VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Pavel Mazánek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MODELOVÁNÍ A DETEKCE ÚTOKU SLOWDROP
MODELING AND DETECTION OF SLOWDROP ATTACK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Pavel Mazánek

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Marek Sikora

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Pavel Mazánek ***ID:*** 185933
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Modelování a detekce útoku SlowDrop

POKYNY PRO VYPRACOVÁNÍ:

Tato práce je zaměřena na nedávno objevený slow DoS útok s názvem SlowDrop. Úkolem je podrobně nastudovat a popsat současný stav problému a vytvořit funkční model útoku pro následné testování. Dalším úkolem je vytvořit testovací prostředí, ve kterém se prokáže funkčnost modelu a zranitelnost běžně používaných webových serverů. Posledním úkolem je návrh, implementace a testování metodiky, která bude schopna detekovat přítomnost tohoto útoku v počítačové síti.

DOPORUČENÁ LITERATURA:

[1] CAMBIASO, Enrico, Giovanni CHIOLA a Maurizio AIELLO. Introducing the SlowDrop Attack. Computer Networks [online]. 2019, 150, 234-249 [cit. 2019-04-18]. DOI: 10.1016/j.comnet.2019.01.007. ISSN 13891286. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1389128619300210

[2] SUROTO, Suroto. A Review of Defense Against Slow HTTP Attack. JOIV: International Journal on Informatics Visualization [online]. 2017, 1(4), 127-134 [cit. 2019-06-06]. DOI: 10.30630/joiv.1.4.51. ISSN 2549-9904. Dostupné z: http://joiv.org/index.php/joiv/article/view/51

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Marek Sikora

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Práce je zaměřena na nedávno objevený slow DoS útok s názvem SlowDrop. Zabývá se podrobným nastudováním a popisem současného stavu problému DoS útoků obecně i útoku konkrétního. Z teoretického základu vychází při implementaci vlastního modelu SlowDrop útoku, který je následně testován v různých scénářích a vzešlé výsledky jsou analyzovány a kriticky diskutovány. Dále se práce snaží o návrh obrany proti SlowDrop útoku, rozebírá doporučená nastavení na straně serveru a metodiky pro snížení hrozby stavu DoS jak obecně, tak i z pohledu SlowDrop útoku. Navržená metodika obrany proti této hrozbě je následně zkoušena a hodnocena. V neposlední řadě je datový provoz SlowDrop útoku porovnáván s datovým provozem legitimního klienta se špatným připojením, které se tento útok snaží napodobit. Z uvedené komparace jsou v závěru vyvozena finální východiska práce.

KLÍČOVÁ SLOVA

DDoS, DoS, HTTP, IDS, IPS, pomalý, SDA, TCP, útok, zahazování, ztráta

ABSTRACT

The work's main topic is a recently published slow DoS attack called SlowDrop. The work focuses on the subject of describing the current state of the DoS problem as a whole and the SlowDrop attack as well. It works with this theoretical basis during the implementation of it's own SlowDrop attack model. This model is tested in various scenarios and the outcome results are analyzed and constructively discussed. Furthermore defensive mechanisms against this threat and DoS attacks in general are proposed, specific methods shown and configurations recommended. These methods are followingly tested and evaluated. Last but not least the traffic of a SlowDrop attacker and a legitimate client with bad connection, which the SlowDrop attack is trying to immitate, are compared. From this comparison final conclusions of this work are drawn.

KEYWORDS

attack, DDoS, DoS, drop, HTTP, IDS, IPS, loss, SDA, slow, TCP

MAZÁNEK, Pavel. *Modelování a detekce útoku SlowDrop*. Brno, Rok, 75 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Marek Sikora

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Modelování a detekce útoku SlowDrop“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Marku Sikorovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Diplomová práce se zabývá problematikou relativně nového pomalého DoS útoku s názvem SlowDrop publikovaného v roce 2019[1]. Cílem této práce je teoreticky i prakticky rozvést útok SlowDrop, uvést informace nutné k pochopení jeho fungování, realizaci, testování a také následnému návrhu obrany proti němu.

Na teoretické i praktické poznatky získané v rámci práce, která vychází primárně z jediného zdroje – návrhu útoku[1], je kriticky nahlíženo, jsou hodnoceny a následně konstruktivně komentovány za účelem zlepšení realizace modelu SlowDrop útoku a obrany proti němu[2].

V úvodních teoretických kapitolách se práce zabývá problematikou DoS/DDoS útoků obecně, kategorizuje je a na základě tohoto rozdělení uvádí specifické příklady reprezentující dané kategorie. Mimo jiné k těmto rozdělením uvádí také nejčastější způsoby realizace obrany proti příkladovým DoS útokům, jelikož obranou proti klíčovému SlowDrop útoku se práce zabývá v pozdějších kapitolách.

Práce rozděluje uvedené útoky do různých kategorií z toho důvodu, že se v následujících částech snaží o zařazení SlowDrop útoku do těchto pomyslných kategorií[3],[4], které nejsou tak přesně vymezeny vzhledem k rozličnosti existujících DoS útoků, aby bylo jejich snadné a jednoznačné zařazení umožněno.

V hlavní teoretické částí práce se pojednává o teoretické přípravě SlowDrop útoku, jeho principech, základních myšlenkách, inspiracích z předchozích publikovaných pomalých DoS útoků, ale také o jeho slabinách a omezeních, které jsou optimálně využity jak při realizaci útoku samotného, tak obrany proti němu.

Hlavní teoretická část práce také rozvádí klíčové parametry, na jejichž základě SlowDrop útok operuje a jejichž pochopení a přizpůsobení pro konkrétní případy je klíčové k úspěšnému dosažení stavu odepření služeb (DoS) na straně oběti za co nejmenšího rizika detekce.

Konkrétní parametry realizace tohoto útoku jsou teoreticky diskutovány, následně implementovány a testovány. Optimalizace těchto parametrů pro konkrétní situační scénář je jedním z klíčových aspektů pro průběh tohoto útoku, který by měl být při správné implementaci zaměnitelný s chováním legitimního klienta služeb serveru cílové oběti.

V první praktické části se práce zabývá jak obecně realizací SlowDrop útoku v praxi, tak jednotlivými konkrétními kroky průběhu útoku. Na začátku praktické části je představeno testovací prostředí, ve kterém vývoj a následné testování implementace SlowDrop útoku probíhá. Následně jsou rozebrány jednotlivé fáze praktické realizace SlowDrop útoku, jejich praktická omezení a doporučení.

Největší důraz je kladen na fázi útoku, ve které probíhá zahazování, na jehož základě stojí celý princip útoku SlowDrop. Následně jsou uvedeny způsoby realizace

zahazování, které byly v rámci práce testovány a vyzkoušeny. Tyto způsoby jsou následně diskutovány na základě jejich vhodnosti pro implementaci SlowDrop útoku a je vybrán způsob jeho realizace v rámci této práce.

Dále se práce věnuje porovnání testovacích scénářů modelů útoku realizovaného jeho autory[1] a scénářů realizovaných v rámci této práce. Diskutuje poznatky získané z těchto testování a vyvozuje závěry ovlivňující správné nastavení modelu SlowDrop útoku pro jednotlivé scénáře. Důraz je v rámci této práce kladen na dosažení stavu DoS na straně serveru oběti po požadovanou dobu určeného testovacího scénáře za jeho podmínek.

Provoz testovaných scénářů je následně analyzován a teoretické poznatky relevantní pro optimalizaci i omezení realizace SlowDrop útoku jsou uvedeny. Zejména jsou rozvedeny nástroje a vlastnosti protokolů, se kterými SlowDrop útok pracuje a zneužívá jich, a jejichž parametry jsou hlavním limitem tohoto útoku.

Práce se v poslední praktické části zabývá návrhy na implementaci IDS – systému detekce SlowDrop útoku. Uvádí kroky podniknuté pro zabezpečení serveru oběti před SlowDrop útokem, jak obecná konfigurační nastavení serveru, tak vybrané obrané a detekční software komponenty, o které byl server oběti rozšířen.

Konečné části práce se věnují porovnání legitimního provozu klienta se špatným připojením realizovaného pomoci WAN emulátorů a provozu SlowDrop útoku, který se legitimního klienta snaží napodobit za účelem snížení rizika detekce. V kontextu této komparace jsou následně uvedeny závěry vyplývající pro vznik signatury a návrh obrany proti tomuto útoku.

1 Kategorizace DoS útoků

Teoretická část práce se zabývá tématy, která se obecně týkají DoS útoků jako takových. Na začátku je rozděluje do již v praxi rozpoznaných a zavedených kategorií[3], [4] na základě jejich klíčových vlastností, principů a odlišností mezi nimi.

Pro charakterizaci konkrétního DoS útoku, kterým se práce dále bude zabývat, je nutné jej identifikovat a zařadit do již známých kategorií z důvodu zlepšení přehlednosti (do míry, které lze dosáhnout).

Níže uvedená dělení DoS útoků se používají již od vydání prvotních publikací věnujících se tomuto tématu a také jsou definovány organizacemi zabývajícími se IT bezpečností[5], avšak s plynutím času, vývojem technologií a objevování nových způsobu, jak docílit DoS stavu, se ovšem tyto kategorie obecně mění, rozšiřují, různě kombinují a tím pádem se také komplikují ve svých vymezeních.

Níže uvedené rozdělení do skupin a kategorií podle určitých vlastností DoS útoku je tedy poměrně povrchové a často není stoprocentně přesné z důvodu komplexnosti konkrétních útoků, které mohou spadat do více kategorií zároveň anebo jsou například tak unikátní, že tvoří kategorii sami o sobě.

V rámci rozdělení jsou uvedeny i příklady z praxe demonstrující rozdělení do kategorií za účelem zlepšení přehlednosti. Problematice dělení DoS útoků (zejména pomalých) podle jejich vlastností se podrobně věnuje např. [3].

1.1 Počet útočících zařízení

Nejvíce v laické mluvě zaměňované a zároveň nejvíce na první pohled zřejmé rozlišení DoS útoků je podle počtu útočících zařízení, tedy podle počtu zařízení, která generují škodlivý provoz cílený na oběť způsobující stav DoS.

Rozlišujeme tedy tzv. nedistribuované a distribuované DoS útoky, neboli v zpopularizovaných anglických zkratkách DoS – *Denial of Service* a DDoS – *Distributed Denial of Service* útoky.

1.1.1 Nedistribuované

Jedná se o DoS útoky, při kterých útočník disponuje pouze jedním zařízením, které generuje škodlivý provoz. Je logicky odvoditelné, že v případě kybernetického útoku za cílem dosažení stavu odepření služeb u oběti hraje často výpočetní síla útočníka nebo počet útočníků významný faktor.

Proto jsou zpravidla nedistribuované útoky podceňovány v celosvětovém měřítku, avšak nelze opomíjet škálu jejich schopností například v kontextu menších útoků cílených na zařízení nebo servery, které jsou opatřeny pouze slabou nebo žádnou

ochranou proti DoS útokům, jako jsou například koncová zařízení běžných uživatelů nebo menších firem.[1]

Dosažení stavu DoS na straně oběti je zpravidla neúměrně méně náročné pro stranu útočníka oproti tomu, jak náročné je bránit se na straně oběti. Z tohoto důvodu si specializovanou ochranu proti DDoS útokům často zajišťují pouze velké firmy, které mají svá vlastní rozsáhlá IT oddělení, nebo společnosti, jejichž služby musí být dostupné i v případě, že by na ně byl realizován DoS útok.

Skutečnost, že z jednoho domácího počítače v cenové relaci desetitisíců korun českých, kterým dnes disponuje značná část domácností, je útočník schopný dosáhnout stavu odepření služeb na serveru, jehož pořizovací cena často přesahuje statisíce jestliže není ochrana proti DoS útokům udržována a aktualizována[2], jen poukazuje na to, jak důležité je zabývat se obranou proti DoS útokům .

V neposlední řadě je důležité říci, že jakýkoliv nedistribuovaný DoS útok se může stát DDoS útokem při propojení více útočných zařízení za účelem uskutečnění daného útoku, čímž se může například dosáhnout stavu DoS u oběti rychleji, je možné jej udržet po delší dobu, pokud to je záměrem útočníka, nebo může být zejména hůře detekovatelný, čímž se proti němu lze ze strany oběti hůře bránit.

Detekce nedistribuovaných útoků

Slabinou nedistribuovaných útoku je často ten fakt, že pocházejí z jednoho zařízení a zpravidla je tedy jejich provoz charakterizován společnou charakteristikou, kterou jim dává zařízení generující DoS útok.

Obecně bývají nedistribuované DoS útoky generovány za účelem vyčerpání specifických zdrojů na straně oběti, avšak pokud se tento zdroj bude snažit vyčerpat jediné útočníkovo zařízení, je relativně jednoduché nastavit na straně serveru oběti (nebo v jeho lokální síti) pravidla pro omezení toho, co si jednotlivé zařízení může vzhledem k oběti nárokovat za zdroje (počet TCP spojení, přidělené IP adresy, apod.).

V neposlední řadě je při detekci nedistribuovaného DoS útoku na místě vždy přidat útočníkovo zařízení na tzv. *černou listinu* (angl. *black list*) na základě nějaké určující charakteristiky, kterou často bývá např. IP adresa.

[1] V práci [1] je navrhováno použití i hardwarově nenáročného zařízení jako Raspberry Pi pro realizaci DoS útoku, potažmo obraně proti nim.

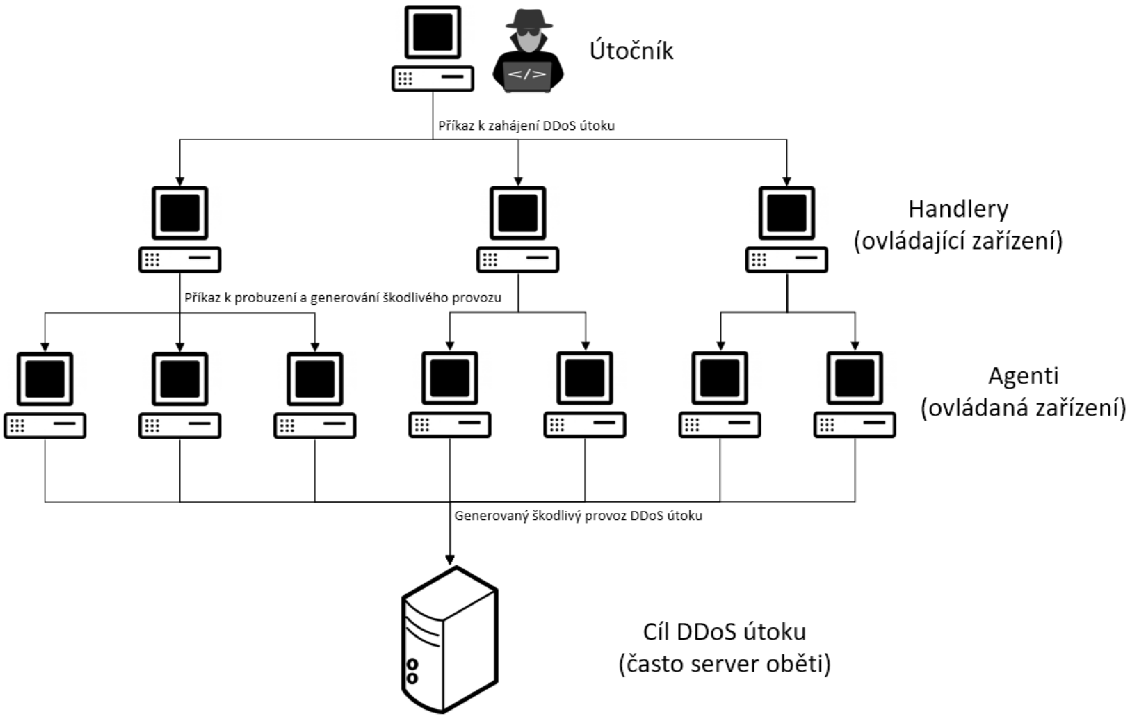
[2] Přičemž zařízení specializovaná na jejich ochranu jsou taktéž finančně nákladná. Např. hardwarové filtry (firewally) společnosti *paloalto networks* se cenově pohybují v řádech statisíců dolarů. Dostupné z: https://www.paloaltonetworks.com/products/product-selection

1.1.2 Distribuované

Ve světě známější a škodlivě potentnější distribuované DoS neboli DDoS útoky spočívají v zapojení více zařízení jakožto generátorů škodlivého provozu směrem k cílové oběti. Tyto útoky, ačkoliv tomu v některých případech není potřeba, jsou často koordinované z více zařízení, která řídí více podřazených zařízení a zadávají jim příkazy, na základě kterých tato zařízení již konají útok – již přímo generují a zasílají škodlivý provoz (pakety, datagramy).

Zmíněným zařízením, která na základě žádosti útočníka koordinují útok a rozdávají rozkazy, se dle používaného *agent–handler* modelu[6] anglicky říká *handler* od slovesa *handle*, které v češtině znamená *ovládat* nebo *řídit*.

Zařízením, která tyto rozkazy poslouchají a na jejich základě vykonávají útok se říká *agents*, *bots* nebo *zombie*, přičemž české překlady těchto pojmů jsou zřejmé. Výraz, u kterého je ale na místě jeho zmínění v tomto kontextu a bližší upřesnění, je tzv. *botnet* neboli síť botů.



Obr. 1.1: Model agent-handler

Počet zařízení, která může mít jeden DDoS útočník k dispozici se může pohybovat i v řádech tisíců (potenciálně více), přičemž tento počet zařízení se odvíjí nejčastěji od počtu zařízení infikovaných malware (*malicious software* – česky *škodlivý software*) daného útočníka nejčastěji v podobě nejčastěji specifického druhu trojského koně, což je typ malware, který se na první pohled jeví jako legitimní software

nebo skrývá svoji škodlivou povahu, avšak dokáže převzít kontrolu nad zařízením (nebo jeho částí) bez vědomosti jeho majitele a následně v případě DDoS generovat škodlivý provoz na pozadí běžného užívání zařízení.

Díky této schopnosti, kdy zařízení se může chovat přirozeně po dobu jeho užívání, avšak na přání útočníka se „probudí“ a stává se součástí DDoS útoku, se tato zařízení označují za, jak bylo dříve zmíněno, *zombie*.

V době psaní této práce je hrozba DDoS útoků stále velmi zřejmá a nelze ji podceňovat na celosvětovém měřítku. Jako aktuální demonstrativní příklad DDoS útoku (mimo již zmíněné DDoS útoky, které probíhají denně po celém internetovém prostoru[3]) může posloužit například útok na populární vývojářskou platformu GitHub[7], která čelila v roce 2018 útoku o velikosti 1,35 terabitů za sekundu (*Tbps*) a 130 miliónů paketů za sekundu.

GitHub naštěstí byl díky jejich DDoS ochraně od společnosti Akamai připraven na útok i většího rázu, avšak, jak sám viceprezident společnosti Akamai – Josh Shaul, řekl: *„It's one thing to have the confidence, it's another thing to see it actually play out how you'd hope.“*[7]. Volně česky přeloženo: *„Je jedna věc být sebevědomý, druhá věc je, jestli vše opravdu dopadne podle očekávání.“*.

Pro porovnání a představu toho, jak se s časem posouvají i možnosti DDoS útočníků je také potřeba uvést útoky z ledna roku 2019 na jednoho z klientů společnosti Imperva o velikosti 500 milionů paketů za sekundu a z dubnu roku 2019 proti jinému klientovi téže společnosti, který ve své největší intenzitě generoval přes 580 milionů paketů za sekundu[7].

Pouze z těchto dvou demonstrativních příkladů je možné zpozorovat trend ve zvyšování intenzity a tedy i závažnosti hrozby DDoS útoku. V kontextu DDoS na celosvětovém měřítku nezůstává opomíjena ani Česká republika, kde v roce 2017 byly pod DDoS útokem stránky českého statistického úřadu při přepočítávání hlasů parlamentních voleb[8].

V neposlední řadě je potřeba zmínit, že v dnešní době lze DDoS útok objednat, ať už za nelegálními (například k znepřístupnění služeb konkurenční společnosti) nebo legálními účely. Příkladovým legitimním účelem objednání DDoS útoku může být najmutí tzv. *stres-testerů*, které provedou DDoS útok připravený na míru na základě objednávky a zadaných parametrů[9].

Tyto služby, nazývané *DoS as a Service*, jsou široce nabízeny na internetu[10], avšak před tím, než člověk nebo společnost těchto služeb využije, tak je třeba rozsáhlých příprav a komunikace s ISP. Tato služba může také být součástí penetračního testování na objednávku a je na místě zákazníka na užití těchto metod předem

[3]Existují webové stránky sledující průběh těchto útoků v reálném čase, jako například digitalattackmap.com, cybermap.kaspersky.com a další.

upozornit (nebo se domluvit na jejich rozsahu, vymezení síťové oblasti apod.), jelikož může při jejich úspěchu nastat stav odepření služeb nebo i poškozen důležitý hardware, což nemusí být v zájmu zákazníka.

Detekce distribuovaných útoků

Oproti nedistribuovaným DoS útokům nelze jednoduše omezit to, co si může konkrétní zařízení dovolit vzhledem k oběti (síti oběti), nebo přímo zakázat již detekovanému škodlivému zařízení generování provozu ovlivňujícího síť.

Prevence DDoS útoků často spočívá v aplikaci plošných pravidel pro celou síť, kritickou infrastrukturu (určení tzv. demilitarizované zóny) nebo rozložení této infrastruktury do více míst (geograficky míněno), pak jestliže je jeden server (shluk) poskytující službu v DoS stavu, může jej zastoupit server záložní a naopak.

1.2 Způsob útoku

Kromě počtu útočících zařízení lze DoS útoky dělit také podle principu útoku, který za útokem stojí, neboli čeho útok zneužívá, aby dosáhl stavu odepření služeb na straně oběti. Podle toho, jak DoS útoky fungují, je lze dělit na *záplavové* (neboli anglicky typu *flood*) a *zneužívající zranitelností* (anglicky nazývané typu *exploit* nebo *crash*).

1.2.1 Záplavové

Nejrozšířenějším typem DoS (DDoS) útoků jsou útoky záplavové, které spočívají v *zahlcení* oběti nebo určité služby na straně oběti. Mezi ty nejznámější patří:

- **ICMP flood** – také známé jako *ping flood* nebo *ping of death*. Útočník zahlcuje oběť *ICMP echo-requesty*, na které oběť (v případě nenastavené obrany proti tomuto útoku) musí odpovídat *ICMP echo-reply*, čímž dochází k zahlcení.
- **SYN flood** – útok zneužívající povahy *three-way handshake* TCP protokolu. Útočník navazuje spojení s obětí (odesílá pakety s příznakem SYN, odsud název), ale nikdy jej neuzavře kompletně, tedy nedokončí *three-way handshake*. Útočník se tak snaží obsadit obsadit všechna spojení, aby legitimní uživatelé se již nemohli připojit.
- **RST/FIN flood** – útok také zneužívající TCP protokolu, avšak zneužívající paketů s příznakem RST/FIN. Pakety s tímto příznakem resetují/ukončují aktivní TCP spojení, tudíž záměrem útočníka je narušit veškerá legitimní spojení oběti s uživateli. Tento útok již vyžaduje k efektivní účinnosti určité znalosti o aktivních spojení, které se ale dají zjistit pomocí tzv. *sniffingu* (odposlouchávání provozu) nebo odhadnout.

- **HTTP flood** – jedná se o více typů útoků pracujících s dotazy protokolu HTTP. Nejčastěji bývají zneužívány HTTP requesty typu GET (získání dat z webového serveru oběti) a POST (např. nahrání/vložení dat útočníka na webové stránky oběti). Známým příkladem těchto útoků je zejména HOIC (High Orbit Ion Cannon) používaný v minulosti skupinou *Anonymous*[11].
- **ARP flood** – útok zneužívající ARP protokolu, který se stará o zjištění linkových (nejčastěji MAC) adres z adres síťových (nejčastěji IP). Síť oběti v tomto případě útočník zahltí ARP dotazy nebo podvrženými odpověďmi na ně, čímž dochází k přeplnění tzv. *ARP cache*, kde jsou tyto záznamy ukládány, a k obecnému zahlcení sítě oběti vzhledem k broadcastové povaze protokolu.
- **DNS flood** – útok zneužívající protokolu DNS, který se stará o získání IP adresy z jednodušeji zapamatovatelného DNS názvu, tedy doménového jména. Tento útok ale může být cílen pouze na DNS servery, a to tak, že je na daný server útočníkem zasíláno tak velké množství dotazů, že nedokáže odpovídat dotazům od legitimních uživatelů a dochází tím pádem ke stavu DoS.
- **DHCP starvation** – útok zneužívající protokolu DHCP, který dynamicky nabízí (a přiděluje/rezervuje po dobu *lease time*) na základě dotazu *DHCP discover* žadateli IP adresu a další údaje potřebné k působení v síti (maska sítě apod.). Útočník se snaží o vyčerpání adresového prostoru DHCP serveru nebo o jeho hardwarové vytížení v takové míře, že dochází ke stavu DoS.

V případě nedistribuovaného záplavového útoku je obrana na první pohled zřejmá a poměrně jednoduše realizovatelná[4] – zakázání provozu od útočícího zařízení (nejčastěji formou zahazování paketů pocházejících z určitého zdroje, IP/MAC adresy).

V případě DDoS záplavového útoku nejde však v praxi zablokovat všechny útočící stroje (zvláště, když jsou jejich počty v řádech tisíců a více), obrana se tedy stává znatelně složitější a je potřeba aplikovat pokročilejší metodiku často v podobě obranných algoritmů a nastavení síťových pravidel.

Detekce záplavových DoS útoků

Detekce záplavových útoků prostřednictvím systému detekce – IDS probíhá na základě tzv. anomálií v provozu[12],[13]. Jestliže normální provoz mezi legitimními uživateli a serverem, který jim poskytuje služby, má svůj charakteristický průběh a podobu, tak pomocí heuristiky, dynamicky či staticky nastavených prahů nebo metod strojového učení lze systému IDS určit, kdy ještě chráněný systém pracuje za normálních podmínek a kdy ne, tedy kdy je pod vlivem DoS útoku.

[4]Nebereme v potaz schopnost útočníka podvrhovat zdrojové adresy, která modelovou situaci činí komplikovanější.

1.2.2 Zneužívající zranitelnosti

DoS útoky, u kterých nezáleží primárně na množství generovaného škodlivého provozu, ale na vektoru útoku, tedy zneužité zranitelnosti, jsou v provedení protistranou záplavovým útokům v otázce poměru *kvantita vs. kvalita*.

Nelze je definovat obecně, jelikož i záplavové útoky zneužívají určitých zranitelností jako např. specificky navržených protokolů a tudíž i všechny záplavové útoky prakticky spadají do této kategorie. Je ale potřeba rozlišit útoky, které zneužívají zranitelností a využívají velký provoz k dosažení DoS stavu u oběti od těch, které nevyžadují prakticky žádnou výpočetní sílu na straně útočníka.

Tyto útoky jsou už z jejich povahy poměrově výskytem mnohem vzácnější, jelikož vygenerovat úspěšný záplavový útok není s průměrným hardware zdaleka tak složité jako najít zranitelnost, která ještě nebyla opravena nebo ošetřena na straně oběti, a dosáhnout jejím prostřednictvím stavu DoS. Pokud jsou provedeny správně, tak jsou zpravidla tyto útoky oproti záplavovým logicky méně nápadné a tím pádem se proti nim i hůře brání.

Mezi tyto útoky spadají mimo jiné zejména tzv. *pomalé* neboli *slow* DoS útoky, mezi které spadá i útok SlowDrop, kterým se tato práce primárně zabývá. Avšak i tyto útoky mohou vykazovat rysy záplavových DoS útoků. Více se těmto útokům bude práce věnovat v pozdějších kapitolách.

Detekce DoS útoků zneužívajících zranitelnosti

Jelikož se tyto útoky při sledování provozu sítě zpravidla neprojeví prostřednictvím překročení prahových hodnot nebo skoky v grafech analýzy provozu, tak se detekují na základě jejich specifických znaků – signatur, v systému detekce se jedná anglicky o tzv. *signature-based IDS*. Signatura je specifický vzor či atribut útoku, na základě kterého jej IDS rozezná od legitimního provozu. Může se jednat například o specificky upravenou hlavičku IP paketu nebo HTTP požadavek.

Příkladů toho, jak může signatura DoS útoku vypadat je stejně tak mnoho, jako těchto DoS útoků a jejich podob, kterých je každým dnem více a více. V případě *signature-based IDS* je tedy hlavní podstatou udržovat aktuální databázi známých signatur útoků, aby na ně mohl detekční systém reagovat.

Ovšem signaturu útoku je možné vytvořit a do pravidel IDS začlenit až po tom, co daný útok někdo detekuje, jeho signaturu zformuluje a potvrdí se, že se pravidlo na základě sepsané signatury aplikuje správně pouze na zamýšlený škodlivý provoz – nedetekuje legitimní provoz jako útok. Proces vytváření těchto pravidel pro IDS na základě signatur je tedy oproti detekování anomálií v provozu značně složitější a vyžaduje hlubší pochopení principu daného útoku.

1.3 Cíl útoku

Následně lze DoS/DDoS útoky rozdělit podle toho, jakým způsobem dochází na straně oběti ke stavu odepření služeb legitimním uživatelům, tedy na základě nedostatku čeho již nemůže oběť dále poskytovat své služby.

Je potřebné útoky dělit na základě tohoto parametru, jelikož na první pohled se může zdát, že je zřejmé, o co se daný útok snaží – jaký zdroj na straně oběti se snaží vyčerpat, ale může se stát, že tento odhad je špatný a jedná se až o sekundární zdroj, který se snaží legitimním klientům odepřít, nebo je těchto zdrojů služeb vícero a na základě jejich kombinovaného vyčerpání dochází ke stavu DoS.

Opět se tedy lze setkat se sofistikovanějšími útoky, které cílí svojí aktivitu na větší počet zdrojů na straně oběti, čímž mají větší potenciál škodlivosti.

1.3.1 Šířka pásma

Pravděpodobně nejjednodušeji realizovatelné a také pochopitelné jsou útoky mířené na šířku pásma, přes které oběť poskytuje své služby. Jedná se o nejstarší typy DoS/DDoS útoků, které se snaží vyčerpat síťový zdroj šířky pásma na straně oběti, anglicky tzv. *bandwidth*.

Jestliže útočník obsadí veškerou šířku pásma (nebo její valnou většinu), přes které oběť posílá data spojená se službami, které poskytuje, tak legitimní uživatel se s obětí daného útoku nebude moct spojit nebo jeho připojení a následná komunikace bude značně zpomalená, často až za hranici praktické použitelnosti takového připojení, na základě čehož se i v praxi může stát, že oběť ukončí toto spojení legitimního uživatele, i když jeho spojení záměrně nijak škodlivé není a ke stavu DoS záměrně nepřispívá.

Detekce útoků na šířku pásma

Proti tomuto typu útoků se nejčastěji brání servery nastaveními, kterými určí, jakou šířku pásma věnují jednotlivým spojením s klienty. Toto nastavení vychází z parametrů jako je kvalita připojení serveru (serverů) jakožto takového, limit současných spojení a dalších – této technice se říká anglicky tzv. *rate-limiting*, tedy ovládání toku provozu.

Jestliže i přes aplikování těchto pravidel dochází k vyčerpání šířky pásma serveru, často aplikovanými možnostmi nápravy je navýšit tento parametr např. přidáním nových serverů nebo tzv. *load-balancing* (vyrovnání zátěže) mezi servery.

I přes veškeré snahy poskytovatele služby se však může stát i v legitimních situacích, že se jeho servery dostanou do stavu DoS vzhledem k neočekávaně velkému zájmu o jeho služby (např. při spuštění serverů nové populární hry).

1.3.2 Hardware

Útoky cílené na technickou vybavenost oběti se snaží vyčerpat nejčastěji různé druhy pamětí nebo výpočetních kapacit. Útočník se může snažit například vytížit výpočetní sílu oběti různorodými požadavky do takové míry, že technická komponenta procesoru neboli CPU nebude tyto požadavky fyzicky stíhat zpracovávat.

Útoky tohoto typu je možné směřovat nejen přímo na oběť, jejíž služby chce útočník znepřístupnit, ale je možné je směřovat například i na ochranné mechanizmy, kterými cílová oběť disponuje, jako jsou třeba detekční a prevenční systémy – *IDS* a *IPS*, za účelem zvýšení zranitelnosti oběti samotné.

Mimo výpočetní sílu oběti lze útočit také na paměti, ve kterých si oběť ukládá data, se kterými aktuálně pracuje nebo která bude zpracovávat. Jedná se například o RAM, různé typy vyrovnávacích pamětí, bufferů a zásobníků. Tyto DoS útoky se snaží dané paměti zaplnit do takové míry, kdy nové požadavky na uložení dat do těchto pamětí již nemohou být zpracovány a buď čekají v nějakém typu fronty, která neustále narůstá, nebo jsou přímo zahazovány.

V případě, že fronta neustále narůstá a legitimní požadavky jsou díky obrovskému množství požadavků útočníka umísťovány až na konec fronty, tak dochází k jejich vyřizování až po dlouhé době, kdy už může být (jak bylo dříve zmíněno) spojení ukončeno na základě toho, že je pro praktické užití příliš pomalé.

V případě, kdy už je i prostor fronty vyčerpán a nové požadavky jsou přímo zahazovány na straně oběti, tak legitimní požadavek nedostane ani příležitost k tomu se dostat ke paměti, ze které by byl zpracován, tudíž dochází také ke stavu odepření služeb.

Ukázkovým příkladem, který cílí na vyčerpání kapacity bufferu, je v minulosti velice známy a nebezpečný *Ping smrti* (anglicky *Ping of Death*), který cílil na zneužití chyby v protokolu TCP/IP. Nástroj ping spadající pod ICMP protokol se běžně používá pro zjištění, zda vzdálená stanice nebo server pracuje. Typicky má zpráva nástroje ping velikost 56 bajtů (64 když počítáme i hlavičku) z maximálních možných 65 535 bajtů – toto maximum je určeno standardem RFC 791 [14], který pokrývá IP – *Internet Protocol*.

Tento útok zneužívá principů *fragmentace*, kdy velikost přenášených datových jednotek se mění mezi jednotlivými uzly přenosu a vrstvami spojení na základě předem určených parametrů užitých komunikačních protokolů, díky čemuž je možné odeslat škodlivý paket.

Při zaslání paketu, který je na straně útočníka pomocí pole *offset* IP hlavičky velikostně nastaven tak, že přesahuje maximální povolenou velikost IP paketu, tak dochází na straně oběti při zpětném skládání daného paketu do bufferu k *přetečení bufferu* (angl. *buffer overflow*), což může například vypnout, zmrazit nebo také

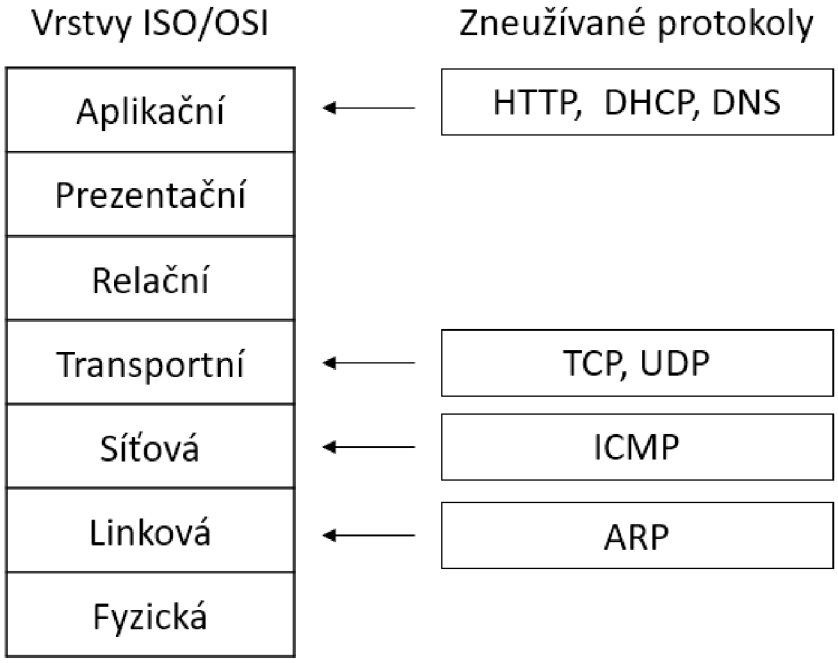
restartovat operační systém na straně oběti, čímž je dosaženo stavu odepření služeb.

Detekce útoků na hardware

Nejčastější forma obrany proti těmto útokům spočívá v již zmíněných IDS založených na signaturách, jejichž databáze signatur se musí udržovat aktuální za účelem udržení jejich efektivity. Jako příklad obrany proti zmíněnému *Pingu smrti* může být např. nastavena signatura v IDS analyzující pole offsetu v hlavičce IP paketu.

1.3.3 Protokoly různých vrstev

Na základě toho, jakého protokolu jaké vrstvy daný útok zneužívá lze DoS útoky také rozdělit. V modelu ISO/OSI [15] se nachází 7 vrstev, na kterých pracují různé protokoly pro zajištění vícestranné komunikace, ze kterých jsou některé typickými cíli DoS útoků.



Obr. 1.2: ISO/OSI protokoly nejčastěji zneužívané DoS útoky

Jak lze z obrázku vidět, DoS útoky je možné nalézt jak na vrstvách nižších, tak vyšších. Dá se říci, že útoky na nižší vrstvy jsou starší, rozšířenější a často i lépe detekovatelné, což ale nemusí vždy znamenat, že je proti nim snazší obrana. Jedná se zpravidla o útoky záplavové zneužívající protokolů čtvrté (TCP a UDP), třetí (ICMP) nebo druhé (ARP) vrstvy. Tyto útoky jsou nejčastěji záplavového typu *flood* a snaží se vyčerpat šířku pásma nebo existující nastavená omezení zmíněným

protokolů (např. maximální počet TCP spojení serveru, počet echo requestů, na které oběť zvládá odpovědět, nebo zahlcení prostředí sítě ARP dotazy).

Tyto útoky by se daly považovat za méně sofistikované oproti DoS útokům na vrstvy vyšší vzhledem k tomu, že jsou zpravidla jednodušší na realizaci, což je ale na druhou stranu činí mnohem rozšířenějšími, a proto není na místě je podceňovat při přípravě preventivních opatření vlastních serverů apod.

Tato práce se podrobněji zabývá i útoky na vyšší vrstvy, zejména tedy na vrstvu aplikační, kde funguje především velmi rozšířený protokol HTTP. V předchozích kapitolách práce byly ovšem příkladově uvedeny útoky nejen na protokol aplikační vrstvy HTTP, ale také na protokoly DNS nebo DHCP.

Jelikož se v praxi lze ale nejčastěji setkat se situací, kdy cílem útočníka není znepřístupnit služby DNS nebo DHCP serveru, nýbrž webové stránky poskytující služby patřící cílové oběti (ať už konkurenční společnosti nebo komukoliv jinému), tak nejrelevantnějším protokolem na aplikační vrstvě pro další analýzu v této práci je protokol HTTP a DoS útoky směřované na něj.

Útoky směřované na protokol HTTP (a protokoly aplikační vrstvy obecně) jsou často cílené na vyčerpání zdrojů operačního systému serveru oběti nebo konkrétní služby na něm běžící. Jelikož tyto útoky pracují na vyšších vrstvách, tak ochranné mechanismy (IDS a IPS) sledující provoz na vrstvách nižších nemusí postihovat útoky tohoto typu, což je činní o to více nebezpečnými.

1.4 Způsob spojení

Na základě toho, jaký protokol čtvrté vrstvy konkrétní DoS útok využívá, je lze rozlišovat na nespojované a spojované. Zpravidla se nejčastěji lze setkat s protokolem TCP v případě spojovaných a s protokolem UDP v případě nespojovaných.

V případě spojovaných útoku musí být mezi zařízením (zařízeními v případě DDoS) útočníka a zařízením oběti vytvořeno spolehlivé spojení, přes které následně probíhá provoz. Na základě toho, jaké vlastnosti a stádia tohoto spojení se konkrétní útok snaží dosáhnout stavu DoS u oběti, lze tyto útoky dále rozlišovat.

V případě nespojovaných útoku k vytvoření žádného takového spojení pro zasílání provozu mezi útočníkem a obětí nedochází, a proto se na základě tohoto parametru nedají dále rozlišovat.

1.4.1 Nespojované

Útoky využívající protokol UDP nejčastěji spočívají v zahlcení oběti – jsou záplavového typu (např. DNS nebo DHCP flood). V případě protokolu UDP a jeho přenosu nevzniká spolehlivé spojení mezi útočníkem a obětí, a využívá se obecně na přenášení

dat o menších velikostech (např. DNS žádosti a odpovědi), která nejsou spolehlivě doručena (zpravidla se nekontroluje jejich přijetí), tedy se jedná o nespolehlivý protokol – nezaručuje úspěšné doručení dat.

Tato nespolehlivost je ale například v případě protokolu DNS ošetřena na aplikační vrstvě, která dohlíží na to, že na základě DNS žádosti o získání IP adresy z doménového jména se dotazujícímu zařízení dostane odpovědi od DNS serveru.

Nejčastěji se v případě UDP záplavových útoků realizuje konkrétní útok tak, že je zasláno velké množství UDP datagramů na náhodné porty oběti, která následně na základě toho, jestli nějaká aplikace naslouchá nebo nenaslouchá na daném portu, musí zaslat odpovědi nebo paket ICMP *Destination Unreachable*, který značí to, že na portu žádná aplikace nenaslouchá. Zpracování a odesílání všech těchto paketů má za účel vytížit zařízení oběti a způsobit u ní stav odepření služeb.

Obrana proti nespojovaným DoS útokům

V případě toho, že daný DoS útok pracuje na principu vytížení oběti prostřednictvím ICMP zpráv, tak je možné nastavit na zařízení potenciální oběti maximální povolený počet odeslaných ICMP zpráv za čas, čímž se limituje to, jak moc se jejich prostřednictvím dané zařízení může vytížit.

Jako další způsob ochrany proti těmto útokům je vhodné vypnout/zakázat nepoužívané porty, čímž se datagramy mířené na ně přímo zahazují a není na ně potřeba odpovídat. V praxi IT bezpečnosti je obecně považováno za rozumné vypnutí portů, které by mohly být útočníkem zneužity (jako například port služby TLS), proto zakazování služeb/portů, které by se mohly projevit v budoucnu jako problematické, je obecně na místě.

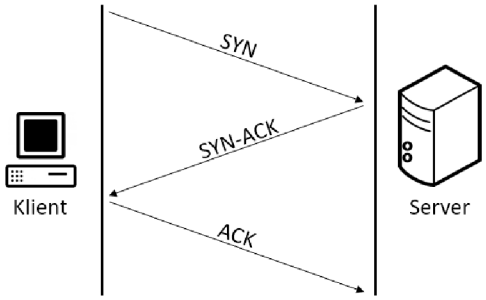
Ne všechny nespojované DoS útoky fungují na podobném principu, proto se nejedná o jednoznačné řešení, avšak určená politika spravování portů je každopádně pozitivní přínos pro zvýšení zabezpečení systému sítě.

1.4.2 Spojované

Tyto útoky zpravidla zneužívají protokolu TCP, který se vyznačuje svým mechanismem navázání spojení – tzv. *three-way handshake* (česky *tří-směrné podání ruky*), při kterém je navázáno TCP spojení, přes které je mezi útočníkem a obětí zasílán provoz a eventuálně je spojení ukončeno. Cílem útočníka může být kterékoliv z těchto tří stádií TCP spojení – navázání, průběh nebo ukončení, a podle toho také lze tyto útoky rozdělovat.

DoS útoky cílené na navázání TCP spojení

Mechanismus *three-way handshake* slouží k navázání TCP spojení a vypadá následovně:



Obr. 1.3: Navázání TCP spojení pomocí three-way handshake

Při normálním průběhu TCP spojení je prvně ze strany legitimního uživatele zaslán paket s příznakem SYN, následně server odpovídá paketem s příznaky SYN-ACK a nakonec uživatel odpovídá pouze příznakem ACK. Na základě tohoto procesu je navázáno TCP spojení a může probíhat spolehlivý přenos dat mezi uživatelem a serverem.

Typickým příkladem zneužití tohoto procesu je velice rozšířený DoS útok SYN flood, který byl již nastíněn. Při SYN flood v praxi útočník zasílá z mnoha podvržených IP adres na server oběti TCP pakety s příznakem SYN. Server následně odpovídá na všechny tyto žádosti o navázání spojení vlastními pakety s příznaky SYN-ACK a otevírá dané spojení (rezervuje jej pro danou IP adresu).

Každý server má však softwarově i hardwarově omezený počet TCP spojení, které může mít otevřené, naslouchat na nich a spravovat je. Proto když útočník vyčerpá počet těchto spojení, tak legitimnímu uživateli již nemůže být další otevřeno a dochází k odepření služeb.

DoS útoky cílené na průběh TCP spojení

Útoky předchozího typu měly za cíl navázat co nejvíce TCP spojení, avšak tyto útoky ne vždy fungují, jelikož servery potenciálních obětí mohou být relativně jednoduše (často i jsou) nastaveny na ukončení spojení po určité době, jestliže není dokončen *three-way handshake* paketem s příznakem ACK ze strany útočníka.

Jelikož ten často zdrojové IP adresy podvrhl a pouze z nich odeslal inicializační paket TCP spojení s příznakem SYN, tak spojení nebývá navázáno. Může se ale stát, že útočník navázání spojení dokončí a na řadu přichází zneužití již konkrétního průběhu TCP spojení.

Nejčastěji se lze setkat s útoky na průběh TCP spojení, které se snaží udržet co nejvíce TCP spojení obsazených za vynaložení co nejmenších zdrojů ze strany útočníka tak, aby na legitimního uživatele již nezbylo volné TCP spojení a byly mu odepřeny služby.

Čím více je mechanismus udržení TCP spojení sofistikovaný a nerozeznatelný od legitimního připojení, tím nebezpečnější je daný DoS útok ho využívající. Do této kategorie spadá většina DoS útoků, tzv. pomalých *Slow DoS* útoků, kterými se práce bude dále zabývat a také jejich principy a mechanismy, které užívají k udržení TCP spojení.

DoS útoky cílené na ukončení TCP spojení

Jestliže se DoS útok nesnaží vyčerpat kapacitu souběžných TCP spojení oběti k odepření služeb legitimním uživatelům, tak se může snažit zneužít mechanismů, které zasahují do již běžících TCP spojení legitimních uživatelů.

Příkladem mohou být již zmíněné záplavové DoS útoky *RST/FIN flood*, které se snaží zasíláním škodlivých paketů se stejnojmennými příznaky zasáhnout do již existujících legitimních spojení.

Často se ale stává, že dojde ke stavu DoS z důvodu, že server oběti nestíhá požadavky zpracovávat nebo se vyčerpá šířka pásma oběti, jelikož se jedná o příznaky, které používají i legitimní spojení, tudíž je nelze jednoduše zahazovat.

V případě těchto útoků může být obecně cílem server poskytující služby jako celek – útočník se náhodně snaží narušit všechna existující TCP spojení serveru, nebo pouze konkrétní spojení uživatele, které útočník chce narušit – útočník potřebuje informace o daném spojení k tomu, aby jej narušil (např. IP adresu uživatele), které získá např. prostřednictvím již zmíněného odposlechu provozu angl. *sniffingu*.

2 Příprava SlowDrop útoku

V této hlavní teoretické části se práce zabývá SlowDrop útokem, který byl navržen a zveřejněn v roce 2019 výzkumným týmem pod vedením *Enrico Cambiasa*[1], který okomentoval zveřejnění návrhu SlowDrop útoku takto:

> *Although the proposal of a novel threat may be unusual, we believe that knowing in advance offensive tools is fundamental in order to properly design efficient detection and mitigation systems. The proposed work should therefore not be considered as the release of a tool for cyber-criminals, but instead an essential resource for the network security world, providing researchers an important element to properly investigate the denial of service phenomenon.*

Ve zkratce (přeloženo do češtiny) se vyjadřuje o tom, že publikování návrhu tohoto útoku nemá sloužit jako návod pro realizaci kriminálních aktivit, nýbrž má být odborníky zajímající se informační bezpečností nabádat k další práci v této oblasti a případnému vývoji obrany proti útokům tohoto rázu.

V následujících kapitolách tato práce zařazuje SlowDrop útok do kategorií uvedených v předchozích částech na základě jeho vlastních specifik, věnuje se přirovnání a porovnání SlowDrop útoku s jeho předchůdci a inspiracemi z kategorie Slow DoS útoků (anglicky *Slow DoS Attacks* – SDA) a uvádí analýzu rozdílností mezi nimi – výhody a nevýhody oproti uvedeným útokům jsou taktéž uvedeny.

Zejména je v této teoretické části kladen důraz na specifika a vlastnosti SlowDrop útoku, které je potřeba v rámci práce podrobně popsat, pochopit a kriticky zanalyzovat, jelikož se jedná o myšlenkové jádro pro vytvoření funkčního modelu útoku a následně efektivní obrany proti němu. Jsou uvedeny hlavní myšlenky a principy, které za útokem SlowDrop stojí a které jej v teorii činí obzvláště nebezpečným a unikátním.

Velký důraz je kladen v hlavní části práce zejména jednotlivým parametrům útoku, které je potřeba vzájemně k sobě vztahovat a následně určovat jejich poměrové hodnoty. To, jak je model útoku SlowDrop úspěšný nebo neúspěšný v dosažení stavu DoS na straně oběti, je totiž přímo závislé na nastavení vhodných parametrů pro konkrétní situaci – typ oběti, způsob připojení atd.

V neposlední řadě jsou v této kapitole nastíněna omezení SlowDrop útoku vycházející z jeho principů a hlavních myšlenek. Ačkoliv se jedná o novou a teoreticky velmi potentní hrozbu v oblasti DoS útoků, tak se nejedná v žádném případě o všestranný nástroj využitelný za jakýchkoliv podmínek na jakýkoliv cíl oběti.

2.1 Kategorické zařazení SlowDrop útoku

Podle počtu útočících zařízení můžeme tento DoS útok zařadit mezi typ útoků, které spadají do skupiny útoků, u kterých není zpravidla potřeba je realizovat distribuovaně, avšak distribuovaná varianta je samozřejmě preferovaná za účelem zvýšení efektivity (jako u všech DoS útoků).

Pomalé DoS útoky se obecně značí tím, že na jejich realizaci není potřeba značné množství paketů, jelikož se snaží být co nejméně nápadné (vypadat jako běžný, legitimní provoz) a jedná se o útoky na aplikační vrstvu ISO/OSI, tudíž IDS sledující nižší vrstvy je nedetekují.

Pomalé DoS útoky neboli anglicky *Slow DoS attacks (SDA)* jsou povahově útoky zneužívající zranitelností, avšak mají určité vlastnosti i záplavových útoků – snaží se někdy i přetížit fronty serveru oběti až do stavu odepření služeb, kdy oběť již nemůže obsloužit více uživatelů a zahazuje jejich požadavky.

Přesto nejsou SDA obecně považovány za útoky záplavového typu, jelikož hlavními myšlenkami je jejich nenápadnost (skrytí mezi legitimní provoz) a nízké nároky na útočníka – tzv. filozofie útoku *low and slow*[3] (česky *pomalu a nenápadně*), proto se označují za *exploit based* útoky cílené na aplikační vrstvu, nejčastěji protokol HTTP. SlowDrop útok, ačkoliv dle jeho autorů spadá pod SDA, tak zneužívá mechanismů zejména nižších vrstev – TCP protokolu transportní vrstvy modelu ISO/OSI.

Cílem pomalých DoS útoků je obecně zneužívat různých implementací mechanismů časovačů *timeout* – nastavená doba, po kterou server čeká na určité akce, než ukončí vyřizování konkrétního požadavku. Udržení TCP spojení, které vyřizuje daný požadavek, se serverem oběti je klíčové pro dosažení stavu DoS.

Nemusí se však jednat o jediný zdroj oběti (cíl útoku), který se snaží konkrétní útok vyčerpat. V případě SlowDrop útoku se primárně cílí na vyčerpání počtu současných TCP spojení, které dokáže server oběti najednou obsluhovat, nejedná se však o jediný zdroj, které tento útok může potenciálně vyčerpat.

V neposlední řadě, je potřeba SlowDrop útok a SDA obecně zařadit mezi útoky spojované, tedy využívající slabin TCP protokolu a jeho vlastností, nikoliv UDP. V případě protokolu UDP se totiž nelze bavit o proměnných jako různé časovače a limitovaný počet soustavných spojení mezi útočníkem a obětí, které se snaží útočník udržovat a znemožnit tak navázání spojení (poskytnutí služeb) legitimním uživatelům.

2.2 SlowDrop oproti typickým SDA

V předchozím textu práce bylo uvedeno, že pomalé DoS útoky obecně zneužívají nastaveného časovače *timeout* na straně serveru oběti, který se snaží zneužít, specificky cílí na něj a pracují s ním jako s proměnnou pro svou realizaci.

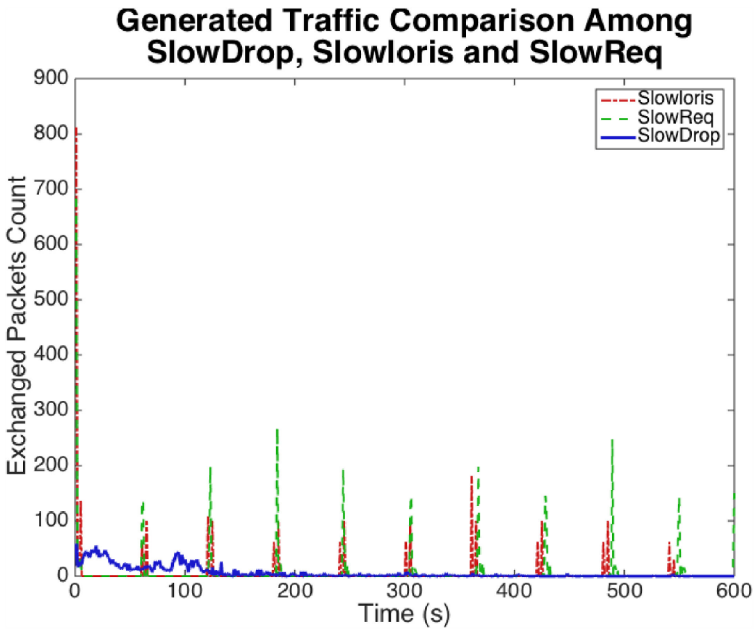
Přestože SlowDrop se řadí mezi SDA, tak pod tuto generalizaci doslovně nespadá, jelikož s touto proměnnou přímo v jeho parametrech přímo nepracuje a snaží se udržet TCP spojení i bez jeho znalosti.

Samozřejmě je vhodné pro útočníka nastavený časovač *timeout* znát, jelikož ten průběh SlowDrop útoku může přímo ovlivnit, avšak filozofie tohoto útoku nevychází ze znalosti této proměnné.

Jelikož SlowDrop oproti ostatním pomalým DoS útokům nepracuje s proměnnou *timeout*, tak není provoz tohoto útoku cílen na konkrétní časy v průběhu TCP spojení se serverem oběti. Průběhy typických SDA se totiž dají časově rozdělit podle toho, zda-li útočník průběžně zasílá škodlivé pakety útoku serveru oběti nebo ne.

Dělí se tak na dobu aktivity a neaktivity útočníka, kdy u většiny SDA je doba aktivity útočníka nejčastěji zaměřena právě okolo času *timeout*, přičemž po zbytek času bývá útočník neaktivní.

SlowDrop útok se oproti ostatním typickým SDA vyznačuje nepřetržitým provozem mezi útočníkem a obětí, tudíž útočník vykazuje neustálou průběžnou aktivitu pro udržení TCP spojení.



Obr. 2.1: Průběh SlowDrop útoku oproti jiným SDA[1]

[1]Obrázek je dostupný z [1] pod licencí Creative Commons BY-NC-ND 4.0

Na první pohled se tedy graf průběhu typického SDA výrazně liší od průběhu útoku SlowDrop, kdy v případě typického SDA lze jasně identifikovat tzv. *spike* (česky *špičku*) v provozu, které jsou útočníkem strategicky nastaveny podle znalosti časovače timeout na straně serveru tak, aby za co nejmenšího vynaloženého generovaného provozu od útočníka bylo TCP spojení udrženo aktivní.

V případě SlowDrop útoku tyto špičky v provozu neexistují, čímž se jeví jako více legitimní a lze jej hůře identifikovat jako škodlivý.

Dalším specifikem SlowDrop útoku je již dříve zmíněné zneužívání mechanismů nižších vrstev. Oproti typickým SDA, které zneužívají zranitelností protokolu HTTP protokolu, který pracuje na aplikační vrstvě.

SlowDrop pracuje s datovým provozem na straně útočníka na vrstvách nižších tak, že při průběhu TCP spojení po jeho úspěšném navázání již praktická část SlowDrop útoku spočívá v práci s daty přenosu na nižších vrstvách, tedy ještě před tím, než mají možnost se dostat na aplikační vrstvu útočníka a zde být potenciálně interpretovány aplikačním démonem útočníka, který provoz generuje.

Tento proces bude podrobněji rozveden v dalších částech práce, jelikož se jedná o samotný princip SlowDrop útoku.

2.3 Inspirace a předchůdci

Pro pochopení toho, čím byla idea SlowDrop útoku inspirována je potřeba teoreticky rozvést i další pomalé DoS útoky (SDA), které již existují a pracují v některých ohledech na stejných principech (udržení TCP spojení apod.).

Následující podkapitola se věnuje nastínění toho, jak jednotliví SDA předchůdci SlowDrop útoku pracují a jak se snaží dosáhnout stavu DoS na straně oběti. Na konci podkapitoly je následné shrnutí poznatků z porovnání těchto SDA oproti SlowDrop útoku, jejich podobností, výhod a nevýhod.

2.3.1 Slow GET – Slowloris

Jeden z nejznámějších pomalých DoS útoků díky jeho používání hacktivisty (kombinace slov hacking a aktivista) ze skupiny Anonymous se vyznačuje zneužíváním podvržených HTTP GET požadavků. Jedná se tedy o útok zneužívající slabin protokolu aplikační vrstvy – HTTP.

Útok byl navržen Robertem Hansenem[16] a při jeho prvním uvedení byl nazván jako Slowloris (česky outloň – druh noční poloopice význačné svým *pomalým* chováním).

Po navázání TCP spojení se útočník snaží tato spojení udržet (obsadit) po co nejdelší dobu tak, že zasílá nevalidně ukončené HTTP GET požadavky – hlavička

tohoto požadavku není řádně ukončena (typicky je místo posledních dvou řádků ukončených znaky `\r\n\r\n` toto ukončení obsaženo pouze jednou `\r\n`).

Běžná odpověď webového serveru na toto chování je započetí časového limitu (jistý druh zmíněného parametru časovače), během kterého server čeká na dokončení tohoto nekompletního HTTP GET požadavku. Těsně před vypršením tohoto času zašle útočník tzv. *keep-alive* zprávu, která toto spojení udržuje navázané a zmíněný časovač resetuje.

Tento proces zasílání *keep-alive* (česky *drž-naživu*) zpráv, které často obsahují náhodně vygenerované nesmyslné řetězce jako např. `X-a: b\r\n` nenesoucí žádnou informaci a pouze prodlužující původní HTTP GET požadavek, může trvat při nesprávném nastavení serveru oběti teoreticky až do nekonečna.

Těmito nedokončenými HTTP GET požadavky jsou postupně obsazena veškerá TCP spojení a dochází ke stavu odepření služeb na straně oběti.

2.3.2 Slow POST – R.U.D.Y

Jak může na první pohled z názvu útoku vyplývat, tento SDA se vyznačuje zneužíváním HTTP POST požadavků. Pracuje tedy stejně jako předchozí uvedený pomalý DoS útok na principech zneužívajících HTTP protokol pracující na aplikační vrstvě.

Tento útok je také známý jako nástroj, který ho realizuje, pod zkratkou R.U.D.Y. – *R-U-Dead-Yet*, tedy „*Are you dead yet?*“ neboli česky „*Už jsi mrtvý?*“. Jedná se zcela jistě o otázku útočníka směřovanou na server oběti.

Útok spočívá v principu na úplně stejné bázi jako útok Slow GET – Slowloris, pouze využívá jiného požadavku HTTP protokolu. Avšak právě tento rozdíl, kdy tento útok využívá využívá požadavků typu POST oproti předchozímu útoku, který využíval požadavky GET, klade při bližší analýze jistá omezení.

Tento útok může pracovat pouze se servery, které umožňují přijímání HTTP POST požadavků – dovolují zasílání dat na server. Tento útok tedy nejčastěji zneužívá různých typů formulářů (nejčastěji realizovaných pomocí *php*), vyhledávacího pole, vkládání souborů apod.

Tento požadavek na umožnění zasílání HTTP požadavků typu POST, ač v praxi útočníka pravděpodobně neomezí svojí nepřítomností, klade teoretickou překážku k provedení tohoto útoku.

Další odlišností od útoku předchozího je kompletnost HTTP hlavičky, kdy v případě tohoto útoku je hlavička řádně ukončena zmíněným dvojitým odřádkováním. Princip tohoto útoku nevychází z toho, že útočník neúplně dokončí hlavičku a server čeká na to, co vlastně útočník specifikuje, že chce serveru zaslat pomocí požadavku POST.

Prodlužování splnění tohoto požadavku vychází z toho, že útočník v legitimně ukončené hlavičce POST požadavku určí pomocí pole `Content-length`, které je nastaveno na nepřiměřené vysoké hodnoty (např. v řádech stovek milionů), to, že bude oběti zasílat obrovské množství dat.

Po prvotním odeslání tohoto HTTP POST požadavku server opět startuje časovač (*timeout*), po jehož dobu čeká na příjem dat, která jsou předmětem daného požadavku. Podoba *keep-alive* zprávy v případě tohoto útoku bude mít nejčastěji podobu náhodně vygenerovaného shluku dat nebo třeba i jednoho symbolu.

Po zaslání těchto dat je časovač stejně jako v případě předchozího útoku resetován, spojení TCP udrženo (obsazeno) a útočník se přibližuje k dosažení stavu DoS na straně oběti.

2.3.3 Apache Range Header

V případě tohoto útoku zasílá útočník škodlivé HTTP požadavky, která žádají data od serveru ve specifických rozsazích. Server oběti následně musí alokovat paměti v odesílacích bufferech pro každý z vyžádaných rozsahů, čímž dochází k vyčerpání zdrojů na straně serveru.

Tomuto útoku ze skupina pomalých DoS útoků tedy primárně nejde o obsazení TCP spojení, jako tomu bylo u předchozích případů.

Útok Apache Range Header zneužívá toho, že HTTP požadavky musí být všechny vyřizovány sekvenčně tak, jak jsou uvedeny v hlavičce a to samostatně, přestože třeba obsahují převážně stejná data – např. bude vyžádána sekvence specifických n bajtů a následně ve stejné požadavku bude vyžádána stejná sekvence n rozšířená o jeden bajt, tedy $n + 1$. Server musí alokovat paměť pro odeslání obou těchto bajtových sekvencí zvlášť.

Útok byl rozšířeně zveřejněný v podobě Perl skriptu s názvem *killapache.pl* [17], který generoval žádost obsahující bajtové sekvence, přičemž některé z nich ani nemusí být validně definovány, avšak strana serveru se jimi přesto musí zabývat a vynakládat na jejich vyhodnocení své výpočetní síly.

Tento útok zejména porušuje specifikace RFC normy číslo 2616 [18], která se zaměřuje na protokol HTTP 1.1 a říká: „*If the last-byte-pos value is present, it MUST be greater than or equal to the first-byte-pos in that byte-range-spec, or the byte-range-spec is syntactically invalid.*“. Česky zkráceně: „*Jestliže je specifikována poslední pozice bytové sekvence, tak musí být rovna nebo vyšší než pozice první, jinak není sekvence validní.* “

Kromě toho, že se tento útok snaží vytížit zdroje serveru vyžádáním specifických bajtových sekvencí a vytíženým odesílacích bufferů, tak se snaží v zmíněné skriptové implementaci také vytížit procesor serveru vyžádáním komprese zasílaných sekvencí

ve formátu `gzip`. Útok se tedy snaží vytížit vícero zdrojů serveru oběti zároveň za účelem dosažení stavu DoS.

2.3.4 Slow Read

Dalším SDA je útok, který nezneužívá přímo zranitelností HTTP protokolu, nýbrž vlastností TCP spojení – nepracuje tedy se zranitelnostmi aplikační vrstvy ale transportní. Princip útoku spočívá na straně útočníka v záměrném zmenšování tzv. *TCP okna* (angl. *TCP window*), které určuje, kolik bajtů může server útočníkovi odesílat bez toho, aniž by musel čekat na potvrzení přijetí.

Útok probíhá tak, že útočník naváže TCP spojení se serverem oběti, zažádá si o zaslání určitých dat např. prostřednictvím požadavku HTTP GET a následně se snaží toto TCP spojení, přes které jsou data posílána, udržet (obsadit) po co nejdelší dobu manipulací se zmíněným TCP oknem, které může dosáhnout až nulové velikost (což je sice podezřelé, ale v legitimní situaci může znamenat, že klient je zaneprázdněn, právě čte již zaslaná data a okno se za chvíli může zvětšit), tudíž server další data neodesílá, avšak spojení je stálo udržováno.

Velikost TCP okna může být ze strany útočníka (i serveru oběti) totiž upravována i za průběhu odesílání dat, čímž může útočník zmenšovat objem dat, který mu má server zasílat a následně čekat na zprávy o jejich přijetí.

Tento útok simuluje situaci, kdy legitimní klient je z nějakého důvodu (např. špatného připojení nebo nedostatečně výpočetní síly) omezený v tom, jak rychle jeho zařízení zvládá zpracovávat příchozí data od serveru, tudíž může být složité tento útok rozlišit od legitimních připojení.

V případě realizace tohoto útoku je dobré znát velikost odesílacího bufferu serveru oběti a žádat o data přesahující jeho velikost, aby se vyžádaná data nevlezla do paměti socketu, přes který jsou data odesílána, aby těchto odesílání (a spojených dotazování na obsazenost socketu[2]) mohlo proběhnout více a útok mohl být realizován.

2.3.5 Slow Next

Tento pomalý DoS útok je stejného původu jako útok SlowDrop, kterým se tato práce primárně zabývá – pochází od stejného vědeckého týmu pod vedením *Enrico Cambiasa* a byl zveřejněn v roce 2015 [19].

Princip tohoto útoku je tedy stejně koncipován na myšlence napodobení legitimního uživatele. Konkrétně tento útok zneužívá toho, že TCP spojení je udržováno po

[2]Tohoto principu se zneužívá ve více druzích DoS útoků, které využívají metodu vytížení socketů tzv. Sockstress TODO Zdroj

splnění konkrétní žádosti ještě určitou dobu (*timeout*), po které je ukončeno, pokud neproběhne další žádost ze strany klienta. Tento útok se tedy řadí také mezi útoky, které zneužívají určitých časových nastavení serveru a jeho TCP spojení.

Útok probíhá tak, že útočník legitimně naváže TCP spojení a následně zašle legitimní HTTP požadavek, na který je mu následně serverem oběti odpovězeno. Následně útočník zneužívá časovače tzv. *Wait timeout*, čeká a před tím, než vyprší, tak zašle další legitimní požadavek. Principiálně tento útok nelze rozeznat od legitimního klienta, který pouze čeká mezi zasíláním legitimních požadavků na konkrétním TCP spojení.

Jelikož je chování tohoto útoku prakticky zaměnitelné s chováním legitimního klienta, který náhodně využívá navázané spojení s obětí, tak ho lze rozeznat jen na základě toho, jak je implementováno zneužití konkrétního *timeout* parametru časovače, tedy jestli je např. útok nedostatečně sofistikovaně navržen tak, že zašle nový legitimní požadavek vždy před vypršením *timeout* ve stejný čas na rozdíl od legitimního klienta, který by vnesl do tohoto chování jistou míru náhodnosti.

Tento útok se tedy jako většina předchozích obsadit veškerá TCP spojení se serverem k dosažení stavu DoS, při kterém server již nemůže navázat spojení s legitimními zájemci o poskytnutí služeb.

2.4 Základní myšlenky

SlowDrop útok se soustředí na zneužití protokolů, které využívají pro přenos na transportní vrstvě spolehlivý TCP protokol. Útočník realizující SlowDrop simuluje klienta, který je ke službám serveru oběti připojen přes nekvalitní a nespolehlivé připojení, jako může být např. špatné bezdrátové připojení.

Takovému klientovi nedorazí veškeré vyžádané pakety od serveru – ztratí se nebo se nad opravitelnou míru poškodí z důvodů špatného připojení, což následně nutí tento server zasílat nedoručené pakety znovu, dokud požadavek klienta není splněn.

V případě SlowDrop útoku je ale tento stav pouze simulován, útočník je zpravidla připojen k serveru prostřednictvím dostatečně kvalitního spojení a pouze simuluje jeho nekvalitnost tím, že příchozí pakety zahazuje (anglicky *drop*), na základě čehož je útok pojmenován.

Jelikož tento koncept napodobuje legitimního uživatele tak, že lze jen těžko rozeznat ze strany oběti na základě průběhu útoku a analýzy provozu, jestli se jedná o útočníka nebo o legitimního uživatele pouze se špatným připojením, tak se jedná principiálně o útok velmi podobný výše uvedeným útokům Slow Read a Slow Next.

V literatuře se tyto útoky někdy označují za tzv. *meta SDA*[3] právě z toho povahového rysu, že je až prakticky nemožné tyto útoky rozlišit od legitimních uživatelů

na základě analýzy provozu jak na aplikační, tak na síťové i transportní vrstvě, kde obrané mechanismy (firewall, IDS a IPS) jsou nejčastěji nasazeny.

V praxi se tedy může stát, že při obraně před těmito útoky bude spojení omylem zrušeno i legitimním uživatelům služeb oběti, což je cílem útočníka.

Koncepčně je tento útok „ošemetný“ (pro absenci lépe výstižného slova), ale o to sofistikovanější, v tom ohledu, že je potřeba udržet TCP spojení s obětí a zároveň vynaložit co nejmenší množství zdrojů ze strany útočníka.

Jelikož ten ale musí generovat průběžný provoz, jak bylo dříve zmíněno, tak odhad toho, jaký je co nejmenší potřebný provoz pro udržení TCP spojení a dosažení stavu DoS na straně oběti, je poměrně složitý vzhledem k množství relevantních proměnných (požadovaná doba DoS stavu, míra zahazování, šířka pásma a další) a konkrétním nastavením na straně serveru oběti (zejména časovače *timeout*).

Jak již bylo dříve naznačeno, vyčerpání počtu TCP spojení serveru není jediným možným cílem SlowDrop útoku. Jedním z myšlenek SlowDrop útoku je také vyčerpání síťových a hardwarových zdrojů na straně oběti tím, že předmětem žádosti ze strany útočníka bude vyžádání zaslání dat většího objemu, jejichž odesílání může tyto zdroje vyčerpat při neoptimálním nastavení serveru.

V případě, že útočník v rámci TCP spojení žádá oběť o zaslání velkého objemu dat a ta se neustále snaží o vyhovění této žádosti odesíláním daných dat, tak může DoS stav nastat i v souvislosti s vyčerpáním kapacity odesílacích bufferů na straně oběti.

V tomto ohledu SlowDrop využívá i principů podobných dříve uvedenému Apache Range Header útoku, avšak staví se k nim jinak.

2.4.1 Zahazování

Ačkoliv zahazování paketů na straně útočníka probíhá na nižší – transportní vrstvě, tak se nejedná o DoS útok pouze na nižší vrstvy.

Cílem útoku je naslouchající aplikační démon služby na straně oběti, jedná se tedy v tomto ohledu o typický pomalý DoS útok, jelikož předmětem zahazování na straně útočníka jsou pouze pakety obsahující pro útok relevantní data – data pocházející ze služeb aplikační vrstvy, nejčastěji tedy protokolu HTTP.

To, jak konkrétně bude zahazování paketů na straně útočníka probíhat, je klíčovým aspektem pro správné nastavení SlowDrop útoku.

Jelikož se pomalé DoS útoky vyznačují také svojí nenáročností na zařízení útočníka, tak je potřeba, aby byly pakety zahozeny ještě předtím, než mohou začít zaměstnávat zdroje na straně útočníka, tedy než mohou být interpretovány aplikační démonem.

Dvě možnosti realizace se nabízejí v tomto případě:

- Zahazování paketů před tím, než se mohou dostat k cílovému zařízení (útočníkovi).
- Zahazování paketů na zařízení útočníka před tím, než jejich data mohou být interpretovány aplikačním démonem.

Oba tyto přístupy jsou možné a prakticky realizovatelné. Otázkou při výběru z těchto dvou možností je existence zařízení nebo nástroje, které by se staralo o zahazování paketů v první variantě, jako je například určitý druh proxy firewallu nebo jiného síťového filtru v podobě samostatného zařízení, přes který bude provoz útočníka filtrován.

Útok by v ohledu zahazování bylo možné nastavit „napevno“ tzv. *deterministicky* tak, že by bylo přesně určeno například kolik paketů za sebou má být na straně útočníka zahozeno, než bude na paket odpovězeno.

Tento přístup, ačkoliv zlepšuje přehled útočníka o tom, jak přesně útok probíhá, který paket byl a který nebyl zahozen, narušuje základní myšlenku simulace klienta, který komunikuje se serverem prostřednictvím nespolehlivého připojení.

Šlo by tedy teoreticky zahazovat pakety na základě jejich sekvenčních čísel v provozu, jednalo by se o realizaci útoku jednodušší a předvídatelnější, avšak na straně oběti by se detekčním/preventivním a obraným systémům mohlo zdát podezřelé, že konkrétnímu spojení je/není doručen každý n-tý paket.

Z tohoto důvodu je tento přístup k zahazování paketů pro realizaci SlowDrop útoku sice možný, avšak kontraproduktivní a autorem útoku nedoporučený. Z výše uvedených důvodů je tedy nutné určit to, jak budou pakety obsahující data aplikační vrstvy zahazovány, a to neurčitě, nedeterministicky.

Na straně útočníka je tedy potřeba nastavit zahazování paketů náhodně, avšak nad ním stále musí mít útočník takovou míru kontroly, aby toto náhodné zahazování nezpůsobilo ukončení TCP spojení a tedy uvolnění zdrojů, které se útočník snaží obsadit k dosažení stavu DoS.

Samozřejmě, že je ze strany útočníka možné sledovat, zda-li spojení bylo ukončeno a na základě toho reagovat např. započetím nového TCP spojení, avšak udržení již existujících spojení je lepší variantou jak z důvodů udržení konstantnosti útoku, tak snížení možnosti detekce ze strany oběti, která již může vyhodnotit opakovaná spojení jako podezřelá.

2.5 Parametry útoku

Jak již bylo dříve zmíněno, tak SlowDrop útok na rozdíl od typických SDA nepracuje se znalostí serverové proměnné *timeout*, což zavádí první aspekt neinformovanosti do tohoto útoku. Za druhý takový aspekt by se dal považovat hlavní princip zahazování

paketů na straně útočníka, který v konečném důsledku nemá přehled o tom, jaké pakety jsou zahazovány a tedy jak přesně SlowDrop útok probíhá.

Například u zmíněného útoku Slow GET – Slowloris má útočník pod kontrolou, že opravdu oběti zasílá nevalidně ukončené hlavičky požadavku typu HTTP GET. Jediným spolehlivým ukazatelem, zda-li SlowDrop útok v praxi probíhá podle představ útočníka je tedy to, jestli dochází ke stavu odepření služeb na straně oběti a jsou udržována požadovaná TCP spojení.

Šlo by ovšem namítnout, že lze sledovat a případně logovat na straně útočníka to, zda-li jsou pakety úspěšně zahazovány (a ne nad přijatelnou míru v konkrétním TCP spojení), to ovšem zatěžuje zařízení útočníka a nevychází tedy z principů SDA.

V rámci této části se práce bude věnovat parametrům[1], které má útočník pod kontrolou a může je nastavit tak, aby bylo stavu DoS dosaženo pokud možno po co nejdelší dobu trvání a za co nejmenšího vynaložení zdrojů ze strany útočníka.

Tyto parametry se ovšem mění podle konkrétních situací – softwarových a hardwarových dispozic útočníka i oběti, způsobu/vlastností jejich vzájemného připojení a také neznámého časovače *timeout* na straně oběti.

2.5.1 Poměr zahazování

Ústředním parametrem, který útočník musí nastavit, je poměr příchozích paketů od oběti. Jelikož dříve bylo určeno, že pakety není vhodné prakticky zahazovat na základě např. sekvenčních čísel, tak budou pakety zahazovány náhodně, přičemž útočník určí, jakou šanci má určitý paket na to, že bude zahozen.

Poměr zahazování bude nadále pro zlepšení přehlednosti označován proměnnou D (jako *drop rate* – poměr zahazování). Tato proměnná se bude pohybovat na škále od 0 do 1, kdy $D = 0$ znamená, že záměrně nebude zahozen žádný příchozí paket od oběti – dokonalé připojení, a $D = 1$ reprezentuje situaci, kdy jsou záměrně zahazovány všechny příchozí pakety od oběti.

Je zřejmé, že ani jeden extrém není reálnou možností v provedení SlowDrop útoku, jelikož $D = 0$ nezahazuje pakety vůbec a $D = 1$ bude vést k ukončení spojení ze strany oběti, jelikož útočník nebude vyvíjet na základě nedoručených paketů žádnou aktivitu.

Z definice poměru zahazování můžeme logicky usoudit, že čím vyšší hodnota parametru D je za podmínek, že TCP spojení není ukončeno ze strany serveru vzhledem k neznámé proměnné *timeout*, tím reálně méně náročné je tento útok na zařízení útočníka a tím delší dobu bude dané TCP spojení trvat, tedy o to déle bude útočníkem obsazováno spojení, které by za normálních podmínek mohl užít legitimní uživatel, jemuž jsou služby serveru odepřeny.

Každý server je však nastaven jinak v ohledu tolerance toho, kolik potvrzení přijetí paketů na straně útočníka vyžaduje, aby neukončil dané spojení, přičemž na straně útočníka je v reálné situaci nepraktické sledovat to, zda-li server spojení ukončil (což může i určitý čas trvat) a následně navazovat spojení nové, jak bylo zmíněno dříve.

Proto je vhodné vybírat parametr D pečlivě a spíše začít na nižších hodnotách a postupně ho zvyšovat, ačkoliv to může být více technicky náročné na zařízení útočníka, než začít s vysokou mírou zahazování a být detekován.

2.5.2 Doba udržení TCP spojení

Další přímo závislou proměnnou je doba, po kterou chce útočník udržet aktivní TCP spojení a zamezit tak legitimním uživatelům k připojení k oběti. Jinými slovy se jedná o požadovanou délku trvání stavu DoS ze strany útočníka, který se nejčastěji bude snažit o to, aby toto trvání dosahovalo co nejvyšších časových hodnot.

Tato proměnná se v rámci této práce bude označovat jako T a bude určována v *sekundách*.

2.5.3 Požadovaná data

Jelikož obsazení a vyčerpání všech možných TCP spojení není jediným možným cílem SlowDrop útoku, tak je potřeba se zabývat i velikostí dat, o která si útočník od oběti žádá. Čím větší objem těchto dat je, tím větší má SlowDrop útok potenciální šanci k dosažení stavu DoS u oběti.

V první řadě se od velikosti objemu vyžádaných dat odvíjí to, jak moc budou vytíženy odesílací buffery/vyrovnávací paměti služby na straně oběti, tedy zdali budou zvládat odesílat takové množství dat vícero různým klientům.

Za druhé, čím větší objem dat bude muset oběť útočníkovi odesílat, tím logicky déle bude přenos těchto dat trvat, což je v případě SlowDrop útoku ještě záměrně prodlouženo jejich zahazováním. Toto částečné zahazování je umožněno, jelikož data nejsou odesílána jednorázově, ale jsou patřičně fragmentována pro přenos přes různé komunikační vrstvy prostřednictví jejich protokolů.

Požadovaná data budou označována R (jako *requested* – požadovaná) a budou určována v *bajtech* (B).

V této souvislosti je také potřebné uvést proměnné, které budou označovat data, která na základě požadovaných dat bude oběť ve skutečnosti odesílat útočníkovi. Jelikož potřebujeme pracovat zejména s proměnnými, ke kterým můžeme vztahovat poměr zahazování, tak chceme vyjádřit data odesílaná směrem k útočníkovi v podobě paketů, jež se jím budou zahazovat, ale i ve velikosti těchto paketů.

Proto data odesílaná ze strany oběti směrem k útočníkovi v případě, že je bude potřeba vztáhnout k ostatním proměnným jakožto jednotlivé pakety, budou označována proměnnou P (jako pakety), kdy jejich množství lze vyjádřit jejich počtem.

Také budu pakety označovány proměnnou S (jako *sent* – odeslané), když velikost jednotlivých paketů bude potřeba vyjádřit v jednotkách *bajtů* (B).

2.5.4 Šířka pásma útočníka

Posledním relevantním parametrem pro útok SlowDrop, nad kterým má útočník kontrolu je šířka pásma, které věnuje přijímání paketů od oběti, jinými slovy které aktivně vynaloží na realizaci SlowDrop útoku.

Již z povahy pomalých DoS útoků je zřejmé, že se útočník bude tuto proměnnou snažit minimalizovat, jelikož SDA mají být na zařízení útočníka nenáročná a nemají ho, a tedy i jeho síťové připojení, vytěžovat. Tento parametr bude práce označovat proměnnou B (jako *bandwidth* – šířka pásma) a bude jej měřit v *bajtech za sekundu* (B/s)

2.5.5 Vztahy jednotlivých parametrů a proměnných

Následující část práce se bude věnovat konkrétním vztahům mezi jednotlivými proměnnými parametrů útoku[1], jelikož jejich pochopení je klíčové pro úspěšnou realizaci útoku.

$$P_{real} = \frac{P_{ideal}}{1 - D} \tag{2.1}$$

Rovnice 2.1 udává množství paketů, které musí být reálně odeslány útočníkovi ze strany serveru na splnění jeho požadavku – P_{real}. Jako proměnné vstupují do této rovnice P_{ideal} – počet paketů, který by byl potřeba k splnění požadavku za ideálních bezeztrátových přenosových podmínek ($D = 0$), a samotný parametr D, který určuje poměr zahazování, přičemž se bude pohybovat v rozmezí $0 < D < 1$.

$$Q = \frac{B}{R} \tag{2.2}$$

Druhá rovnice 2.2 se zabývá specifiky nastavení jednoho konkrétního spojení, které je potřeba zohlednit při navazování více spojení. Parametr Q vyjadřuje počet paketů za sekundu, které může konkrétní spojení pojmout vzhledem k šířce pásma určené danému škodlivému spojení – B a vyžádanému objemu od serveru oběti – R.

$$T = \frac{P_{real}}{Q} = \frac{P_{ideal} R}{B(1 - D)} \tag{2.3}$$

Poslední rovnice 2.3 vyjadřuje jedny z nejdůležitějších údajů pro útočníka, tedy potřebná nastavení proměnných pro dosažení DoS stavu na straně oběti po dobu T. Jedná se o kombinaci předchozích dvou vztahu, které spolu dohromady dávají tuto konečnou rovnici, do které všechny vstupují svými vlastními parametry.

Vidíme tedy, že čas T lze vyjádřit jako poměr reálného množství přenesených paketů P_{real} a proměnné Q vyjadřující počet paketů za sekundu procházejících jedním konkrétním škodlivým TCP spojením. Také lze tento čas vyjádřit pomocí komplexnějšího vztahu poměru mezi proměnami $P_{ideal}R$ a $B(1-D)$, které byly již rozvedeny.

2.6 Omezení SlowDrop útoku

Z výše uvedených kapitol práce je po zamyšlení zřejmé, že se nejedná o útok univerzální, aplikovatelný za jakýchkoliv okolností a podmínek. Naopak jeho úspěšnost a optimalizace závisí na velmi specifickém nastavení a zohlednění zmíněných proměnných. Principiálně však tento útok klade určitá omezení, která nelze překonat ani správným nastavením těchto parametrů.

TCP spojení

Tento útok lze použít pouze na protokoly aplikační vrstvy (zpravidla HTTP), jež užívají pro svůj přenos na transportní vrstvě spolehlivého spojovaného TCP protokolu. Jelikož tento útok vychází ze zneužití povahy tohoto protokolu, kdy se oběť snaží vyhovět požadavku útočníka prostřednictvím uzavření TCP spojení, nemůže být aplikován zejména na nespolehlivý a nespojovaný protokol UDP.

V případě zahazování provozu UDP protokolu by se totiž oběť nesnažila tento nedostatek svými mechanismy napravit. Toto omezení na TCP protokol je ovšem charakteristické pro valnou většinu pomalých DoS útoků, jelikož tyto útoky daného protokolu zneužívají.

Požadovaná data

To, jaký je největší možný požadovaný objem dat (např. soubor obrázku nebo videa) ze strany útočníka od serveru oběti je významným limitem SlowDrop útoku. Od této proměnné se totiž přímo odvíjí to, zda-li vůbec může být SlowDrop útok realizován a případně na jak efektivně dlouho může být obsazeno konkrétní TCP spojení – dosaženo na straně oběti stavu DoS.

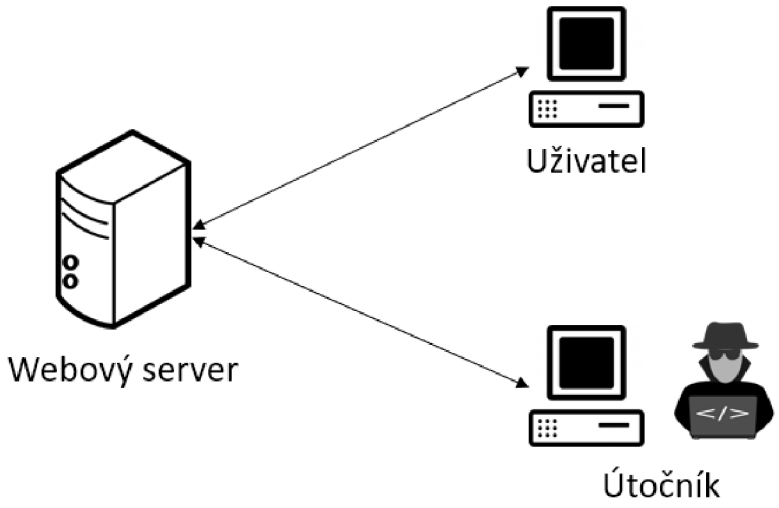
Jako usnadnění řešení tohoto problému existují nástroje dostupné na straně útočníka, jež zjistí data obsažená na straně serveru (včetně adresářového rozložení atd.), o která může útočník případně žádat. Jako příklad může sloužit např. penetrační nástroj DirBuster[20], který dokáže zobrazit i skryté složky a jejich obsah.

3 Model SlowDrop útoku

Tato část práce se zabývá praktickou realizací funkčního modelu útoku SlowDrop a jeho následným aplikováním v testovacím prostředí. V rámci praktické části bylo vyzkoušeno více možných způsobů realizace tohoto útoku, jejichž implementace jsou zahrnuty do textu této práce. Hlavní podmínkou pro výběr těchto způsobů realizace je jejich možná reálná implementace na straně útočníka splňující dříve zmíněné požadavky kladené na SlowDrop útok jakožto pomalý DoS útok.

3.1 Příprava testovacího prostředí

Topologie sítě testovacího prostředí se skládá ze stanice útočníka, stanice uživatele (variabilní) a webového serveru oběti.



Obr. 3.1: Topologie sítě testovacího prostředí

Bohužel se valná většina testování modelu SlowDrop v rámci této práce realizovala v domácím prostředí, které bylo omezeno pouze na jeden stolní počítač, tudíž je celé testovací prostředí pouze virtualizované na tomto hardware. Tento fakt zasahoval reálně pouze do časového omezení testování, kdy virtualizované stroje (zejména při testování průběhu útoku) často měly větší dobu odezvy nebo selhávaly. K těmto podmínkám byl autor práce donucen pandemií COVID-19, která autorovi neumožnila využít laboratorního prostředí fakulty.

Na druhou stranu toto domácí virtualizované prostředí vytvořené na míru mělo výhodu kompletní kontroly nad parametry virtualizovaných strojů. Za zmínku stojí také uvést to, že pomalé DoS útoky z jejich povahy nevyžadují na straně útočníka

vysoké hardwarové nároky, avšak pro serverovou část se projevilo náročné testování vyčerpání limitu TCP spojení.

Stolní počítač, na kterém je testovací prostředí virtualizované prostřednictvím programu VMware, disponuje 4-jádrovým procesorem i5-2400, 16 GB RAM a operačním systémem Windows 10.

Jakožto webový server oběti byl virtualizován OS Kali Linux 2019.3 s přidělenými 4 GB RAM. Na tomto virtualizovaném OS byl nainstalovaný Apache server verze 2.4.41 v jeho výchozím nastavení (které se v rámci testování adaptovalo), do jehož kořenových složek byly vloženy obrázky různých velikostí objemu dat za účelem poskytnutí většího objemu dat pro předmět HTTP GET požadavku pro SlowDrop útok. Na tomto operačním systému byl také nainstalován program Wireshark pro zlepšení analýzy provozu sítě.

Jako stroj útočníka je opět virtualizován OS Kali Linux 2019.3 s 4 GB RAM. Na tomto virtualizovaném stroji je opět nainstalován program Wireshark pro analýzu síťového provozu a program DirBuster pro zjištění obsahu webového serveru.

Stanice legitimního uživatele je realizována virtualizovaným OS Windows 7 nebo Ubuntu 18.04 s 4 GB RAM. Tento přístroj má variabilní OS na základě potřeb testování modelu útoku.

Kontrola testovacího prostředí

V rámci přípravy testovacího prostředí byl nainstalován na virtualizovaný OS útočníka Kali Linux nástroj pro generování pomalých DoS útoků pomocí příkazu `apt-get install slowhttptest` zvaný *SlowHTTPTest* [21], alternativou může být také např. [22]. Následně byl nástroj použit na ověřený správného zapojení a konfigurace prostředí tak, že byly otestovány zmíněné pomalé útoky (jako např. Slow Headers– Slowloris a Slow Read) na základě podkladů autora nástroje. Útoky na Apache server verze 2.4.41 ve výchozím nastavení probíhaly bez dedikovaného IDS/IPS podle očekávání (docházelo ke stavu DoS) a funkčnost testovacího prostředí tak byla ověřena.

3.2 Realizace modelu SlowDrop útoku

Jelikož realizace SlowDrop útoku je možné uskutečnit vícero způsoby, tak je důležité eliminovat na začátku ty implementace, které v praxi nebudou použitelné pro valnou většinu útočníků. Následující část práce se tedy bude zabývat výběrem ideálního řešení implementace SlowDrop útoku.

3.2.1 Před zahájením SlowDrop útoku

Jednou z důležitých prerekvizit realizace úspěšného SlowDrop útoku je přítomnost dat většího objemu na straně serveru oběti, o která může útočník žádat prostřednictvím HTTP GET požadavku.

Jestliže je útočníkovi předem znám např. soubor obrázku/videa, který je většího datového objemu a může o něj požádat prostřednictvím HTTP GET požadavku, tak se touto otázkou nemusí zabývat. Ne vždy však má útočník tyto informace, proto jsou v rámci této práce nastíněny dva přístupy, jak dále v praxi postupovat.

První možností útočníka je tento soubor většího datového objemu nalézt buď laickým zkoumáním webu v prohlížeči nebo použitím nástrojů jako dříve zmíněného DirBuster zjistit obsah webových stránek na serveru oběti včetně skrytých složek a souborů. Úspěšná délka trvání stavu DoS v případě SlowDrop útoku se přímo odvíjí od velikosti vyžádaného souboru, proto čím větší nalezený soubor, o které lze zažádat prostřednictvím požadavku HTTP GET, o to úspěšnější následný útok může být.

Druhou možností je nahrání vlastního souboru většího datového objemu na webový server oběti prostřednictvím HTTP POST požadavku a následné požádání (prostřednictvím HTTP GET požadavku) o jeho zaslání. Ne každý webový server však umožňuje např. vkládání souborů do jeho paměti, proto je vhodné opět provést zběžnou laickou analýzu prostředí webové stránky, jestli toto umožňuje.

O trochu pokročilejší přístup se může útočník pokusit otevřením vývojářské konzole prohlížeče prostřednictvím stisknutí klávesy F12 (pro většinu prohlížečů), kde lze zobrazit např. PHP formulář dostupný na webové stránce a formu toho, jaké požadavky typu POST přijímá. Nakonec i k tomuto účelu může sloužit program DirBuster, který může najít soubory s příponou *.php*, které mohou implikovat přítomnost zmíněných formulářů.

3.2.2 Průběh SlowDrop útoku

Princip SlowDrop útoku spočívá v zahazování příchozích paketů od serveru oběti v takové míře, že jsou udržována TCP spojení mezi útočníkem a daným serverem. Udržení těchto spojení by mělo být pro přístroj útočníka co nejméně náročné, zatímco server oběti by se měl stále co nejvíce snažit vynaložit svá TCP spojení (a sekundárně odesílací buffery socketů) na úspěšné splnění požadavků útočníka. Toto je možné pouze za dříve splnění zmíněných podmínek v předchozí podkapitole.

Pro zlepšení přehlednosti průběhu SlowDrop útoku budou následně rozepsány jeho jednotlivé kroky z pozice útočníka:

1. Navázání TCP spojení se serverem pomocí *three-way handshake*.
2. Zaslání HTTP GET požadavku o zaslání dat většího objemu.

3. Přijímání vyžádaných dat, přičemž poměrná část z nich je záměrně zahazována za účelem prodloužení doby, po která je nutné je zasílat, ale zároveň udržovat tak škodlivé TCP spojení.
4. V případě splnění HTTP GET požadavku (vyžádaná data byla i přes jejich zahazování úspěšně doručena) je možné opakovat útok od 2. kroku. Toto opakování je dle autora doporučeno nekonat okamžitě po dokončení předchozího požadavku, ale je na místě počkat za účelem zmenšení rizika detekce útoku. Při tomto čekání ovšem nemusí docházet ke stavu DoS na straně oběti.

Na první pohled je vidět, že kroky číslo 1 a 2 není nutné více podrobněji rozvádět, jelikož se jedná o předmět navázání standardního TCP spojení prostřednictvím *three-way handshake*, které již bylo rozvedeno v teoretické části, a následného zaslání HTTP GET požadavku serveru oběti, který není podvržený nebo upravený, tedy není nijak atypický od chování legitimního klienta.

Nicméně je na místě do jisté míry upravit hlavičku HTTP GET požadavku za účelem zlepšení efektivity útoku a snížení šance detekce.

3.2.3 Realizace zahazování u SlowDrop útoku

V rámci této části práce bude rozvedeno, jakými způsoby bylo zkoušeno uskutečnění realizace zahazování přijímaných paketů na straně útočníka.

V teoretické části práce bylo zmíněno, že existují dva různé způsoby, jak realizovat zahazování paketů před tím, než má aplikační démon na straně útočníka možnost je interpretovat (v zájmu co nejmenšího vytížení stroje útočníka), a to prostřednictvím dalšího zařízení, které bude ještě před příchodem do zařízení útočníka provoz filtrovat a zahazovat, nebo na nižších vrstvách stroje útočníka. Pro druhé řešení bylo rozhodnuto z následujících důvodů:

- V praxi je pohodlnější spravovat pouze jedno zařízení pro realizaci útoku, pokud hodláme realizovat útok nedistribuovaně.
- Pokud by byl útok podniknut v distribuované formě, tak je reálně velmi nepraktické umísťovat ke každému útočícímu zařízení další zařízení, které se bude věnovat pouze filtraci/zahazování provozu.
- Šiřitelnost konečného řešení a jeho implementace se násobně zjednoduší.

Bylo tedy rozhodnuto pro zahazování paketů až v prostřednictví stroje útočníka, a to prostřednictvím filtrace provozu na nižších vrstvách a jeho následného zahazování, tak aby nedošlo k vytížení aplikačního démonu na straně útočníka.

Linux Traffic Control

Tento nástroj je automaticky obsažen v systémech na bázi Debian Linux v balíčku *iproute*, který lze případně doinstalovat příkazem `apt-get install iproute` a využívá se k tzv. tvarování provozu.

Pomocí tohoto nástroje lze pro jednotlivá rozhraní upravit parametry jako odezvu, šířku pásma a hlavně ztrátovost – zahazování. Jelikož na straně je v případě SlowDrop útoku v zájmu upravovat především parametry šířky pásma B a poměr zahazování D, tak se může zdát tento nástroj ideálním kandidátem pro realizaci tohoto útoku.

Zde je ukázkový příkaz, který na rozhraní *eth0* nastaví zahazování 10 % neboli $D = 0.1$ vztaženo k proměnným z teoretické části:

```
tc qdisc add dev eth0 root netem loss 10 %
```

Následující ukázkový příkaz nastavuje na rozhraní *eth0* šířku pásma neboli v proměnných z teoretické části B na 1 Mb/s:

```
tc qdisc add dev eth0 root tbf rate 1mbit
```

Ač lze pomocí tohoto nástroje nastavit pro útočníka potřebné parametry poměrně ideálně, tak má poměrně zásadní nedostatek, a to ten, že lze pomocí něj ovlivnit pouze provoz vycházející z daného rozhraní (a také ne pouze socketu).

Ačkoliv byla funkčnost tohoto nástroje otestována v testovacím prostředí i na straně serveru a zahazování paketů bylo možné realizovat i pomocí tohoto nástroje, tak v praxi zahazování za účelem dosažení stavu DoS je možné provést pouze na straně útočníka.

Dle dokumentace tohoto nástroje je možné jej použít i na filtraci provozu příchozího na dané rozhraní, avšak není přímo optimální volbou oproti ostatním testovaným způsobům zahazování pro realizaci útoku SlowDrop zejména kvůli tomu, že potenciální informace o zahazování paketů předává do vyšších vrstev ISO/OSI, čímž informuje a zatěžuje aplikačního démona na straně útočníka.

iptables

Velmi populární softwarový nástroj na platformě Linux, se kterým se pracuje prostřednictvím příkazové řádky a umožňuje nastavování pravidel firewall Linux kernelu (jádra) – ten poskytuje framework *Netfilter*, který umožňuje práci se síťovými pravidly. Prostřednictvím příkazů `iptables` se manipuluje s tabulkami *Xtables*, které definují řetězce pravidel, podle kterých se nakládá s pakety odesílanými, příchozími a průchozími.

Pomocí pravidel *iptables* lze nastavit řetězce pravidel jako *filtr*, který nakládá různě s pakety směřujícími dovnitř – INPUT, ven – OUTPUT a pokud dané zařízení

slouží není zdrojem nebo cílen provozu, ale slouží pouze k přesměrování (jako např. router) – FORWARD.

Pro zahazování paketů z konkrétní IP adresy (např. 1.2.3.4.) vypadá příkaz pro použití *iptables* jako paketového filtru následovně:

```
iptables -A INPUT -s 1.2.3.4. -j DROP
```

Tento příkaz však zahazuje veškerý provoz pocházející z dané IP adresy – $D = 1$, a proto není vhodný pro realizaci SlowDrop útoku, při kterém je potřeba zahazovat pakety pouze v určitém poměru.

V pozdějších etapách vypracování této práce byly nalezeny *iptables* moduly s názvy *ipt_random* a *ipt_statistic*. Primární účel těchto modulů slouží k tzv. *loadbalancingu* – rozložení zátěže provozu mezi více zařízení (serverů), prostřednictvím např. přesměrování 50 % provozu na jeden nebo na druhý server.

Modul *ipt_random* však zavádí pro tento útok klíčovou vlastnost příkazům *iptables*, umožňuje jim aplikovatelnost pouze s určitou šancí. Lze tedy nastavit pomocí tohoto modulu pravidlo, které umožňuje např. zahazování paketů z určité zdrojové IP adresy (opět příkladově 1.2.3.4.) s určitou šancí (např. $D = 0,5$), že se pravidlo aplikuje.

```
iptables -A INPUT -s 1.2.3.4. -m statistic

--mode random --probability 0.5 -j DROP
```

Tento přístup k zahazování paketů byl v rámci práce ve finále vybrán a nejvíce testován, jelikož se zdá být jak nejpraktičtějším, tak nejvíce přehledným (vzhledem k uživatelskému prostředí *iptables*).

Obdobně pro tento přístup může být použito nástroje *nftables*, který je alternativou *iptables*, ovšem i tento nástroj má v tomto ohledu dokumentaci dostupnou až po extenzivním hledání, jelikož možnosti nástroje *nftables* umožňující primárně zmíněný *load-balancing*, nejsou dokumentovány nikde na jeho hlavních stránkách[1].

NFQUEUE

Přístup, který byl adaptován autory a původci útoku SlowDrop, tedy *Enrico Cambiasem* a jeho výzkumným týmem, je užití tzv. NFQUEUE jakožto cíle pro již zmíněné *iptables*. NFQUEUE je možný cíl *iptables*, který umožňuje příchozí pakety umístit do tzv. fronty, která může následně být spravována v uživatelském prostředí na aplikační vrstvě při propojení software s *libnetfilter_queue* k frontě poskytnuté z *iptables* (výchozí fronta je 0).

[1]Dostupných na adrese: https://netfilter.org/projects/nftables/

Paket se z paketového filtru *iptables* dostane do uživatelského prostředí prostřednictvím protokolu *nfnetlink*, kde v dané aplikaci následně jednotlivé pakety čekají ve frontě na aplikování pravidel.

Problém s implementací přímo navrženou původcem tohoto útoku byl pro autora v rámci této práce následující. V dané kapitole článku věnovaném realizaci útoku SlowDrop uvádí výzkumný tým Enrico Cambiasa volbu NFQUEUE jako vybraný způsob zahazování a citují při výběru tohoto způsobu výzkumný článek[23] zabývající se akcelerací *iptables* prostřednictvím NFQUEUE a použití paralelizace na procesorech grafických karet – GPU.

Rozhodovací rychlost při paralelizovaných procesech rozhodování týkajících se síťových pravidel aplikovaných na pakety v NFQUEUE je dle tohoto článku na GPU více než *43krát* vyšší než na běžném CPU při použití *iptables*.

Při pokusech o vlastní implementaci v rámci této práce pomocí programovací jazyku *python* pro vytvoření programu v uživatelském prostředí, který bude rozhodovat o paketech, které mu budou poskytnuty frontou prostřednictvím NFQUEUE knihovny *python-nfqueue*, se nedařilo implementovat tuto frontu správně.

Program napsaný v jazyce *python* dokázal zpracovávat frontu přeposlanou z *iptables* prostřednictvím cíle NFQUEUE, avšak pravděpodobně ne dostatečně efektivně a rychle, na základě čehož byla fronta neimplementována správně a další pakety do ní zaslané z *iptables* byly v lepším případě zahazovány (ne účelně pro realizaci SlowDrop útoku, ale nad útočníkem určenou míru D) a v horším případě způsobovaly selhání programu rozhodujícího nad frontou paketů.

Autor této práce se tedy rozhodl pro realizaci zahazování příchozích paketů přímo pomocí interakce s *iptables* ve spojení se zmíněnými moduly oproti navrženému přístupu autorů útoku.

K tomuto závěru ho vede, mimo vlastní zkušenosti s pokusy o realizaci zahazování paketů dle autorů útoku, zejména otázka toho, proč by v rámci SlowDrop útoku, jehož primárním účelem je nezatěžovat zařízení útočníka a zahazovat pakety na nižších vrstvách (nejlépe dokonce mimo zařízení útočníka), byly pakety z nižších vrstev přeposílány na úroveň aplikační vrstvy (ne aplikačnímu démonu, který je původcem HTTP GET požadavku), kde až následně jsou zahozeny.

Stejně tak tyto pakety mohou být zahozeny již v prostředí firewallu *iptables* místo jejich přeposílání do fronty NFQUEUE. Kromě vlastností větší kontroly a režie (které nejsou přímo vyžadovány) nad zahazovanými pakety je totiž tento navrhovaný přístup dle autora této práce kontraproduktivní.

3.3 Realizace modelu útoku SlowDrop

Tato podkapitola se věnuje praktické realizaci modelu útoku SlowDrop, použitým softwarovým komponentám a nastavitelné variabilitě modelů útoku.

3.3.1 Software modelu útoku SlowDrop

Jako programovací jazyk byl vzhledem k doporučení autorů útoku a zkušeností autora této práce zvolen python. Jelikož velká část práce byla věnována snaze o implementaci zpracováváním paketové fronty NFQUEUE pomocí knihovny *python-nfqueue*, tak i další programování modelů útoku v tomto programovacím jazyce pokračovalo.

Samotný útok je realizován formou skriptu, který vyžaduje interpreter python verze 3.8.x a při jeho spuštění zobrazí návod k jeho použití (lze zobrazit příznakem -h). Samotný skript je rozdělený do jednotlivých částí na základě fází útoku SlowDrop uvedených dříve:

1. Přípravná fáze útoku spočívá v realizaci zahazování (hlavní charakteristiky útoku SlowDrop) pomocí přidání směrovacích pravidel do *iptables* na základě argumentů vložených útočníkem.
2. Hlavní fáze útoku generování provozu útoku ve vláknech je realizována pomocí knihovny *requests*, která slouží ke generování a správě žádostí HTTP protokolu, a knihovny *threading*, která slouží k paralelizaci procesů v jazyce python a práci s vlákny.[2]
3. Konečná fáze útoku sleduje stav těchto žádostí a na základě volby útočníka může po jejich splnění generovat nové žádosti po nastaveném čase nebo ne. Splnění žádostí je nežádaný jev při realizaci útoku SlowDrop, proto je umožněno a doporučeno za takových okolností průběh útoku ukončit a generovat nový s výběrem nových parametrů.

Optimalizace software modelu útoku SlowDrop

Za účelem ztížení detekce byla do realizace útoku, tedy generování provozu, přidána nastavitelná hlavička HTTP žádosti.
Tato hlavička je přizpůsobena následovně:

- *User-Agent* je generován z náhodného seznamu, ve kterém je obsažena řada prohlížečů různých zařízení a operačních systému. Při výchozím generování žádostí by za jiných okolností bylo zobrazeno generování pomocí python skriptu.

[2]Bohužel python za pomoci této knihovny (ani jiné) aktuálně neumožňuje zavírání vláken s probíhajícími procesy, jelikož se jedná o typicky špatnou činnost z ohledu správy paměti, tudíž ukončování probíhajícího útoku generuje výjimky.

- *Cache-Control* parametr je nastaven na neukládání cache.
- V poli *Accept-Encoding* je specifikována prefence pro zvýšení fragmentace (za účelem zlepšení průběhu útoku SlowDrop) za použití komprese pomocí algoritmů *gzip, deflate*.
- V zájmu udržení spojení TCP co nejdéle byla nastavena specifikace pole *Connection* na hodnotu *keep-alive*.

Následně lze přizpůsobit model útoku SlowDrop na základě parametrů v různých ohledech potřeby vzhledem k podmínkám týkajících se jednotlivého scénáře útoku:

- Počet vláken generujících HTTP GET žádosti – v případě nebráněného serveru je možnost vyčerpat počet TCP spojení serveru oběti.
- Časový rozestup mezi generováním jednotlivých vláken (s jistou mírou náhodnosti) – při příliš rychlém generování se zvyšuje riziko detekce útoku.
- Časový interval, po který bude vlákno uspáno po splnění žádosti jím vygenerované, než zašle cílové oběti žádost novou – nízké hodnoty zvyšují šanci detekce útoku ale udržují TCP spojení obsazené.

V neposlední řadě byla do python skriptu přidána možnost manuálního odstranění přidaných *iptables* pravidel v případě, že útok bude přerušen nesprávnou cestou (např. počítač/virtuální stroj se zasekne a musí být ukončen).

4 Testování modelu útoku SlowDrop

Tato část práce se věnuje testování modelu SlowDrop útoku realizovaného prostřednictvím zahazování paketů metodami uvedenými v předchozí kapitole. Samotné testování v rámci této práce probíhá pouze ve virtualizovaném prostředí, které odpovídá drátovému propojení mezi koncovými zařízení – serverem oběti a strojem útočníka/uživatele. Parametry tohoto propojení se mění na základě potřeb testovaného modelu.

4.1 Testování modelu útoku jeho autory

Mimo již uvedené odlišnosti týkající se přístupu k zahazování paketů – klíčového faktoru realizace modelu útoku, se mezi testovacími podmínkami v případě autorů útoků a autora této práce nachází klíčové odlišnosti, které je potřeba zohlednit v modelové situaci vzhledem k jejich relevantnosti vůči šanci na dosažení stavu DoS prostřednictvím útoku SlowDrop.

4.1.1 Tři různé testovací scénáře

Autoři útoku specifikovali jejich testovací prostředí tak, že se jedná o dvě zařízení propojená různými typy připojení. Jedno zařízení generuje útok a (nejspíše) disponuje OS na bázi Linux, jelikož užívá prostředí *iptables*. Druhé zařízení běží na bázi OS Linux a je na něm webový server Apache2 verze 2.2 ve výchozím nastavení bez obraných modulů, který byl nakonfigurován pro každou modelovou situaci jinak.

Cílem autorů v každé modelové situaci bylo dosáhnout pomocí útoku SlowDrop stavu DoS trvajícího 600 sekund, tedy 10 minut. Vzhledem k tomuto požadavku na relativně dlouhodobé udržení DoS stavu na straně serveru oběti se rozhodli pro umístění souboru o velikosti 700MB[1].

Hlavním úkolem autorů práce bylo dle nich najít co nejvyšší hodnotu míry náhodného zahazování paketů, při které ještě spojení TCP nebude ukončeno předčasně ze strany serveru. V neposlední míře je potřeba zmínit, že autoři útoku také pracovali s výchozí hodnotou proměnné časovače *timeout* pro Apache server verze 2.2, tedy 300 sekundami – 5 minutami, což činní přesně polovinu požadované doby trvání DoS stavu. Proč je právě proměnná časovače *timeout* klíčová pro reálné úžití útoku SlowDrop bude rozvedeno v následujících kapitolách práce.

[1] V reálné situaci dle autora této práce relativně nepředstavitelná podmínka pro úspěšnou realizaci útoku.

Vysokorychlostní LAN

V tomto scénáři byla upravena konfigurace Apache serveru oběti tak, že dokáže obsluhovat souběžně 256 TCP spojení (maximum Apache serveru) oproti výchozímu nastavení 150 současných spojení.

Cílem tohoto útoku bylo obsadit veškerá dostupná spojení serveru a udržet je po zmíněnou požadovanou dobu při poměru zahazování $D = 0,9$, tedy s šancí na zahození příchozího paketu 90 %.

Výsledkem tohoto scénáře bylo dosažení DoS stavu po dobu 593 sekund, kdy bylo zrušeno jedno spojení z 256 vygenerovaných. Na celkové generování útoku bylo průměrně potřeba 652 Kbps šířky pásma – parametr B útoku.

V tomto scénáři se tedy zejména prokázalo to, že při udržení nízkých přenosových rychlostí (šířek pásma) jednotlivých spojení je tento útok velmi nenáročný na šířku pásma ze strany útočníka a umožňuje dosáhnout DoS stavu po požadovanou dobu.

Drátová připojení různých rychlostí

Druhý testovací scénář pracuje s výchozí konfigurací 150 současných TCP spojení serveru a třemi různými šířkami pásma mezi útočníkem a serverem oběti. Cílem těchto útoků bylo opět obsadit veškerá dostupná spojení serveru a udržet je po zmíněnou požadovanou dobu při poměru zahazování $D = 0,9$.

1. První šířka pásma simulující scénář lokální sítě LAN (zaměnitelné s předchozím scénářem) byla specifikována šířkou pásma $B = 925$Mbps. Stejně jako v předchozím scénáři bylo stavu DoS dosaženo po požadovanou dobu.
2. Druhým scénářem specifikovaným šířkou pásma $B = 93,8$ Mbps měla být dle autorů napodobena situace WAN sítě. Po uplynutí výchozího časovače *timeout*, tedy po 300 sekundách, bylo ukončeno ze strany serveru 110 spojení (přes 70 %) a postupně následovaly další – uvolňoval se stav DoS.
3. Ve třetím scénáři napodobujícím dle autorů podmínky náhodného internetového připojení byla nastavena šířka pásma na $B = 321$Kbps. Po vypršení časovače *timeout* byla ukončena veškerá spojení generovaného útoku a stav DoS byl ukončen.

Z výše uvedených provedených scénářů a jejich výsledku lze vyvodit, že:

- Časovač *timeout* na straně serveru bude proměnnou, která je pro SlowDrop určující k jeho úspěchu, pokud chceme jeho výkon maximálně optimalizovat.
- Zahazovací míra $D = 0,9$ je pro úspěšné dosažení stavu DoS za užití horších připojení s menší šířkou pásma příliš vysoká, ač je jakkoliv výhodná pro hardware útočníka ve smyslu charakteristik pomalých DoS útoků.

Bezdrátová připojení různých rychlostí

Třetí testovací scénář pracuje opět s výchozí konfigurací 150 současných TCP spojení serveru a třemi různými šířkami pásma mezi útočníkem a serverem oběti, avšak toto spojení je tomto scénáři realizováno bezdrátově. Cílem těchto útoků bylo opět obsadit veškerá dostupná spojení serveru a udržet je po zmíněnou požadovanou dobu při poměru zahazování $D = 0,9$.

Jelikož výsledky tohoto scénáře jsou velmi podobné výsledkům předchozího drátového připojení, pouze se liší rychlostmi, tak je na místě je akorát krátce shrnout a vyvodit závěry.

Scénář pracoval se třemi rychlostmi bezdrátového připojení – 60,4 Mbps, 23,3 Mbps a 1,19 Mbps, přičemž opět nejlepší připojení stačilo k udržení DoS stavu, střední variantě byla většina spojení ukončena po 300 sekundách a nejhoršímu připojení byla všechna spojení po uplynutí časovače ukončena.

Z provedení tohoto scénáře si lze vzít dle autora zejména to, že bezdrátové sítě jsou oproti jejich drátovému protějšku více nevyzpytatelné vzhledem k ztrátě dat, což je pro SlowDrop útok klíčovou charakteristikou, kterou je potřeba na straně útočníka mít v mezích pod kontrolou.

4.2 Testování modelu útoku v rámci této práce

Testování modelu útoku se v rámci této práce soustředilo primárně na schopnosti útoku udržet stálé TCP spojení k dosažení DoS stavu na straně oběti za pozměňování klíčové charakteristiky – zahazovacího poměru příchozích paketů, tedy parametru D.

Zatím co autoři útoku se více soustředili na zjišťování toho, co útok dokáže za různých podmínek týkajících se šířky pásma a přenosové rychlosti, tato práce se rozhodla jít jiným směrem.

4.2.1 Revize modelu útoku autorů

Vzhledem k tomu, že se jedná přímo o testování modelu útoku SlowDrop a jeho parametrů, tak si autoři útoku (dle názoru autora této práce nevhodně) vybrali jako jeho primární cíl při testování Apache server verze 2.2 s jeho výchozí hodnotou časovače *timeout* 300 sekund a zvolili si jako požadovanou dobu dosažení DoS stavu dvojnásobek této hodnoty.

Teoreticky vzato je tedy již od začátku garantovaná doba dosažení stavu DoS v jejich scénáři 300 sekund po úspěšném započetí TCP spojení, načež pro úspěšné dosažení DoS stavu po celou cílenou dobu 600 sekund stačí prakticky útočníkovi

pouze jedenkrát tento časovač vhodně resetovat. Stejně tak autoři práce zvolili za zdroj požadovaný útočníkem v rámci HTTP GET požadavku 700 MB soubor (v jednom případě i 4 GB), přičemž s takto velkým zdrojem se v praxi nesetkáme jen tak na každém serveru potenciální oběti.

Jak již bylo ale dříve řečeno, účelem výzkumu autorů útoku bylo primárně testovat, jaké šířky pásma si útočník může dovolit aplikovat pro generování útoku, díky čemuž bylo dokázáno to, že se SlowDrop útok opravdu může zařadit mezi tzv. pomalé DoS útoky, jelikož lze uskutečnit i za podmínek nízkých šířek pásma útočníka.

Nicméně autoři práci sami uvádějí, že ačkoliv oni sami adaptovali vysoký parametr náhodného zahazování ve výši 90 %, tedy statisticky pouze každý 10 není zahozen, tak naznačili, že by bylo vhodné se zabývat i nižšími hodnotami parametru D vzhledem ke kvalitě připojení mezi obětí a útočníkem.

4.2.2 Porovnání, přizpůsobení a odůvodnění vlastního modelu

V testovacím modelu v rámci této práce je vzat jako cíl útoku Apache server verze 2.4, který má také ponechané výchozí hodnoty, ovšem výchozí hodnota časovače *timeout* této verze Apache serveru je v porovnání s předchozí verzí méně tolerantní a je nastavena na 60 sekund. V testovacích modelech této práce se autor rozhodl vzhledem k výše uvedeným změněným podmínkám pro testování více scénářů na základě proměnných:

- Zahazovací poměr SlowDrop útoku – parametr D.
- Počet současně udržovaných TCP spojení mezi útočníkem a jeho cílem.
- Velikost požadovaného souboru – parametr R.

Obecně se autor této práce snažil veškeré hodnoty a parametry co nejvíce přiblížit reálným situacím, tak aby se (ne)prokázala míra hrozby SlowDrop útoku za podmínek, se kterými by se v praxi útočník mohl setkat.

Požadovaná doba trvání stavu DoS

Vzhledem k tomu, že autory práce byla nastavena tato hodnota jako dvojnásobek hodnoty časovače *timeout* strany serveru, který je hlavním faktorem, na základě kterého serveru ukončí TCP spojení ve scénáři SlowDrop útoku, tak byla zvolena požadovaná hodnota stavu DoS na pětinásobek časovače *timeout* a zároveň symbolickou hodnotu 300 sekund, tedy 5 minut.

Šířka pásma a kvalita připojení

Jako výchozí hodnotu kvality připojení – šířky pásma B mezi virtualizovaným strojem útočníka se autor práce rozhodl zvolit 10 Mbps. Tato hodnota byla nastavena

přímo pomocí virtualizačního nástroje VMware a potvrzena testem provedeným nástrojem pod názvem *iperf*, který testuje kvalitu spojení mezi zařízeními.

Velikost požadovaného souboru

Velikost souborů, které útočník může požadovat od oběti, se autor této práce rozhodl také omezit oproti autorům útoku, a to na 3 různé varianty (řekněme úrovně obtížnosti pro útočníka) – 1,2 MB, 12 MB a 120 MB. Tyto tři různé varianty také dále systematicky rozdělují prezentované výsledky testování modelu útoku.

Počet TCP spojení

V neposlední řadě je potřeba zmínit, že modely útoku nanejvýš pracovaly s 20 současnými TCP spojeními generovanými útočníkem směrem k serveru oběti.

Odůvodnění tohoto faktu spočívá v tom, že pokud je server byť už jen minimálně zabezpečen proti hrozbě DoS útoku, tak má nastavený limit maximálního počtu současných TCP spojení s jednou IP adresou jakožto úplně základní obranu proti vyčerpání zdroje možných TCP spojení.

Druhé odůvodnění tohoto rozhodnutí spočívá v tom, že hrozbu vícero současných TCP spojení dokázali již autoři útoku, proto byly primárně testovány scénáře s 1, 10 a 20 současnými TCP spojeními mezi útočníkem a serverem oběti.

Zahazovací poměr

V rámci testování modelu útoku byly testovány různé hodnoty zahazovacího poměru SlowDrop útoku – parametru D. Jako inkrementální krok byla zvolena hodnota 10 %, přičemž při vysoké míře ukončených spojení nebo při dosažení požadované doby stavu DoS bylo testování ukončeno v rámci daného scénáře.

5 Výsledky testování útoku SlowDrop

Během testování modelu útoku SlowDrop byly realizovány celkově tři scénáře odlišné od sebe ve velikosti dostupného zdroje jakožto předmětu HTTP GET žádosti, pomocí které útok probíhá.

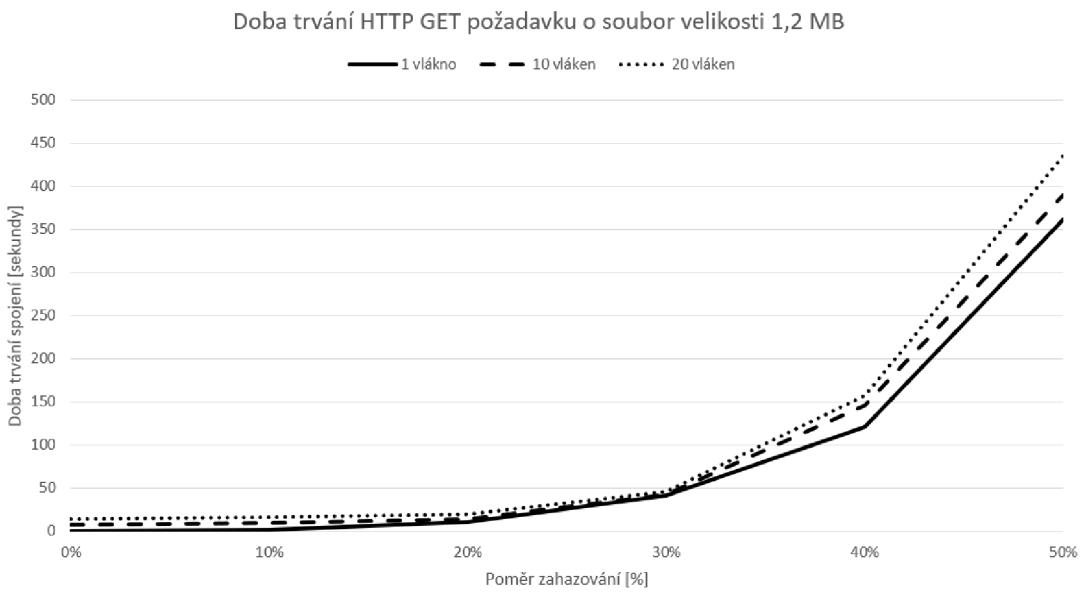
Cílem každého scénáře bylo udržet navázaná TCP spojení po požadovanou dobu 300 sekund (pětinásobek časovače *timeout*), přičemž šířka pásma mezi útočníkem a cílovým serverem byla nastavena na 10 Mbps. Postupně byl zvyšován zahazovací poměr – hlavní charakteristika SlowDrop útoku, tak, aby TCP spojení nebylo ukončeno ze strany serveru z důvodu neodpovídání klienta (útočníka).

Provoz byl analyzován pomocí dat dostupných přímo ze skriptu útoku a také z *.pcap* souborů programu Wireshark běžícího na serveru oběti.

5.1 Výsledky měření

Následující grafy zobrazují dobu trvání TCP spojení pro jednotlivé scénáře útoků – od nejmenšího souboru po největší. Doba trvání TCP spojení je počítána jako průměr dob, kterou trvá serveru oběti splnit žádost HTTP GET jednotlivého vlákna za jejich současného běhu (scénáře 10 a 20 vláken), v případě jednotlivého vlákna byla žádost HTTP GET realizována jediným vláknem desetkrát za sebou a doby zprůměrovány.

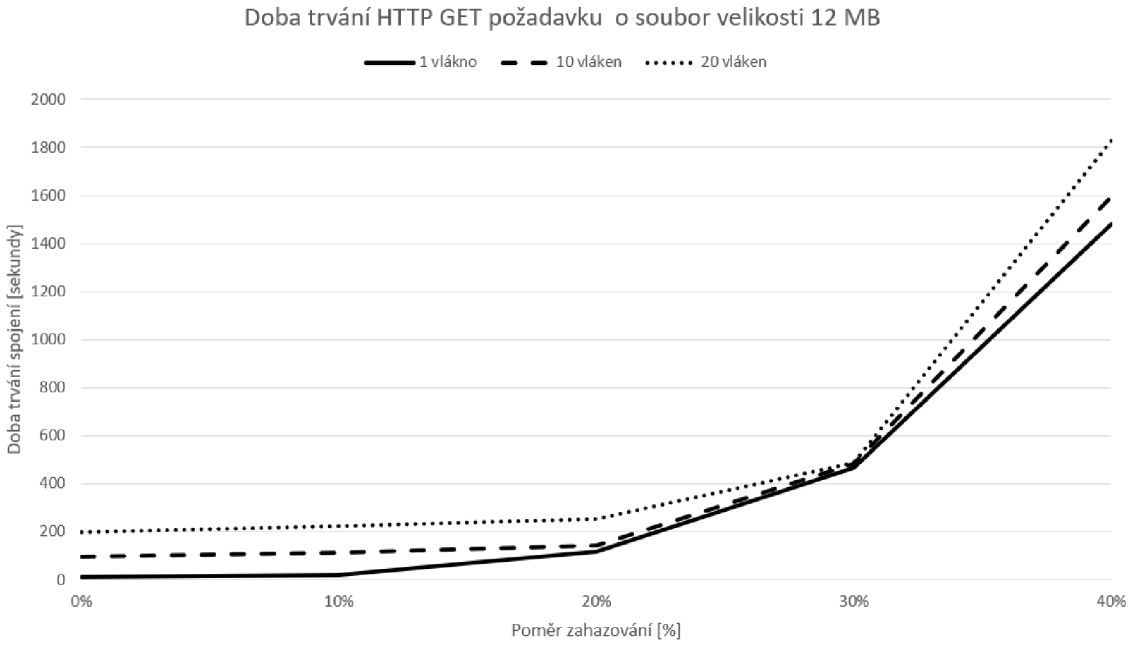
Scénář 1 – velikost souboru 1,2 MB



Obr. 5.1: Scénář 1 – žádaný zdroj velikosti 1,2 MB

V případě tohoto scénáře lze z grafu vyčíst, že pro udržení TCP spojení po požadovanou dobu 300 sekund je potřeba poměrně vysoký poměr zahazování příchozích TCP paketů – $D = 0,5$, přičemž na počtu souběžných TCP spojení (vláken) tolik nezáleží, jelikož přenos tohoto souboru není přímo limitován šířkou pásma. Takto vysoká míra zahazování má však svá úskalí, která budou uvedena.

Scénář 2 – velikost souboru 12 MB



Obr. 5.2: Scénář 2 – žádaný zdroj velikosti 12 MB

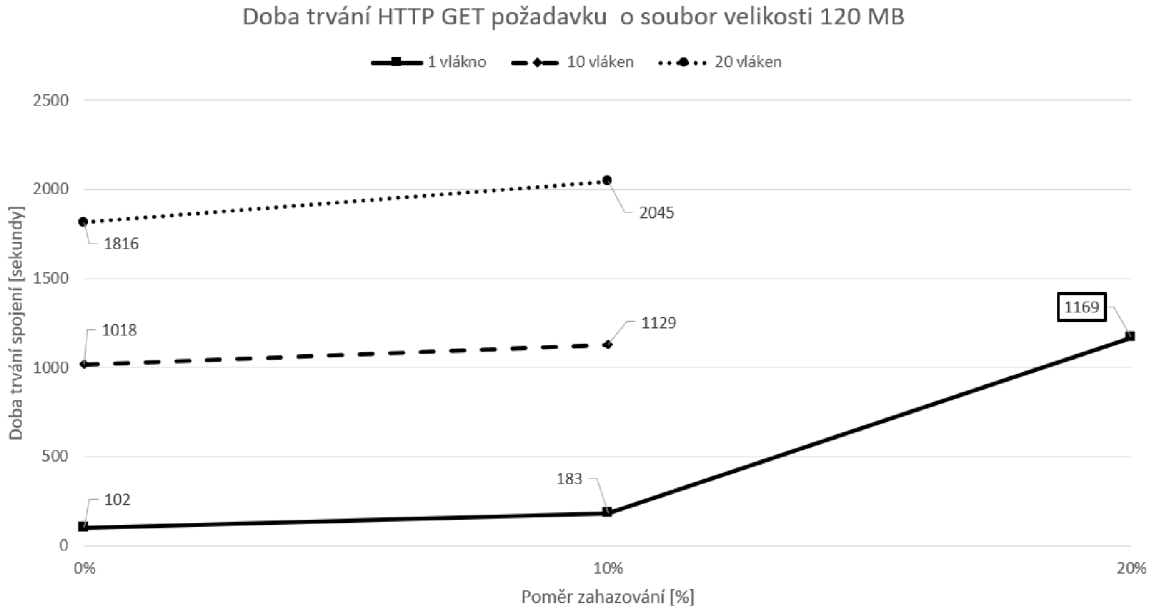
V případě tohoto scénáře se lze setkat s nižším potřebným poměrem zahazováním pro všechny vláknové varianty k dosažení trvání TCP spojení po požadovanou dobu, a to s $D = 0,3$, kdy průměrná doba na splnění HTTP GET žádosti pro všechny vláknové scénáře je cca 470 sekund.

Se zvyšující se hodnotou parametru D lze pozorovat (stejně jako v předchozím scénáři), že doba na splnění HTTP GET žádosti až nepřiměřeně roste, což je žádoucí efekt z pohledu útočníka.

V neposlední řadě je na místě si povšimnout podobnosti grafů tohoto scénáře a předchozího, kdy ve scénáři s větším souborem můžeme pozorovat střídmější stoupání křivky se zvyšující se hodnotou parametru D.

Scénář 3 – velikost souboru 120 MB

Z grafu posledního scénáře doplněného o detailnější popisky lze vyčíst, že pro udržení TCP spojení po požadovanou dobu 300 sekund není v praxi za nastavené šířky pásma 10 Mbps v případě vícera simultálních vláken potřeba žádného zahazování a splnění HTTP GET žádostí bude samo o sobě trvat přes 1000 sekund. Stejně tak malý poměr zahazování jako $D = 0,1$ nebude mít takřka vliv na tento provoz.



Obr. 5.3: Scénář 3 – žádaný zdroj velikosti 120 MB

V případě jedné probíhající HTTP GET žádosti je například tato doba navýšena o zhruba stejnou dobu jako tomu bylo u varianty s deseti vlákny.

Rapidní nárůst doby pro splnění HTTP GET požadavku lze však pozorovat při iteračním kroku z poměru zahazování 10 % na 20 % ve variantě jediného vlákna díky opakovanému přenosu paketů (TCP Retransmission a fragmentaci), přičemž vícevláknové varianty vyčerpaly hardwarové kapacity testovacího prostředí po cca 3000 sekundách (50 minutách).

5.2 Analýza provozu a výsledků měření

Výše uvedené grafy dobře ukazují potenciál hrozby, která tkví v zahazování příchozího provozu TCP spojení, zejména ve scénářích s většími požadovanými soubory. Avšak hlavní princip, okolo kterého pracuje útok SlowDrop a se kterým musí pracovat při výběru proměnných, na nich ukázán není.

Teoreticky by pro SlowDrop útok v praxi bylo nejlepší vzít ten nejvyšší zahazovací poměr možný, jako např. $D = 0,9$, pro který se rozhodli autoři útoku, avšak při

vyšších hodnotách tohoto parametru dochází až příliš často v provozu k tzv. *TCP Retransmission* (opakování přenosu TCP paketu), které má za úkol napravit chyby způsobené neúspěšným doručením TCP paketů (odesílatel – server neobdrží zpět náležité pakety s příznakem ACK).

V souvislosti s TCP Retransmission funguje v TCP protokolu časovač *Retransmission timeout*, který je při správném nastavení parametrů SlowDrop útoku jeho největším pomocníkem, avšak je také jeho největší hrozbou, jelikož si jde protichůdně přímo proti časovači *timeout*.

5.3 TCP Retransmission a Timeout (RTO)

Nejdříve je vhodné ukázat jev, kterého se SlowDrop útok snaží skrze funkci TCP Retransmission dosáhnout pro jeho nejvíce optimalizovaný průběh – co nejdelší udržení TCP spojení. Je na místě zopakovat, že Apache časovač *timeout*, se kterým testovací prostředí pracuje, je 60 sekund.

Tento časovač zjednodušeně řečeno udává (podrobněji viz např. [24],[25]), že pokud během 60 sekund kdykoliv v průběhu plnění žádosti o zaslání dat neobdrží server potvrzení přijetí paketu (ACK) a jeho odesílací buffer je plný, tak ukončí probíhající žádost (a s ní spojené TCP spojení – nežádoucí jev pro SlowDrop útok). Je tedy na místě, aby minimálně jedním paketem s příznakem ACK útočník odpověděl během tohoto časovače – je tedy potřeba, aby i „měl na co odpovědět“.

REF	192.168.198.128	192.168.198.129	TCP	78	[TCP Window Update] 57242 → 80 [ACK] S
0.000014325	192.168.198.129	192.168.198.128	TCP	1514	80 → 57242 [ACK] Seq=1042561 Ack=350 W
0.015593508	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
0.223237574	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
0.655290852	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
1.487350386	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
3.151293866	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
6.606603991	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
13.263475581	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
26.575249959	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
54.222646927	192.168.198.129	192.168.198.128	TCP	1514	[TCP Retransmission] 80 → 57242 [ACK]
54.223145497	192.168.198.128	192.168.198.129	TCP	66	57242 → 80 [ACK] Seq=350 Ack=1042561 W

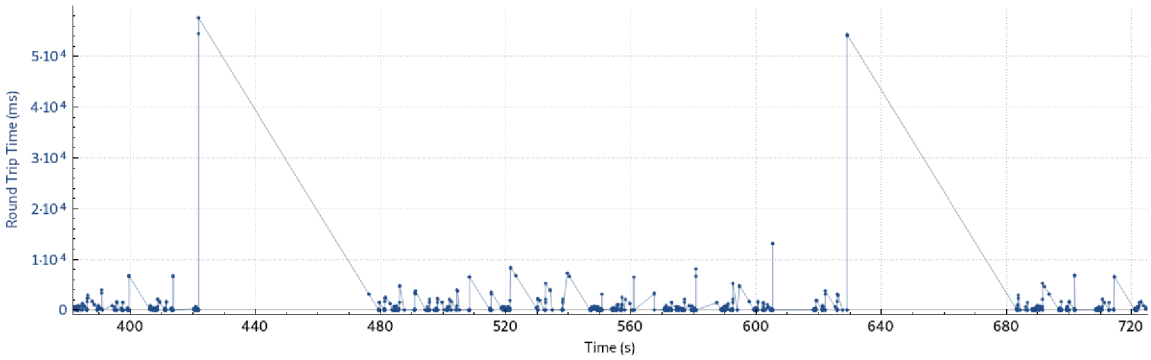
Obr. 5.4: Vhodná míra TCP Retransmission pro SlowDrop útok

Jak lze vyčíst ze snímku analýzy provozu programu Wireshark, doba mezi odeslanými pakety od útočníka je cca 54 sekund, což je v tomto scénáři skoro maximální možná doba, kterou si útočník může dovolit „mlčet“, aniž by jeho spojení bylo ukončeno.

Dále lze ze snímku vyčíst, že doba určená hodnotou časovače *Retransmission Timeout (RTO)* – doba uplynulá mezi odesláním jednotlivých TCP paketů s příznakem TCP Retransmission, se s každou iterací zdvojnásobuje, což je potřeba vzít v úvahu v nastavování parametrů vlastního generování útoku. Počáteční hodnota

časovače RTO vychází z tzv. *Round trip time (RTT)*, což je doba mezi odesláním a přijetím potvrzení o doručení daného odeslaného paketu.

Graf průběhu zvyšovaní a snižování této hodnoty je taktéž možné vygenerovat pomocí programu Wireshark, přičemž následující graf zobrazuje ideální scénář pro SlowDrop útok (situaci znázorněnou výpisem výše).



Obr. 5.5: Průběh změn RTT vhodný pro SlowDrop útok

Výše zobrazený snímek je pouze výsekem z názorného TCP provozu SlowDrop útoku, kdy můžeme pozorovat 2 okna trvající dobu (ideálně pro scénář nastaveného časovače) menší než 60 sekund, kdy se zvyšuje RTT, jelikož útočník nezasílá potvrzující ACK pakety, avšak vhodně před uplynutím časovače znovu vykonává činnost.

Situace, které se snaží útočník vyhnout, je znázorněna na následujícím snímku z programu Wireshark.

REF	192.168.198.128	192.168.198.129	TCP	78 [TCP Window Update] 57248 → 80 [ACK] S
0.000020218	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
0.204418409	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
0.636157435	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
1.468093836	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
3.131905713	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
6.555734712	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
13.212081005	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
26.524888634	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
54.684093477	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]
107.931731266	192.168.198.129	192.168.198.128	TCP	1514 [TCP Retransmission] 80 → 57248 [ACK]

Obr. 5.6: Nevhodná míra TCP Retransmission pro SlowDrop útok

Na tomto snímku lze pozorovat situaci nadměrného zahazování (při nastavení $D = 0,5$), kdy se hodnota časovače RTO dostává na příliš vysoké hodnoty a pakety TCP Retransmission mezi sebou mají příliš velké časové rozestupy. V tomto scénáři je plnění žádosti ze strany serveru oběti ukončeno na základě časovače *timeout*.

5.3.1 Zhodnocení testování modelu útoku SlowDrop

V průběhu testování modelu útoku SlowDrop vyšlo najevo, proč je tento útok tak specifický, potenciálně velmi škodlivý a zneužitelný. Hlavní pointou zhodnocení testování je, že útok SlowDrop je tak variabilní, že jej lze až nebezpečně přizpůsobit na míru, stejně tak se lze ale při nastavování tohoto útoku zmýlit a nebude plnit svůj účel.

SlowDrop útok má oproti tradičním DoS útokům (i těm pomalým) relativně velké množství proměnných, které je potřeba vzít v úvahu před jeho aplikovaním a nelze ani jednu z nich opomenout:

- Počet vláken generujících HTTP GET žádosti.
- Velikost souboru požadovaného HTTP GET žádostí.
- Zahazovací poměr D.
- Šířka pásma B mezi útočníkem a jeho serverem.

V neposlední řadě je také na místě mít při realizaci útoku SlowDrop na mysli faktory, které jeho průběh budou nepřímo i přímo ovlivňovat, jako:

- Fragmentace provozu mezi útočníkem a serverem.
- Kvalita spojení mezi útočníkem a serverem.
- Časovač *timeout* serveru.
- Fyzická vzdálenost mezi útočníkem a serverem – RTT (ovlivňující RTO).

Při podmínkách testovacího prostředí (šířka pásma 10 Mbps, Virtualizované stroje ve VLAN, atd.) uvedeného výše se autorovi této práce osvědčil zahazovací poměr 30 až 40 %, tedy $D =< 0,3; 0,4 >$, jelikož při vyšším zahazovacím poměru se při delším trvání TCP spojení zvyšoval počet předčasně ukončeným spojení na cca 30 %, což je pro průběh SlowDrop útoku nežádoucí.

Zároveň také při testování modelu útoku vyplynula vzhledem k jeho variabilitě a unikátnosti komplikovanost návrhu detekce tohoto útoku na základě signatury, již může být obtížné navrhnout.

Samotní autoři tohoto útoku totiž při jeho tvorbě nepřišli na univerzální způsob detekce, pouze doporučují omezení počtu TCP spojení serveru s jedním klientem, což je jedním ze základních nastavení pro obranu proti DoS útokům obecně.

6 Detekce SlowDrop útoku

Jak bylo na začátku této práce uvedeno, návrh tohoto útoku byl autory zveřejněn zejména za účelem připravení IT společnosti na hrozby podobného druhu do budoucna. Jestliže pochopíme principy, které stojí za funkcionalitou daného útoku, tak návrh obrany proti němu je následně o to snazší.

Následující část práce se věnuje návrhům obrany proti SlowDrop útoku, krokům podniknutým pro obecné zabezpečení serveru v testovacím prostředí a jak byly efektivní proti útoku SlowDrop.

6.1 Obecné zabezpečení systémů

Za účelem obecného zvýšení odolnosti serveru Apache verze 2.4 vůči DoS útokům (i pomalým) v testovacím prostředí byly podniknuty tyto základní doporučené kroky:

- Omezení počtu TCP spojení z jedné adresy pomocí *iptables*.
- Rozšíření Apache serveru o bezpečnostní modul *evasive*.

Omezení počtu TCP spojení z jedné adresy – iptables

Pravidlo (jeho přidání pomocí příkazové řádky terminálu) pro omezení počtu TCP spojení z jedné IP adresy na 20:

```
iptables -I INPUT -p tcp --dport 80
-m connlimit --connlimit-above 20 -j DROP
```

Toto pravidlo primárně zabraňuje vyčerpání zdroje počtu TCP spojení na straně serveru při scénáři, kdy útok je generován nedistribuovaně – nejedná se o DDoS útok. V případě distribuovaného útoku, kdy teoreticky každé jednotlivé TCP spojení pochází z jiné zdrojové IP adresy, tak je toto pravidlo bez jakéhokoliv dopadu.

V takovém případě je obrana proti jakémukoliv (distribuovanému) DoS útoku násobně ztížena. Toto platí i pro DDoS variantu útoku SlowDrop, kdy existuje velké množství přístrojů generujících provoz tohoto útoku.

Apache moduly proti DoS a DDoS útokům

Následně byl server Apache rozšířen[26] o modul s názvem *mod_evasive*, který slouží jako základní zdarma dostupný modul pro ochranu Apache serverů před DoS a DDoS útoky. Příkaz pro instalaci modulu vypadá pro Debian/Ubuntu systémy následovně:

```
apt-get install libapache2-mod-evasive
```

Tento modul po jeho nainstalování, připojení (*enable*) a konfiguraci sleduje podezřelou aktivitu, která je často spojována právě se snahou útočníka dosáhnout stavu

DoS na chráněném serveru. Konfigurační soubor tohoto modulu se ve výchozím nastavení nachází ve složce:

```
/etc/apache2/mods-enabled/evasive.conf
```

V rámci tohoto souboru lze nastavit sledování klíčových prvků pro obranu proti klasickým DoS útokům jako:

- Podezřele vysoký počet žádostí o stejný zdroj (webovou stránku, soubor, další).
- Počet žádostí za jednu sekundu od jednoho klienta.
- Jestli IP na blacklistu dále generují žádosti (prodlužují záznam na blacklistu).

Tento modul mimo detekci a obranu proti podezřelým aktivitám uchovává (samozřejmě podle přání a nastavení administrátora serveru) také logy o těchto incidentech a může administrátora o nich upozorňovat emailem. Dále udržuje tzv. dočasné *blacklisty* (černé listiny), kde zaznamenává IP adresy spojené s podezřelou aktivitou a na základě pravidel nové žádosti od těchto adres tyto žádosti zamítá nebo nechává určitý čas čekat.

Po nakonfigurování a nastavení proměnných tohoto modulu, které jsou zpočátku nastaveny poměrně velmi agresivně (mají relativně vysokou šanci na tzv. *false-positive* – nesprávné identifikace legitimního spojení jako spojení útočníka), lze otestovat funkčnost tohoto modulu skriptem *test.pl* v jazyce *perl* dostupným ve složce *example*s modulu.

V neposlední řadě byl server Apache dočasně rozšířen i o modul *mod_security* od nadace OWASP, který ač je velmi užitečným modulem pro zabezpečení webových serveru a webových aplikací, tak je primárně orientován na útoky zneužívající specifických zranitelností známých v IT společnosti zejména ze seznamu OWASP Top 10 (SQL injection, XSS, atd.), ne na (pomalé) DoS útoky.

Přítomnost tohoto modulu se při testování SlowDrop útoku prokázala nadbytečnou, avšak jeho užitečnost při zabezpečení proti hrozbám nejčastějších útoků na internetu ze seznamu OWASP Top 10 je nepopiratelná a jeho užití lze jen doporučit.

6.2 Implementace IDS

Přímo jako program pro detekci na základě signatur[27] (i vlastních vytvořených) a určených pravidel pro rozpoznání vniku nežádaného provozu byl zvolen open source software Suricata. Suricata pracuje v reálném čase na detekci a prevenci škodlivého provozu na určitém rozhraní na základě jejího nastavení.

Je na místě ji přímo implementovat na dedikované filtrační zařízení, na kterém bude fungovat komplexní firewall jako např. PfSense, který data z IDS Suricata

dokáže vhodně interpretovat a následně s nimi nakládat (např. zobrazit je administrátorovi ve svém uživatelském prostředí). V testovacím prostředí v rámci této práce byl ovšem software Suricata spuštěn přímo na webovém serveru potenciální oběti.

Jako alternativa pro IDS Suricata může být také velice rozšířený a populární Snort[28], který je ovšem starší a využívá vždy pouze jednoho procesního vlákna procesoru, zatímco Suricata podporuje tzv. *multithreading*.

Jednotlivá pravidla (signatury), na základě kterých tyto IDS detekují škodlivý provoz, jsou z IDS Snort použitelná i pro IDS Suricata. Z těchto důvodů bylo rozhodnuto pro IDS Suricata.

Software Suricata byl nastaven na sledování rozhraní *eth0* webového serveru, přes který prochází komunikace mezi útočníkem a webovým serverem (také jej sleduje program Wireshark) a následně byla otestována jeho funkčnost nástrojem *ping* využívající protokolu *icmp*.

Ukázkové testované pravidlo pro upozornění uživatele o zjištění *icmp* provozu na sledovaném rozhraní vypadá následovně:

```
alert icmp any any -> any any (msg: "ICMP";)
```

Upozornění v parametru *msg* se při detekci *icmp* provozu následně zobrazí v terminálu, kde je spuštěn software Suricata. Pro správné fungování pravidel (signatur) v IDS Suricata musí tato pravidla (soubory s příponou *.rules*) být náležitě umístěna ve složce *rules* v domovské složce tohoto IDS a také musí být povolena v konfiguračním souboru.

V konfiguračním souboru je také potřeba definovat domovskou a externí síť, přičemž definice domovské sítě v testovacím prostředí obsahovala pouze IP adresu serveru oběti.

6.2.1 Testování existujících signatur

V rámci testování byla zkoušena pro detekci SlowDrop útoku již existující pravidla dostupná na internetu. V případě IDS Suricata se jedná zejména o soubory *emerging-dos.rules* a *emerging-web_server.rules* dostupné z webu *rules.emergingthreats.net*, kde jsou spravována nejrůznější aktuální pravidla a signatury pro IDS Snort a Suricata. Obdobou pro tento soubor pravidel pro IDS snort je balík *dos.rules*.

Vhodnou poznámkou pro správně fungoování těchto IDS je, že je na místě zakomentovaná pravidla (signatury) v těchto souborech odkomentovat pro jejich správné fungování.

Bohužel v těchto pravidlech nejsou typicky obsažena pravidla pro detekci pomalých DoS útoků. Proto je potřeba prohledávat nejrůznější pravidla pod klíčovými slovy jako „slow“ a dalšími.

Po širším hledání i v pravidlech IDS Snort se podařilo autorovi práce najít pravidla pro zachytávání pomalých útoku Slowloris, Slow POST, Slow RANGE, Slow READ a pomalých útoků nástroje SlowHTTPTest, který byl již v rámci této práce testován za účelem ověření funkčnosti testovacího prostředí.

Tato pravidla rozpoznávají uvedené pomalé útoky na základě identifikace anomálií a parametrů v podvržených hlavičkách HTTP žádostí:

- Slowloris – Content-Length
- Slow POST – Keep-Alive a Content-Length
- Slow RANGE – rozsahy specifické pro nástroj Killapache generující tento útok
- Slow READ – malá velikost TCP Window přes akceptovatelou dobu 60 sekund
- Útoky generované nástrojem SlowHTTPTest – Referer (záměrně nástrojem nastavené rozpoznatelně)

Pro ověření funkčnosti IDS vzhledem k těmto pravidlům byl opět použit nástroj SlowHTTPTest a upozornění uživatele na základě těchto pravidel bylo úspěšně ověřeno.

Žádná z již existujících signatur bohužel podle očekávání nedetekovala provoz útoku SlowDrop.

6.2.2 Návrh signatury útoku SlowDrop

Při zamýšlení se nad návrhem signatury pro rozpoznání provozu SlowDrop útoku je nutné se nejdříve zamyslet nad již existujícími signaturami pomalých DoS útoků, co mají společného, co je odlišuje a zdali jde něco z nich použít pro detekční signaturu útoku SlowDrop.

Všechny z výše uvedených signatur rozpoznávají škodlivý provoz na základě hlavičky HTTP žádosti, která je v některých parametrech útočníkem upravena oproti provozu legitimního uživatele. SlowDrop útok v teorii může užívat naprosto legitimních hlaviček HTTP GET žádostí, tedy na základě jeho hlavičky jej nejde odlišit od legitimního provozu. Hlavní myšlenkou útoku SlowDrop je totiž dokonalé napodobení legitimního uživatele a znemožnění detekce útoku.

Další vhodné zamyšlení nad signaturou pro SlowDrop útok je analýza toho, co vlastně SlowDrop útok dělá záměrně škodlivého, když nepodvrhuje hlavičku HTTP žádosti. Odpověď je jeho hlavní charakteristikou, zahazuje náhodně příchozí pakety od serveru. Pakety jdoucí od serveru ke klientovi se ale mohou ztratit i legitimnímu klientovi se špatným připojením. Je tedy na místě analyzovat to, čím a zdali se vůbec liší provoz SlowDrop útoku od takového legitimního klienta.

6.2.3 Porovnání SlowDrop a legitimního špatného připojení

Nástroje pro simulaci špatného připojení (někdy označované jako WAN emulátory) se využívají zejména pro testování výkonu webových aplikací a serverů vzhledem k tomu, jak kvalitně budou jejich služby poskytovány klientům s nedokonalým připojením (např. nekvalitní bezdrátové připojení). Pro simulaci těchto legitimních klientů byly vybrány následující nástroje:

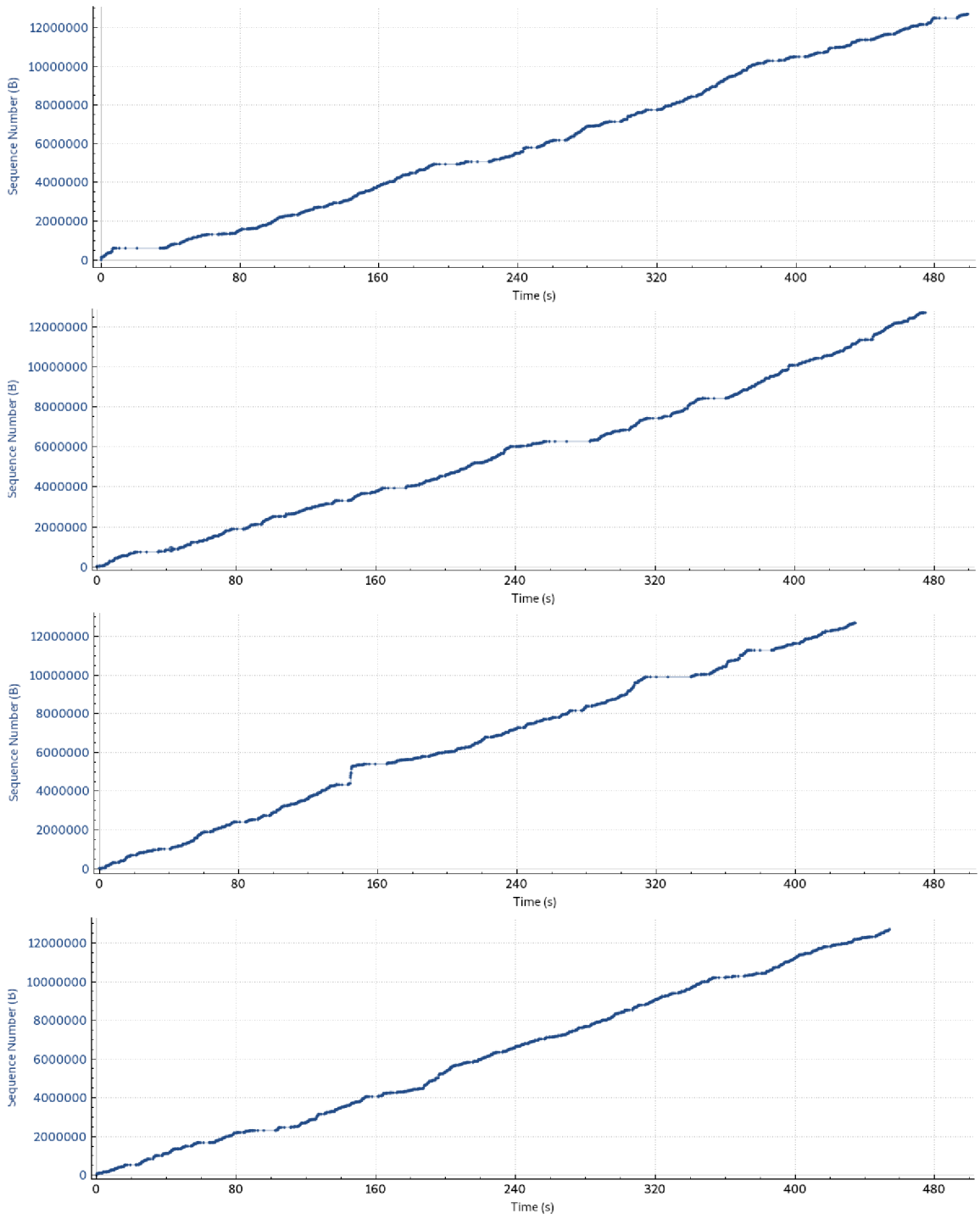
- clumsy 0.2 pro Windows
- SoftPerfect Connection Emulator pro Windows
- traffic control/netem pro Linux (na straně serveru oběti)

Pro testování těchto scénářů byly použity virtuální stroje OS Windows i Linux legitimního uživatele se stejnou šířkou pásma jako ve scénáři s útočníkem – 10 Mbps. Nástroje clumsy a SoftPerfect Connection Emulator jsou intuitivní nástroje s přívětivým uživatelským prostředím, ve kterých pouze stačí vybrat hodnoty potřebné pro simulaci nekvalitního připojení. Nástroj *traffic control* je, jak již bylo dříve uvedeno, obsažen v balíčku *iproute* a jeho nastavování probíhá skrze příkazy v terminálu, jejichž příklady byly taktéž ukázány.

Testován byl provoz probíhající skrze nativní internetové prohlížeče legitimního uživatele (Firefox a Internet Explorer) při simulované ztrátovosti provozu (*packet loss, drop rate*) 30 % a velikosti požadovaného souboru 12 MB.

6.2.4 Výsledky porovnání a jejich zhodnocení

V následujících snímcích programu Wireshark je zobrazen průběh přenosu útoku SlowDrop, který generuje jedno vlákno a zahazovací poměr je nastaven na $D = 0,3$, v porovnání se třemi výše uvedenými scénáři napodobujícími legitimními uživateli se simulovaným nekvalitním připojením. Snímky ukazují průběh TCP spojení z pohledu Time Sequence – průběh odesílání paketů (jejich sekvenčních čísel) v průběhu času spojení.



Obr. 6.1: Porovnání Time Sequence HTTP GET žádostí. Pořadí shora dolů: SlowDrop, clumsy, SoftPerfect Connection Emulator, traffic control/netem.

Při analýze nejen ukázaných grafů (lišících se pouze v dobách trvání, která se při daných poměrech zahazování přirozeně pohybují v těchto rozmezích), ale i dalších výstupů programu Wireshark, které je nadbytečné zde ukazovat (všechny jsou si podobné a spolu zaměnitelné), je zřejmé, že nelze lidským okem odlišit průběh SlowDrop útoku od simulovaných špatných připojení.

Při dalším zamyšlení je zřejmé, proč tomu tak je. SlowDrop útok ve své povaze vlastně vůbec není „útok“, jedná se pouze o další simulaci záměrně špatného připojení, nad kterou ale má útočník kontrolu.

Hlavička HTTP žádosti je legitimní, chování stroje útočníka je legitimní, dokonce může (dle implementace) útočník legitimně zobrazit po dokončení HTTP GET žádosti její obsah.

SlowDrop útok není podobný svou povahou žádnému předchozímu pomalému DoS útoku a pokud je vůbec na místě o něm mluvit jako o útoku, tak se jedná o zvláštní druh DoS útoku spočívající v ovládání podmínek spojení mezi útočníkem a obětí. Identifikace takto škodlivého spojení by již byla v dnešní době předmětem strojového učení (viz např. [29],[30]) jakožto nástroje pro analýzu provozu, což potvrzuje i autor útoku v zakončení práce tomuto útoku věnované.

Škrcení spojení a tvarování provozu (angl. *throttling, traffic shaping*), se, jak již bylo řečeno, běžně užívá pro testování tzv. WAN emulací, avšak jeho použití jako nástroje pro realizaci DoS útoku však teoreticky zdaleka nekončí pouze u zahazování příchozích paketů, jako tomu je u útoku SlowDrop.

Fantazii se v tomto ohledu navrhování dalších způsobů realizace DoS útoků pomocí těchto metod meze nekladou. Autor této práce by tedy rád v budoucnu viděl (a bude se tématu věnovat a sledovat jeho vývoj) další vědecké články a výzkumné práce na toto téma, jelikož literatury na téma užití angl. *rate-limiting* a tvarování provozu obecně jako prevence proti DoS útokům je mnoho, avšak použití těchto metod jako nástroje ke generování DoS útoků a dosažení stavu DoS je prozatím téma zdá se neprobádané.

Závěr

Diplomová práce se zabývala problematikou pomalého DoS útoku s názvem SlowDrop publikovaného v roce 2019[1]. Cílem této práce bylo teoreticky i prakticky rozvést útok SlowDrop, uvést informace nutné k pochopení jeho fungování, realizaci, testování a také následnému návrhu obrany proti němu.

Teoretické i praktické poznatky získané v rámci práce a původní práce autorů útoku[1] byly kriticky hodnoceny a následně konstruktivně komentovány za účelem zlepšení realizace modelu SlowDrop útoku a obrany proti němu[2] v rámci této práce.

V úvodních teoretických kapitolách se práce zabývala problematikou DoS/DDoS útoků jako takovou, kategorizovala je a na základě tohoto rozdělení uvedla specifické příklady útoků daných kategorií. Nebyl opomenut ani aspekt obrany proti těmto útokům, který byl teoreticky rozveden a poznatky z něj přeneseny do pozdějších částí práce.

V hlavní teoretické částí práce pojednávala o teoretické přípravě SlowDrop útoku, jeho principech, základních myšlenkách, inspiracích v již existujících pomalých DoS útocích. Uvedeny byly i slabiny a omezení tohoto útoku, které jsou v praktických částech práce názorně rozvedeny a demonstrovány.

Hlavní teoretická část práce rozvedla klíčové parametry, na jejichž základě útok SlowDrop operuje a jejichž přizpůsobení je klíčové pro specifické scénáře útoku. Konkrétní parametry realizace tohoto útoku byly teoreticky diskutovány, následně implementovány a testovány. Optimalizace těchto parametrů byla v praktických modelových situacích následně rozvedena.

V první praktické části se práce zabývala jak obecně realizací SlowDrop útoku v praxi, tak jednotlivými konkrétními kroky průběhu útoku. Po představení testovacího prostředí, ve kterém vývoj a následné testování implementace SlowDrop útoku probíhalo, byly následně rozebrány jednotlivé fáze praktické realizace modelu SlowDrop útoku.

Největší důraz byl kladen na fázi útoku, ve které probíhá zahazování, na jehož základě stojí celý princip útoku SlowDrop. Různé metody realizace zahazování byly testovány, zhodnoceny a následně byla vybrána optimální implementace pro účely této práce – zahazování v rámci *iptables*.

Následně se práce věnovala testování modelu útoku na cílové oběti v podobě Apache serveru verze 2.4. Byly realizovány scénáře různých podmínek pro potenciálního útočníka SlowDrop útoku. Poměrná část práce je věnována diskuzi nad poznatky získané z těchto testování a závěry z nich plynoucí. Tyto poznatky jsou také kriticky zhodnoceny v kontextu porovnání s původním testováním autory útoku[1]. Jednotlivé parametry pro dané scénáře v souvislosti s optimálním průběhem útoku SlowDrop jsou náležitě uvedeny.

Analýza provozu testovacích scénářů byla následně teoreticky rozvedena a byly prakticky demonstrovány limity SlowDrop útoku v podobě časovače *timeout* na straně serveru oběti v protichůdnosti s časovačem TCP protokolu *Retransmission timeout* (RTO), který je klíčový v opakovaném přenosu – *TCP Retransmission*.

V poslední praktické části se práce zabývala návrhy na implementaci IDS – systému detekce SlowDrop útoku. Kroky podniknuté pro zabezpečení serveru oběti před SlowDrop útokem jsou uvedeny, jak obecná konfigurační nastavení serveru, tak vybrané obrané a detekční software komponenty, o které byl server oběti rozšířen.

Tyto obranné mechanismy zahrnující Apache moduly pro detekci DoS útoků a IDS Suricata obohacený o existující dostupné signatury (včetně signatury určené pro IDS Snort) byly následně testovány, jejich funkčnost potvrzena, avšak aplikovatelnost proti provozu SlowDrop útoku nikoliv. Typické signatury jsou totiž zpravidla založeny na detekci podvržené hlavičky HTTP GET požadavku, která je v případě SlowDrop útoku naprosto legitimní.

Konečná část práce se věnovala porovnání legitimního provozu klienta se špatným připojením realizovaného pomoci vybraných WAN emulátorů a provozu SlowDrop útoku, který se legitimní provoz snaží napodobit.

Na základě analyzované nerozlišitelnosti provozu útoku SlowDrop byl vyvozen závěr, že SlowDrop útok není typickým pomalým DoS útokem, nýbrž se jedná pouze o legitimní provoz na základě HTTP GET žádosti, jehož průběh je upraven nastavením parametrů kvality připojení k cílové oběti. Detekce tohoto útoku je tedy principiálně detekcí nekvalitního připojení, proti němuž již má běžný server sám od sebe výchozí mechanismy a nastavení (zejména časovače), které je možné v jeho konfiguračních souborech upravit v ohledech tolerance k nekvalitním připojení klientů.

Literatura

[1] CAMBIASO, Enrico, Giovanni CHIOLA a Maurizio AIELLO, 2019. Introducing the SlowDrop Attack. *Computer Networks* [online]. Amsterdam: Elsevier, 26 February 2019, *2019*(150), 234-249 [cit. 2020-06-01]. DOI: https://doi.org/10.1016/j.comnet.2019.01.007. ISSN 13891286. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1389128619300210

[2] SUROTO, Suroto, 2017. A Review of Defense Against Slow HTTP Attack. *JOIV: International Journal on Informatics Visualization* [online]. *1*(4), 127-134 [cit. 2020-06-01]. DOI: 10.30630/joiv.1.4.51. ISSN 2549-9904. Dostupné z: http://joiv.org/index.php/joiv/article/view/51

[3] CAMBIASO, Enrico, Gianluca PAPALEO, Giovanni CHIOLA a Maurizio AIELLO, 2013. Slow DoS attacks: definition and categorisation. *International Journal of Trust Management in Computing and Communications* [online]. October 2013, 300-319 [cit. 2020-06-01]. DOI: 10.1504/IJTMCC.2013.056440. ISSN 2048-8378. Dostupné z: http://www.inderscience.com/link.php?id=56440

[4] ALNAKHALNY, Redhwan, Sureswaran RAMADASS a Selvakumar MANICKAM, 2013. A Study on Detecting IMCPv6 Flooding Attack based on IDS. *Australian Journal of Basic and Applied Sciences. 1*(7), 175-181. DOI: https://doi.org/10.22587/ajbas. ISSN 1991-8178. Dostupné také z: https://www.researchgate.net/publication/237065202_A_Study_on_Detecting_ICMPv6_Flooding_Attack_based_on_IDS

[5] SUMNER, Ryan, 2019. What is a DDos Attack? *IBM.com* [online]. New York, USA: IBM [cit. 2020-06-01]. Dostupné z: https://www.ibm.com/cloud/blog/new-builders/video-what-is-a-ddos-attack

[6] KENIG, Ronan, 2013. DDoS Survival Handbook: The Ultimate Guide to Everything You Need To Know About DDoS Attacks. *Radware* [online]. Tel Aviv, Izrael: Radware, 2013 [cit. 2020-06-01]. Dostupné z: https://security.radware.com/uploadedFiles/Resources_and_Content/DDoS_Handbook/DDoS_Handbook.pdf

[7] CRANE, Casey, 2019. The Largest DDoS Attacks in history. *Hashedout: by The SSL Store* [online]. St. Petersburg, FL: The SSL Store, 29 May 29 2019 [cit. 2020-06-01]. Dostupné z: https://www.thesslstore.com/blog/largest-ddos-attack-in-history/

[8] BÁČOVÁ, Petra, 2017. Volební weby byly nedostupné kvůli DDoS útoku. *Český Statistický Úřad* [online]. Česká republika, 22. října 2017 [cit. 2020-06-01]. Dostupné z: https://www.czso.cz/csu/czso/volebni-weby-byly-nedostupne-kvuli-ddos-utoku

[9] KARAMI, Mohammad, Youngsam PARK a Damon MCCOY, 2016. Stress Testing the Booters. *Proceedings of the 25th International Conference on World Wide Web - WWW '16*. New York, New York, USA: ACM Press, 2016, *25*(1), 1033-1043. DOI: 10.1145/2872427.2883004. ISBN 9781450341431. Dostupné také z: http://dl.acm.org/citation.cfm?doid=2872427.2883004

[10] NOROOZIAN, Arman, Maciej KORCZYŃSKI, Carlos Hernandez GAÑAN, Daisuke MAKITA, Katsunari YOSHIOKA a Michel VAN EETEN, 2016. Who Gets the Boot? Analyzing Victimization by DDoS-as-a-Service. *Research in Attacks, Intrusions, and Defenses*. Cham: Springer International Publishing, 2016-09-07, *9*(1), 368-389. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-45719-2_17. ISBN 978-3-319-45718-5. Dostupné také z: http://link.springer.com/10.1007/978-3-319-45719-2_17

[11] NIDECKI, Tomasz Andrzrej, 2019. What is the High Orbit Ion Cannon. *Acunetix* [online]. USA: acunetix, 5 July 2019 [cit. 2020-06-01]. Dostupné z: https://www.acunetix.com/blog/web-security-zone/what-is-high-orbit-ion-cannon/

[12] ZARGAR, Saman Taghavi, James JOSHI a David TIPPER, 2013. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys & Tutorials* [online]. *15*(4), 2046-2069 [cit. 2019-12-09]. DOI: 10.1109/SURV.2013.031413.00127. ISSN 1553-877X. Dostupné z: http://ieeexplore.ieee.org/document/6489876/

[13] CHHABRA, Meghna, Brij GUPTA a Ammar ALMOMANI, 2013. A Novel Solution to Handle DDOS Attack in MANET. *Journal of Information Security* [online]. *04*(03), 165-179 [cit. 2019-12-10]. DOI: 10.4236/jis.2013.43019. ISSN 2153-1234. Dostupné z: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jis.2013.43019

[14] *RFC 791: Internet protocol Darpa internet program protocol specification*, 1981. 1. USA: The Internet Engineering Task Force (IETF). Dostupné také z: https://tools.ietf.org/html/rfc791

[15] ISO, *Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model: ISO/IEC 7498-1:1994*, 1994. 1. USA: ISo.

[16] ROBERT, Hansen, 2009. Slowloris HTTP DoS. *Ha.ckers.org* [online]. USA: ha.ckers, 17. 6. 2009 [cit. 2020-06-01]. Dostupné z: https://web.archive.org/web/20150426090206/http://ha.ckers.org/slowloris

[17] KINGCOPE, Masashi FUJIWARA a Markus NEIS, 2000. Apache Range Header DoS (Apache Killer). *Rapid7* [online]. Boston, Massachusetts, USA: rapid7, 19. 8. 2011 [cit. 2020-06-01]. Dostupné z: https://www.rapid7.com/db/modules/auxiliary/dos/http/apache_range_dos

[18] *RFC 2616: Hypertext Transfer Protocol – HTTP/1.1*, 1999. 1. USA: The Internet Engineering Task Force (IETF). Dostupné také z: https://tools.ietf.org/html/rfc2616

[19] CAMBIASO, Enrico, Gianluca PAPALEO, Giovanni CHIOLA a Maurizio AIELLO, 2015. Designing and Modeling the Slow Next DoS Attack. *International Joint Conference* [online]. Cham: Springer International Publishing, 2015-5-27, 249-259 [cit. 2020-06-01]. Advances in Intelligent Systems and Computing. DOI: 10.1007/978-3-319-19713-5_22. ISBN 978-3-319-19712-8. Dostupné z: http://link.springer.com/10.1007/978-3-319-19713-5_22

[20] PAULI, John, 2013. Web Application Recon and Scanning. *The Basics of Web Hacking* [online]. Amsterdam: Syngress, *2013*(1), 41-62 [cit. 2020-06-01]. DOI: https://doi.org/10.1016/C2013-0-00087-4. Dostupné z: https://doi.org/10.1016/C2013-0-00087-4

[21] SHEKYAN, 2011. SlowHTTPTest: SlowHTTPTest Package Description. *KALI TOOLS* [online]. USA: Offensive Security, 24 August 2011 [cit. 2020-06-01]. Dostupné z: https://tools.kali.org/stress-testing/slowhttptest

[22] GREGR, Filip, 2017. *Generátor kybernetických útoků* [online]. Brno [cit. 2020-06-01]. Dostupné z: http://hdl.handle.net/11012/65884. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací. Vedoucí práce Jan Hajný.

[23] KARIMI, Kayvan, Arash AHMADI, Mahmood AHMADI a Bahram BAHRAMBEIGY, 2013. Acceleration of IPTABLES Linux Packet Filtering using GPGPU. *Symposium on Computer Science and Software Engineering (CSSE)* [online]. Tehra, Iran, 1-9 [cit. 2020-06-01]. DOI: 10.13140/2.1.3047.7763. Dostupné z: https://pdfs.semanticscholar.org/166d/1ca7adf0d25734dacc5898ca330f3b84b98e.pdf

[24] MEDHI, Deep a Karthik RAMASAMY, 2018. *Network Routing: Algorithms, Protocols, and Architectures* [online]. 2. Burlington, MA: Morgan Kaufmann [cit. 2020-06-01]. ISBN 978-0-12-800737-2. Dostupné z: https://doi.org/10.1016/C2013-0-18604-7

[25] F. ANASTASI, Gaetano, Massimo COPPOLA, Patrizio DAZZI a Marco DISTEFANO, 2016. QoS Guarantees for Network Bandwidth in Private Clouds. *Procedia Computer Science* [online]. *2016*(97), 4-13 [cit. 2020-06-01]. DOI: https://doi.org/10.1016/j.procs.2016.08.275. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1877050916320919

[26] Guidelines for Securing Apache Web Servers, 2002. *Network Security* [online]. *2002*(12), 8-14 [cit. 2020-06-01]. DOI: https://doi.org/10.1016/S1353-4858(02)12009-5. ISSN 1353-4858. Dostupné z: http://www.sciencedirect.com/science/article/pii/S1353485802120095

[27] DE SOUSA ARAUJO, Tiago Emilio, Fernando Menezes MATOS a Josilene Aires MOREIRA, 2017. Intrusion detection systems' performance for distributed denial-of-service attack. *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)* [online]. IEEE, 2017, 1-6 [cit. 2020-06-01]. DOI: 10.1109/CHILECON.2017.8229519. ISBN 978-1-5386-3123-2. Dostupné z: http://ieeexplore.ieee.org/document/8229519/

[28] BEALE, Jay, 2006. *Snort: IDS and IPS toolkit*. 2006. Burlington, MA: Syngress. ISBN 978-1-59749-099-3.

[29] ZHANG, Naiji, Fehmi JAAFAR a Yasir MALIK, 2019. Low-Rate DoS Attack Detection Using PSD Based Entropy and Machine Learning. *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)* [online]. IEEE, 2019, 59-62 [cit. 2020-06-01]. DOI: 10.1109/CSCloud/EdgeCom.2019.00020. ISBN 978-1-7281-1661-7. Dostupné z: https://ieeexplore.ieee.org/document/8854045/

[30] KACHAVIMATH, Amit V, Shubhangeni Vijay NAZARE a Sheetal S AKKI, 2020. Distributed Denial of Service Attack Detection using Naïve Bayes and K-Nearest Neighbor for Network Forensics. *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)* [online]. IEEE, 2020, 711-717 [cit. 2020-06-01]. DOI: 10.1109/ICIMIA48430.2020.9074929. ISBN 978-1-7281-4167-1. Dostupné z: https://ieeexplore.ieee.org/document/9074929/

Seznam symbolů, veličin a zkratek

DDoS	Distributed Denial of Service – Distribuované odepření služeb
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoS	Denial of Service – Odepření služeb
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System – Systém detekce vniknutí
IPS	Intrusion Prevention System – Systém prevence vniknutí
ISP	Internet Service Provider – Poskytovatel internetových služeb
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Seznam příloh

A Použití skriptu útoku SlowDrop

Požadavky pro užití a testování

Strana serveru – cílová oběť:

- Zařízení, na kterém bude spuštěn server (Apache, CentOS, atd.), které bude umístěno v testovacím prostředí – LAN kontrolované laboratoře.
- Soubory větších velikostí dostupné prostřednictvím HTTP GET požadavků umístěné na serveru.
- (Volitelné) Systém prevence/detekce škodlivého provozu (IPS/IDS) a další nástroje pro analýzu datového provozu jako např. Wireshark.

Strana útočníka – zařízení generující škodlivý provoz SlowDrop útoku:

- Zařízení generující škodlivý provoz pomocí přiloženého skriptu SlowDrop útoku. Toto zařízení musí disponovat nástrojem *iptables*, čili je vhodné použít OS na bázi Linux.
- Python interpreter verze 3.8.X pro spuštění skriptu SlowDrop útoku.
- Spojení mezi strojem útočníka a cílové oběti, přes které bude veden škodlivý provoz útoku.

Příkazy pro užití skriptu útoku

Jako první krok je potřeba otevřít terminál ve složce, ve které se nachází soubor slowdrop_mazanek.py. Tento soubor skriptu je náležitě doplněn v jeho obsahu pomocí vysvětlivek v angličtině, které komentují jednotlivé funkce a fáze generování útoku.

Pro zobrazení vysvětlivek *help* k jednotlivým parametrům příkazů pro skript útoku lze zadat do terminálu příkaz:

```
python3 slowdrop_mazanek.py -h
```

Pro generování útoku s výchozími hodnotami zadejte do terminálu příkaz:

```
python3 slowdrop_mazanek.py
-url 'http://1.2.3.4/picture.jpg'
```

Výše uvedený příkaz generuje provoz SlowDrop útoku (žádosti HTTP GET o soubor picture.jpg) oproti cílovému serveru oběti s příkladovou IP adresou 1.2.3.4.

Dále tento příkaz pracuje s výchozími proměnami této implementace SlowDrop útoku, které lze dle potřeby testovaného scénáře upravovat (přepínače uvedeny v závorkách):

- 20 vláken generujících HTTP GET žádosti. (-t 20)

- Náhodné časové rozestupy mezi generováním nových vláken – 0 až 3 sekundy, vrchní float hodnota. (-tgd 3)
- 30 sekund čekání před generováním nového HTTP GET požadavku po dokončení předchozího v rámci jednoho vlákna (-gn 30)
- 40 % zahazovací poměr, klíčový parametr SlowDrop útoku, float hodnota mezi 0 a 1 (-d 0.40)

Útok v konzoli terminálu informuje v angličtině (s cílem zvýšení přístupnosti uživatelů) o stavu jeho průběhu, počtu splněných HTTP GET žádostí a jejich časovém trvání.

Ukončení útoku se v terminálu provádí zasláním SIGINT (neboli ctrl+c), které „zabije“ všechna živá vlákna generující provoz útoku a odstraní přidaná pravidla z *iptables*. Ukončování živých vláken je z pohledu správy paměti procesů nesprávné, tudíž tato akce zobrazuje výjimky. Bohužel v jazyce Python aktuálně neexistuje úhledná možnost ukončení živých probíhajících vláken.

Pro případ selhání útoku SlowDrop v jeho průběhu bez řádného ukončení byla přidána možnost manuálního odstranění pravidel z *iptables* pomocí příkazu (pro výchozí zahazovací poměr -d 0.4):

```
python3 slowdrop_mazanek.py -rem 0.40
```

V případě toho, že je potřeba spouštět skript útoku prvotně lokálně (bez připojení k internetu), tak je potřeba nainstalovat knihovnu *requests* (Python knihovna zabývající se HTTP požadavky), která není automaticky obsažena. Příkaz pro instalaci knihovny pomocí nástroje *pip* vypadá následovně:

```
pip install requests
```

B Přiložené soubory

- `/` ..... kořenový adresář zipové složky přiložených souborů
  - `slowdrop_mazanek.py` ..... soubor skriptu útoku v jazyce Python
  - `README.md` .. textový soubor s informacemi o útoku a návodem k užití v angličtině