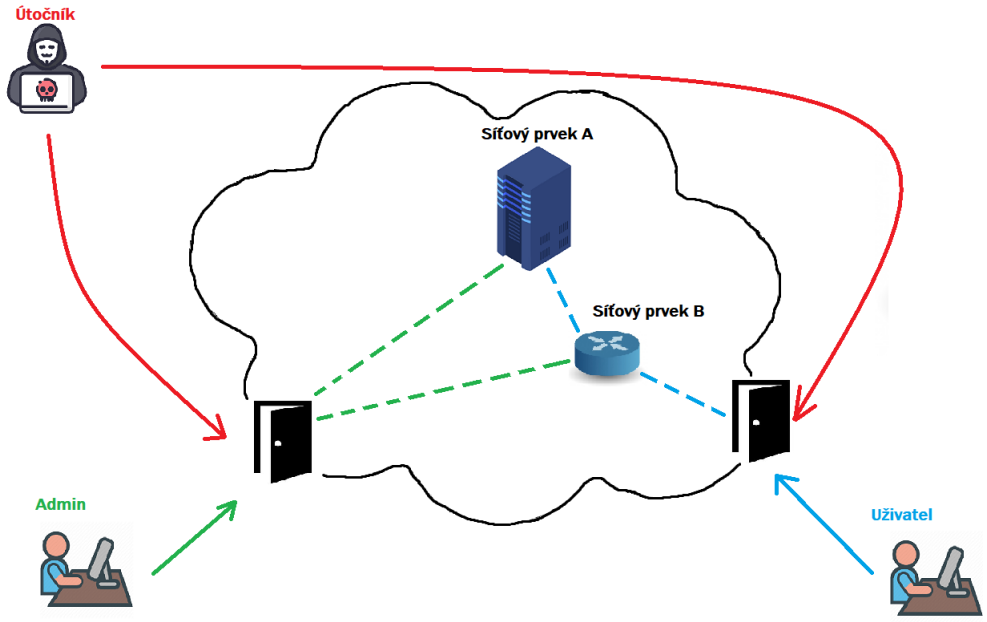
4.1 Případy užití

Před samotným definováním přesných požadavků, bude představeno několik případů užití aplikace. Případy užití nám pomohou lépe pochopit potřeby uživatele aplikace pro simulování DoS útoků.

Případ užití A

Chceme zaútočit na službu A. Pro útok si zvolíme například útok typu SYN Flood. Dále si uvědomíme, že jakmile administrátoři zaznamenají, že probíhá útok, pokusí se sjednat nápravu. Ke službě se pravděpodobně budou připojovat pomocí VPN/SSH. Spustíme tedy ještě druhý útok, který bude zaměřený na SSH, aby ztížil přístup administrátorům.

Na obrázku 4.1 je zobrazen scénář takového útoku. Máme nějakou infrastrukturu, kde hlavní úlohou je poskytnout službu (služba je na obrázku označena jako síťový prvek A). Klient se k této službě připojuje přes přístupový bod, případně přes další prvky v síti. Ty jsou pro běžného uživatele často skryté. Důležité pro něj je, že se dostane na službu. V celé infrastruktuře může běžet více služeb. V infrastruktuře se také vyskytují interní zařízení, ke kterým běžní uživatelé internetu nemají přístup, jako například firewall nebo DDoS ochrana (na obrázku jsou reprezentovány síťovým prvkem B). Na tyto prvky se administrátoři připojují přes SSH, tedy přes jiný přístupový bod (gateway). Pokud útočník zaútočí na oba tyto přístupové body, znemožní uživateli přístup na službu. Současně znemožní i přístup do systému administrátorům, kteří se snaží připojit, aby mohli zastavit útok.



Obrázek 4.1: První případ užití nástroje.

Případ užití B

Chceme otestovat chování zařízení pro DDoS ochranu. Předpokládejme, že útočník při útoku použije stejnou skupinu zdrojových IP adres (prefix) jako legitimní klient. Chceme sledovat, jak se v této situaci DDoS ochrana zachová. Konkrétně zda ochrana zakáže celý prefix a zamezí tak přístup i legitimnímu klientovi, nebo jestli se podaří odstínit pouze útočníka a klientský provoz zachovat.

Obrázek 4.2 znázorňuje chování ve dvou různých situacích. V horní části můžeme vidět jak se klient i útočník snaží komunikovat se serverem, ale DDoS ochrana zachytí útok a zakáže celý prefix. V druhé, spodní části, dochází k situaci, kdy opět klient i útočník komunikují ze stejného prefixu, DDoS ochrana útok opět zachytí, ale tentokrát zablokuje přístup pouze útočníkovi.

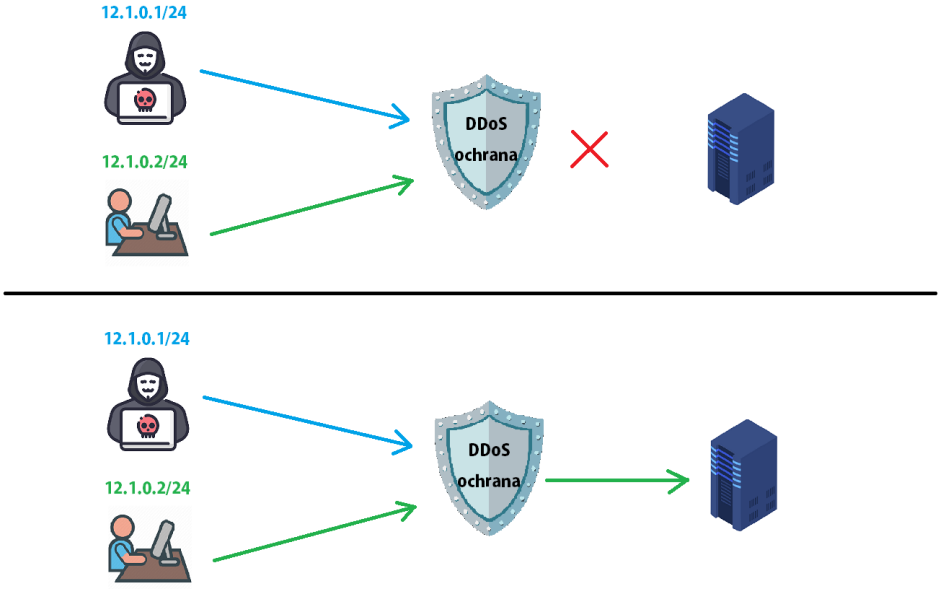
Případ užití C

Spravujeme síť a chtěli bychom ji chránit před DDoS útoky pomocí software pro DDoS ochranu nebo jen za použití pravidel na firewallu. Vytvořili jsme nějaké postupy pro mitigaci DDoS útoku a chtěli bychom ověřit, které fungují nejlépe pro konkrétní útok. Případně zda postupy vůbec fungují a jestli není potřebné doplnit další pravidla.

Chceme tak vygenerovat co největší počet různých používaných útoků s různými parametry, abychom dokázali ověřit fungování našich konfigurací a připravenost ochrany pro infrastrukturu.

Případ užití D

Mějme reálnou veřejně dostupnou službu provozovanou na internetu, například nějaký webový server. Tuto službu chceme otestovat, ale nemáme nebo nechceme použít specifické izolované prostředí. Musíme tedy simulovaný útok provést přes internet (samozřejmě



Obrázek 4.2: Druhý případ užití nástroje.

s vědomím poskytovatelů, všech správců sítě po cestě, atd.). Na zařízení v laboratorním prostředí, odkud je simulace spouštěna, může být také k dispozici více síťových rozhraní, kdy výstupy všech nemusí směřovat do internetu. Chceme tak mít možnost zvolit, kterým síťovým rozhraním budou pakety generovány.

Případ užití E

Síťová infrastruktura a její obrana jsou stále vyvíjeny a zdokonalovány. Vznikají nové pokročilejší protokoly, které by měly být bezpečnější a obecně ve všech ohledech lepší. S tím, jak se zlepšuje obrana proti DoS, se ale vyvíjí i používané DoS útoky. Proto je pro vytvářený testovací nástroj nutná jednoduchá rozšiřitelnost, aby nástroj dokázal simulovat nejpoužívanější útoky a sloužil ke svému účelu, tj. testování systémů proti realistickým DoS útokům. Chceme tak mít možnost rozšiřovat již implementované typy útoků, například přidávat nové typy amplifikací nebo záplav. Je také vhodné, aby bylo možné jednoduše doplnit zcela nový typ útoku.

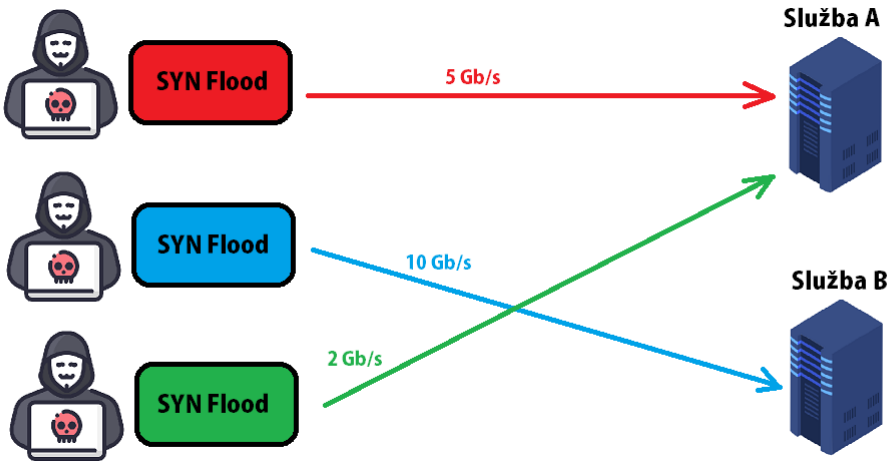
4.2 Požadavky na řešení

Z předešlých případů užití nám vyplývá několik požadavků. Systém na simulaci DDoS útoků musí podporovat určité funkcionality a mít určité vlastnosti. Některé lze odvodit z představených případů užití. V následujících odstavcích se pokusíme shrnout všechny funkcionality a vlastnosti, které by měl výsledný software na provádění DDoS útoků zahrnovat.

Současné generování více různých útoků

Samozřejmým požadavkem je generování útoku na vybraný cíl. Kromě toho chceme být také schopni simulovat multi-vector útok. To znamená, že nástroj by měl poskytovat rozhraní pro souběžné spuštění více útoků. Měli bychom být také schopni spustit více instancí stejného

útoku, ale s různými parametry (v případě potřeby i stejnými, pokud to z pohledu užití bude smysluplné). Konkrétně chceme mít možnost, například měnit cíl útoku. Na obrázku 4.3 můžeme vidět příklad takové situace. V jednom běhu testu můžeme chtít vytvořit tři útoky, kde zvolíme, že všechny útoky budou typu SYN Flood. Na obrázku jsou tyto tři různé útoky odlišeny barvami. Červený útočí rychlostí 5 Gb/s na službu A. Na stejnou službu útočí i zelený SYN Flood, který je pomalejší s rychlostí 2 Gb/s. Na službu B pak útočí poslední, modrá záplava rychlostí 10 Gb/s.



Obrázek 4.3: Příklad současného generování více různých útoků.

Vytvoření scénáře útoku

Když vyžadujeme možnost vytvářet více útoků současně, bylo by také přínosné dokázat generovat útoky v návaznosti na sobě. Chceme mít tedy možnost nastavit útoky tak, aby po ukončení jednoho útoku byl spuštěn hned další. Případně můžeme mezi jednotlivými útoky určitou dobu čekat. Společně s předchozím požadavkem pak dokážeme vytvořit rozsáhlejší a komplexnější scénáře útoku.

Parametry útoku

Každý útok má své konkrétní parametry. Jedním z důležitých a prvních, který je potřeba zvolit, je typ. Podle typu se pak odvíjí jaká další nastavení bude útok vyžadovat, protože například pomalé a záplavové útoky se v některých parametrech mohou odlišovat.

Důležitým parametrem je i síla útoku. Ta může být definována různým způsobem. U záplavových útoků ji můžeme definovat:

- počtem odeslaných paketů za sekundu (pps),
- počtem odeslaných bitů za sekundu (bps),
- nebo i procentuálním vytížením linky, pomocí které útok probíhá

Tyto zmíněné metriky lze v podstatě používat u každého typu útoku. U některých se nám hodí jiný způsob zadávání. Pro amplifikační útoky můžeme chtít volit amplifikační faktor

a počet simulovaných odražečů. U pomalých útoků pak můžeme sledovat počet současně vytvořených spojení.

Aby bylo možné efektivně testovat různé služby, je potřebné mít možnost nastavit cílovou adresu útoku. Jako adresu lze použít přímo doménové jméno (pokud ho služba má), například www.vutbr.cz. Každé službě ovšem nemusí být nějaké doménové jméno přiřazeno. V takovém případě by byla jako adresa cíle použita IP adresa. Současně s IP adresou bychom mohli chtít měnit i cílový port, protože specifické služby komunikují pouze na určitém portu.

Kromě cílové adresy můžeme také vyžadovat možnost měnit zdrojovou adresu. S tou některé útoky nepracují přímo, například pro SYN Flood je zdrojová adresa nepodstatná. Není pouze vhodné, aby byla adresa stejná jako pravá adresa útočníka. Došlo by tak k ustavení spojení, což je v případě útoku nežádoucí. Změnou zdrojové adresy ovšem můžeme simulovat falšování adresy útočníkem (spoofing). Spoofing lze využít i při simulaci SYN Flood útoků.

Parametrem souvisejícím se zdrojovou adresou je použití botů. V teoretické části práce bylo popsáno, že DDoS útoky v dnešní době hojně využívají právě sítě botů (tzv. botnet). Bylo by tak vhodné, aby navrhované prostředí dokázalo modelovat použití botnetů, tedy abychom dokázali nastavit počet IP adres, ze kterých budou pakety generovány, respektive jaké budou paketům vytvářeným v testovacím prostředí přiřazovány zdrojové adresy. Můžeme tak například nejprve zvolit podsíť, ze které mají být zdrojové adresy vybírány, a následně podle počtu botů z této podsítě náhodně vybrat příslušný počet IP adres. Ty pak budou použity jako zdrojové adresy útoku.

Dalším důležitým parametrem je doba trvání útoku. Chceme umět ovládat jak dlouho budou aktivní jednotlivé instance útoků. Společně s parametrem, který určuje zpoždění útoku, tak dokážeme modelovat vytvoření určitého scénáře útoku.

U pomalých útoků chceme granularitu čekání ještě zvýšit. A to konkrétně tak, že nastavíme zpoždění mezi jednotlivými odeslanými pakety. Tím pak dokážeme útok optimalizovat a i s odesláním malého množství paketů způsobit výpadek služby. Bylo by také vhodné, aby byl tento interval proměnlivý, například modelován jako maximální odchylka. Toto by pak mohlo ztížit detekci útoku a kvalitněji tak otestovat zabezpečení služby.

Zapojení do infrastruktury

Kromě testování odolnosti služby na lokální síti chceme mít možnost testovat službu i kdekoliv na internetu, samozřejmě s povolením od jejich správců. K tomuto účelu nám slouží především cílová IP adresa a porty, které byly diskutovány v dřívějších odstavcích. Ve většině případů máme přichystané specifické testovací prostředí a infrastrukturu s tím spojenou, to znamená, že máme nakoupené, připojené a správně nakonfigurované síťové karty, které slouží k různým účelům. Infrastruktura tak nemusí nutně sloužit jen k testování DDoS útoků. Při simulacích útoků chceme mít možnost určit, kterým síťovým rozhraním útočné pakety odejdou. Můžeme chtít tyto pakety kontrolovaně odeslat do internetu nebo naopak je poslat pouze na vlastní zařízení, aby nezpůsobovaly výpadky v reálných sítích. Proto bychom chtěli, aby námi vyvíjené prostředí dokázalo nastavit, kam má být který tok paketů odeslán, neboli kterou odchozí/výchozí bránu má použít. Při použití TRex generátoru takového chování docílíme nastavením MAC adresy zařízení, na které budeme přesměrovávat generovaný provoz. Pro ostatní aplikace, které provoz rovnou generují přes výchozí bránu nastavenou v operačním systému, je možné vytvořit virtualizované prostření na úrovni operačního systému, například pomocí namespace [15].

Simulace serveru

Prostředí by mělo být schopno simulovat komunikaci ze serveru, tj. chtěli bychom mít možnost vytvořit instanci zdroje paketů, který se bude chovat jako server, služba, atd., na kterou je možno zaútočit. Server by měl také být schopen správně odpovídat legitimním požadavkům. Funkcionalita simulovaného serveru je užitečná i v případě, že chceme testovat nějaké zařízení nebo techniku zajišťující DDoS ochranu. V případě, že chceme testovat právě tyto prostředky pro mitigaci útoků, můžeme vytvořit simulovaný server, který se chová jako klasický.

Každé DDoS útoky cílí na nějakou specifickou vlastnost a předpokládají jiné chování serveru. To znamená, že i simulovaný server by mělo být možné spustit ve více režimech a případně i s různými parametry. Například pokud používáme útok typu SYN Flood, předpokládáme, že bude server správně odpovídat na klientovy požadavky o navázání TCP spojení. Naopak pokud použijeme ICMP Flood, předpokládáme, že server bude odpovídat na tyto dotazy adekvátním způsobem. Dále v případě pomalých útoků chceme, aby server nejprve spojení navázal a následně probíhala klasická HTTP komunikace.

Simulace legitimního provozu

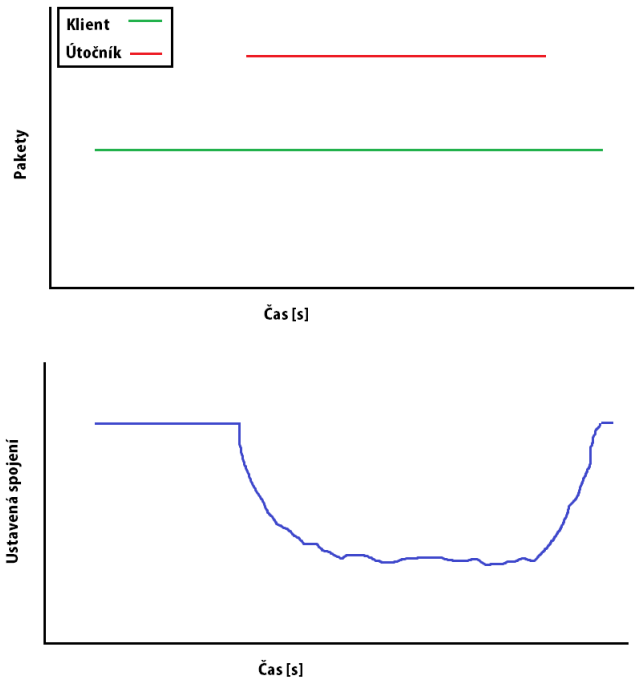
Kromě různých variant útoků by v prostředí mělo být možno vytvořit instanci legitimního klienta. Vytvoření provozu z legitimního klienta přináší užitek při sledování dostupnosti oběti útoku. Můžeme tak například sledovat průběh chování serveru, před, během a po útoku. Na obrázku 4.4 v horní části vidíme jak v čase klient posílá konstantní množství paketů (klient je znázorněn zelenou barvou). Útočník, červený průběh, začne po určité době útočit. Na spodní části obrázku pak vidíme schopnost serveru odpovídat na dotazy. Ta je v první fázi, kdy komunikuje pouze klient, stabilní a server zvládá odpovídat na všechny požadavky. Jakmile začne útok, vidíme, že schopnost serveru navázat všechna spojení značně klesá. Schopnost serveru ustavovat legitimní spojení je obnovena až po uplynutí určité doby od ukončení útoku.

Jak již bylo diskutováno, každý útok je jiný a je tedy pokaždé potřeba ověřovat funkčnost napadené služby jiným způsobem. Bylo by tedy vhodné, aby vytvářené prostředí dovolovalo simulovat klienta, a zároveň aby bylo možné klienta spustit v různých verzích, čili s podporou různých funkcí:

- TCP klient – Chtěli bychom mít možnost ověřovat dostupnosti nejen pro samotné ustavení spojení (kontrola pro SYN Flood), ale i dostupnost pro služby postavené nad tímto protokolem jako HTTP, HTTPS, OpenVPN, atd.

- UDP klient – Některé útoky používají UDP pakety. Služby, na které jde útok právě pomocí tohoto protokolu, je potřeba skenovat také tímto stejným protokolem, abychom dosáhli správných výsledků o dostupnosti. Jedná se například o DNS nebo NTP dotazy. Útoky nejsou ve většině případů vedeny přímo proti těmto službám. Pokud je služba použita pouze jako odražeč, nesnaží se útok zahltit přímo ji. UDP klienta můžeme použít, abychom ověřili dostupnost služeb, které se nacházejí v napadené síti. Pod útokem tedy není prvek (DNS, NTP server) samotný, ale dochází například k volumetrickému útoku na celou síť.

- ICMP klient – Jednoduchá instance klienta, která dokáže generovat ICMP dotaz na službu. Tento typ klienta se hodí pro ověřování dostupnosti služby při použití útoku typu ICMP Flood.



Obrázek 4.4: Demonstrace použití klienta pro sledování chování serveru při útoku.

Rozšiřitelnost

Kromě obranných mechanismů se vyvíjí i útoky. Testovací prostředí by tak mělo umožňovat snadné přidávání nových typů útoků. U některých variant útoků může docházet k různým modifikacím, a proto chceme být schopni jednoduše upravovat i již existující útoky, tj. přidávat parametry, rozšiřovat o nové typy amplifikací či záplav, atd.

Kromě samotných útoků chceme mít možnost doplňovat další funkcionality i pomocným službám, jakými jsou výše popsaný legitimní klient a server. Chceme tak mít možnost jednoduše doplnit další funkcionality a implementace nových aplikačních protokolů i pro legitimního klienta a server, aby dokázali správně simulovat nově vznikající reálné služby.

Vyhodnocení testu

Nedílnou součástí jakéhokoliv testování je sběr statistik a vyhodnocení výsledků. Proto je i v našem prostředí potřebné sbírat statistiky o průběhu jednotlivých simulacích útoku. Z těchto statistik pak budeme chtít vyvozovat určité závěry, například jak se útok daří, jak dobře je síť/služba chráněná, zda přidat další obranné mechanismy, atd.

Jednou ze základních statistik, kterou lze sledovat u každého útoku, je množství vygenerovaného provozu. Statistiky chceme typicky měřit s určitou frekvencí, díky čemuž získáme data průběžně. Z těchto dat je následně možné vytvořit graf průběhu útoku v čase. Grafy pak můžeme mezi sebou porovnávat a sledovat, co se ve kterých časových úsecích událo.

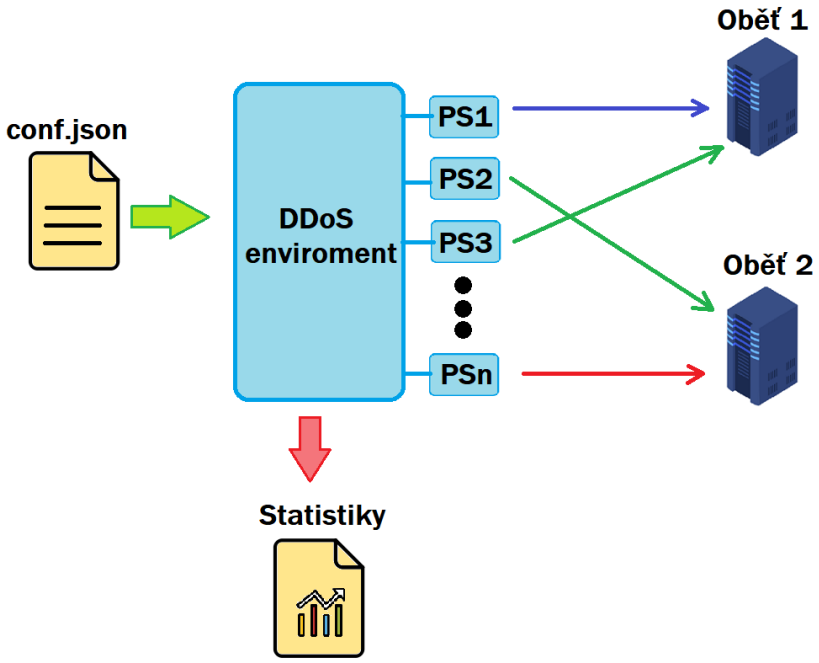
Pro některé specifické útoky, například low and slow útoky, může být zajímavou statistikou, kromě obyčejné informace o počtu vygenerovaných paketů, počet vytvořených spojení.

Počet spojení můžeme sledovat i u SYN Flood útoku. Kdy při použití vlastního testovacího klienta, diskutovaného dříve, můžeme sledovat kolik spojení se daří ustavit během útoku a kolik při klidovém stavu.

Detailnější statistiky budou potřebné pro útok typu Pulse Wave, kdy je vhodné sledovat jak jednotlivé útoky probíhají v čase. Chtěli bychom tedy jako výsledek získat souvislý graf s vygenerovaným provozem za celou dobu útoku. V grafu by též bylo užitečné označit, který útok probíhal v kterou dobu.

4.3 Návrh fungování aplikace

Konfigurační soubor specifikuje, které útoky, zdroje paketů (packet sources) budeme používat. Jak je zobrazeno na obrázku 4.5, konfigurační soubor vstupuje do programu, který spouští jednotlivé zdroje paketů. Podle definovaného konfiguračního souboru, jsou spuštěny příslušné zdroje paketů se specifikovanými nastaveními. V příkladu jsou jednotlivé zdroje paketů označeny jako PS1 až PSn, kde číslovka značí pořadové číslo instance. Tyto zdroje paketů pak pracují v závislosti na implementaci a podle nastavení, které bylo dodáno v konfiguračním souboru. Dále je naznačeno jak generátory posílají provoz na dvě různé oběti. V reálném scénáři může být cílů více, nebo i méně. Modrou šipkou je pak ukázán útok typu SYN Flood o síle 10 Gbps. Zelenou klientský (legitimní) provoz rychlostí 1 Gbps. Oba zdroje komunikují s obětí jedna. Poslední (červený) provoz značí útok pomocí DNS amplifikace o síle 5 Gbps na oběť dva. Na konci běhu programu by měly být poskytnuty statistiky o průběhu jednotlivých útoků, například ve formě grafů. Na obrázku 4.5 je toto chování naznačeno jako výstup programu. Specifika jednotlivých částí budou dále diskutována v rámci kapitoly.



Obrázek 4.5: Příklad spuštění aplikace.

4.4 Podporované útoky

Z předchozích kapitol jsme získali přehled o používaných útocích, které budeme v tomto nástroji implementovat. V této sekci budou probrány detailnější specifikace pro jednotlivé kategorie útoků.

Záplavové útoky

Pro tento typ útoku se jeví jako vhodné implementovat celkem čtyři útoky. SYN, ACK, RST a ICMP Flood. První tři zmíněné mají podobnou funkcionalitu. Všechny využívají TCP protokol. Jedinou změnou je příznak v TCP hlavičce. ICMP Flood funguje taktéž na stejném principu, jen nepoužívá TCP protokol. U všech Flood útoků se jeví jako vhodné použít TRex generátor, konkrétně v bezstavovém režimu. Chceme totiž generovat velké množství paketů, k čemuž se výkonný TRex hodí. Bezstavový režim použijeme, protože nechceme, aby reálně docházelo k ustavení spojení se serverem, na který útočíme. Bylo by to proti podstatě útoku. Při modelování provozu, který má být poslán na oběť, nám poslouží TRex API implementované knihovnou testsuite [34]. To nám umožní poměrně jednoduše vyrobit potřebné pakety podle zadaných specifikací.

Díky TRex API nebude složité získávat statistiky o průběhu útoku. Dokážeme sbírat kumulativní data o poslaných paketech a bitech. Ta pak bude potřebné upravit, aby představovala data za sekundu.

Pomalé útoky

Pro tento typ útoku se podařilo nalézt několik nástrojů [61] [40] [8]. Jako nejnadějnější se jeví slowhttptest [40], protože podporuje nejvíce z požadovaných parametrů.

U těchto typů útoků bychom chtěli sledovat počet ustavených spojení, abychom mohli odhadnout zda v příštím útoku počet spojení zvýšit nebo stačí menší počet. Současně můžeme i v závislosti na předchozím sledovat, jaká je dostupnost pro legitimní uživatele. Chceme tedy mít přehled jak počet útočných spojení ovlivňuje dostupnost služby.

Amplifikace

Naším cílem je používat amplifikaci bez odražečů, tedy posílat útoky přímo. Budeme tak simulovat již odražený provoz bez reálných paketů od útočníka. Pakety vytvořené útočníkem, například speciálně vytvořený dotaz na DNS, by se k oběti stejně nedostaly. Proto budeme v rámci prostředí generovat pouze provoz, který by byl generován službou po zpracování požadavku od útočníka, tj. již amplifikovaný provoz. Tento postup volíme i proto, abychom zbytečně nezatěžovali jiné služby. Navíc budeme mít útok více pod kontrolou a nebudeme závislí na hledání vhodných odražečů.

Simulaci odražečů provedeme pomocí paketů s odpovědmi od reálných služeb [10]. Pokud tedy chceme modelovat útok pomocí DNS amplifikace, použijeme paket, který obsahuje právě odpověď na DNS dotaz.

K provedení útoku využijme TRex, který použije pakety jako šablonu, pomocí které vytvoříme nový provoz. Jakmile bude vyřešeno vytváření paketů, je konstrukce útoku stejná jako pro Flood útoky.

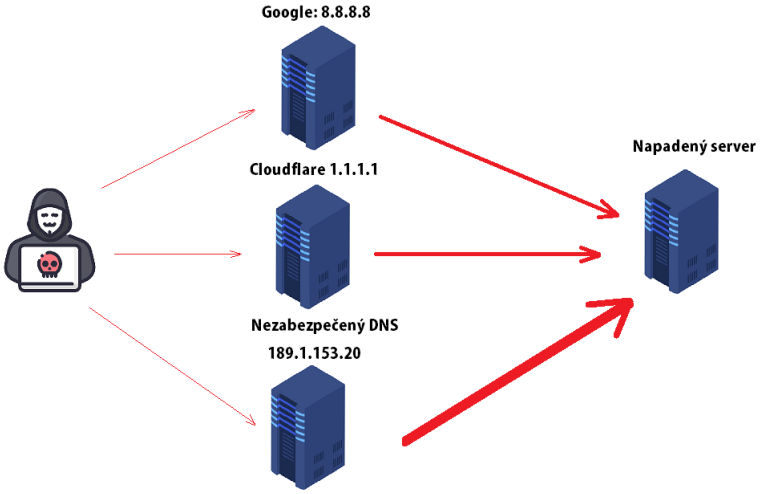
Nástroj by měl při použití amplifikace uživateli umožnit vybrat typ paketů, kterými se bude odpovídat. Výběr paketu simuluje výběr typu amplifikačního útoku tj. zneužitého

odražeče. Pokud tedy chceme modelovat útok DNS amplifikací, vybereme paket obsahující odpověď DNS serveru.

Servery se službou (DNS, NTP, ...) mohou na útočníkovy dotazy odpovídat různě. Pokud je server správně nakonfigurován, nepošle znásobený provoz, ale pouze jednoduchou odpověď. I tyto situace by ve vytvářeném nástroji měly být zohledněny. Bude tak do jisté míry simulován i výběr odražeče pro útok. V reálném provozu generovaném útočníkem, se mohou vyskytovat i odpovědi správně nakonfigurovaných serverů. Proto je vhodné mít možnost modelovat i ty. Na obrázku 4.6 vidíme situaci kdy útočník používá tři různé odražeče:

- 1.1.1.1 – Zabezpečený DNS společnosti Cloudflare odpovídá pakety typu Standard query response not implemented ANY. Pakety mají velikost přibližně 75 bytů.
- 8.8.8.8 – Zabezpečený DNS společnosti Google odpovídá pakety typu Standard query response ANY. Pakety mají velikost přibližně 62 bytů.
- 189.1.153.20 – Nezabezpečený DNS server odpovídá na dotaz, čímž vytvoří amplifikaci. Pakety mají velikost přibližně 1 500 bytů, což je přibližně 24krát větší objem dat než u předchozích paketů.

V příkladu na obrázku je také různou šířkou čar znázorněna velikost odpovědi, kdy nejsilnější čára značí největší odpověď.



Obrázek 4.6: Vliv použití různých odražečů na vliv velikosti odpovědi.

Nejvíce používané amplifikace (DNS, NTP, Memcached, ...) mohou být v programu podporovány ve formě předpřipravených paketů s příslušnými odpověďmi. Pokud by potřeboval uživatel vytvořit jiné typy amplifikací, může si pakety, které by chtěl použít, připravit sám a přidat je k již dostupným. Mělo by tak být možné spustit jakoukoliv amplifikaci.

Uživatel také může zvolit více různých zdrojů odrazu, tj. vytvořit podobnou situaci jaká je demonstrována na obrázku 4.6. V případě, že chceme použít více zdrojů odrazu, jeví se jako vhodné také určit, v jakém poměru budou jednotlivé zdroje využity. Jde tedy o objem paketů, který jednotlivé odražeče vygenerují. Chceme být například schopni zadat, že odražeč, který generuje nezesílenou odpověď, generuje méně provozu než odražeč, u kterého k amplifikaci dojde. Pokud tedy zvolíme například poměr 2:1 s odražeči 189.1.153.20 a 8.8.8.8, můžeme očekávat provoz 10 Gb/s z 189.1.153.20 a 5 Gb/s z 8.8.8.8.

DNS Water Torture

Pro DNS Water Torture se také podařilo nalézt existující nástroje [14] [20].

K vybrané implementaci bude potřebné doplnit sběr statistik. Pro tento typ útoku můžeme sledovat jednak objem odeslaných dat (pps, bps), ale také počet odeslaných požadavků. Chceme také sledovat zda se útok daří. K tomuto účelu můžeme implementovat klienta, který se bude doptávat na určený DNS záznam (ten na který útočíme). Buď může žádat přímo o záznam, kdy se ptáme přímo DNS, nebo se pokusí dostat na stránku, což znamená, že bude prováděna simulace běžného uživatele internetu, který například na stránce hledá informace. V obou případech je potřebné dát pozor, aby si klient neuchovával odpověď DNS serveru a nenačítal tak data z mezipaměti.

Carpet Bombing

Útok technikou Carpet Bombing má dvě hlavní specifika, zjišťování limitů DDoS ochrany a útočení na celou zvolenou podsíť.

Carpet Bombing je tak spíše technika než jiný druh útoku. Techniku je možné využít ve spojení s jinými útoky. Můžeme například zvolit, že budeme útočit pomocí SYN Flood, zjistíme si konkrétní limity DDoS ochrany chránící cílovou síť a následně na celou síť zaútočíme.

Při aplikaci této metodiky do praxe je tak spíše nutné přidat potřebné funkcionality ke konkrétním útokům. Útoky, které budou podporovat tuto techniku, tak musí implementovat:

1. Zjišťování limitů DDoS ochrany – Pro každý typ útoků nás mohou zajímat jiné parametry DDoS ochrany. Pro záplavové útoky to může být maximální objem odeslaných dat. Pro pomalé útoky, které naopak negenerují takový objem dat, se tento aspekt při mitigaci nepoužije. V případě pomalých útoků nás tak budou zajímat jiné informace sloužící k mitigaci právě tohoto konkrétního útoku.

2. Útočení na celou zvolenou podsíť – Každý útok, u kterého chceme podporovat metodiku Carpet Bombing, musí dokázat zaútočit na více cílů současně. Nejlépe na celou zvolenou podsíť.

Pulse Wave

Opět se jedná spíše o metodiku útoku než využití nové konkrétní zranitelnosti. Je tak potřebné implementovat mechanismus, který dovolí spouštět již implementované útoky v pulzech. Musíme tak mít možnost určit, které typy útoků budou spouštěny v konkrétní instanci Pulse Wave útoku. Dále potřebujeme zadávat jiné specifikace útoku, například jaké budou intervaly mezi útoky, jak dlouho mohou jednotlivé útoky trvat, atd.

Důležité je také sbírat statistiky. Kromě celkového vygenerovaného provozu potřebujeme znát výsledky konkrétních dílčích útoků. Je také užitečné sledovat, kolik kterých útoků bylo použito.

Chceme také znát posloupnost jednotlivých útoků a jejich konkrétní parametry, tedy jak jednotlivé útoky probíhaly za sebou a jaké měly nastavení. Užitečné by mohlo být vidět také graf, který ukazuje právě i charakteristické pulzy. Ten by mohl být sestaven ze sledování bitů/paketů za sekundu v čase přes celý běh Pulse Wave útoku. Bylo by tedy vhodné při běhu Pulse Wave útoku sbírat průběžně statistiky o bitech/paketech generovaných útokem za sekundu.

4.5 Konfigurační soubor

V předchozích sekcích bylo popsáno konfigurační nastavení, které by bylo vhodné pro konkrétní útoky. Zde bude vysvětleno jak lze tyto abstraktní požadavky mapovat na konkrétní nastavení v konfiguračním souboru.

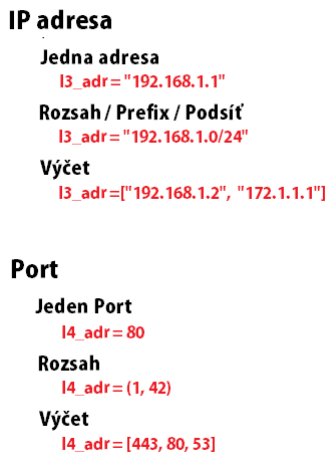
Pro zadávání konfigurace jsme zvolili formát JavaScript Object Notation (JSON). Abychom mohli spustit simulaci, potřebujeme mít možnost nastavit, které zdroje paketů chceme využít. Do konfiguračního souboru můžeme definovat tolik zdrojů paketů, kolik jich chceme v rámci prostředí spustit. Každý zdroj paketů má vlastní nastavení, která se specifikují přímo při definici konkrétního zdroje paketů. V příkladu 4.1 vidíme vytvoření dvou zdrojů paketů s jednoduchými nastaveními. Prvním zdrojem paketů je generátor útoku SYN Flood se základním nastavením. Druhým zdrojem je pak simulovaný legitimní klient. Všechny specifikovatelné vlastnosti budou detailněji popsány dále v této kapitole.

```
"packet_source0": {
    "type": "syn_flood",
    "settings": {
        "src": "152.2.1.0/24",
        "dst": "42.5.2.0/26",
        "pkt_len": 256,
        "rate": "15Gbps",
        "duration": 3600,
        "delay": 5
    }
},
"packet_source1": {
    "type": "client",
    "settings": {
        "dst": "42.5.2.0/26",
        "conn_rate": 1000,
        "duration": 3600
    }
}
```

Výpis 4.1: Zjednodušený příklad definice dvou zdrojů paketů.

V konfiguračním souboru by měla být jednotlivá nastavení mapována následujícím způsobem (každé nastavení se pak v reálném konfiguračním souboru specifikuje pro každý zdroj paketů zvlášť):

- Typ – Je prvním zadávaným parametrem. Podle něj se následně vytvoří příslušný zdroj paketů s definovaným nastavením. Samotný parametr určující typ se do konfiguračního souboru zadává pod klíčovým slovem `type`. Další nastavení jsou v konfiguračním souboru specifikována ve slovníku s názvem `settings`.
- Nastavení cíle útoku – Nastavujeme pomocí specifikace IP adresy a portu. IP adresu lze nahradit konkrétním doménovým jménem (např. vutbr.cz). Chceme mít možnost zadat i více adres. V případě IP adresy můžeme určit výčtem nebo zvolením prefixu, tj. podsítě. Pro port opět můžeme zvolit pouze výčet nebo i rozsah portů. Na obrázku 4.7 můžeme vidět příklady definice pro jednotlivé zmíněné způsoby zadávání. Pro nastavování cílové IP adresy se používá parametr `l3_dst` pro port je to pak `l4_dst`.
- Nastavení zdroje útoku – Je stejné jako u předchozího. Pro zadávání parametru se používají podobná klíčová slova (pouze změna na klíčové slovo `src`), tj. `l3_src` a `l4_src`.



Obrázek 4.7: Způsoby zadávání IP adres a portů.

- Botnet – Specifikuje počet použitých botů. Jejich IP adresy se náhodně zvolí z podsítě. Ta je určena parametrem nastavujícím zdrojovou IP, tj. parametr výše. V konfiguračním souboru je toto nastavení definováno jako `bot_cnt`. Aby bylo možné tento parametr využít, je nutné zadat zdrojovou IP adresu určením prefixu.
- Kontrola odchozího rozhraní – Pro TRex útoky stačí specifikovat MAC adresu výchozí brány. Pro útoky, které jsou realizovány za pomocí socketů, použijeme namespace, který je potřebné před spuštěním aplikace nakonfigurovat. Následně se jeho název předá jako parametr do konfiguračního souboru. Dokážeme tak ovládat, kterým rozhraním pakety odejdou. Pro některé útoky tak nastavujeme cílovou MAC adresu (`dst_mac`) a pro některé název namespace (`namespace`), který chceme použít.
- Čas běhu, čekání, atd. – Všechny časy (čas běhu, čas čekání před spuštěním útoku, čas čekání mezi odeslanými pakety, ...) jsou zadávány v sekundách. Čas běhu (`duration`) a čekání před spuštěním dalšího útoku (`delay`) jsou dostupné pro všechny útoky. Čas čekání mezi odeslanými pakety (`interval_between_data`) je relevantní pouze pro pomalé útoky.
- Síla útoku – Sílu útoku lze pro útoky prováděné pomocí TRex generátoru specifikovat pomocí bitů nebo paketů za sekundu. Můžeme ji také zadat procentuálním využitím používané linky. Pro pomalé útoky se síla definuje počtem souběžných spojení a počtem spojení, která jsou vytvořena za sekundu. U TRex útoků se pro rychlost útoku používá nastavení `rate` společně s parametrem určujícím velikost paketu (`pkt_len`). Pro pomalé útoky je pak použit parametr určující, kolik spojení má být celkově udržováno (`connection_cnt`) a kolik spojení má být vytvořeno za sekundu (`connection_per_second`).
- Klient a server – Pokud uživatel vyžaduje vytvoření simulace serveru a legitimního klienta, může vytvořit zdroj paketů, který představuje právě tyto dvě entity. Klient a server tak nejsou přímo parametry některého útoku nebo samotného prostředí, ale jsou specifikovány jako speciální typ zdroje paketů. Samotná definice typu klienta (TCP, HTTP, ...), respektive serveru, je definována pomocí nastavení `client_type` a `server_type`.

- Současné generování více útoků – Současného generování útoků je možné docílit specifikováním požadovaných zdrojů paketů v konfiguračním souboru. V konfiguračním souboru je možné specifikovat jakýkoliv počet zdrojů paketů. Záleží pouze na volných zdrojích zařízení, na kterém jsou generátory provozovány.

- Vytvoření scénáře útoku – Pokud bychom chtěli vytvořit konkrétní scénář útoku, můžeme využít toho, že každému zdroji paketů můžeme nastavit délku trvání a zpoždění před spuštěním. Když tedy potřebujeme po doběhnutí útoku A trvajícího 10 vteřin, spustit útok B, můžeme pro útok B nastavit zpoždění spuštění na 10 sekund, čímž dojde ke spuštění útoku B až po ukončení útoku A.

- Volba odražeče – Pro amplifikační útoky je možné použít stejná nastavení jako pro ostatní útoky realizované pomocí TRex generátoru. Pro amplifikační útoky je ovšem důležitý typ odrazu. Bylo tak doplněno nastavení, které umožňuje zvolit, který typ odražeče bude použit. Konkrétně se jedná o nastavení `modes`, které má význam pouze pro amplifikační útoky a je zadáváno jako výčet všech druhů paketů, které chceme použít.

4.6 Statistiky

Statistiky, které budou zaznamenávány, závisí i na konkrétním útoku. Obecně lze u každého útoku měřit bity a pakety za sekundu (bps a pps). Při provozování DoS útoků pomocí TRexe, budeme tyto statistiky vyčítat pomocí API z informací, které si udržuje samotný generátor. Konkrétně budeme vyčítat statistiky, které zaznamenávají kumulativní počet odeslaných bitů/paketů. Z toho pak dopočítáme bps/pps. Kumulativní statistiky jsou takové, které se měří za celou dobu běhu, tj. měříme například, kolik bitů bylo celkem posláno. Naproti tomu průběžné statistiky jsou měřeny s ohledem na čas. Pokud spustíme útok trvající deset vteřin, při vyhodnocování pak v případě průběžných statistik zjistíme, že v prvních pěti sekundách bylo posláno 50 Gb dat a v druhých pak jen 10 Gb. Kumulativní data by nám v tomto případě ukázala pouze souhrnný počet odeslaných bitů za deset sekund, tedy 60 Gb.

U útoků, které nejsou realizovány pomocí TRexe, budeme muset provoz počítat pomocí jiných nástrojů. Můžeme například počítat, kolik bitů/paketů bylo odesláno ze síťového rozhraní, kterým byl útok generován.

Pokud se v rámci testování rozhodneme použít simulovaného klienta, umožní nám to získat další doplňkové statistiky, konkrétně úspěšná spojení a latenci za daný čas. Obě tyto statistiky jsou získávány průběžně. Dokážeme tak vidět časový průběh jejich změn. Můžeme díky tomu srovnávat jak se spojení (ne)daří realizovat při útoku a v klidovém režimu.

Pro pomalé útoky pak chceme vědět jak jsou vytvářena spojení. Případně jak často se daří oběti útočná spojení ukončovat. Chtěli bychom také vědět, ve kterém okamžiku přestala služba komunikovat, tj. kdy bylo dosaženo DoS.

Kromě grafů, ze kterých uživatel okamžitě vizuálně zjistí, jak útok probíhal, by bylo vhodné poskytnout statistiky také ve formě nezpracovaných dat, konkrétně ve formátu CSV nebo JSON. Data pak může uživatel použít k dalším analýzám, zpracováním, atd. Konkrétní sledované statistiky se budou odvíjet od typu útoku. Obecně bude u dat první sledovanou informací čas. Následně budou zaznamenávány další užitečné hodnoty, jako počet odeslaných bitů, paketů, atd. Data budeme zaznamenávat se zadanou vzorkovací frekvencí, která je určena pro každý zdroj paketů pomocí parametru `sampling_rate`.

Kapitola 5

Implementace

Prostředí pro simulaci DDoS útoků jsme se rozhodli implementovat v jazyce Python. Konkrétně za pomoci modulu PyTest [31]. Python je použit, protože dovoluje rychlý vývoj, přičemž samo prostředí nemusí poskytovat extrémní výkon, protože generátory, u kterých je rychlost odesílání důležitá, nejsou implementovány přímo v tomto jazyce. Navíc API k TRex generátoru je také dostupné právě pomocí jazyka Python.

Modul PyTest používáme také z důvodu, že některé pomocné funkce pro jednoduché použití generátoru TRex jsou implementovány právě za pomocí funkcionalit modulu PyTest. Kromě toho můžeme také využít funkcionalit pro automatickou správu a uvolňování zdrojů.

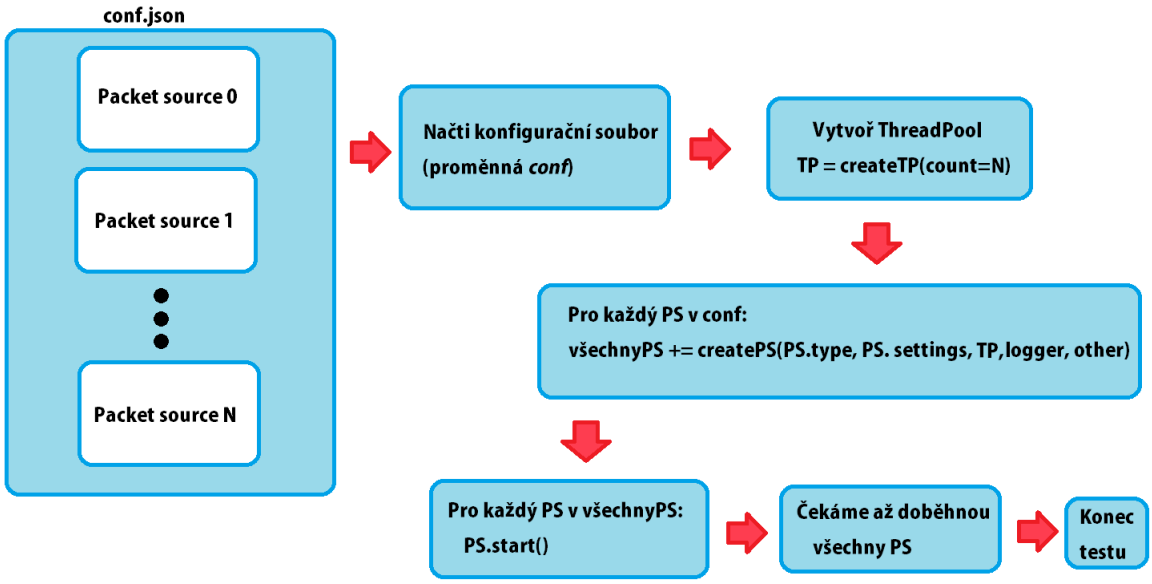
5.1 Celková struktura programu

Na obrázku 5.1 je zobrazen postup při vytváření zdrojů paketů. V levé části obrázku je zjednodušený příklad konfiguračního souboru s N zdroji paketů, který je při spuštění programu načten a použit v dalších fázích. Po načtení souboru už máme informaci o tom, kolik zdrojů paketů bude vytvářeno (v příkladu na obrázku N). Dojde tak k vytvoření ThreadPool (TP), který bude obsahovat N určených vláken, která potřebujeme, protože chceme, aby bylo možné útoky spustit asynchronně.

Následně jsou instancovány jednotlivé generátory paketů. Pro jejich vytvoření je potřeba předat jim informace z konfiguračního souboru a to konkrétně typ (`type`) a jejich specifikované parametry (`settings`). Dále je předán ThreadPool, logger a další zdroje jako například TRex, které jsou v obrázku skryty za položkou other. Při vytvoření instance se inicializují potřebné proměnné do stavu zadaného konfiguračním souborem, nebo do výchozího stavu, pokud nebyla hodnota určena. Následně se vykonají akce specifické pro typ útoku. Pro záplavové útoky je například před spuštěním útoku vytvořen profil provozu. Specifická nastavení se vykonají právě při inicializaci útoku.

Posledním krokem je spuštění všech vytvořených generátorů. Všechny generátory paketů běží nezávisle na sobě v asynchronním režimu, tj. každý útok běží ve vlastním vlákně. Hlavní vlákno počká až všechny definované útoky dokončí generování. Jakmile všechny zdroje paketů úspěšně dokončí svoji práci, dojde k uložení statistik a grafů a test je ukončen.

Všechny útoky jsou podtřídou třídy PacketSource, která implementuje společné metody pro nastartování a zastavení útoku, obecné zpracování statistik, atd. Na obrázku 5.2 je zobrazen zjednodušený diagram tříd pro vybrané zdroje paketů. Všechny zdroje paketů, které je v prostředí možné vytvořit jsou zobrazeny v pravé části ve žlutém rámečku. Hlavní třída PacketSource (na obrázku modře) obsahuje i další atributy a metody, než



Obrázek 5.1: Vytváření zdrojů paketů.

které jsou v příkladu zobrazeny, ale pro přehlednost ukázky byly vybrány pouze nejdůležitější z nich. Jsou to metody pro spuštění (`start_paket_source`) a zastavení zdroje paketů (`stop_paket_source`), uložení statistik (`save_stats`), spuštění aplikace v namespace (`execute_in_namespace`), nastartování generování (`start_paket_source`) a spuštění měření statistik (`start_measurement`).

Třída samotná implementuje ve výchozím stavu i metody pro záplavové útoky. Konkrétní záplavové útoky tak definují metody pro nastavení TRex generátoru, tj. `trex_setup`. Pro ostatní typy útoků je nutné funkce `start_paket_source` a `save_stats` upravit.

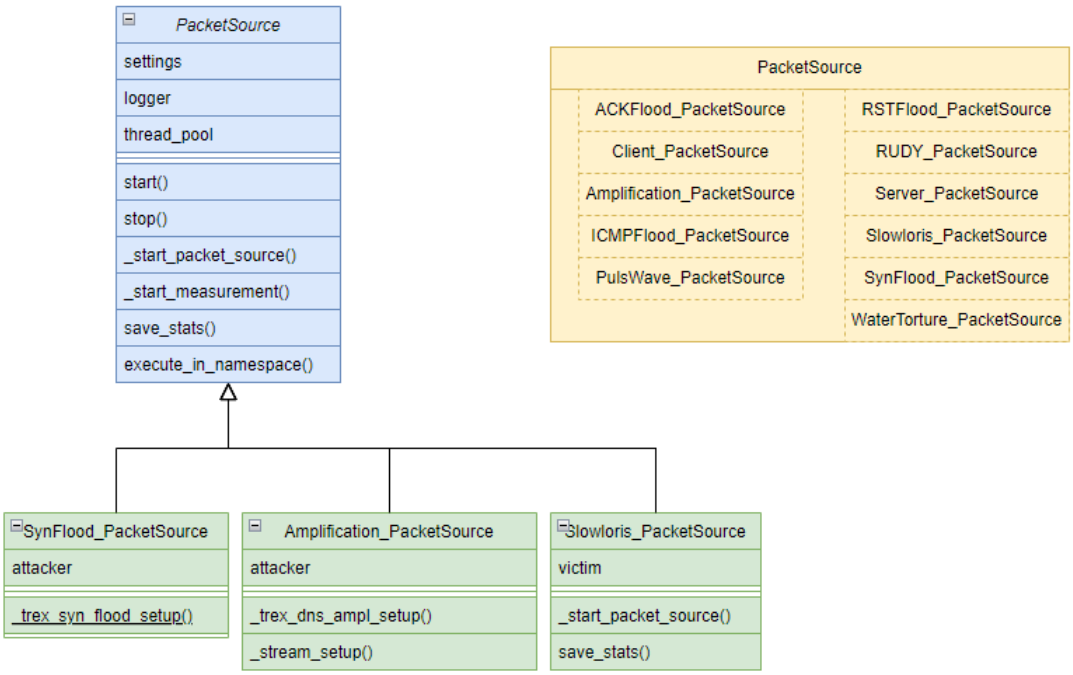
Metody `start` a `stop` jsou zamýšleny jako rozhraní pro spouštění a ukončování zdroje paketů. Startovací metoda ve svém těle volá `start_paket_source`. Tato metoda se postará, aby došlo ke spuštění konkrétního zdroje paketů. Metoda pro zastavení se stará o korektní zastavení zdroje paketů a získání statistik právě pomocí metody `save_stats`.

Metoda `execute_in_namespace` je specifická pro implementace, které nepoužívají TRex generátor, a zajišťuje spouštění programu v namespace. Metoda `start_measurement` je naopak používána právě jen s podtřídami, které TRex využívají. Jejím zavoláním dojde ke spuštění měření statistik.

Dceřiná třída `SYNFlood_PacketSource` oproti rodičovské implementuje navíc metodu `trex_syn_flood_setup`. Ta se volá při inicializaci a stará se o připravení TRex generátoru (nastavení adres, výběr použitého protokolu, ...). Vidíme, že třída má také oproti bázové třídě i atribut navíc (attacker), který uchovává právě nakonfigurovaný TRex generátor. Amplifikace jsou navrženy stejně, jen implementují jednu další metodu (`stream_setup`), která vytváří stream paketů.

Třída pro útok typu Slowloris není implementována pomocí TRex generátrou a vidíme tak, že přepisuje implementace metod `start_paket_source` a `save_stats`. Navíc uchovává jiný atribut (victim), který značí oběť útoku.

Abychom měli pod kontrolou, kterým rozhraním budou pakety odeslány, stačí nám při použití TRex generátoru určit MAC adresu zařízení, na které budeme přesměrovávat gene-



Obrázek 5.2: Diagram tříd s výčtem všech dostupných generátorů paketů.

rovaný provoz. Pokud jsou použity externí implementace, které využívají sockety, využijeme tzv. namespace [15].

Do námi vytvořeného namespace lze přiřadit síťová rozhraní, která chceme používat. Následně můžeme ve stejném namespace spustit aplikace, které budou pracovat s rozhraními dostupnými v daném namespace. Víme například, že na zařízení máme dvě síťová rozhraní, kde A má přístup do internetu a B je připojena pouze do lokální sítě. Při simulaci chceme zajistit, že se veškerý provoz odešle pouze na lokální síť. Pokud aplikaci spustíme bez nastavení namespace, bude provoz generován z výchozího rozhraní, tj. A. Když vytvoříme namespace, do kterého přidáme pouze rozhraní B a ve stejném namespace pak spustíme i požadovanou aplikaci, docílíme tak toho, že aplikace bude mít pro komunikaci do sítě přístupné pouze rozhraní B. Tím tak zajistíme, že se aplikace bude chovat podle našich požadavků.

Vytváření samotného namespace není součástí aplikace, ale je jej nutné vytvořit před spuštěním testovacího prostředí. Je tomu tak především z důvodu flexibility programu. Pokud bychom vytváření namespace integrovali přímo do simulačního prostředí, byl by namespace dostupný pouze při běhu aplikace a dovoloval by pouze předdefinované funkcionality. Tím by mohlo docházet k omezování uživatele. Zvolili jsme tak přístup, kdy si uživatel nakonfiguruje namespace sám před spuštěním testu a do konfiguračního prostředí už předá pouze název vytvořeného namespace, kdy každému útoku může být předán jiný namespace.

Pro základní vytvoření namespace je k prostředí dodán skript, který se o to postará. I méně zkušený uživatel by tak měl mít možnost jednoduše používat útoky, které namespace vyžadují. Neomezujeme tedy ani zkušené uživatele, kteří mohou chtít nakonfigurovat složi-

tější síťování pomocí namespace, ale ani méně zkušené uživatele, pro které by už samotné vytváření namespace mohlo být přítěží.

5.2 Implementované zdroje paketů

V této podkapitole budou podrobněji popsány implementace pro jednotlivé třídy zdrojů paketů.

Záplavové útoky

Pro záplavové útoky je použit bezstavový TRex režim, protože je nežádoucí, aby došlo k vytvoření spojení. Před spuštěním samotného testu je získán formát ukládaných dat, tj. statistik. Dále je při přípravě útoku potřebné nakonfigurovat cílovou MAC adresu a VLAN, čili informace potřebné pro to, aby bylo možné provoz poslat kamkoliv po síti. Následně je vytvořen TRex stream s nastaveními z konfiguračního souboru (adresy, délka paketu, ...). Stream je potřebný pro generování provozu. Funguje v podstatě jako vzor, podle nějž je následně vytvářen provoz (pomocí Field engine komponenty jsou modifikovány IP adresy a další specifikované položky). Pokud je využito nastavení pro zadávání botů, je potřebné, aby byla zdrojová IP adresa zadána jako podsíť, například 1.2.3.4/5. Ze zadané podsítě je následně náhodně vybráno tolik IP adres, kolik bylo požadováno botů. Tímto způsobem dokážeme simulovat použití botnetu. Pokud nastavení botů není využito, ale adresa je zadána stejným způsobem, tj. pomocí podsítě, generují se pakety v plném rozsahu adres. Jednotlivé záplavové útoky jsou odlišeny při vytváření streamu, kdy je nastaven příslušný příznak pro TCP hlavičku (RST, ACK, SYN). Při vytváření streamu je dostupné nastavení, jež určuje, který z protokolů čtvrté vrstvy použijeme. V tomto poli můžeme také zvolit použití ICMP protokolu. Tímto způsobem tak dokážeme implementovat i ICMP Flood. Některá data z TRex jsou kumulativní, tj. sbíráme statistiky za celou dobu běhu. Data jsou tak před generováním grafů předzpracována a převedena na informace, které ukazují jak se hodnoty mění v čase.

Klient a server

K vytvoření klienta i serveru je využito stavového režimu TRex. Pro správné fungování je potřebné určit program, neboli sekvenci paketů, které se mají poslat a které očekávat. Tímto programem se pak klient/server řídí při zpracování provozu. Program, který bude v rámci běhu klienta vykonáván, lze určit v konfiguračním souboru. Ve výchozím nastavení je použit TCP klient, tj. pokud vytvoříme klienta a nedefinujeme jeho typ nebo definujeme neplatný typ, použije se TCP klient. Pro doimplementování vlastního programu stačí přidat funkci, která vytvoří program pro klientskou a serverovou část.

Klient také získává více statistik, proto je přidána rozšířená implementace funkce pro získání formátu dat a funkce pro získání samotných dat. Protože získáváme více statistik, můžeme generovat více grafů. K tomu bylo potřebné doplnit funkce, které ze získaných dat grafy vygenerují. V rodičovské třídě je implementována funkce na vykreslování grafů, ale zpracovává pouze data o počtu odeslaných bitů a paketů, protože tyto statistiky sbírají všechny dceřiné zdroje paketů.

Při použití stavového režimu nastavujeme profil komunikace, tedy jakým způsobem se má klient/server chovat. Při nastavování profilu předáváme informace o tom jaký program

budeme používat, kolik spojení bude ustavováno za sekundu, adresy, atd. Všechna zmíněná nastavení jsou měnitelná přes konfigurační soubor.

Server má v podstatě stejnou strukturu a požadavky. Je tedy implementován velice podobně jako výše popsaný klient. Rozdílem jsou pak samozřejmě grafy a získávaná data.

Výše popsaným způsobem je v testovacím prostředí dostupný TCP a HTTP klient a server. Prostředí dovoluje spustit i ICMP klienta, který není implementován pomocí TRex generátoru, ale je vytvořen pomocí knihovny icmplib [33]. Na rozdíl od předchozích typů klientů, generuje především statistiky o úspěšnosti ICMP požadavků, tedy například celkový počet úspěšných a neúspěšných požadavků, jitter, round-trip time (RTT), atd.

Amplifikace

Amplifikační útoky jsou opět implementovány pomocí bezstavového režimu TRex generátoru. V případě amplifikačních útoků bylo potřebné vytvářet pakety jiným způsobem než je použit u záplavových útoků. API, dostupné pro vytváření paketů, totiž nepodporuje všechna nastavení. Amplifikovaný provoz tak není možné vytvořit stejným způsobem jako tomu bylo u záplavových útoků. Stream pro amplifikační útoky je vytvářen pomocí vzorového paketu, který reprezentuje reálný odraz. Tento paket je tedy použit jako šablona, ve které se provedou příslušné úpravy (změna IP adres, portů, ...). Vzorový paket je ve formátu PCAP a pokud je v tomto souboru více paketů, je vybrán pouze první z nich.

Nastavení v konfiguračním souboru jsou podobná jako u záplavových útoků (zdrojové adresy, rychlost, atd.), ale nenastavuje se délku paketu, protože ta je odvozena ze vzorového paketu. Pro amplifikační útoky je navíc použito nastavení, které určuje jaký vzorový paket bude pro amplifikaci využit. V současném stavu je podporována DNS amplifikace s třemi druhy odrazu. Všechny typy jsou popsány v předchozí kapitole, která se zabývá návrhem.

Přidání dalších druhů amplifikace není složité. Stačí nalézt vhodný paket provozu, který chceme simulovat, a přidat ho k ostatním paketům, které jsou již podporované. Uveďme si příklad pro přidání NTP amplifikace:

1. Rozhodli jsme se přidat NTP amplifikaci. Potřebujeme tedy získat (najít nebo vytvořit) paket, který právě požadovanou NTP amplifikaci obsahuje.

2. Upravíme PCAP soubor, aby obsahoval pouze jeden paket. Bude se s ním tak lépe pracovat a získáme menší soubor, který se rychleji načte, atd.

3. Námi vytvořený paket přidáme do složky s ostatními pakety používanými testovacím prostředím.

4. Přidáme novou alternativu pro vybírání paketů pro amplifikaci, tj. zvolíme jméno, které bude použito pro výběr tohoto paketu v konfiguračním souboru, a přidáme cestu k souboru.

Pokud chceme používat při generování VLANy, musí být ve vzorovém paketu také nastaveny (pokud nechceme, naopak tam být nesmí). Field engine totiž upravuje pouze položky v paketu a nedokáže přímo přidat nebo odebrat hlavičky. Přidat VLAN do paketu je možné buď pomocí jednoduchého přiloženého skriptu, nebo ručně například pomocí knihovny Scapy. Ať už se uživatel rozhodne jakkoliv, úpravu paketů je nutné provést před spuštěním samotného testovacího prostředí.

V konfiguračním souboru můžeme definovat i více zdrojových paketů naráz. Můžeme třeba požadovat, aby byly generovány pakety pro nebezpečný provoz současně s pakety od

zabezpečených odražečů, tj. bude generován amplifikovaný provoz společně s neamplifikovaným. Není ovšem zaručeno v jakém poměru budou data odesílána. V současném stavu se vytvoří dva streamy, které se budou posílat v limitu nastavené rychlosti. Tedy pakety budou brány jako součást jednoho provozu, tj. pokud nastavíme rychlost 5 Gb/s bude poslána kombinace všech zvolených paketů. Nedokážeme tedy například definovat, že nebezpečný odraz má být posílán s menší intenzitou než provoz vytvořený podle bezpečného odrazu. Poměr odeslaného provozu není sice možné určit přímo, ale můžeme tuto funkcionalitu simulovat spuštěním více zdrojů paketů, tj. pokud chceme posílat kombinaci různých paketů, vytvoříme pro každý typ nový zdroj paketů a ke každému přiřadíme jeden určitý typ amplifikačního paketu.

Pro generování grafů a získávání statistik pak platí stejné vlastnosti jaké jsou popsány u záplavových útoků.

Pomalé útoky

Tyto útoky jsou v testovacím prostředí realizovány pomocí externí implementace. Před samotným spuštěním simulace je tedy nutné zkompilovat aplikaci pro tento typ útoku. Toto můžeme udělat buď ručně nebo přiloženým skriptem. Samotné spuštění útoku je realizováno pomocí Python knihovny subprocess [32]. Externí aplikace je spuštěna s předdefinovanými nastaveními, jako jsou uvedena v příkladu na Gitu [40] u zdrojového kódu. Způsob, jakým je externí aplikace spouštěna, nelze jednoduše měnit bez zásahu do kódu testovacího prostředí. Můžeme ovšem měnit parametry těchto nastavení (spojení za sekundu, interval mezi pakety, ...) v konfiguračním souboru.

Simulaci pomalých útoků lze spustit v předem definovaném namespace. Ten lze vytvořit uživatelem zvoleným způsobem před spuštěním aplikace. Do konfiguračního souboru se pak pouze vloží název namespace, který chceme použít. K vytvoření namespace lze také použít přiložený skript.

Použitá implementace pomalých útoků generuje vlastní statistiky do CSV souboru. Ten je po ukončení simulace načten a zpracován do podoby grafu.

DNS Water Torture

Stejně jako pomalé útoky je DNS Water Torture realizován pomocí externí aplikace. Pro tuto implementaci bylo potřebné provést malé úpravy, například přidáni některých knihoven a doimplementování kontrolování délky prefixu navržené v připomínkách v Git repositáři projektu [44].

Opět je potřebné před použitím aplikaci zkompilovat. Aplikace je spuštěna s fixními argumenty příkazové řádky, jejichž přesné hodnoty lze opět nastavit přes konfigurační soubor. Konkrétně se jedná o počet spojení, interval mezi pakety a typ DNS dotazu. Aplikace lze stejně jako pomalý útok spustit v namespace.

Použitá externí implementace neposkytuje žádný sběr statistik. V tomto případě je tedy alespoň implementována velmi jednoduchá funkce pro získávání statistik o provozu na síťovém rozhraní v použitém namespace.

Carpet Bombing

Tato metodika není přímo implementována jako zvláštní útok, ale u všech útoků vytvořených pomocí TRex generátoru lze nastavit rozsah cílových IP adres. Tím lze v podstatě dosáhnout jedné z částí techniky Carpet Bombing.

Druhá část techniky Carpet Bombing, tedy zjišťování limitů DDoS ochrany, není v současné době plně automatická. K zjišťování limitů testované služby můžeme ovšem použít simulovaného klienta. To nám umožňuje spouštět útoky různou silou a právě pomocí klienta sledovat, zda došlo k zamezení přístupu, tj. DoS.

Plně automatická verze zjišťování limitů DDoS ochrany je nad rámec zadání práce, ale může být případně do prostředí doplněna později. Díky tomuto rozšíření by mohlo být možné testovat schopnost DDoS ochrany zachytávání právě i „sondovací“ pakety.

Pulse Wave

Pulse Wave je metoda, která dokáže spustit všechny dříve popsané útoky (samozřejmě kromě sebe). V konfiguračním souboru tak definujeme všechny parametry, které by mohly útoky potřebovat. Navíc definujeme kolik útoků se má provést. Abychom mohli správně provést útoky, potřebujeme, aby byly vykonávány sekvenčně. Prostředí z toho důvodu implementuje i přepínač, díky kterému je možné spouštět útoky i sekvenčně. Pulse Wave útok tak nejprve vybere útok, který se má spustit, spustí ho a počká na jeho ukončení. Pokud ještě nebylo dosaženo maximálního počtu spuštěných útoků, spustí další útok. Každý útok generuje svoje statistiky, je tak možné sledovat jak útok probíhá. Pulse Wave samotný pak generuje informace o jednotlivých útocích.

Kapitola 6

Výsledky

Tato kapitola se zabývá testováním vytvořeného prostředí. U útoků jsme převážně testovali jejich průběh a celkovou funkčnost, případně parametry, které se vyplatí používat. V této kapitole jsou pak shrnuty poznatky z tohoto testování společně s příklady spuštění jednotlivých útoků s konkrétními parametry a výsledky útoků. Nejprve je popsáno testovací prostředí a nástroje, použité pro testování vytvořené implementace simulátoru DDoS útoků. Následně jsou diskutovány konkrétní typy útoků, kde u každého útoku jsou nejprve popsána důležitá nastavení a následně je ukázán a popsán graf s výsledky daného útoku.

6.1 Popis testovacího prostředí

K testování bylo použito několik zařízení. První zařízení, které poskytuje celkem čtyři síťové karty podporující DPDK. Dvě jsou schopné posílat provoz rychlostí 100 Gb/s a druhé dvě 25 Gb/s. Na tomto zařízení budeme spouštět útoky, které využívají TRex generátor. Tedy všechny záplavové a amplifikační útoky. Pro testování těchto útoků použijeme simulovaný server, který je možné vytvořit v našem prostředí. Současně využijeme i možnost vytvoření klienta, díky čemuž dokážeme získat dodatečné statistiky o útoku. Pro všechny testy, kde se využije simulovaný klient a server, budou použity následující parametry entit:

1. Klient
    - IP klienta – 150.0.0.0/24
    - Spojení za sekundu – 1000
    - Doba běhu generátoru – 10 sekund
2. Server
    - IP serveru – 100.0.0.0/26
    - Doba běhu generátoru – 20 sekund

Pomalé útoky byly testován na reálné webové službě. Jedná se o jednoduchou webovou stánku hostovanou na cloudu. Tato webová stránka byla vytvořena právě za účelem testování těchto útoků, které generují poměrně malý objem provozu. Služba běží ve virtualizovaném prostředí na operačním systému Ubuntu 22.04 s jedním CPU. Poskytuje šířku pásma 0.48 Gb/s a 1 GB RAM. Pro srovnání stroj, na kterém běží záplavové útoky (případně i simulovaný server, klient, ...), má 20 jader a přibližně 45 GB paměti. Samotné

pomalé útoky jsou také spuštěny z tohoto zařízení, tj. všechny zdroje paketů, které jsou vytvořeny v rámci testovacího prostředí pro DDoS útoky, jsou generovány z tohoto výkonného zařízení.

6.2 Testovací případy

V této podkapitole budou popsány testovací útoky, které byly provedeny pomocí vytvořeného simulátoru DoS útoků. Všechny útoky byly provedeny pouze na simulovaný server nebo službě, která byla k tomuto účelu určena.

Záplavové útoky

Nástroj pro simulaci DDoS útoků podporuje několik různých záplavových útoků. Jejich způsob spouštění a statistiky jsou ale velice podobné. Proto zde bude uveden příklad spuštění SYN Flood útoku. Konkrétně jsme spustili dva útoky, které se liší objemem vygenerovaného provozu.

Typ útoku: SYN Flood

Parametry testu:

- Zatížení linky – 10 Gb/s a 1 Mb/s
- Doba běhu generátoru – 10 sekund
- Čekání před útokem – 5 sekund
- Počet botů – 100000

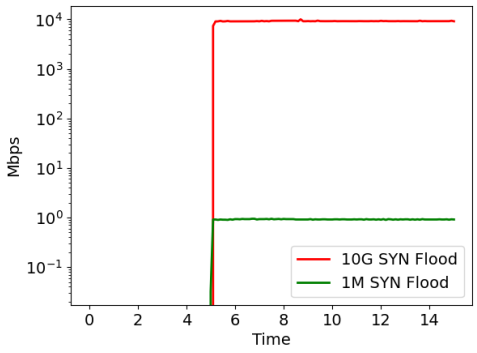
Útok byl proveden na simulovaný server. K získání dodatečných statistik byl také použit simulovaný klient. Na grafech 6.1 lze vidět jak útoky probíhaly. Server a klient byli nastartováni hned po spuštění testu, útoky vždy až po pěti sekundách. Z grafu 6.1a vidíme, že útoky opravdu běží rychlostech 10 Gb/s (červená) a 1 Mb/s (zelená) jak bylo specifikováno.

Na grafu 6.1b vidíme, že se nejprve spojení daří ustavovat. Po pěti sekundách ovšem započne útok. Můžeme si tak všimnout, že právě od času pět sekund klient začíná mít problém navázat spojení se serverem (červená). Toto je způsobeno probíhajícím útokem. Díky tomu dokážeme říct, že se útok daří. Útočníkovi se tak podaří zamezit přístupu ke službě klientovi. V případě slabšího útoku (zelená) k žádnému výkyvu schopnosti klienta navázat spojení nedochází. Můžeme tak říci, že útok, který generuje 1 Mb provozu za sekundu, testovaný server zvládne ustát bez narušení své provozuschopnosti.

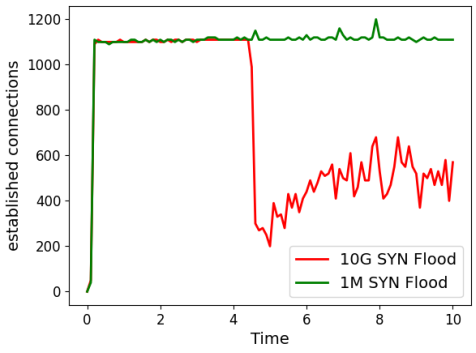
Amplifikace

Pro tento typ útoku jsme zvolili dva testovací případy. Oba s následujícími parametry:

- Zatížení linky – 5 Mp/s
- Doba běhu generátoru – 10 sekund
- Čekání před útokem – 5 sekund



(a) Rychlost SYN Flood.



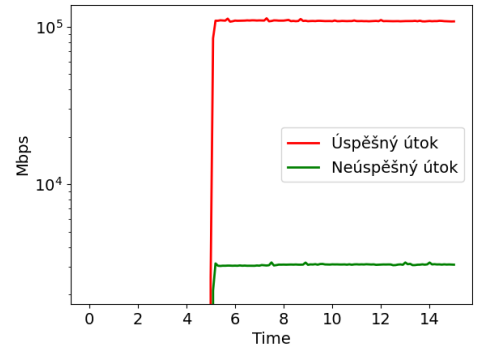
(b) Ustavená spojení klientem.

Obrázek 6.1: SYN Flood útok.

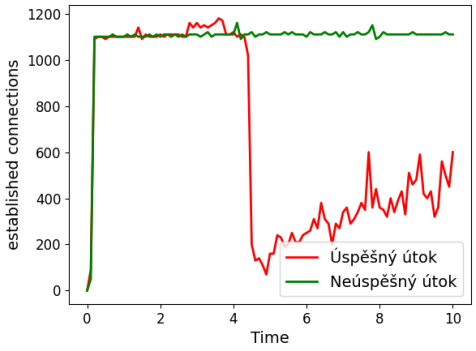
Případy užití se však liší použitými šablonami pro vytvářený provozu. První používá typ pro zabezpečený odražeč, kde nedochází k amplifikaci. Druhý pak používá paket s nebezpečným odrazem, kdy dochází k amplifikaci s faktorem přibližně 24.

Na grafech 6.2 můžeme vidět oba amplifikační útoky, jednak z pohledu vygenerovaného provozu 6.2a a jednak z pohledu klienta, tj. jak se klientovi daří navázat spojení se server během útoku 6.2b. Z grafu 6.2a můžeme také vidět, že provoz vytvořený ze vzoru, který je správně amplifikovaný (červená), generuje při stejném počtu paketů mnohonásobně větší provoz.

První útok, který se z pohledu útočníka nepodařil (zelená), protože byl použit provoz z odražeče, který neprovádí amplifikaci, přičemž druhý útok (červená) se už daří lépe. Vidíme, že z pohledu klienta došlo k situacím, kdy nedokázal navázat spojení.



(a) Rychlosti při různých typech amplifikačních útoků.



(b) Ustavená spojení při různých typech amplifikačních útoků.

Obrázek 6.2: Amplifikační útoky.

Pomalé útoky

Byly testovány oba implementované pomalé útoky, tj. Slowloris a R.U.D.Y. Při provádění tohoto scénáře nebyl použit simulovaný server ani klient, ale útok byl veden na reálnou službu.

Typ útoku: Slowloris a R.U.D.Y.

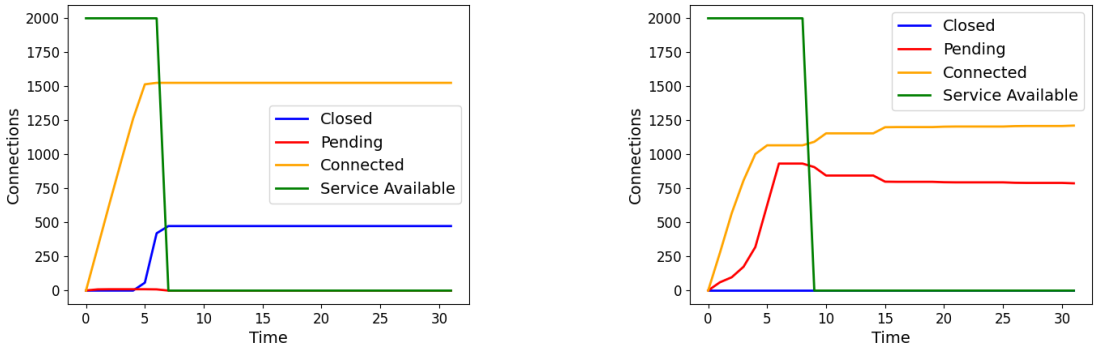
Pro oba pomalé útoky byla použita podobná nastavení:

- Spojení za sekundu – 600
- Celkem spojení – 2000
- Doba běhu generátoru – 30 sekund

V grafu 6.3a vidíme, že po ustavení 1500 útočných spojení, dojde k zamezení přístupu ke službě. V tomto případě je dostupnost služby znázorněna zelenou čárou, která nabývá pouze hodnot 2000, tj. maximum nastavených spojení, a 0. Jedná se tedy pouze o binární proměnou, která je vizualizována spojitě.

V grafu je jistá prodleva mezi nedostupností služby a počtem ustavených spojení (žlutá čára). Ta může být způsobena zpožděním při zjišťování dostupnosti služby, která probíhá každé tři vteřiny. Můžeme si také všimnout, že jsme nastavili počet spojení na 2000, ale je udržováno pouze 1500. Z grafu ale můžeme odvodit, že spojení jsou postupně zavírána, protože čára zobrazující počet ukončených spojení (modrá), je od jisté chvíle na konstantní hodnotě 500. Což souhlasí s námi nastaveným počtem celkových spojení, který byl definován na 2000.

Výsledky útoku typu R.U.D.Y. můžeme vidět na grafu 6.3b. Vidíme, že útok má pomalejší náběh než předchozí útok Slowloris. Vidíme také, že se spojení pomaleji ustavují. Přičemž jich je stále udržováno přibližně 1500 i když je nastaveno 2000, tj. nastává podobná situace jako u předchozího útoku. Limit serveru je zřejmě 1500 souběžných spojení, přičemž vidíme, že některá spojení čekají na ustavení. To může být způsobeno nižší výkonností serveru při ustavování spojení pro čtení. V jisté chvíli jsou vyčerpána všechna spojení (i čekající), což zabrání legitimnímu uživateli službu používat, a tím dojde k zamezení přístupu legitimnímu klientu.



(a) Slowloris útok na reálný server.

(b) R.U.D.Y. útok na reálný server.

Obrázek 6.3: Pomalé útoky.

DNS Water Torture

Tento útok byl testován pouze na lokální síti. Zaměření tohoto testování bylo spíše na samotné chování útoku, chování aplikace v namespace a sběr statistik z namespace.

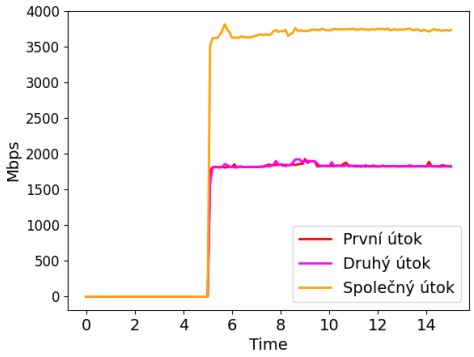
Otestovali jsme tak funkčnost externího generátoru sledováním struktury provozu pomocí nástroje tcpdump a wireshark. Mimo jiné se právě projevil problém definovaný v repositáři tohoto nástroje [44], který upozorňuje na nevalidní velikost přidávaného prefixu do domény. Pokud by tak nebyla aplikována navrhovaná změna, větší množství vytvořených paketů by bylo nevalidních, což by mohlo narušit průběh útoku. Útok by tak při stejných nastaveních mohl vykazovat nekonzistentní výsledky.

Multi-vector útok

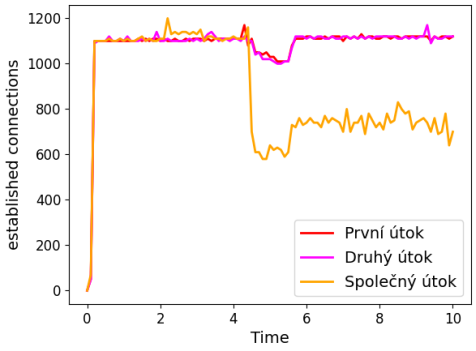
Pro tento test jsme vybrali dva SYN Flood útoky, kdy každý ze záplavových útoků cílil na odlišnou část podsítě simulovaného serveru. Oba útoky měly následující parametry:

- Síla SYN Flood – 2 Gb/s
- Doba běhu generátoru – 10 sekund
- Čekání před útokem – 5 sekund

V grafech 6.4 vidíme jak multi-vector útok probíhal. Pro srovnání jsou zobrazeny i průběhy samostatných útoků (červená a fialová). Hlavní sledovaný multi-vector útok je pak v grafech zobrazen žlutou barvou. Vidíme, že samostatné útoky způsobily pouze malý dočasný výpadek služby. Jakmile se ovšem útoky spojily, tj. byly použity oba naráz, došlo k výraznému zhoršení schopnosti klienta navázat spojení se serverem.



(a) Rychlost při společném útoku oproti útokům samostatně.



(b) Ustavená spojení klientem při společném útoku oproti útokům samostatně.

Obrázek 6.4: Multi-vector útok.

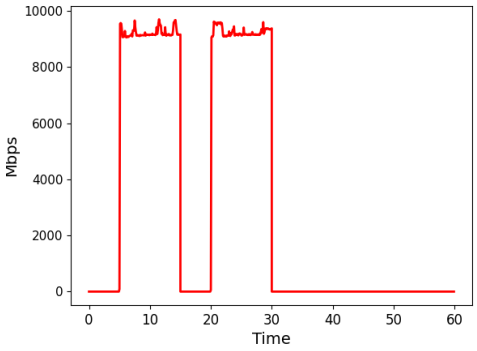
Pulse Wave

Posledním testovacím případem byl Pulse Wave útok. Spustili jsme dvakrát po sobě stejný SYN Flood. Útok byl proveden na simulovaný server. Opět jsme měli k dispozici simulovaného klienta pro dodatečné statistiky. Útok byl spuštěn s následujícími parametry:

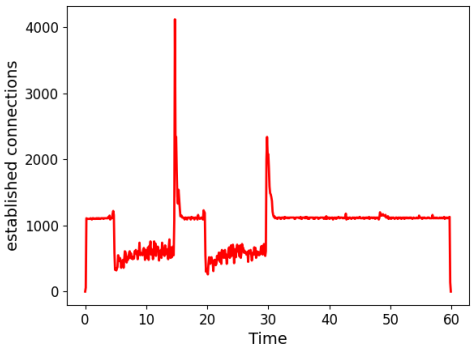
- Čekání před útokem – 5 sekund
- Délka trvání jednotlivých útoků – 10 sekund
- Prodleva mezi útoky – 5 sekund
- Síla útoku – 10 Gb/s

V grafech 6.5 můžeme sledovat pro tento útok charakteristické pulzy. A to jak v grafu zobrazujícím rychlost generování dat záplavovým útokem 6.5a, tak pak hlavně v chování klienta při navazování spojení se serverem 6.5b. Vidíme, že první pulz začíná přibližně po pěti sekundách, tj. v době kdy začal první útok. Druhý pak začne ve dvacáté sekundě běhu simulace, opět na začátku běhu druhé záplavy. Toto chování tak odpovídá nastavením útoku, kdy předpokládáme, že první útok běží od páté do patnácté vteřiny scénáře. Druhý pak od dvacáté do třicáté sekundy. Tento předpoklad pak vidíme i v grafu.

Vidíme také skokový nárůst úspěšných spojení po ukončení útoku, tj. patnáctá a třicátá vteřina v grafu 6.5b. Ten je způsoben tím, že klient při neúspěchu při navázání spojení znovu pošle požadavek na připojení. Přičemž se ale stále vytváří nová spojení. Po ukončení útoku tak dojde k nahromadění požadavků, které se naráz splní.



(a) Generovaný objem dat během trvání Pulse Wave simulace (60 sekund).



(b) Ustavených spojení klientem během trvání Pulse Wave simulace (60 sekund).

Obrázek 6.5: Průběh Pulse Wave útoku zobrazen na grafu klientských spojení.

Kapitola 7

Závěr

Práce byla zaměřena převážně na fenomén DDoS útoků. Ty jsou v dnešní době, kdy už se nikdo neobejde bez internetu, stále běžnější. Byly popsány obecné principy fungování DoS útoků. Byl také vysvětlen rozdíl mezi základním DoS útokem a jeho distribuovanou variantou DDoS. Bylo diskutováno fungování nejpoužívanějších DDoS útoků za poslední dva roky. Jedná se například o útoky záplavou a amplifikační útoky. Byly popsány také pomalé útoky a technika Carpet Bombing. Všechny tyto zmíněné útoky byly také implementovány do testovacího prostředí.

V neposlední řadě byly popsány i generátory provozu, které jsou pro vytvoření testovacího prostředí nezbytné. Cílem bylo nalézt vhodný generátor provozu pro vybrané typy útoků. Pro realizaci záplavových a amplifikačních útoků byl zvolen generátor Cisco TRex, který je postaven nad DPDK frameworkem, díky čemuž dovoluje generovat provoz o vysoké rychlosti i nad komoditním hardware. Tím pádem není vytvořené řešení závislé na žádném specifickém a drahém hardware, jež vyžadují komerční generátory, jako například Spirent Test Center. Pro pomalé a ostatní útoky, u kterých není potřebné generovat tak velké objemy provozu, byla použita již existující implementace. Do vytvořeného prostředí tak byly integrovány právě volně dostupné implementace Slowloris, R.U.D.Y. a DNS Water Torture.

Vytvořené testovací prostředí je implementováno v jazyce Python. Umožňuje generování více útoků současně, dokážeme tak simulovat i multi-vector útoky. Díky možnosti konfigurace časování jsme schopni vytvářet scénáře útoku, tj. spouštět útoky po sobě v zadaných intervalech.

Kromě generování útoků, dokáže nástroj také simulovat legitimní klientský provoz. Můžeme tak sledovat dostupnost služby během útoku, případně jak moc se v době útoku kvalita služby zhoršila. Nástroj umožňuje i simulaci serveru, který je pod útokem. To je užitečné při použití systému k testování nějaké DDoS ochrany. Lze tak porovnat, jaká část útoku se dostane na server v případě aktivní či neaktivní ochrany.

Vytvořený testovací nástroj dovoluje i celou řadu parametrizací útoků. Kromě síly daného útoku, ať už počtem paketů, bajtů, nebo otevřených spojení, lze definovat počet botů generujících takový útok. U amplifikačních útoků lze definovat, jaký typ odrazu má být použit. Kromě zranitelných strojů lze simulovat i odraz od služeb, které nezpůsobují amplifikaci.

Simulační nástroj je snadno rozšiřitelný. Co se týče amplifikačních útoků, stačí přidat PCAP vzorek a tím zařídit možnost simulování jiného typu odrazu například QUIC, který se začíná v poslední době ve větší míře také objevovat, i když jeho zastoupení není tak výrazné. Nové útoky jde jednoduše přidávat jako novou implementaci abstraktní třídy, která zastřešuje všechny podporované útoky.

Tvorba prostředí probíhala ve spolupráci se sdružením CESNET. Počítá se s praktickým využitím výsledku v rámci aktivit CESNET, ať už pro testování odolnosti vlastní interní infrastruktury nebo infrastruktury do CESNET připojených nebo i externích organizací. V neposlední řadě bude prostředí využité také při vývoji systému DDoS ochrany DDoS Protector, který sdružení CESNET vyvíjí. Možným pokračováním práce je samozřejmě rozšiřování prostředí o další skupiny a typu útoků, ale také například zdokonalení sběru statistik. S novými statistikami je potom možné dále implementovat plnohodnotnou automatizaci Carpet Bombingu a autonomně detekovat, kdy došlo k zahlcení cílového systému a automaticky přizpůsobovat parametry útoku, aby byl útok optimální z pohledu použitých zdrojů vzhledem k způsobeným škodám.

Literatura

[1] *About Scapy*. [cit. 2024-01-14]. Dostupné z: https://scapy.readthedocs.io/en/latest/introduction.html#about-scapy.

[2] Brooks, R. R. R. *Distributed denial of service attacks : real-world detection and mitigation*. First edition. Boca Raton: CRC Press, 2020. ISBN 978-1-138-62681-2.

[3] Damon, E., Dale, J., Laron, E., Mache, J., Land, N. et al. Hands-on denial of service lab exercises using SlowLoris and RUDY. In: [online]. říjen 2012 [cit. 2023-12-11]. DOI: 10.1145/2390317.2390321. Dostupné z: https://dl.acm.org/doi/pdf/10.1145/2390317.2390321.

[4] *DDoS Attack Vectors* [online]. Netscaut, 2023 [cit. 2023-12-07]. Dostupné z: https://www.netscout.com/threatreport/ddos-attack-vectors/.

[5] *DNS Amplification* [online]. Imperva [cit. 2023-12-05]. Dostupné z: https://www.imperva.com/learn/ddos/dns-amplification/.

[6] *DNS Water Torture: how not to drown in this tsunami of requests* [online]. Tarlogic, květen 2023 [cit. 2023-12-10]. Dostupné z: https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/.

[7] *DPDK Project*. [cit. 2024-03-24]. Dostupné z: https://www.dpdk.org/.

[8] Erb, B. *Simple slowloris in Python*. [cit. 2024-03-25]. Dostupné z: https://github.com/vs-uulm/slowloris-ng.

[9] Ganti, V. a Yoachimik, O. *A Brief History of the Meris Botnet* [online]. Cloudflare, listopad 2021 [cit. 2023-11-26]. Dostupné z: https://blog.cloudflare.com/meris-botnet/.

[10] Haaijer, L. *DDoS Packet Capture Collection*. 2022 [cit. 2024-03-15]. Dostupné z: https://github.com/StopDDoS/packet-captures/blob/main/amp.UDP.DNSANY.pcap.

[11] *Hacktivists step back giving way to professionals: a look at DDoS in Q3 2022* [online]. Kaspersky, listopad 2022 [cit. 2023-11-24]. Dostupné z: https://www.kaspersky.com/about/press-releases/2022_hacktivists-step-back-giving-way-to-professionals-a-look-at-ddos-in-q3-2022.

[12] *Hidden threat of Pulse Wave DDoS attacks* [online]. DDOS-GUARD [cit. 2024-01-14]. Dostupné z: https://ddos-guard.net/en/blog/hidden-threat-of-pulse-wave-ddos-attacks.

[13] *How to DDoS | DoS and DDoS attack tools* [online]. Cloudflare [cit. 2024-01-07]. Dostupné z: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/how-to-ddos/.

[14] *Implementation of DNS Water Torture DDoS attack.* [cit. 2024-01-14]. Dostupné z: https://github.com/nicovell3/DNSWaterTorture.

[15] *Ip-netns manual page.* [cit. 2024-03-24]. Dostupné z: https://man7.org/linux/man-pages/man8/ip-netns.8.html.

[16] *Ixia-C.* [cit. 2024-01-14]. Dostupné z: https://github.com/open-traffic-generator/ixia-c.

[17] *IxNetwork* [online]. Keysight [cit. 2024-01-14]. Dostupné z: https://www.keysight.com/us/en/products/network-test/protocol-load-test/ixnetwork.html.

[18] *Kaspersky unveils an overview of IoT-related threats in 2023* [online]. Kaspersky, září 2023 [cit. 2023-11-24]. Dostupné z: https://www.kaspersky.com/about/press-releases/2023_kaspersky-unveils-an-overview-of-iot-related-threats-in-2023.

[19] *Killnet Hackers Launch DDoS Attack on U.S. Federal Tax Payment System Website* [online]. Oreanda-News, červenec 2022 [cit. 2023-11-26]. Dostupné z: https://www.oreanda-news.com/en/it_media/killnet-hackers-launch-ddos-attack-on-u-s-federal-tax-payment-system-website/article1436125/.

[20] Kowalski, M. *Modified version of DNS-Flood tool.* [cit. 2024-03-25]. Dostupné z: https://github.com/mikolaj-kow/dns-flood-ng.

[21] Krčmář, P. *Českým bankám nefungují weby a služby, útoky DDoS přicházejí ze zahraničí* [online]. Root [cit. 2023-11-24]. Dostupné z: https://www.root.cz/zpravicky/ceskym-bankam-nefunguji-weby-a-sluzby-utoky-prichazeji-ze-zahranici/.

[22] Kupreev, O., Gutnikov, A. a Shemlev, Y. *DDoS attacks in Q3 2022* [online]. SECURELIST by Kaspersky, listopad 2022 [cit. 2023-11-24]. Dostupné z: https://securelist.com/ddos-report-q3-2022/107860/.

[23] *Memcached DDoS attack* [online]. Cloudflare [cit. 2023-12-07]. Dostupné z: https://www.cloudflare.com/learning/ddos/memcached-ddos-attack/.

[24] Morales, D. *What is Memcached?* [online]. Medium, leden 2021 [cit. 2023-12-07]. Dostupné z: https://medium.com/swlh/what-is-memcached-d1498623db3b.

[25] Nicholson, P. *Five Most Famous DDoS Attacks and Then Some* [online]. A10, 2022 [cit. 2024-03-24]. Dostupné z: https://www.a10networks.com/blog/5-most-famous-ddos-attacks/.

[26] *NoName057(16): Pro-Russian Hacktivist Group* [online]. Radware, 2023 [cit. 2023-11-26]. Dostupné z: https://www.radware.com/cyberpedia/ddos-attacks/noname057(16)/.

[27] *NTP amplification DDoS attack* [online]. Cloudflare [cit. 2023-12-05]. Dostupné z: https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/.

[28] *Overview of TRex* [online]. Cisco TRex [cit. 2024-01-02]. Dostupné z: https://trex-tgn.cisco.com/trex/doc/trex_manual.html#_overview_of_trex.

[29] *Ping (ICMP) flood DDoS attack* [online]. Cloudflare [cit. 2023-12-10]. Dostupné z: https://www.cloudflare.com/learning/ddos/ping-icmp-flood-ddos-attack/.

[30] *PRINCIPLES AND CHARACTERISTICS OF TCP REFLECTION ATTACKS* [online]. NSFocus, červenec 2021 [cit. 2023-12-07]. Dostupné z: https://nsfocusglobal.com/principles-and-characteristics-of-tcp-reflection-attacks.

[31] *Python framework PyTest.* [cit. 2024-04-23]. Dostupné z: https://docs.pytest.org/en/8.1.x/.

[32] *Python knihovna subprocess.* [cit. 2024-03-30]. Dostupné z: https://phttps://docs.python.org/3/library/subprocess.html.

[33] *Python package icmplib.* [cit. 2024-03-30]. Dostupné z: https://pypi.org/project/icmplib/.

[34] *Python package lbr_testsuite.* [cit. 2024-03-24]. Dostupné z: https://pypi.org/project/lbr-testsuite/.

[35] Raghavan, S. V. *An investigation into the detection and mitigation of denial of service (DoS) attacks : critical information infrastructure protection.* New Delhi: Springer, 2011. ISBN 978-81-322-0276-9.

[36] Raviv, I. *DDoS Carpet-Bombing – Coming In Fast And Brutal* [online]. Radware [cit. 2024-01-14]. Dostupné z: https://www.radware.com/blog/ddos-protection/2023/07/ddos-carpet-bombing-coming-in-fast-and-brutal/.

[37] Rijswijk Deij, R. van, Sperotto, A. a Pras, A. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In: *Proceedings of the 2014 Conference on Internet Measurement Conference.* New York, NY, USA: Association for Computing Machinery, 2014, s. 449–460. IMC '14. DOI: 10.1145/2663716.2663731. ISBN 9781450332132. Dostupné z: https://doi.org/10.1145/2663716.2663731.

[38] *R U Dead Yet? (R.U.D.Y.) attack* [online]. Cloudflare [cit. 2023-12-11]. Dostupné z: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/r-u-dead-yet-rudy/.

[39] *R.U.D.Y. (R-U-Dead-Yet?)* [online]. Imperva [cit. 2023-12-11]. Dostupné z: https://www.imperva.com/learn/ddos/rudy-r-u-dead-yet/.

[40] Shekyan, S. *SlowHTTPTest.* [cit. 2024-03-03]. Dostupné z: https://github.com/shekyan/slowhttptest.

[41] *Slowloris DDoS attack* [online]. Cloudflare [cit. 2023-12-11]. Dostupné z: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/.

[42] *Spirent TestCenter Platforms, Brochure* [online]. Spirent, 2018 [cit. 2024-01-03]. Dostupné z: https://www.spirent.com/assets/u/spirent_testcenter_platform_brochure.

[43] *SYN flood attack* [online]. Cloudflare [cit. 2023-12-07]. Dostupné z: https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/.

[44] Szanto, I. C. *Reduce the maximum label size.* [cit. 2024-03-25]. Dostupné z: https://github.com/mikolaj-kow/dns-flood-ng/pull/2.

[45] *Tcpdump and libpcap.* [cit. 2024-04-05]. Dostupné z: https://www.tcpdump.org/manpages/pcap.3pcap.html.

[46] *TCP/IP vs OSI Model – Difference Between Them.* [cit. 2024-01-14]. Dostupné z: https://vitolavecchia.altervista.org/wp-content/uploads/2019/06/Caratteristiche-e-livelli-del-modello-TCP-IP-nelle-reti-di-telecomunicazioni-960x492.png.

[47] *The Pktgen Application.* [cit. 2024-01-14]. Dostupné z: https://pktgen-dpdk.readthedocs.io/en/latest/.

[48] Tran, D. *Prostředí pro testování zařízení umožňujících ochranu před DoS útoky.* Brno, CZ, 2020. [cit. 2024-01-02]. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/129336.

[49] *TRex Advance stateful support* [online]. Cisco TRex [cit. 2024-01-02]. Dostupné z: https://trex-tgn.cisco.com/trex/doc/trex_astf.html.

[50] *TRex Realistic Traffic Generator* [online]. Cisco TRex [cit. 2024-01-02]. Dostupné z: trex-tgn.cisco.com.

[51] *TRex Stateless support* [online]. Cisco TRex [cit. 2024-01-02]. Dostupné z: https://trex-tgn.cisco.com/trex/doc/trex_stateless.html.

[52] Vicens, A. *Pro-Russian cybercriminals briefly DDoS Congress.gov* [online]. CyberScoop, červenec 2022 [cit. 2023-11-26]. Dostupné z: https://cyberscoop.com/killnet-congress-ddos-russia-hacktivist/.

[53] Wagnon, J. *The DNS Water Torture Attack* [online]. F5 DevCentral, říjen 2018 [cit. 2023-12-10]. Dostupné z: https://www.youtube.com/watch?v=sXtrRpsq1vc&ab_channel=F5DevCentral.

[54] *What is a DDoS attack?* [online]. Cloudflare [cit. 2024-01-07]. Dostupné z: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/.

[55] *What Is A DDoS Attack?* [online]. Radware [cit. 2024-01-07]. Dostupné z: https://www.radware.com/cyberpedia/ddospedia/ddos-meaning-what-is-ddos-attack/.

[56] *What is a low and slow attack?* [online]. Cloudflare [cit. 2023-12-10]. Dostupné z: https://www.cloudflare.com/learning/ddos/ddos-low-and-slow-attack/.

[57] *What Is a Slowloris DDoS Attack?* [online]. Akamai [cit. 2023-12-11]. Dostupné z: https://www.akamai.com/glossary/what-is-a-slowloris-ddos-attack.

[58] *What Is an ICMP Flood DDoS Attack?* [online]. Akamai [cit. 2023-12-10]. Dostupné z: https://www.akamai.com/glossary/what-is-icmp-flood-ddos-attack.

[59] *What is the Mirai Botnet?* [online]. Cloudflare [cit. 2023-11-26]. Dostupné z: https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/.

[60] *What is the Mirai botnet?* [online]. Malwarebytes [cit. 2023-11-26]. Dostupné z: https://www.malwarebytes.com/what-was-the-mirai-botnet.

[61] YALTIRAKLI, G. Slowloris. *Github.com*. 2015, [cit. 2024-01-14]. Dostupné z: https://github.com/gkbrk/slowloris.

[62] YOACHIMIK, O. *DDoS Attack Trends for 2022 Q1* [online]. Cloudflare, duben 2022 [cit. 2023-11-24]. Dostupné z: https://blog.cloudflare.com/ddos-attack-trends-for-2022-q1/.

[63] YOACHIMIK, O. *Mantis - the most powerful botnet to date* [online]. Cloudflare, červenec 2022 [cit. 2023-11-26]. Dostupné z: https://blog.cloudflare.com/mantis-botnet/.

[64] YOACHIMIK, O. a PACHECO, J. *DDoS threat report for 2023 Q3* [online]. Cloudflare, říjen 2023 [cit. 2023-11-24]. Dostupné z: https://blog.cloudflare.com/ddos-threat-report-2023-q3/.