Princípy informačnej bezpečnosti

Zvýšenie odolnosti webových aplikácií proti útokom typu DDoS, za pomoci horizontálneho škálovania

Miroslav Hájek

Akademický rok: 2020 / 2021

Fakulta informatiky a informačných technológií,
Slovenská technická univerzita v Bratislave

Obsah

1 Dostupnosť ako bezpečnostný atribút

Zabezpečenie nepretržitého prístupu k webovým službám je očakávanou a takmer nevyhnutnou požiadavkou pre akýkoľvek významnejší informačný systém. Predstavuje neoddeliteľnú súčasť obrazu o spoľahlivosti ich prevádzkovateľov pôsobiacich vo virtuálnom priestore internetu. Aspekt dostupnosti sa prejavuje tým, že údaje sú k dispozícii pre autorizovaných používateľov okamžite a bez nečakaných obmedzení [1]. Odlišná interpretácia definuje pojem dostupnosti ako ochranu proti zlomyseľnému zatajovaniu informácií. Spoločným menovateľom pre oba tieto pohľady je dôraz na všadeprítomnosť služby v ľubovoľnom čase potreby. Taký stav je vskutku ideálny, ale je možné sa mu aspoň priblížiť predovšetkým identifikáciou bodov zlyhania alebo miest prieniku a ich následným systematickým eliminovaním.

Pre bezpečné nakladanie s informáciami nestačí samotná dostupnosť, ale zároveň je potrebné pri návrhu a prevádzke systémov myslieť aj na dôvernosť a integritu údajov. Spolu tvoria tradičný model informačnej bezpečnosti označovaný ako tzv. *CIA triáda* (Confidentiality, Integrity, Availablity), ktorý sa často spolieha na vyváženosť a rovnocennosť týchto troch prvkov [1]. Nutno poznamenať, že to úplne neplatí, pretože nežiadaným znemožnením prístupu k zdrojom sa ich neporušenosť a zabezpečenosť proti neoprávnenému prezeraniu, či úprave, stáva bezpredmetná. O dostupnosť sa teda opierajú všetky ďalšie bezpečnostné predpoklady, ktoré má systém napĺňať.

1.1 Kľúčový zabezpečovatelia dostupnosti

Komunikačné technológie často tvoria chrbtovú kosť väčšiny moderných biznisov pričom ich hlavnou úlohou je sprostredkovanie informácií naprieč organizáciou a podieľajú sa na riadení podnikových procesov. Okrem ľudského kapitálu sa spoliehajú na tri kľúčové prvky: *softvér, hardvér a počítačovú sieť* [1]:

Softvér: Softvér je najkritickejším komponentom spomedzi vymenovaných, pretože na základe príkazov v kóde programov je ovládaný hardvér a sieťové zariadenia. Všetky potenciálne útoky a ich dopady musia byť riešené primárne na úrovni softvéru. Na napadnutie sú využívané zraniteľnosti systému, sprístupnené prelomením nedostatočne zabezpečeného verejného rozhrania služieb. Najčastejším cieľom útočníkov je dosiahnutie kontroly nad zariadením alebo vyvolaním chaosu vo fungovaní prevádzky.

Zlyhanie programového vybavenia nemusí byť iba v dôsledku nepriaznivých vonkajších vplyvov, ale tiež sa prejavujú chyby spôsobené nekorektným návrhom alebo implementáciou systému s odchýlkami od požadovaného správania. Tieto chyby sú vnesené neúmyselne najčastejšie programátorom. Počas behu aplikácie môžu nastať zlyhania operačného prostredia zapríčinené nedostatkom pamäte pri alokácii, zaplneniu diskového úložiska alebo uviaznutím systému.

Hardvér: Poruchy hardvéru bývajú zriedkavejšie, ale o to podstatne fatálnejšie pre celkový chod, keď výpadok nie je adresovaný redundanciou komponentov. Prinavrátenie do funkčného stavu znamená výmenu zariadenia za iné prevádzkyschopné, či už dočasnou úpravou fyzickej infraštruktúry alebo neodkladnou montážou náhrady. Duplikáciou napr. diskov cez RAID dosiahneme síce vyššiu dostupnosť ale za cenu integrity z dôvodu zdvojením dát [1], preto je vždy potrebné mať na pamäti vyváženosť bezpečnostných vlastností navzájom.

Sieť: Obmedzením počítačových sietí je ich priepustnosť podmienená šírkou prenosového pásma a réžiou spotrebovanou na obsluhu zvolených komunikačných protokolov. Problémy nastávajú v situáciach, kedy sa zahltí sieťová linka. Za normálnych okolností rešpektujú uzly v sieti signalizáciu zaznačenú do paketov a prispôsobia rýchlosť vysielania, čím sa po istom čase uľaví náporu. Útočník, ktorý chce saturovať serverové pripojenie to pochopiteľne nerešpektuje a preto by sa nadmerná premávka mala presmerovať a filtrovať. Pokiaľ bude smerovač vystavený niekoľkonásobnej záťaži než je schopný podporovať, určite vyvolá straty paketov a zvýšenú latenciu.

1.2 Dôvody viktimizácie prevádzkovateľov

Nedostupnosť služieb máva za následok, buď priame škody v podobe finančných strát alebo vplýva na pošramotenie nadobudnutej reputácie u klientov, ktorí si zlyhanie môžu spájať so stratou spoľahlivosti a dôveryhodnosti služby. Poškodenie reputácie sa radí so 47% medzi najväčšie obavy firiem počas kybernetického útoku [2]. Ďalšími zreteľmi starostí sú potenciálna strata ziskov s 21%, obmedzená dostupnosť s 12% a zníženie produktivity počas útoku na 7%. Čím väčší poskytovateľ a používanejšia webová stránka, tým rozsiahlejšie sú potenciálne dopady pri neočakávanom vyradení z prevádzky. Zároveň dochádza k adekvátnemu navýšeniu zriadenej odolnosti investovanej do predchádzania a aktívnej defenzívy proti útokom.

Motivácie a dôvody stojace za aktivitami úmyselne narúšajúcimi dostupnosť zvolených obetí v podobe webových aplikácií sa líšia od prípadu k prípadu, ale sú zhrnuteľné do nasledujúcich okruhov [3] [4]:

- *Kapitálové zisky* - nadobudnutie finančnej odplaty od objednávateľa alebo nekalé snahy o potopenie konkurencie v ekonomickej súťaži predstavujú významný hnací faktor pre útočníkov. Hlavným zámerom je lepšie peňažné zabezpečenie sa nehľadiac na taktiky vynaložené na tento účel. Úspešná realizácia vyžaduje značné technické zručnosti, pretože firmám, ktoré sú hlavných cieľom, ide o veľa. Zvykne sa dotýkať komerčných webstránok alebo služieb finančných inštitúcií (zaznamenané útoky na HSBC, BTC a Ethereum burzy), serverov a sieťových zariadení poskytovateľov

webhostingu alebo internetového pripojenia (Deutsche Telekom, OVH, Dyn) alebo serverov herných spoločností (Steam, Blizzard, EA Sports) [3]

- *Pomsta* - prevažne frustrovaný jednotlivci snažiaci sa o odplatu za vnímanú nespravodlivosť, ktorú podľa nich prevádzkovateľ pácha. Počas operácie Payback z roku 2010 bol odplatou hackerskej skupiny Anonymous, za blokovanie stránok s torentami a pirátskym softvérom, útok odoprenia služieb na organizácie chrániace autorské práva. V decembri zhodili stránky MasterCard, Visa, Paypal a iných, ktoré vydržiavali donácie organizácii Wikileaks, pretože publikovala prísne tajné informácie americkej vlády.
- *Ideologické a politické presvedčenie* - útočník sa snaží dať hlasno najavo svoj nesúhlas s ideovo protichodnými názormi a postojmi znefunkčnením alebo poškodením platformy prostredníctvom ktorej oponent pôsobí alebo šíri svoj svetonázor. Takáto forma útokov sa označuje tiež za hacktivizmus, ktorého prívrženci hájia slobodu slova a právo na súkromie proti nadmernému sledovaniu. Pre predstavu napádaných webstránok boli prominentné útoky roku 2016 cielené na skupiny ako Black Lives Matter, Ku Klux Klan, Wikileaks, oboch prezidentských kandidátov v USA, taliansku a írsku vládu alebo európsku komisiu. [3]
- *Demonštrácia schopností* - ide o experimentálne útoky so zámerom hackera vyskúšať si nové techniky alebo predviesť svoje kompetencie.
- *Kybernetický terorizmus* - útočník je súčasťou vojenskej alebo teroristickej operácie s cieľom poškodenia nepriateľovi. Kritická infraštruktúra štátu prestavuje najčastejšie zasahovaný cieľ.

1.2.1 Teória rutinných aktivít

Výber primeranej obete má taktiež svoj podiel na úspešnosti škodlivého zásahu útočníka do prevádzky, pretože ten sa výrazne nelíši od konvenčného zločinu. Medzi obvykle spomínané zdôvodnenia páchania kriminality zo sociologickej perspektívy patrí kriminologická teória známa pod názvom *teória rutinných aktivít*. Vysvetľuje predpoklady, aké musí daný subjekt spĺňať nato, aby bol zasiahnutý. Vyslovuje, že zločin sa udeje vtedy, keď motivovaný útočník, so sklonmi na páchanie trestnej činnosti, príde do stretu s objektom ponechaného bez prítomnosti schopného strážcu [5].

Pokiaľ sú vytvorené priaznivé okolnosti na prelomenie do systému je veľká šanca, že sa motivovaný útočník bude snažiť o zneužitie bezpečnostnej diery. Na druhej stane pri absencii jediného kritéria, sa buď znižujú šance na viktimizáciu alebo sa úplne vylúčia. Podľa *teórie racionálneho jednania* koná útočník pri zvažovaní realizácie svojho činu racionálne, hoc sa jedná o obmedzenú rozumnosť a síce z uhľa pohľadu delikventa ide o cieľavedomé rozhodnutie, kde pozitívny obnos prevažuje nad možnými rizikami uskutočnenia.

Vhodnosť zamerania sa na zvolenú infraštruktúru pre potenciálneho páchateľa je zachytiteľné kritériami *VIVA* (Value, Inetria, Visibility, Accesibility) [3]. V kontexte útokov odmietnutia služby sa pod *hodnotou* rozumie dôležitosť rozbitia cieľa pre útočníka, teda či daný internetový portál alebo herný server dosahuje dostatočné zisky, aby ich odstavením bola spôsobená dostatočná škoda. *Zotrvačnosťou* sa myslí odpor, ktorý kladie infraštruktúra voči útoku rozličnými bezpečnostnými mechanizmami. Medzi priamočiare praktiky patrí udržiavanie aktualizovanému systému so zaplátanými zraniteľnosťami alebo obmedzením počtu dopytov z jednej adresy. Služby s veľkou zotrvačnosťou sú schopné ustáť väčší nápor v prípade napadnutia. *Viditeľnosť* predstavuje rozsah verejnej prístupnosti a známosti webstránky. *Prístupnosť* značí jednoduchosť v dosiahnutí vytýčených sieťových uzlov, ktoré majú predstavovať obeť, použitou taktikou útoku bez povšimnutia. Rovnako sa spája so schopnosťou nezanechať stopy na mieste činu následkom neprítomnosti mechanizmu monitorovania a detekcie narušenia.

Relatívne vysokou hodnotou, viditeľnosťou a prístupnosťou a nízkou zotrvačnosťou sa cieľ stáva exponovanejší a tým žiadanejší pre útočníka. Oproti klasickému zločinu, kedy býva nutná prítomnosť páchateľa a obete na jednom mieste, kyberzločin dovoľuje útočníkovi pôsobiť cez internet takmer od hocikiaľ a maskovať sa proti odhaleniu.

1.2.2 Psychologická predeterminácia útočníka

Pokiaľ by nejestvovali indivíduá so zámerom druhému spôsobiť ujmu vyplývajúcu z nastolených motivačných faktorov nebolo by ani potrebné sa výrazne zaoberať zvyšovaním odolnosti informačných služieb, či brániť voči kriminalite ako takej. Určité vzorce ľudského správania naznačujú, že je prakticky nemožné sa pred týmito spoločenskými javmi vyhraniť.

Teória diferenciálnej asociácie tvrdí, že v spoločnosti existujú paralelne tak prosociálne, ako aj asociálne normy, postoje a spôsoby správania [6], čím sa vysvetľujú trestné činy dostatočne zabezpečených jedincov strednej vrstvy. Dochádza u nich k stotožňovaniu sa s antisociálnymi prístupmi na ceste za osobným úspechom. Zároveň páchatelia podvedome zľahčujú následky sociálneho zlyhania v spojitosti s stanovenými antikriminálnymi normami. V snahu neutralizovať svoje konanie popierajú zodpovednosti skrývaním sa za bezvýchodiskovosť situácie, neuznávajú význam obete prenášajúc naň vinu a ohraďujú sa konaním v záujme vyššieho princípu. Na základe teórie etiketovania je delikvencia len momentálny stav osobnosti, ktorý pramálo súvisí s psychickými vlastnosťami a správaním.

Predpokladom na spáchanie kybernetického zločinu sú okrem úvodných pohnútok aj požadované technické zručnosti útočníka. Ak pracuje človek radšej sám na seba, a hoc aj nedopatrením má záznam v registri trestov, je niekedy motivovaný vlastnou otáznejšou minulosťou voči bežným klientom sa uchýliť k predávaniu alebo prenajímaniu kompromitovaných strojov. Hacker si časom vybuduje ilegálny biznis, ktorým si dokáže zarobiť

uspokojivý obnos [7]. Využívané programy sú vymieňané alebo predávané sprostredkovane cez fóra, sú reklamované a tamojšími administrátormi overované podobne ako bežný komerčný softvér. Na fórach prebiehajú diskusie a objavujú sa návody k schodným spôsobom ako si dostupný malvér sprevádzkovať a upraviť podľa potreby.

Vykonanie útoku so zámerom obmedziť dostupnosť cudzieho systému je podľa platnej legislatívnej úpravy na Slovensku trestným činom podľa *§247a Trestného zákona* a obdoby sú zavedené takisto v iných právnych poriadkoch (napr. v USA - Computer Fraud and Abuse Act a 18 U.S.C. § 1030, v Nemecku - Strafgesetzbuch: §303b Computersabotage). Činnosť neoprávneného zásahu do počítačového systému spadá v podstate do rovnakej oblasti ako poškodenie cudzieho majetku s trestami odňatia slobody pri preukázaní, od šiestich mesiacov až po 10 rokov podľa závažnosti [8].

2 Anatómia útokov Denial of Service

Internet predstavuje prostredie sprístupňujúce na jednej strane ohromnú kvantitu služieb, ale zároveň sprostredkúva útočníkom širokú paletu nástrojov umožňujúcich ich odstavenie. Útoky odoprenia služby - *Denial of Service (DoS)* - spôsobujú nežiaduci zásah do schopnosti legitímneho používateľa na prístup k zdrojom dostupných v počítačovej sieti. Zneprístupnenie sa realizuje vyčerpaním šírky pásma linky alebo systémových prostriedkov obete, či už CPU, operačnej pamäti alebo priepustnosti vstupno-výstupných operácií [4]. Pokiaľ sa na útoku podieľa značný počet zariadení označuje sa ako distribuovaný DoS skrátene DDoS.

Útoky typu DDoS sú na internete obrovským problémom, napriek snahe vyvíjať neustále lepšie metódy obrany v reakcii na stále sofistikovanejšie modifikácie techník útokov. V architektúre internetu prevládajú prvky so zameraním skôr na efektivitu a spoľahlivosť prenosu paketov medzi koncovými uzlami, než na silné zabezpečenie detailnou kontrolou prenášaného toku. Z distribuovanej povahy a autonómie administrácie nezávislých samostatných sietí, z ktorých je Internet zložený, a ktoré si riadi do veľkej miery každý poskytovateľ pripojenia zvlášť, by bola na systematické celoplošné politiky nevyhnutná ťažko dosiahnuteľná širšia dohoda. Zároveň platí, že akokoľvek je cieľový systém ochránený stále závisí od úrovne zabezpečenia ostatných uzlov v sieti. Okrem rôznorodého vynútenia pravidiel sú ďalšími predpokladmi na degradáciu služieb obmedzené výpočtové zdroje každej entity na trase, hlavne limitované kapacity vyrovnávacích pamätí.

Myšlienka uskutočnenia DoS útoku je pomerne priamočiara. Pokiaľ útočník disponuje väčšou celkovou rýchlosťou pripojenia, je schopný preťažiť linku obete a tým spomaliť spracovanie oprávnených požiadaviek. Prenesene sa uplatňuje zásada, že silnejší pri súboji vyhráva. Strana disponujúca lepším pripojením spravidla predurčí stav dostupnosti služby v kritických momentoch. Spustenie útoku iba z jedného akokoľvek výkonného stroja je pre

útočníkov nevýhodné, pretože zmarenie zlovoľnej činnosti spočíva jednoducho vo vyčítaní adresy pôvodcu odchytávaním premávky, jeho následne zablokovanie a pridanie na čierne zoznamy.

2.1 Botnet

Centralizované prevedenie útoku odoprenia služby je z dnešného pohľadu nepriechodné. Najčastejšie sa preto na znefunkčnenie služby uplatňuje taktika ovládnutia rozsiahlej skupiny zraniteľných počítačov, ktoré dokáže útočník ovládať na diaľku a nasadiť do želanej ofenzívy. Napadnuté zariadenia ani ich užívatelia častokrát netušia, že sa stali súčasťou takéhoto zoskupenia, ktoré sa označuje ako *botnet*. Počítač slúžiaci útočníkovi na naplnenie nekalých úmyslov vzdialeným vykonaných povelov je tzv. *bot*, *zombie*, či *dron*. Boti sa správajú ako hybrid viacerých kybernetických hrozieb s pridanou hodnotou komunikačného kanála so schopnosťou koordinácie cez ovládacie miesta. Šíria sa podobne červom, skrývajú sa pred detekciou ako vírusy a obsahujú útočne metódy toolkitov [9]. Vlastník armády botov je tzv. *botmaster*. Neprávom nadobudnuté výpočtové prostriedky riadi prostredníctvom command and control (C&C) infraštruktúry.

2.1.1 Komunikácia s botmi

DDoS útočné siete používajú spravidla tri typy architektúry: *Agent-Handler*, *Internet Relay Chat (IRC)* a *webovú architektúru* [4] [10]. Model Agent-Handler pozostáva z klientov - útočníkov, ktorí sa pripájajú na tzv. handler so zaneseným softvérovým vybavení na zisťovanie stavu a koordináciu agentov. Umiestňuje sa spravidla do zariadení s veľkým objemom sieťovej premávky a ich strategický výber umožňuje výrazne kamuflovať podozrivú komunikáciu. Terminológia *handler* a *agent* sa zvykne zamieňať s *master* a *démon*.

Medzičlánkom preposielania povelov od botmasterov sa rovnako môže stať verejný IRC server. Vtedy sa jedná o architektúru založenú na protokole Internet Relay Chat. Pôvodný účel využitia botov spočíval pri asistencii moderovania rušných četových miestností IRC kanálov [9]. Jedným z prvých bol bot Eggdrop napísaný už v roku 1993. V tom čase začali vznikať boti so zámerom útočiť na ostatných používateľov a IRC servery. Dovoľovali útočníkovi ukrytie sa za aktivity bota alebo dokonca za botov na viacerých počítačoch, ktorý neboli k útočníkom priamo vystopovateľný. Tým bolo umožnené napádať čím ďalej väčšie ciele.

Agenti sa po pridaní k botnetu ohlásia na dezignovaný IRC kanál a ďalej prijímajú a posielajú správy cezeň. Tento spôsob je lákavý pre jednoduchosť komunikácie v podobe krátkych textových správ príkazov a dostatočnú anonymitu bez silnej autentifikácie. Útočník nemusí udržiavať zoznam dostupných agentov, keďže po prihlásení sa na server vie zobraziť všetkých podriadených botov. Pre zložitejšie odhalenie napomáha využitie známych portov pre IRC (6667/TCP), pomerne veľká prevádzka na známych IRC serve-

roch a technika „preskakovania medzi kanálmi“ (channel hoping), kedy botmaster využíva zvolený IRC kanál iba na krátke obdobie.

Najpoužívanejším modelom je síce pre svoju flexibilitu IRC forma komunikácie, ale v posledných rokoch sa objavujú botnety založené na webových aplikáciach. Boty posielajú webovému serveru pravidelne informácie o svojom stave. Ovládané sú cez komplexné PHP skripty a komunikácia s agentami dokáže byť šifrovaná cez TLS a skrývať sa za bežnú webovú prevádzku na portoch 80/TCP, 443/TCP a tým odolávať tým bežným sieťovým filtrom. Narozdiel od IRC spočíva ich nesporná výhoda v nemožnosti únosu botnetu od svojho pôvodného tvorcu únosom četovej miestnosti.

2.1.2 Šírenie replikáciou škodlivého kódu

Nehľadiac na výber spôsobu komunikácie agentov so svojim command and control uzlom, musí ich sieť byť dostatočne rozsiahla na to, aby spôsobila znateľnejší dopadu na webové služby. Zároveň by mala disponovať metódami vlastnej replikácie sa na priľahlé napadnuteľné počítače. Priebeh rozširovania vplyvu botnetu nad zväčšujúcou sa skupinou hostiteľov sa odohráva v postupných fázach.

V prvom rade musí dôjsť k objaveniu zraniteľných hostov, potenciálnych budúcich botov. Útočník si môže vytipovať vhodnú známu obeť a pokúsiť sa o prevzatie kontroly manuálne, systematickým skúšaním prelomenia známych zraniteľností konkrétneho systému. Automatizované skripty, ktoré sú umiestňované do už nakazených počítačov sa nepotrebujú vopred špecificky zacieliť, ale dokážu si poskladať zoznam IP adries, ktoré bude postupne navštevovať a preverovať preddefinované nezaplátané bezpečnostné diery.

Skenovanie môže prebiehať **náhodne** [11], kedy každý kompromitovaný uzol v sieti generuje postupnosť ľubovoľných IP adries. Technika je použiteľná iba pri IPv4, pretože pri hustote rozloženia obsadených IPv6 adries by bol tento postup výrazne neefektívny. Náhodné skúšanie hostov vytvára veľký objem podozrivej sieťovej premávky smerovanej akiste medzi vzdialenými sieťami, ktoré normálne nekomunikujú, čím sa zvyšuje šanca na odhalenie takejto aktivity. Keďže nedochádza pri skenovaní k synchronizácii medzi infikovanými počítačmi rastie množstvo duplicitných dopytov na rovnaký už preverený koncový uzol s ich zväčšujúcim sa počtom.

Obdržaním zoznamu počítačov (**hitlist**) s ľahko prelomiteľnou obranou dokáže botnet usmerniť svoje šírenie. Známym vyhľadávačom verejných adries IoT zariadení s konektivitou k Internetu a prehľadom známych bezpečnostných dier je `shodan.io`. Kolíziam sondovania sa zabraňuje prerozdelením celého hitlistu na menšie časti, čím sa zabezpečí, že každý agent overí stroje z presne určeného rozsahu. Nevýhoda spočíva v nutnom zostavení celého zoznamu predtým než dôjde k samotnému rozširovaniu útočnej siete. Dôležité je zvolenie vhodnej veľkosti jeho dielov na preposielanie. Ak je zoznam rozsiahly tvorí sa značná sieťová premávka, krátky zoznam zapríčiní malú finálnu populáciu agentov.

Topologické skenovanie nasleduje prirodzene vznikajúce komunikácie objavujúce v sieti, aby sa dosiahlo presnejšie splynutie s bežným tokom paketov. Nakazený hostiteľ v podobe webového servera pošle do prehliadača klientov škodlivý kód a za správnych okolností sa ten dokáže dostať na iné webové servere, ktoré klient prezerá. Spoliehaním sa na správanie používateľov sa výrazne znižuje rýchlosť a úplnosť ovládnutia vyhovujúcich obetí a útočník nedokáže šírenie počítačového červa regulovať.

Predošlé varianty skenovania je užitočné upraviť na prehľadávanie cieľov **v lokálnej podsieti**, čím sa dajú nakaziť náchylné počítače za firewallom a agent pritom neprezrádza svoju lokáciu využívaním nadmernej intersieťovej výmeny správ.

Nachádzanie vektorov prieniku počas prechádzania zoznamom adries je uskutočňované, buď horizontálne, napríklad preverovaním rovnakého otvoreného portu, či mierenej zraniteľnosti naprieč všetkými cieľmi, alebo sa koná vertikálne a síce testovaním širokého spektra malvérom pribalených utilít snažiac sa vniknúť dnu hocako. S vykonávaním vybranej metódy pomalým tempom má útočník príležitosť zostať nebadaný po dlhšiu dobu a ponúka sa mu čas na preverenie možností získania kontroly nad systémom. Po fázach náboru a vykoristení nového bota sa naň prenáša škodlivý kód pochádzajúci z centrálneho úložiska (červ 1i0n) alebo sa siahne zo zariadenia, ktoré bol pôvodcom nákazy v predošlom kroku, tzv. „back-chaining“ (červy Morris, Ramen) [11].

2.2 Klasifikácia typov DDoS útokov

Útok odoprenia služby závisí od schopnosti čo najväčšej alebo špeciálne zameranej sieťovej premávky, aby informačná služba neakceptovala požiadavky legitímnych žiadateľov. Odohráva sa privlastnenie si celej vyhradenej linky alebo výpočtového výkonu prevádzkovateľa útočníkom. Ak je vyústením prevalcovanie serverovej infraštruktúry obete nedokonalosťou zabezpečovacích mechanizmov dochádza k zrušeniu dostupnosti s následkami už uvedenými. Aby sme porozumeli metódam efektívnej obrany je nevyhnutné zatriediť a kategorizovať objavujúce sa hrozby, s ktorými sa ciele DDoS útoku vedia stretnúť. V literatúre existujú rozličné taxonómie separujúce problematiku z rôznych uhlov pohľadu [4] [10] [11] [12].

Základné rozdelenie DDoS útokov spočíva v identifikácii ich primárneho vektora. Webová aplikácia býva zneprístupnená, buď vyčerpaním šírky prenosového pásma hrubou silou záplavy paketov, alebo vyplytvaním systémových prostriedkov sémantickým útokom na komunikačný protokol.

Volumetrické útoky (veľkoobjemové útoky) saturujú kapacitu linky rozmanitou plejádou nálože. Spoločným menovateľom je technika flooding (záplava). Populárnou formou útoku je posielanie UDP datagramov na náhodné porty s úmyslom zapríčiniť overovanie, či sú porty otvorené a spôsobiť reakciu servera signalizačnými správami ICMP Destination port unreachable. So snahou donútiť systém, aby sa venoval predovšetkým záškodníckym

spávam útočníka, pracuje tiež záplava paketmi ICMP Echo Request (Ping) s následnou odpoveďou ICMP Echo Reply. Na VoIP služby je účinným SIP Flood, ktorý zaplaví SIP proxy s falošnými správami pre začatie hovoru SIP INVITE [10]. Webový aplikačný server je možné zahltiť záplavou HTTP(S) požiadaviek GET alebo POST na náhodné alebo existujúce URI webstránky. Dopytovanie sa na neexistujúcu cestu okamžite vráti stavový kód rádu 400 pre chybu klienta, ale rovnako sa bude server musieť zaoberať spracovaním takejto požiadavky, len sa stáva jednoduchšie pozorovateľnou z prístupových logov.

Protokolové útoky sa prižívujú na zraniteľnosti v návrhu komunikačného protokolu na transportnej až aplikačnej vrstve OSI, ktoré spoliehajú na priebežné ukladanie stavových informácii o naviazaných reláciach. Rozšírené sú taktiky na zneužitie časovače stavového automatu protokolu TCP a príznakov v TCP segmentoch, ktorými sa odosielateľ a prijímateľ dohadujú na priebehu výmeny správ.

Počas TCP SYN Flood je doručené také množstvo podnetov na otvorenie spojenia segmentami s príznakom SYN, ktoré vyústi v zaplnenie pamäte vyhradenie na uchovávanie aktívnych relácii. Server je povinný pri zahajovaní TCP spojenia cez 3-way handshake a obdržaní SYN odoslať SYN+ACK a počkať stanovenú dobu. Timeout býva dostatočný na to, aby dokázal útočník ponechať tabuľku relácií zaplnenú iba svojimi podvratnými požiadavkami. Schodnou ochranou je zavedenie tzv. TCP Cookie. Systém po prijatí TCP SYN odošle TCP SYN+ACK a nevytvorí v pamäti žiadnu reláciu [12]. Po prijatí právoplatnej odpovede TCP ACK sa spätne dopočíta TCP sekvencia paketov a až vtedy sa zaháji spojenie.

TCP RST útok sa zameriava na rušenie nadviazaných spojení medzi serverom a klientmi, kedy však je nutné poznať zdrojovú IP adresu klienta, pretože útočník háda začiatkom konverzácie náhodne započaté sekvenčné čísla. Ak uspeje preberie reláciu a zruší ju. Za bežných okolností je také niečo ťažko spáchateľné, lebo TCP spojenia zvyknú mať krátke trvanie a vznikajú ad-hoc.

Nastavením príznaku PSH je serveru nanútené okamžité vyprázdnenie vyrovnávanie pamäte klientovi a odoslanie potvrdzujúcej správy ACK. Pri enormnej hromade takýchto výziev nebude schopný server vybavovať ďalšie požiadavky, čím dôjde k zrušeniu dostupnosti webových a podobných služieb poskytovaných z daného bodu.

Zlomyseľným zásahom do riadenia toku TCP spojenia predchádzajúceho zahlteniam, presnejšie vyžiadaním a udržiavaním nulovej veľkosti okna príjemcu, sa vie útočník obsadiť všetky dostupné spojenia v tabuľke spojení a tým znemožniť nadviazanie komunikácie so serverom ostatným. Určenie veľmi malej nenulovej veľkosti okna spôsobí rozdrobenie odpovedí na veľmi malé fragmenty. Prevenciou býva zapojenie Nagelovho algoritmu (RFC 896) do TCP implementácie, ktorého úlohou je zamedziť veľkej réžii pri posielaní miniatúrneho payloadu.

Na relačnej vrstve modelu OSI je vďačným protokolom na útoky spotrebujúce značný výpočtový výkon Secure Sockets Layer (SSL/TLS). Keďže majorita webových aplikácií

v súčastnosti používa HTTPS je dôležité si uvedomiť, že proces šifrovania spolu s réžiou pri výmene kľúčov v SSL handshaku predstavuje pre server násobnú náročnosť oproti klientovi. Znovu sa naskýta prostý útok záplavou iniciácií TLS spojenia alebo opätovné dohodnutie SSL komunikácie (renegotiation), ktorá zvykne zahŕňať zmenu parametrov šifrovania alebo vyžiadanie certifikátu servera. Riešením je blokovanie takýchto požiadaviek alebo „SSL offloading“ do špecializovaného hardvéru [12].

Spotrebovanie všetkých ponúkaných spojení HTTP protokolu aplikáciou webového servera sú preferované tzv. „low and slow“ útokmi. Agent sa maskuje akoby za veľmi pomalú rýchlosť pripojenia, no v skutočnosti zámerne rozdrobuje svoj dopyt na krátke fragmenty a posiela ich s významným oneskorením, aby držal spojenie otvorené čo najdlhšie. Dôvody existujúcich obmedzení tkvejú v maximálnom počte súborových deskriptorov procesu alebo únosnej hladine bežiacich procesov. Zástupcom tejto skupiny útokov je Slowloris a R-U-Dead-Yet? (RUDY). Z dôvodu malej generovanej premávky, prechádza pomerne ľahko bez povšimnutia, pretože nemá dopad na iné atribúty systému.

Rovnako ako botnety slúžia na zitenzívnenie devastačného prúdu paketov, tak môžu nepriamo zapájať do útoku aj malvérom nenakazené počítače technikami odrazu a zosilnenia. Útoky s odrazom (RDoS a DRDoS) zapríčiňujú poslanie paketov s podvrhnutou zdrojovou adresou cieľa útoku záchytným bodom (pivotom). V domnienke správnosti pôvodcu správy sa odpoveď doručí v konečnom dôsledku na obeť. Samo o sebe to nemá až taký význam, okrem odklonenia nevyhnutného prúdu odpovedí od skutočných spúšťačov požiadaviek na tretie strany. Nastavením cieľovej IP adresy na broadcastovú adresu lokálnej podsiete (L2 alebo L3 OSI) sa útok zosilní, pričom zasiahne všetky počítače v spoločnom broadcastovom segmente siete. Tiež je priechodné odrazenie útoku od viacerých reflektorov. Na týchto princípoch fungujú útoky Smurf a Fraggle.

Domain Name System (DNS) amplifikačné útoky využívajú podstatu odrazeného útoku, ale obsahujú obohatenie zaručeného nárastu veľkosti DNS odpovede voči dopytu. Faktor zväčšenia DNS query response sa pohybuje od 1,1 pri jednom A zázname (example.net), 2,75 v prípade troch AAAA záznamoch (youtube.com) alebo dokonca 3,5 (yahoo.com) pri siedmich štvor-áčkových záznamoch. Protokol DNSSEC ponúka cez otvorené rekurzívne resolvery až 30-násobnú amplifikáciu [12], z dôvodu početnosti vrátených NS záznamov a digitálnych podpisov v DS a RRSIG záznamoch. Uvedenými príkazmi sme objavili odpoveď s 24-násobným zväčšením a podobne iné domény dosahovali amplifikácie bežne v rozsahu 13 - 18-krát:

```
dig +dnssec +trace opendns.com
dig +dnssec @b.root-servers.net opendns.com
```

Každoročný prehľad v trendoch kybernetických hrozieb publikovaných európskou inštitúciou ENISA konštatuje, že takmer 80% všetkých DDoS útokov v treťom kvartáli 2019 boli TCP SYN záplavy [13], stávajúc sa najpopulárnejším typom útoku spolu s DNS od-

razenou amplifikáciou. V apríli 2019 bol zaznamenaný SYN Flood útok s prietokom až 580 miliónov paketov za sekundu. Vyskytujú sa hlavne multivektorové útoky, čím je ich zdolanie komplexnejšie. Zároveň dominovali útoky kratšie ako 10 minút, ktorých bolo 84% zo zaznamenaných. Celkovo došlo k nárastu v počte nahlásených útokov o 241% oproti rovnakom období predošlého roku. [13].

2.3 Ochrana spevnením sieťovej ochrany

Obranné mechanizmy na úspešné zvládnutie útokov odoprenia služby rozlišujeme primárne podľa úrovne pripravenosti reakcie na **proaktívne** a **reaktívne** stratégie [11].

Prevenciou sa zabezpečuje systém proti prieniku priebežným monitorovaním a pravidelným sťahovaním a inštaláciou bezpečnostných záplat. Súčasne sa budujú bezpečnostné politiky organizácie, ktoré rátajú s klasickými vektormi DDoS útokov a premýšľa sa nad adekvátnym minimalizovaním dopadu hrozieb s nimi spojenými. Prakticky osvedčenými riešeniami sú „resource accounting“, čiže účtovanie a limitovanie počtu vyhradených spojení pre každú IP adresu pristupujúcu k službe pri maximálnej frekvencii odpovedí od servera, alebo sa využíva „resource muliplication“, kedy sú zdroje systému duplikované na viaceré zariadenia a prichádzajúca záťaže je vyvažovaná medzi nimi.

Reaktívne spôsoby sa usilujú o zmiernenie útoku počas jeho konania. Pre automatizované riadenie defenzívy by mali byť obranné prvky schopné detekcie útoku odlíšením od typickej sieťovej premávky s ohľadom na odstránenie vlastnej chybovosti pri identifikácii falošných poplachov a prehliadnutí škodlivých činností. Zisťovanie prítomnosti pokusov na odoprenie služieb sa sleduje rozpoznávaním podozrivých vzorov podľa analýzy predošlých útokov alebo pozorovanie anomálií v komunikácií. Strážny komponent je sústredený na preddefinované scenáre alebo sa natrénuje na bežnej premávke a spustí varovanie po prekročení prahových hodnôt.

2.3.1 Remotely Triggered Black Hole

Nežiadúcu premávku je vhodné zlikvidovať na okraji autonómneho systému vzdialene spustiteľnou čiernou dierou (RTBH). Nahlásením zdrojov alebo častejšie cieľa útoku vie správca siete vložiť do spúšťacieho smerovača statickú cestu na virtuálne rozhranie *Null*, ktoré spôsobí zahodenie paketov [14]. Postihnutá služba bude síce odrezaná od Internetu, čiže útočník odoprie dostupnosť pre ostatných používateľov, ale zmiernia sa negatívne dopady na zvyšnú infraštruktúru. Na edge routeroch sa musí vopred nakonfigurovať statická cesta pre ďalší skok stanovený za black hole a zakázať preň odpovedanie o nedosiahnutí cieľa cez ICMP:

```
ip route 192.0.2.1 255.255.255.255 Null0
interface Null0: no ip unreachables
```

Na trigger smerovači sa musí aktivovať BGP politika na presmerovanie cesty k obeti po obdržaní vloženej statickej cesty so spúšťacou značkou (tagom) a povolenie jej distribúcie medzi iBGP peerov a zároveň nepropagovanie mimo autonómneho systému [15]:

```
route-map blackhole
    match tag 66
    set ip next-hop 192.0.2.1
    set origin igp
    set community no-export
router bgp 65535
    redistribute static route-map blackhole
```

Poskytovateľ pripojenia môže na podnet zablokovať premávku na obeť známej IP adresy cez pravidlo na spúšťacom routeri. Uvedená statická cesta musí byť po skončení odobratá zo spúšťacieho routera, ktorý rozpošle BGP route withdrawal svojim iBGP peerom:

```
ip route 172.5.23.1 255.255.255.255 Null0 tag 66
```

2.3.2 IP Spoofing

Na zabránenie spätnej väzby paketov vracajúcej sa naspäť botom útočníka, zvykne pri záplavových útokoch vložiť do paketu sfalšovanú zdrojová adresa. Vychádza z potreby útočníka zostať v anonymite a prípadne zmiasť bezpečnostnú obranu vzbudením dojmu, že nadmerná premávka pochádza od rozptýlených podnecovateľov. Spoofing sa hodí u reflektorových útokov alebo v situáciach, kedy si útočník praje, aby bol vinený za iniciátora určitý počítač. Prostredníctvom zapojenia Ingress a Egress filtrovania sú poskytovatelia pripojenia na trase schopný zabrzdiť škodlivú premávky takmer v zárodku. Ukradnuté zdrojové adresy sa pre navodenie dôveryhodnosti vyberajú z rozsahu verejne smerovateľných IP adries (RFC 1918, RFC 3330). Privátne adresy sa totiž štandardne zahadzujú na smerovačoch registrovaním príslušných ACL pravidiel ako realizácia **Egress filtrovania** Skrátená ukážka odhodí pakety z jedného súkromného rozsahov a prepustí ostatné [16]:

```
access-list 110 deny ip 192.168.0.0 0.0.255.255 any
access-list 110 permit ip any any
```

Podvrhnuté IP adresy sa rozlišujú podľa techniky ich selekcie. Najmenej sofistikovaný postup spočíva vo vygenerovaní náhodného 32-bitového čísla, ktoré bude predstavovať spoofovanú zdrojovú adresu. Vymyslená IP adresa nesmie byť úplne svojvoľná, ale pochádzať z platných podsieti, pre ktoré router má záznam v smerovacej tabuľke alebo sa môže nachádzať niekde po ceste k obeti. **Ingress filtrovanie** (RFC 2827) totiž povoľuje v striktnom režime smerovať iba pakety, pre ktoré existuje vo FIB (Forwarding information base) tabuľke mapovanie reverznej cesty (Reverse Path Forwarding) cez rovnaké rozhranie. Ak nedokáže byť zabezpečené symetrické smerovanie potom „loose“ mód dovoľuje akceptovať spätnú cestu cez ľubovoľné rozhranie smerovača:

```
ip verify unicast reverse-path list            # Strict mode
ip verify unicast source reachable-via any     # Loose mode
```

Naproti bežnej predstavy prevláda snaha o zúžitkovanie validných zdrojových adries botov, tam kde je to uskutočniteľné [11].

2.3.3 Rate limiting a Firewall na Linuxe

Zníženie záťaže na systémové zdroje pridelené na tvorbu paketov odpovede je dosiahnuteľné cez *rate limiting*. Linux oplýva vrámci *procfs* (`/proc/sys/net/ipv4/`) premennými na obmedzenie rýchlosti pri odpovedaní maskou zvolenými ICMP správami `icmp_ratelimit`, `icmp_ratemask` a `icmp_echo_ignore_broadcasts`[1]. Takisto sa odporúča ponechať pravidlá firewallu na najmenšej úrovni priepustnosti smerom dnu za každých okolností ako whitelist. Ak napríklad webová aplikácia beží na porte HTTPS/443 a administrácia vstupuje na server cez SSH/22 zo stáleho rozsahu adries intranetu, môže jednoduchá konfigurácia firewallu vyzerá nasledovne [17]:

```
# Pravidlá pre povolenie prichádzajúcej komunikácie cez SSH, HTTPS
iptables --append INPUT --protocol tcp --dport 443 --jump ACCEPT
iptables --append INPUT --protocol tcp --dport 22 \
         --source 192.168.0.0/24 --jump ACCEPT
# Politiky reťazí pravidiel sú zahodiť všetko okrem odchádzajúcich paketov
iptables --policy INPUT DROP
iptables --policy FORWARD DROP
iptables --policy OUTPUT ACCEPT
```

3 Škálovanie webových aplikácií

Rozvrhnutie architektúry na nasadenie webovej aplikácie, ktorá ustojí legitímnu premávku, ale tiež škodlivo nadmerný prúd vykonštruovaných požiadaviek, vyžaduje balans medzi sústredením prostriedkov na jednom mieste do homogénneho monolitu a ich parcelovaním znásobením počtu inštancií. Návrhár musí nájsť kompromis medzi obstaraním silnejšieho počítača alebo viacerých počítačov. Priklonenie sa k vybranej alternatíve predurčí budúce komplikácie pre rozšírenie platformy hosťujúcej webovú službu.

Vertikálne škálovanie spočíva v navyšovaní výpočtovej sily stroja pridaním, urýchlením alebo zväčšením kapacity hardvérových komponentov - procesora, pamätí, diskového poľa, či sieťovej karty - utesňujúc vzájomnú väzbu dielcov a očakávajúc ich nevyhnutnú vzájomnú kompatibilitu. Uľahčuje sa tým údržba a kontrola systému, zachovanie konzistentnosti dát bez nutnosti navyše častí na zabezpečenie ich integrity. Zároveň sa to

[1] `https://man7.org/linux/man-pages/man7/icmp.7.html`

odráža na nižšej energetickej spotrebe voči mnohým spoločne rovnako výkonným duplikátom. Všetko za cenu o poznanie vyšších obstarávacích nákladov tohto kompaktného balíka s výhľadom na obmedzení rozsah pre upgrade, vytvárajúc jediný bod zlyhania pre prípadný celkový výpadok poskytovanej aplikácie.

Horizontálne škálovanie je naproti vertikálnemu oveľa odolnejšie proti zlyhaniu, keďže sa spolieha na viacero paralelne bežiacich zariadení. Komponenty sú lacnejšie a jednoduchšie na upgrade, pretože pádom systému servera nepostihneme dostupnosť služby ako takej. Nevyhnutnosťou je spravidla zosieťovanie a distribúcia záťaže úloh pomedzi uzly, čím sa zvyšuje oneskorenie a závislosť na externých činiteľov a od prepájacích prvkoch.

3.1 Redundancia nižších vrstiev RM OSI

Odolnosť sieťovej infraštruktúry proti poruchám a výpadkom aktívnych sieťových prvkov, s vedľajším rovnako dôležitým efektom navýšenia dátového prietoku komunikačných spojov, sa utužuje protokolmi agregácie liniek na spojovej vrstve, viaccestným smerovaním na sieťovej vrstve a virtuálnym smerovačom (IPVS) na transportnej vrstve.

Agregácia liniek sa uplatňuje pokiaľ nestačí maximálna prenosová rýchlosť samotného jedného sieťového rozhrania a je nevýhodné, či dokonca nemožné vymeniť sieťovú kartu. Vtedy sa oplatí zoskupiť niekoľko fyzických rozhraní do jedného logického linku, spôsob nazývaný ako „trunking“ alebo „bonding“. Navyšovanie prenosových rýchlostí medzi generáciami technológii na fyzickej vrstve môže takto prebiehať lineárne. Ak je vyžadovaný GigabitEthernet (1 Gbit/s), ale nie je k dispozícii, dokáže Trunked Fast Ethernet zabezpečiť rýchlosti 200 - 800 Mbit/s, v porovnaní s obyčajným 100 Mbit/s Fast Ethernet. Rámce sú striedavo posielané aktívnymi redundantnými linkami vrámci zhluku, čím sa zabezpečí zvýšená dostupnosť a symetrizácia záťaže na participujúcich portoch.

Link Aggregation Control Protocol (LACP), ktorý je súčasťou štandardu IEEE 802.3ad, dovoľuje sieťovému zariadeniu vyjednanie automatického združenia liniek výmenou LACP paketov medzi partnermi. Priebežnými keepalive správami kontroluje LACP priechodnosť spoju pre zamedzenie straty paketov v nefunkčnej linke a overuje chyby spôsobené nesprávnym fyzickým zapojením vznikajúcim prekrížením kabeláže: „loopback links“ alebo „split-trunk“. Prakticky sa na smerovači nastaví kanál do ktorého sú pridané porty[2].

```
interface port-channel 1
   ip address 192.168.0.1 255.255.255.0
interface range g2/0/0-1
   no ip address
   channel-group 1 mode active
```

[2] https://www.cisco.com/c/en/us/td/docs/ios/12_2sb/feature/guide/gigeth.html

Viaccestné smerovanie pridáva spoľahlivosť v miestach, kde vedú do cieľovej destinácie aspoň dve trasy. *Equal-cost multi-path routing* (ECMP) je stratégia smerovania, kedy sa striedavo posielajú pakety cez viaceré ďalšie skoky s rovnako dobrými metrikami cesty. V praxi je tento prístup málo uplatňovaný pre komplikácie viažuce sa k dynamickému výberu spomedzi dostupných smerov na rovnomerné rozloženie záťaže [18]. Rozličné spojenia tiež zvyknú mať rôzne veľkosti MTU (maximum transmission unit) a variabilné oneskorenia vedúce k zbytočnému preusporiadaniu paketov správy mimo poradia. Pri použití mnohých alternatívnych preskakujúcich ciest môže dochádzať k strate paketov.

Zmiernenie popísaných negatív výberu pre next-hop, ale vyžaduje udržovanie si stavu prebiehajúcich tokov alebo zvýšené výpočtové nároky pre voľbu next-hop. *Modulo-N Hash* presmeruje paket na cestu podľa identifikátora toku z hlavičky paketu (najčastejšie zdrojová a cieľová adresa) modulo počet dostupných skokov. Ak dôjde k zmene musí sa upraviť $(N-1)/N$ tokov [18]. *Hash-Threshold* rovnomerne mapuje uzly do výstupu hašovacej funkcie a podľa porovnania hašu identifikátoru toku s hranicami oblasti je zvolený next-hop. Pri zmenách sa upravuje cesta štvrtine až polovici tokov. *Highest Random Weight* počíta hash zakaždým zároveň z hlavičky paketu a kľúča pre next-hop. Zvolí sa ďalší skok s najvyšším výsledným číslom. Za väčšej časovej náročnosti sa mení pri pridaní alebo odobratí cesty, smer už len $1/N$ tokov. Cisco router využívajúci na smerovanie OSPFv2 uplatní ECMP jednoduchým nastavením[3]:

```
router ospf 1
maximum-paths 2
```

Virtuálny smerovač zamedzuje oknám v obsluhe IP adresného priestoru udržiavaním viruálnej IP adresy, pre ktorú eviduje redundantné aplikačné servery na zabezpečenie vysokej dostupnosti služby. Narozdiel od konceptu základnej dostupnosti, kedy sa vyvinie systém spĺňajúci len nevyhnutné funkčné požiadavky, prináša vysoká dostupnosť znásobenie počtu komponentov, aby v prípade údržby, chyby alebo zlyhania komponentu mohol zaujať miesto obsluhy úloh náhradný prvok.

Odstránenie jediného bodu zlyhania nadobúda dve podoby konfiguráciami *Active/Standby* a *Active/Active* [1]. Odlišujú sa v móde prevádzky záložného riešenia, ktoré je buď nečinnou kópiou (Standby) v „Cold“ alebo „Hot“ režime, preberajúce zodpovednosť za spracovanie požiadaviek, iba ak toho hlavný člen nie je schopný, alebo sa neplytvá žiadnymi zdrojmi a záloha je operatívna (Active) zároveň s primárnou infraštruktúrou. Kontinuálna dostupnosť zachádza ešte ďalej so snahou zamaskovať akékoľvek časové prestoje spojené s poruchou, či už zámernou krátkodobou vznikajúcou pri údržbe alebo vážnejšou zapríčinenou útokom.

Virtual Router Redundancy Protocol (VRRP) umožňuje súhru viacerých uzlov pôso-

[3] `https://www.techrepublic.com/article/how-to-configure-equal-cost-multi-path-in-ospf/`

biacich ako VRRP smerovače tvoriace jeden virtuálny router. Spoločne si volia zariadenie zodpovedné za zdieľanú vysoko dostupnú IP adresu. Zariadenie vrámci klastera v roli master má najvyššiu prioritu a v pravidelných intervaloch (najčastejšie 1 - 5 sekúnd) vysiela obežníkové správy *VRRP Advertisiment* na multicastovú adresu 224.0.0.18 [19], na ktorú sú prihlásené cez protokol IGMP všetky uzly v klasteri. Po inicializácií je hlavný uzol povinný oznámiť všetkým zariadeniam v LAN cez Gratious ARP správu zmena vlastníka IP adresy na svoju MAC adresu. Po výpadku, kedy nebol zachytený Advertisiment približne viac ako trojnásobok intervalu, nastupuje backup uzol s najbližšou nižšou prioritou a cez Gratious ARP prehodí IP adresu na seba. Keepalived je softvérový balík využívajúci Linux Virtual Server kernel modul umožňujúc nastaviť dynamický failover virtuálneho smerovača s VRRP [20]:

```
vrrp_instance malina {        # VRRP inštancia
    state MASTER              # štandardný stav inštancie MASTER / BACKUP
    interface eth0            # sieťové rozhranie
    virtual_router_id 1       # VRRP router id pre inštanciu
    priority 100              # priorita VRRP routera 0 - 255
    advert_int 1              # advertisement interval v sekundách
    virtual_ipaddress { 192.168.0.50 } # VRRP virtuálna IP adresa
}
```

3.2 Algoritmy vyvažovania záťaže

Load balancing" means an even distribution of the total load amongst all serving entities Load balancing is very essential in distributed computing systems to improve the quality of service by managing customer loads that are changing over time. The request demands of incoming requests are optimally distributed among available system resources to avoid resource bottlenecks as well as to fully utilize available resources [5]. Load balancing also provides horizontal scaling e.g., adding computing resources in order to address increased loads.

Styčný bod pre vybavenie žiadostí webových klientov je load balancer, ktorý sa snaží rovnomerne prerozdeliť záťaž medzi skupinu serverov, ale celý systém sa stáva náchylným na jeho zlyhanie. Zdvojením load balancera do uzlov spolupracujúcich v klasteri je omnoho jednoduchšie poskytovať vysokú dostupnosť bez obáv o náhlu indispozíciu člena zoskupenia.

3.3 Reverzné proxy na vyvažovanie záťaže

Reverzný proxy server je prostredníkom medzi klientami a veľmi sa pripodobňuje princípu obyčajného proxy servera, snažiacimi sa pristupovať k službe, ktorá býva rozložená na viacerých serveroch.

```
# Table definition
stick-table type ip size 100k expire 30s store conn_cur
# Allow clean known IPs to bypass the filter
tcp-request connection accept if { src -f /etc/haproxy/whitelist.lst }
# Shut the new connection as long as the client has already 10 opened
tcp-request connection reject if { src_conn_cur ge 10 }
tcp-request connection track-sc1 src


backend per_ip_rates
    stick-table type ip size 1m expire 10m store http_req_rate(10s)
frontend fe_mywebsite
    bind *:80
    http-request track-sc0 src table per_ip_rates if METH_POST
    http-request deny deny_status 429 if { sc_http_req_rate(0) gt 100 }
         % Limiting Request Rates
    http-request silent-drop if { sc_http_req_rate(0) gt 100 }

    timeout tarpit 5s   # abusers
    http-request tarpit if { sc_http_req_rate(0) gt 100 }

        timeout http-request 5s
        option http-buffer-request

        # Using ACLs to Block Requests
    http-request deny if HTTP_1.0
    http-request deny if { req.hdr(user-agent) -i -m sub curl }
    http-request deny if { req.hdr(user-agent) -i -m sub -f /etc/hapee
       -1.8/badagents.acl }
    http-request deny unless { req.hdr(user-agent) -m found }

        # Blacklist and Gray list
        http-request deny if { src -f /etc/hapee-1.8/blacklist.acl }
        http-request deny if { src -f /etc/hapee-1.8/greylist.acl } {
           sc_http_req_rate(0) gt 5 }

    tcp-request content reject if { sc_conn_cur(0) gt 1 } || {
       sc_conn_rate(0) gt 5 }
```

```
# Table definition
stick-table type ip size 100k expire 30s store conn_rate(3s)
# Shut the new connection as long as the client has already 10 opened
tcp-request connection reject if { src_conn_rate ge 10 }
```

```
# Table definition
f:
```

```
# Use General Purpose Couter (gpc) 0 in SC1 as a global abuse counter
# Monitors the number of request sent by an IP over a period of 10 seconds
stick-table type ip size 1m expire 10s store gpc0,http_req_rate(10s)
tcp-request connection track-sc1 src
tcp-request connection reject if { src_get_gpc0 gt 0 }
b:
# If the source IP sent 10 or more http request over the defined period,
# flag the IP as abuser on the frontend
acl abuse src_http_req_rate(ft_web) ge 10
acl flag_abuser src_inc_gpc0(ft_web)
tcp-request content reject if abuse flag_abuser
```

```
limit_req_zone $binary_remote_addr zone=one:10m rate=30r/m;
server {
    # ...
    location /login.html {
        limit_req zone=one;
    # ...
    }
}
```

```
limit_conn_zone $binary_remote_addr zone=addr:10m;
server {
    # ...
    location /store/ {
        limit_conn addr 10;
        # ...
    }
}
```

```
server {
    client_body_timeout 5s;
    client_header_timeout 5s;
    # ...
}
```

```
location / {
    deny 123.123.123.0/28;
}
```

```
location / {
    allow 192.168.1.0/24;
    deny all;
}
```

Kód 1: BIND9 nastavenie DNS zóny .home

```
zone "home" {
    type master;
    file "/etc/bind/db.home";
}
```

Kód 2: Priradenie troch DNS A záznamov pre doménu website.home

```
website  IN  A  192.168.0.2
         IN  A  192.168.0.3
         IN  A  192.168.0.4
```

Kód 3: HAProxy základná kostra nastavení

```
global
   log /dev/log local0
   chroot /var/lib/haproxy

defaults
  log global
  option httplog  # tcplog
  mode http       # tcp
```

Kód 4: NGINX základná kostra nastavení

```
worker_processes auto;
events {
   worker_connections 1024;
}
http {  # stream
  access_log /var/log/access.log;
  error_log /var/log/error.log;
}
```

Kód 5: HAProxy: preposielanie požiadavok na pool serverov za reverznou proxy

```
frontend web
    bind *:80
    option forwardfor
    default_backend webservers
```

Kód 6: NGINX: preposielanie požiadavok na pool serverov za reverznou proxy

```
server {
   listen 80;
   server_name _;
   location / {
     proxy_pass http://website;
     proxy_set_header Forwarded
           $proxy_add_forwarded;
   }
}
```

Kód 7: HAProxy: vyvažovanie záťaže najmenej spojení na trojicu HTTP serverov

```
backend webservers
    balance leastconn
    server A 192.168.0.2:80
    server B 192.168.0.3:80
    server C 192.168.0.4:80
```

Kód 8: NGINX: vyvažovanie záťaže najmenej spojení na trojicu HTTP serverov

```
proxy_http_version 1.1;
upstream website {
    least_conn;
    server 192.168.0.2:80;
    server 192.168.0.3:80;
    server 192.168.0.4:80;
}
```

4 Monitorovanie webovej aplikácie

```
<Location "/server-status">
    SetHandler server-status
    Require ip 127.0.0.1
</Location>
```

```
frontend stats
    bind *:8404
    stats enable
    stats uri /stats
    stats refresh 10s
```

```
location = /basic_status
   {
    stub_status;
    allow 127.0.0.1;
    deny all;
}
```

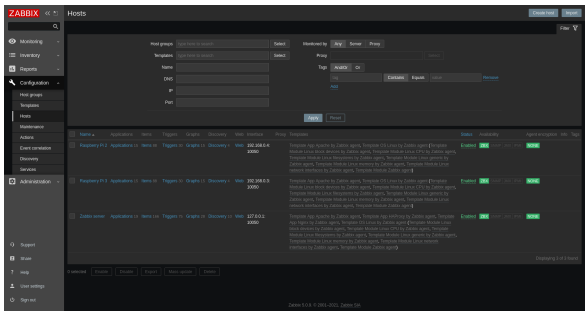
4.1 HTTP hlavičky

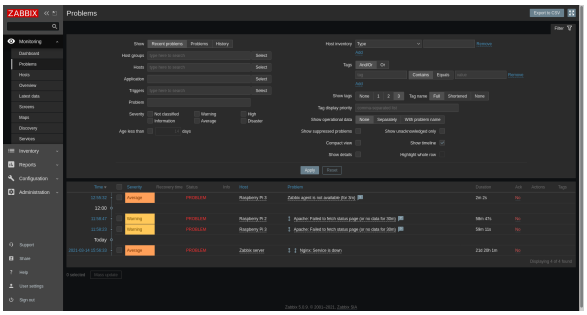
4.2 Metriky

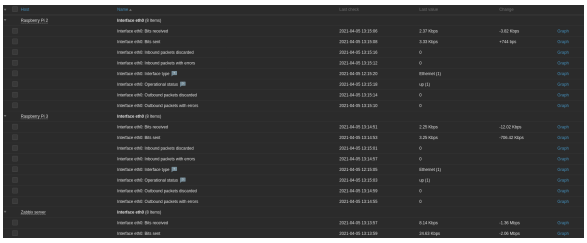
```
# Common Log Format
192.168.0.2 - - [05/Apr/2021 13:45:20 +0200] "GET␣/␣HTTP/1.1" 200 1254

# Apache Custom Log
website.home:80 192.168.0.2 - - [05/Apr/2021 13:45:20.814 +0200] "GET␣/␣
   HTTP/1.1" 200 351 1254 "-"
"Mozilla/5.0"

LogFormat "%v:%p␣%h␣%l␣%u␣%{%d/%b/%Y␣%T}t.%{msec_frac}t␣%{%z}t␣%D␣\"%r\"
   ␣%>s␣%I␣%O␣\"%{Referer}i\"␣\"%{User-Agent}i\"" custom
<VirtualHost *:80>
   DocumentRoot /var/www/html
   CustomLog ${APACHE_LOG_DIR}/access.log custom
</VirtualHost>


# Nginx Custom TCP Log
192.168.0.5 [05/Apr/2021:16:35:30 +0200] TCP 200 465 308 20.960

# Nginx Custom HTTP Log
192.168.0.5 05/Apr/2021:13:32:15 +0200 "GET␣/␣HTTP/1.1" 200
   192.168.0.3:80 1294 0.000 0.040 0.040

log_format fmt '$remote_addr␣$time_local␣"$request"␣$status␣
   $upstream_addr␣$upstream_bytes_received␣$upstream_connect_time␣
   $upstream_header_time␣$upstream_response_time';

log_format fmt '$remote_addr␣[$time_local]␣$protocol␣$status␣$bytes_sent
   ␣$bytes_received␣$session_time';
```

```
# HAProxy Log
haproxy[2550]: 192.168.0.5:45152 [28/Mar/2021:18:07:58.798] web
    webservers/A 0/0/5/23/29 200 2862 - - ---- 9/9/8/8/0 0/0 "GET␣/␣HTTP
    /1.1"
```

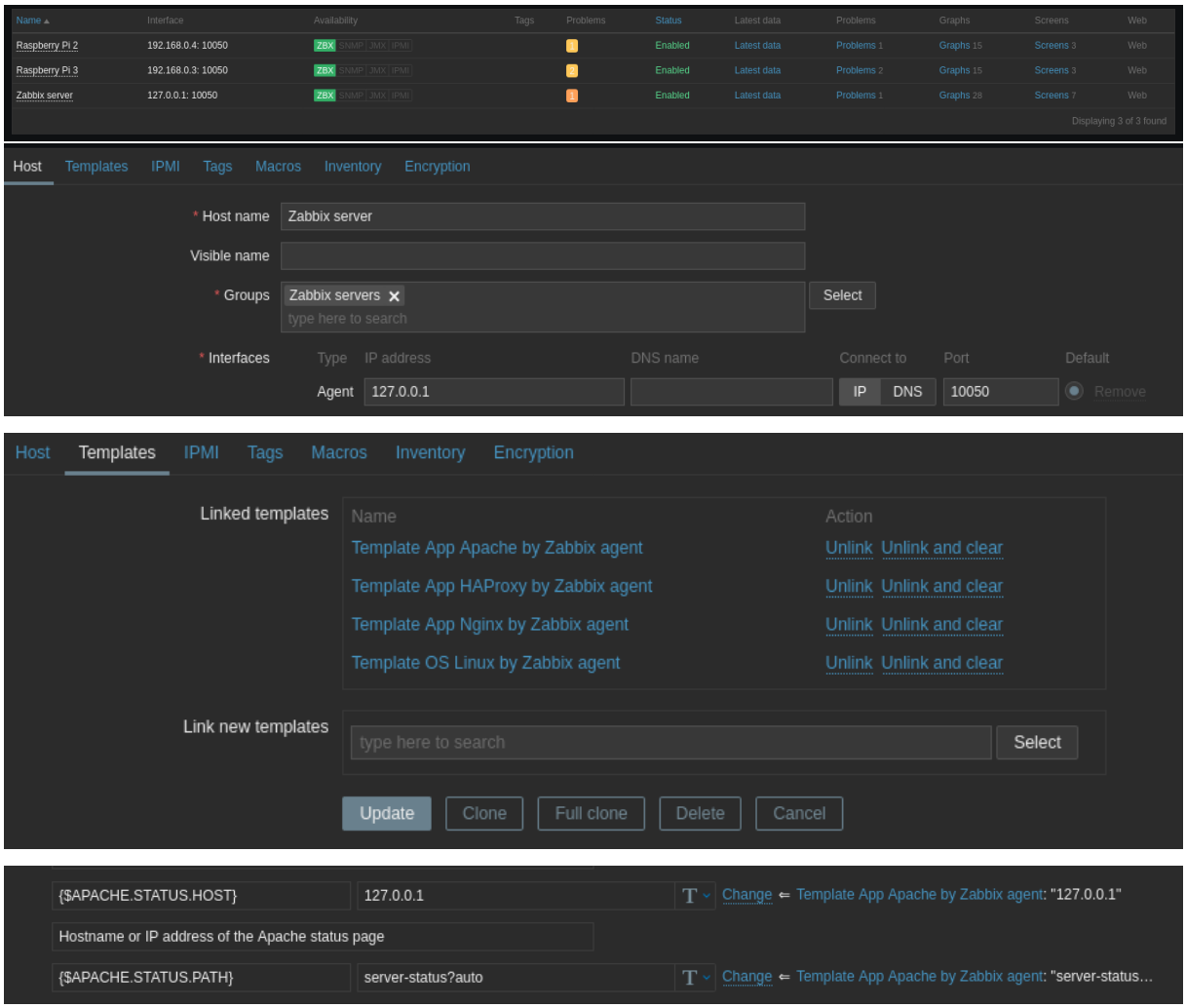
4.3 Zabbix







```
<IfModule security2_module>
    SecRuleEngine on
    ServerTokens Min
    SecServerSignature "PIB␣FIIT␣STU"
</IfModule>
```



5 Simulácie útokov a záťažové testy

5.1 Sieť na experimenty

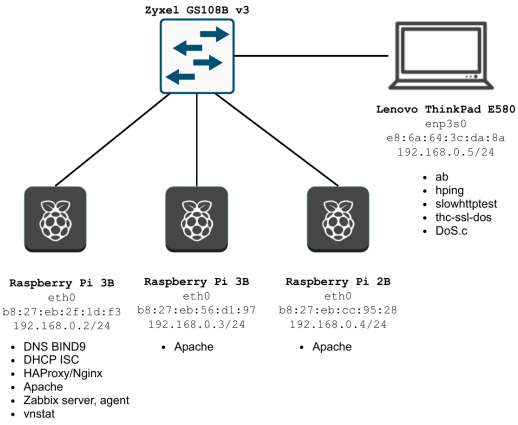
5.2 HPING na DoS útoky záplavou

```
hping 192.168.0.2 --udp --flood
hping 192.168.0.2 --icmp --flood
hping 192.168.0.2 --syn --flood --destport 80
vnstat --traffic
```

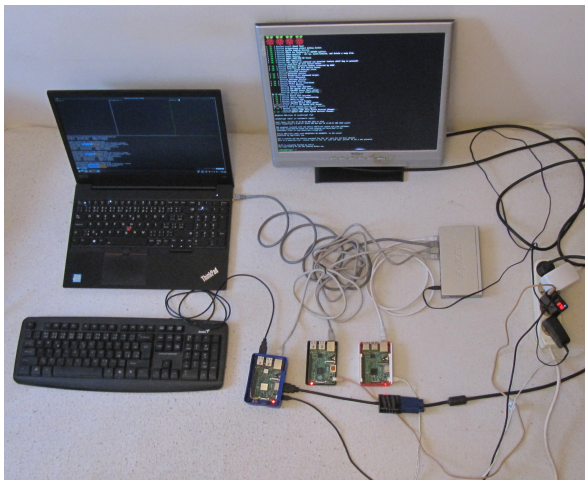
Na náhodné porty 93 paketov/s, nevráti žiadne pakety pri spoofed ip (nevie cez arp nájsť cieľ)

5.3 Analýza prevedenia DoS útoku záplavou

```
int s = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
strcpy(ifr.ifr_name, "enp3s0");
ioctl(s, SIOCGIFINDEX, &ifr);
```



(a) Logická topológia

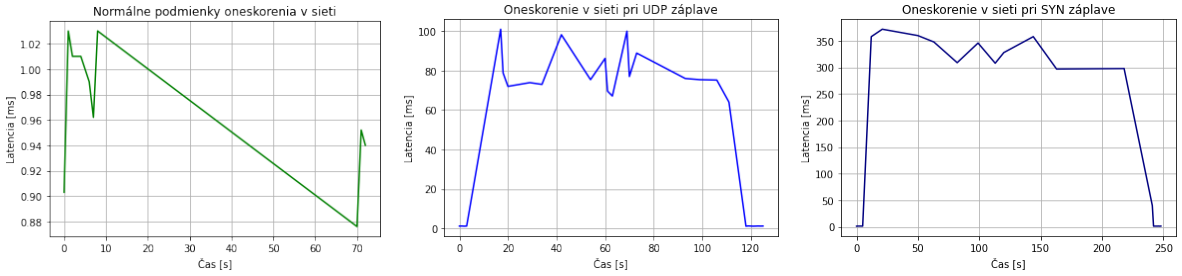


(b) Fyzická topológia

Obr. 4: Topológia izolovanej siete na experimenty

	RX [Mbit/s]	TX [Mbit/s]	RX [pakety/s]	TX [pakety/s]
UDP Flood	54,22	0,001	147345	1
ICMP Flood	45,07	2,21	122483	5534
SYN Flood	45,09	2,87	122539	5118
Slowloris	0,14	0,13	175	170

Tabuľka 1: Dosiahnutá sila útokov s hping odmerané spriemerovaním 5 sekúnd premávky s vnstat na serveri na sieťovej linke s reálnou šírkou pásma 94.2 Mbit/s



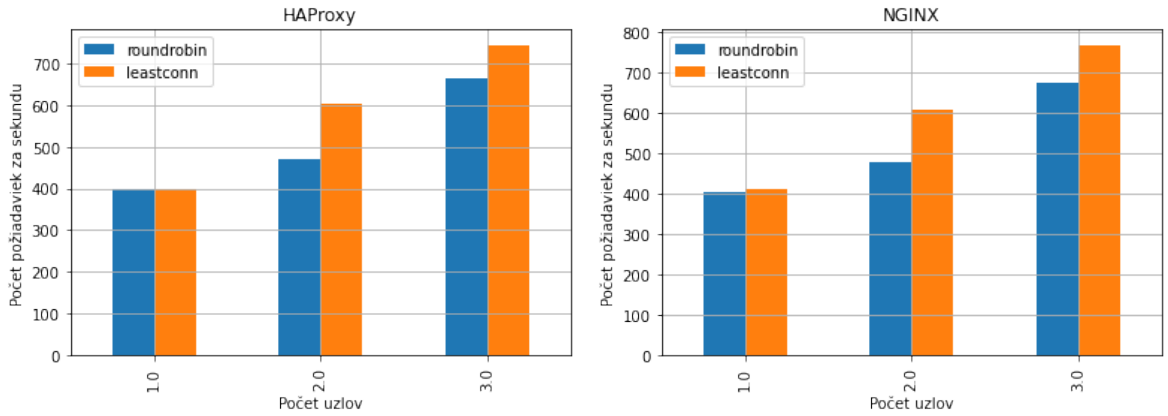

```
setsockopt(s, SOL_SOCKET, SO_BINDTODEVICE, &ifr);
setsockopt(s, IPPROTO_IP, IP_HDRINCL, (char *)&one, sizeof(one));
```

```
in_addr_t ip_random(in_addr_t net, int cidr)
{
    in_addr_t net_mask = (~0 << (32 - cidr));
    in_addr_t host_mask = ~net_mask;
    in_addr_t target_host = 0;
    if (host_mask != 0) target_host = rand() % host_mask;
    return htonl((ntohl(net) & net_mask) | target_host);
}
```

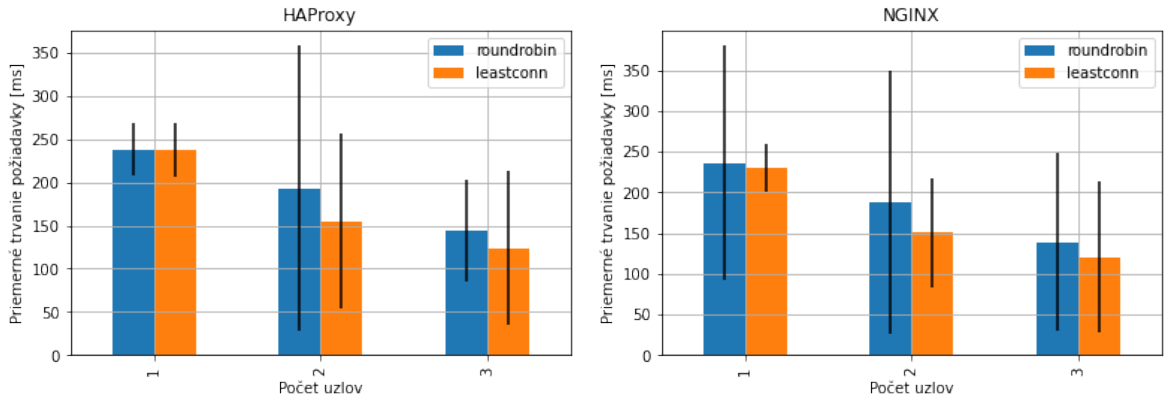
```
int s = socket(AF_INET, SOCK_STREAM, 0);
connect(s, (struct sockaddr *)&victim, sizeof(victim))
snprintf(buffer, LEN, "GET␣/?%d␣HTTP/1.1\r\n", randint(0, 1000));
send(s, buffer, strlen(buffer), 0);
```

```
for (int i = 0; i < CNT; i++) {
    snprintf(buffer, BUF_LEN, "X-a:␣%d\r\n", randint(1, 50000));
    send(sockets[i], buffer, strlen(buffer), 0);
}
sleep(10);
```

5.4 ApacheBench záťažové testy na load balancer

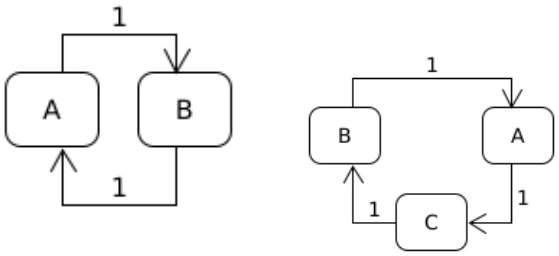


Obr. 6: Počet spracovaných požiadaviek za sekundu pre algoritmy vyvažovania záťaže podľa ApacheBench: `ab -n 1000 -c 100 http://192.168.0.2/`

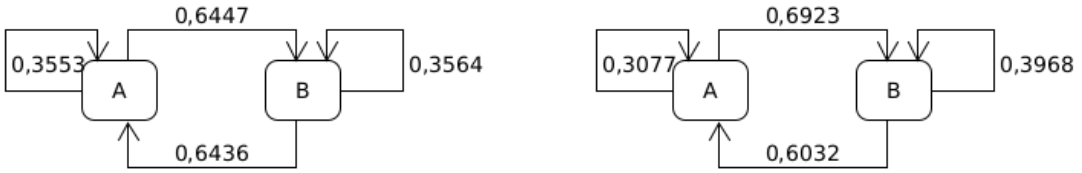


Obr. 7: Priemerné trvanie požiadaviek pre algoritmy vyvažovania záťaže podľa ApacheBench: `ab -n 1000 -c 100 http://192.168.0.2/`

5.5 Slowhttptest Slowloris útok na load balancer

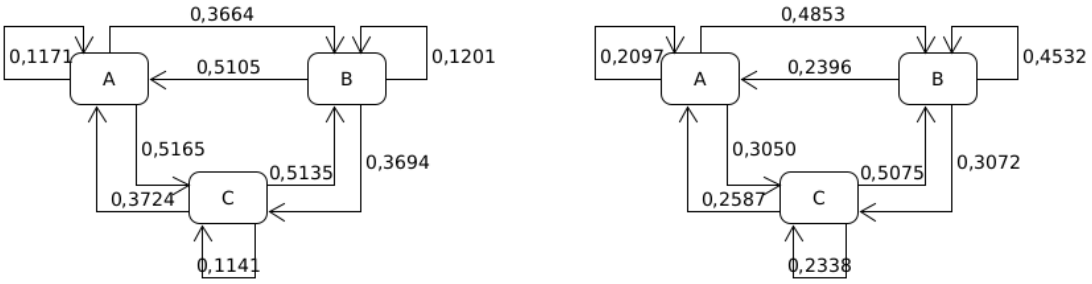


Obr. 8: Markovove reťazce plánovania pridelenia uzlov na obsluhu požiadaviek pri Round Robin aj Least Connections: `ab -n 10 -c 1 http://192.168.0.2/`



(a) Round Robin (b) Least Connections

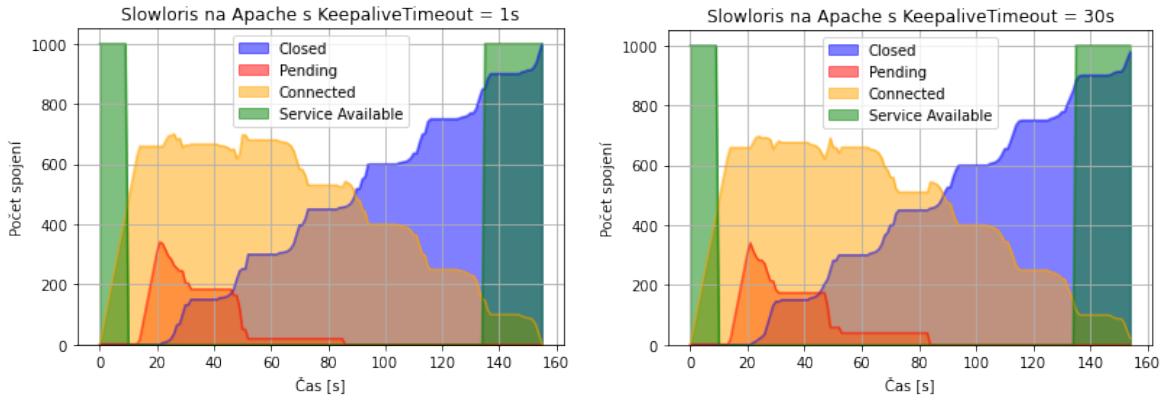
Obr. 9: Markovove reťazce plánovania pridelenia uzlov na obsluhu požiadaviek pri 2 koncových uzloch: `ab -n 10000 -c 1000 http://192.168.0.2/`



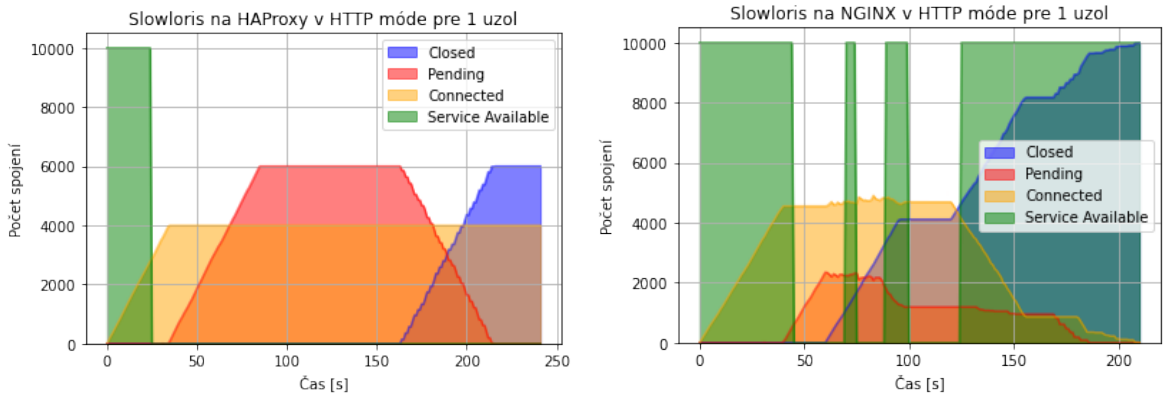
(a) Round Robin (b) Least Connections

Obr. 10: Markovove reťazce plánovania pridelenia uzlov na obsluhu požiadaviek pri 3 koncových uzloch: `ab -n 10000 -c 1000 http://192.168.0.2/`

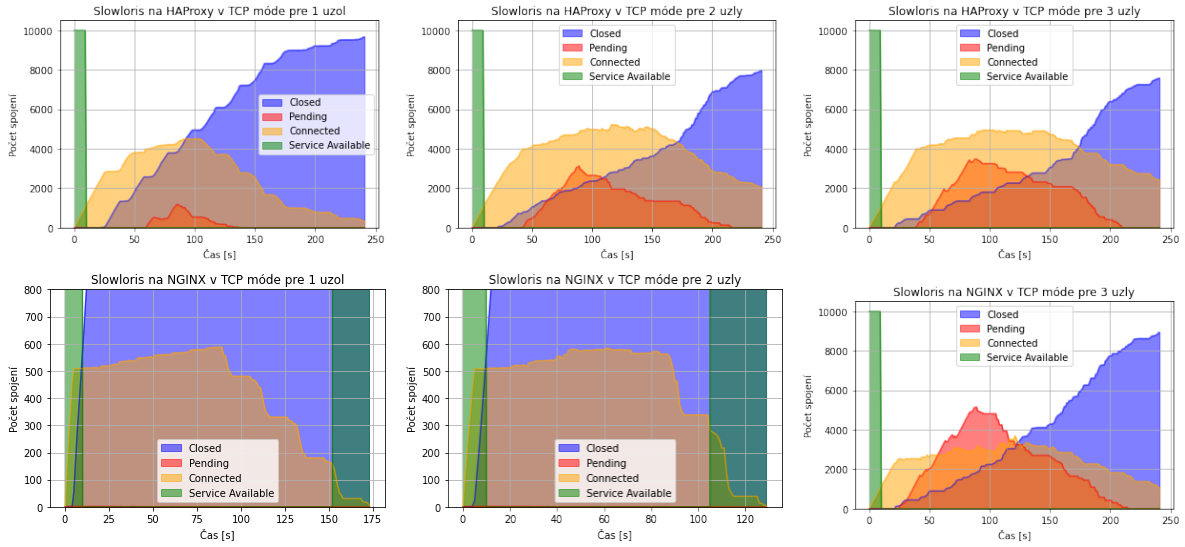
```
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 5
```



Obr. 11: Reakcia webového servera Apache na útok SlowLoris pri zmenách Keepalive timeout na spojenie:
`slowhttptest -H -c 1000 -i 10 -r 50 -l 180 -g -o stat -u http://192.168.0.2/`



Obr. 12: Porovnanie účinkov útoku Slowloris na počet aktívnych spojení a dostupnosť služby pri L7 load balancingu na HAProxy a NGINX:
`slowhttptest -H -c 10000 -i 10 -r 200 -l 240 -g -o stat -u http://192.168.0.2/`



6 Záver

Literatúra

1. MIR, Suhail; QUADRI, Syed. Information Availability: An Insight into the Most Important Attribute of Information Security. *Journal of Information Security*. 2016, roč. 07, s. 185–194. Dostupné z DOI: 10.4236/jis.2016.73014.
2. *DDoS Handbook: The Ultimate Guide to Everything You Need to Know about DDoS Attacks*. Radware, 2016. https://www.radware.com/getattachment/Security/Research/702/Radware_DDoS_Handbook_2015.pdf.aspx.
3. ABHISHTA, Abhishta; HEESWIJK, Wouter van; JUNGER, Marianne; NIEUWENHUIS, Bart; JOOSTEN, Reinoud. Why would we get attacked? An analysis of attacker's aims behind DDoS attacks. *Journal of Wireless Mobile Networks*. 2020, roč. 11, s. 3–22. Dostupné z DOI: 10.22667/JOWUA.2020.06.30.003.
4. PRASAD, K.Munivara; REDDY, A.Rama Mohan; RAO, K.Venugopal. DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms -A Survey. *Global Journal of Computer Science and Technology (GJCST)*. 2014, roč. 14. ISSN 0975-4172. https://core.ac.uk/reader/231149430.
5. COHEN, Lawrence; FELSON, Marcus. Social Change and Crime Rate Trends: A Routine Activity Approach. *American Sociological Review*. 1979, roč. 44. Dostupné z DOI: 10.2307/2094589.
6. HERETIK, Anton. Forenzná psychológia. In: 2. vyd. Sasinkova 5, 815 19 Bratislava: Slovenské Pegagogické Nakladateľstvo, 2004, kap. 4, s. 64–71. ISBN 80-10-00341-7.
7. *Infiltrating a Botnet*. Cisco, 2020. https://tools.cisco.com/security/center/resources/infiltrating_botnet.
8. Zákon 300/2005 Z.z. Trestný zákon. 2021. https://www.slov-lex.sk/pravne-predpisy/SK/ZZ/2005/300/.
9. COOKE, Evan; JAHANIAN, Farnam; MCPHERSON, Danny. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. 2005.
10. ALOMARI, Esraa; MANICKAM, Selvakumar; B. GUPTA, B.; KARUPPAYAH, Shankar; ALFARIS, Rafeef. Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art. *International Journal of Computer Applications*. 2012, roč. 49, č. 7, s. 24–32. Dostupné z DOI: 10.5120/7640-0724.
11. MIRKOVIC, Jelena; REIHER, Peter. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*. 2004, roč. 34, č. 2, s. 39–53. Dostupné z DOI: 10.1145/997150.997156.
12. *Ochrana pred útokmi DDoS (Príručka administrátora)*. CSIRT.SK, 2013. https://www.csirt.gov.sk/doc/DDoS_CSIRT.pdf.

13. ENISA Threat Landscape 2020 - Distributed denial of service. *ENISA*. 2020. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-distributed-denial-of-service.

14. KUMARI, W.; MCPHERSON, D. *Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF)* [Internet Requests for Comments]. RFC Editor, 2009-08. RFC, 5635. RFC Editor. ISSN 2070-1721.

15. *Remotely Triggered Black Hole Filtering - Destination Based and Source Based*. Cisco, 2005. https://www.cisco.com/c/dam/en_us/about/security/intelligence/blackhole.pdf.

16. *Protecting Your Core: Infrastructure Protection Access Control Lists*. Cisco, 2008. https://www.cisco.com/c/en/us/support/docs/ip/access-lists/43920-iacl.html.

17. *Linux Hardening (Príručka administrátora)*. CSIRT.SK, 2013. https://www.csirt.gov.sk/doc/Hardened_v1.pdf.

18. THALER, D.; HOPPS, C. *Multipath Issues in Unicast and Multicast Next-Hop Selection* [Internet Requests for Comments]. RFC Editor, 2000-11. RFC, 2991. RFC Editor. ISSN 2070-1721. http://www.rfc-editor.org/rfc/rfc2991.txt.

19. KNIGHT, S.; WEAVER, D.; WHIPPLE, D.; HINDEN, R.; MITZEL, D.; HUNT, P.; HIGGINSON, P.; SHAND, M.; LINDEM, A. *Virtual Router Redundancy Protocol* [Internet Requests for Comments]. RFC Editor, 1998-04. RFC, 2338. RFC Editor. ISSN 2070-1721. Dostupné tiež z: http://www.rfc-editor.org/rfc/rfc2338.txt.

20. *Keepalived User Guide*. 2021. https://keepalived.readthedocs.io/en/latest/.