VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Minh Hien Nguyen

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

DETEKCE DOS A DDOS ÚTOKŮ ZAMĚŘENÝCH NA WEBOVÝ SERVER
DETECTION OF DOS AND DDOS ATTACKS TARGETING A WEB SERVER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Minh Hien Nguyen**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Karel Kuchař**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Minh Hien Nguyen ***ID:*** 211576
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Detekce DoS a DDoS útoků zaměřených na webový server

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je návrh a implementace detekčních metod probíhajících DoS a DDoS útoků zaměřenými na webový server. Implementované metody a výsledky budou následně prezentovány ve vlastním laboratorním prostředí, kde budou následně i otestovány. V teoretické části práce se student zaměří na rozbor současných DoS a DDoS útoků v souvislosti s útoky zaměřených na webový server (ISO/OSI L5–L7). V rámci praktické části student sestaví vlastní experimentální síť a zprovozní webový server. Mezi legitimním klientem a webovým serverem bude probíhat regulérní komunikace. Tento server následně vystaví vybraným DoS a DDoS útokům. Na základě nashromážděných dat z probíhajícího útoku a legitimní komunikace navrhne detekční, filtrační a mitigační metody těchto útoků (pomocí detekce anomálií a signatur). Tyto metody následně implementuje a otestuje. Všechny výsledky budou prezentovány v experimentálním prostředí na webovém serveru.

DOPORUČENÁ LITERATURA:

[1] JAAFAR, Ghafar A., Shahidan M. ABDULLAH a Saifuladli ISMAIL. Review of Recent Detection Methods for HTTP DDoS Attack. Journal of Computer Networks and Communications [online]. 2019, 2019, 1-10. DOI: 10.1155/2019/1283472. ISSN 2090-7141.

[2] WANKHEDE, Shreekh a Deepak KSHIRSAGAR. DoS Attack Detection Using Machine Learning and Neural Network. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) [online]. IEEE, 2018, 2018, , 1-5. DOI: 10.1109/ICCUBEA.2018.8697702. ISBN 978-1-5386-5257-2.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Karel Kuchař

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zabývá detekcí útoků DoS (Denial of service) a DDoS (Distributed Denial of Service) zaměřených na webový server. Cílem práce je navrhnout detekční metody, které budou následně otestovány. Rozbory útoků podle referenčního modelu ISO/OSI (International Organization for Standardization/Open Systems Interconnection) umožní pochopit znaky jednotlivých útoků. V praktické části se nachází nástroje, které slouží k realizaci útoků, dále generátory legitimního síťového provozu a zabezpečený webový server. Podstatná data jsou vytvořena z probíhajících útoků a komunikací běžných uživatelů a jsou významnou součástí navržených metod. Účelem vyhodnocení dosažených výsledků je zhodnotit efektivnost jednotlivých detekčních metod z pohledu přesnosti určení a časové náročnosti.

KLÍČOVÁ SLOVA

anomálie, DDoS, detekce, DoS, IDS/IPS, mitigace, signatury, strojové učení, útoky, webový server

ABSTRACT

The bachelor thesis deals with the detection of DoS (Denial of service) and DDoS (Distributed Denial of Service) attacks targeting a web server. This work aims to design detection methods, which will be subsequently tested. Analysis of attacks according to the ISO/OSI (International Organization for Standardization/Open Systems Interconnection) reference model will allow an understanding of the features of individual attacks. In the practical part, some tools are used to implement attacks, then there are generators of legitimate network traffic and a secure web server. Substantial data are created from ongoing attacks and communications of ordinary users. These data are an important part of the proposed methods. The purpose of assessing the achieved results is to evaluate the effectiveness of individual detection methods in terms of accuracy and time consumption.

KEYWORDS

anomaly, DDoS, detection, DoS, IDS/IPS, mitigation, signatures, machine learning, attacks, web server

NGUYEN, Minh Hien. *Detekce DoS a DDoS útoků zaměřených na webový server*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 59 s. Bakalářská práce. Vedoucí práce: Ing. Karel Kuchař

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Minh Hien Nguyen
VUT ID autora:	211576
Typ práce:	Bakalářská práce
Akademický rok:	2020/21
Téma závěrečné práce:	Detekce DoS a DDoS útoků zaměřených na webový server

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Karlu Kuchařovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Bezpečnost internetu může ovlivňovat náš každodenní život. Naše osobní data a další informace mohou být sdílená napříč všemi odvětvími. V budoucnu lze očekávat, že dojde k razantnějšímu digitalizování světa. Již nyní dochází k zákeřným útokům, které narušují normální chod mnoha institucí, firem a domácností. Nejčastěji se jedná o odcizení identity, znepřístupnění služby nebo dokonce k zašifrování dat. Pokud se uživatel stane obětí, má málo možností, jak se bránit. Mnohdy nepomůže ani k přistoupení podmínkám útočníka, proto je jediným vhodným řešením přeinstalování operačního systému a následné obnovení dat ze zálohy. Nejlepší opatření tkví v obezřetnosti, v prevenci a v nevyužívání nepodporovaných programů.

Každým dnem přibývá počet zařízení, která vyžadují síťová připojení. Pokud se k tomu přičtou i útoky, je velmi složité se náporům ubránit. Je zde proto důležité včasné zachycení opakujících a nežádoucích jevů již před samotným webovým serverem. Síťový provoz na webových serverech bývá proměnlivý, nejlepším řešením je kvalitně odladěný systém detekce průniku, který bude monitorovat síť v aktuálním čase, a v případě hrozby automaticky zasáhne.

V této bakalářské práci se řeší útoky na webové servery. Cílem je popsat základní útoky, zjistit jejich fungování a na základě zjištěných znalostí vytvořit efektivní opatření. V první části je popsána základní charakteristika nejvíce známých typů a způsobů znepřístupnění sítě. V druhé části je základní dělení a k tomu zkonkretizovaný příklad. Dále jsou útoky rozděleny dle referenčního modelu ISO/OSI od relační vrstvy až po aplikační vrstvu, díky tomuto zpřehlednění lze více porozumět principu konání útoků. V třetí části je sestaven scénář s konkrétní topologii, která nastíní problematiku zahlcených linek a webových serverů. Dále jsou zde popsány generátory legitimního síťové provozu a simulátory útoku. Následně jsou navrženy metody detekce, které se opírají o signatury útoku, anomálie provozu a o strojové učení. Všechny mechanismy jsou při dalším rozboru otestovány. Ve čtvrté části jsou realizovány útoky v uzavřené lokální síti, kde získané výsledky jsou porovnány.

1 Typy odepření služeb

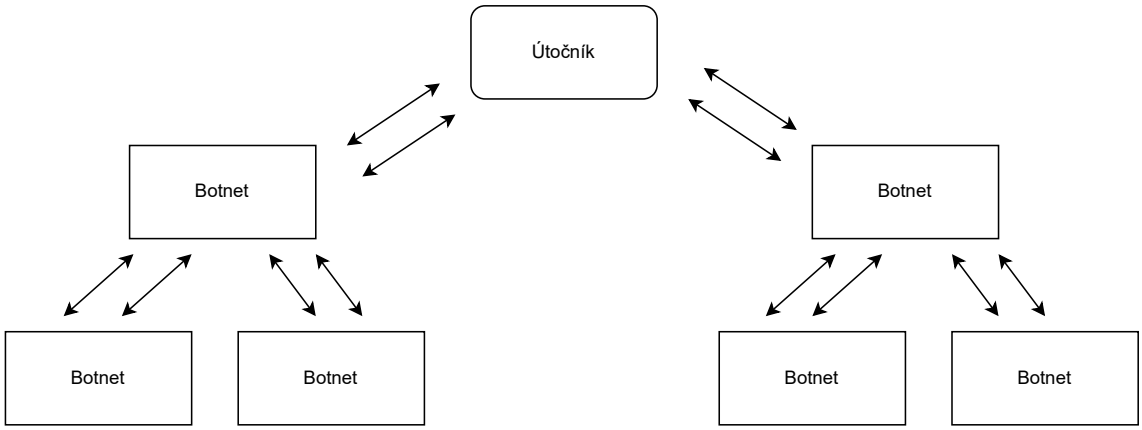
DoS (Denial of Service) je nejzákladnějším příkladem odepření služeb koncových zařízení. Nejčastěji se jedná o výpadek internetové služby, které je způsobeno přehlcením požadavků na cílový zdroj. Zdrojem je jedinec, který sám generuje objemné požadavky. V takových případech je útočník lehce zjistitelný.

Dalším typem je DDoS (Distributed Denial of Service), který se liší hlavně ve způsobu vzniku. Požadavky totiž přicházejí z různých koncových zařízení, následná identifikace je složitá. Kontextem těchto útoků bývá například v soupeření mezi společnostmi, vydírání nebo upozornění aktivistických skupin na sebe [1].

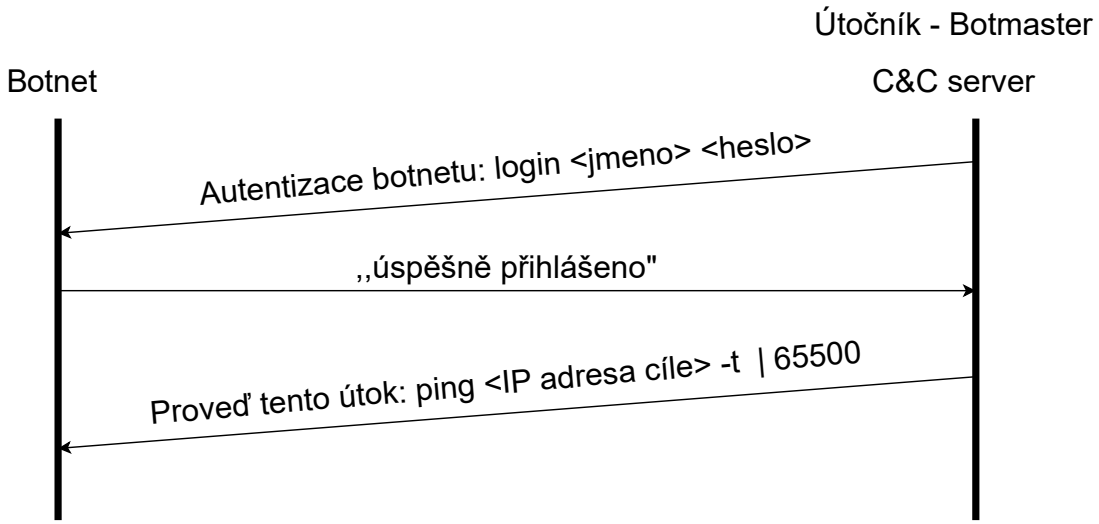
Nejsofistikovanější variantou je DDoS botnet. Dle průzkumu existuje na světě až 22 miliard připojených zařízení k internetu [2]. Cílem útočníků je rozesílat škodlivé kódy na zranitelné počítače, mobilní telefony a chytré hodinky nebo zkrátka na vše, co je připojené k internetu, díky tomu mohou posléze útočníci všechna zařízení dálkově ovládat. Botnet je zákeřný v tom smyslu, že jedná skrytě (vyžaduje malé množství výkonu), oběti (uživatelé) většinou netuší, že se podílejí na nekalých věcech. Například již zasažené koncové zařízení dokáže infikovat další a další. Tento dominový efekt dokáže vytvořit miliónovou armádu tzv. „zombies“, viz obr. 1.1 [3]. Prevence spočívá v aktualizování operačních systémů, a v neotvírání neznámých příloh v elektronické poště [4].
Typy komunikace botnetů:

- Klinet-Server:
  - Command-and-control (C&C) server,
  - IRC (Internet Relay Chat) otevřený protokol pro textovou komunikaci,
  - HTTP/HTTPS (Hypertext Transfer Protocol)/(Hypertext Transfer Protocol Secure) komunikace webového prohlížeče se serverem.
- Peer-to-peer:
  - využití způsobu decentralizace, každý klient je i zároveň serverem.

Command-and-control (C&C) server je mateřskou stanicí všech botnetů. Tento centrální uzel rozdává příkazy, viz obr. 1.2, nebo naopak ukládá získaná data od obětí. Komunikace probíhá nejčastěji pomocí protokolu IRC, který umožní připojení mnoha botnetů k jednomu severu. K autentizaci postačí jednoduché jméno, v některých případech je nutné se ověřit i heslem. Nevýhodou je viditelnost komunikace. Pokud si oběť všimne, že je jeho koncové zařízení ovládané útočníkem. Lze jednoduše vystopovat IP adresu C&C serveru a zablokovat případný další přístup. Proto většina útočníků přechází na systém Tor (The Onion Router), který zajišťuje absolutní anonymizaci spojení. Cesta musí být minimálně 3 skoky (hops) dlouhá. Jednotlivé uzly totiž uchovávají pouze dvě adresy a to adresu dalšího kroku a předešlého kroku. Z důvodu znáhodnění cest je tento průběh velmi pomalý [5].



Obr. 1.1: Popis hierarchie botnetů



Obr. 1.2: Princip ovládání botnetu: útok Ping of death

Formy projevu odepření webové stránky

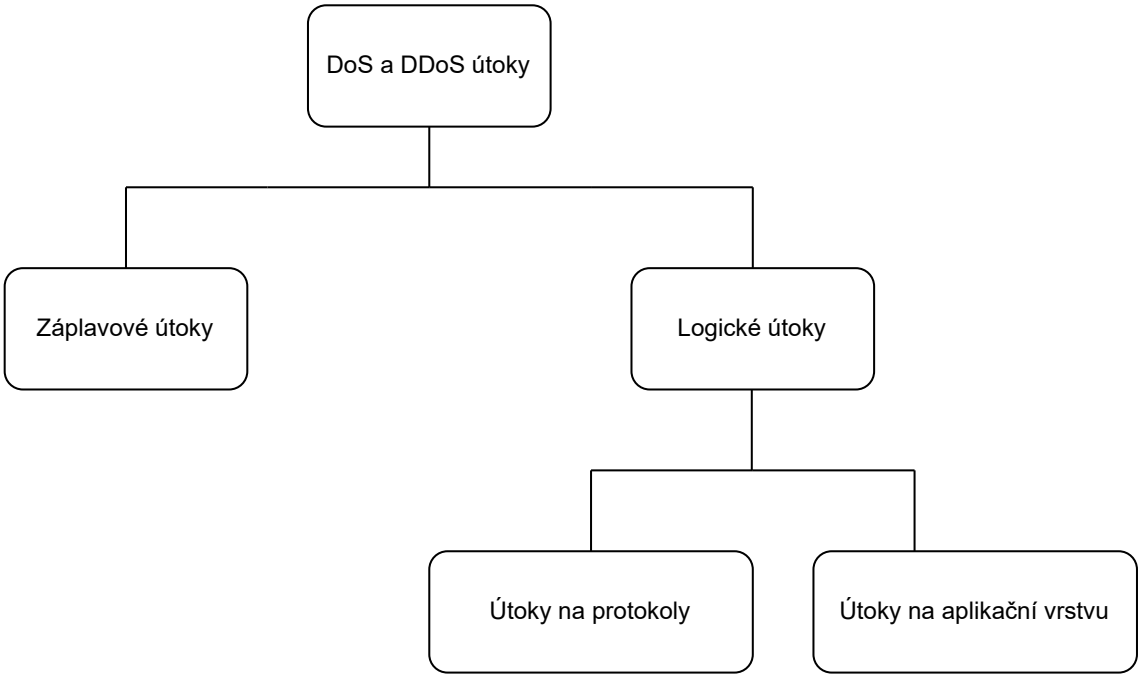
Existuje zde forma upozornění pro běžné uživatele o znepřístupnění služby. Tento způsob je však velmi primitivní a nelze nalézt přesnou příčinu. Dokáže však říci, že chyba je na straně webového serveru nikoliv u klienta. Samotný prohlížeč je vybaven chybovými kódy 5XX [6], které upozorní na podezřelou situaci na serveru:

- 503 Service Unavailable,
- 503 Service Temporarily Unavailable,
- HTTP (Hypertext Transfer Protocol) Error 503,
- HTTP (Hypertext Transfer Protocol) Server Error 503.

2 Základní dělení útoků

DoS a DDoS útoky jsou děleny podle různých znaků a složitosti, viz obr. 2.1. Nejběžnějším způsobem odepření služby na koncových zařízení je záplavový útok, který v jeden určitý okamžik systematicky zahltí linku nejen k webovému serveru, ale i jiné zařízení, které jsou přítomné na cestě jako je přepínač či směrovač. Výhodou může být, že takový velký provoz se nemusí dostat až k cíli, neboť útok může utnout hned první přístupový bod. Škody tedy nemusejí být až tak vysoké.

Na druhé straně jsou naopak důmyslnější útoky, které cílí na konkrétní vrstvu či protokol. Cílem těchto útoků je zvýšit pokud možno nenápadnost a hlavně zmást detekční algoritmy, které mohou být předem nakonfigurované na signatury, a proto je pro útočníky důležité splynutí s legitimním uživatelem nebo drobné vychýlení z pravidel jako je například velikost paketu. Zvláštní znaky mohou způsobit selhání detekčních mechanismů v horším případě i zacyklení celého webového serveru.



Obr. 2.1: Schéma obecného dělení DoS a DDoS útoků

2.1 Záplavové útoky

Záplavové útoky (Flooding attacks) jsou velmi primitivní a zastaralé, ale jejich funkčnost je v dnešní době velmi vysoká. Důvodem je velký počet zařízení v globální síti, které mohou být zneužity útočníky [7]. Princip spočívá ve vytížení ko-

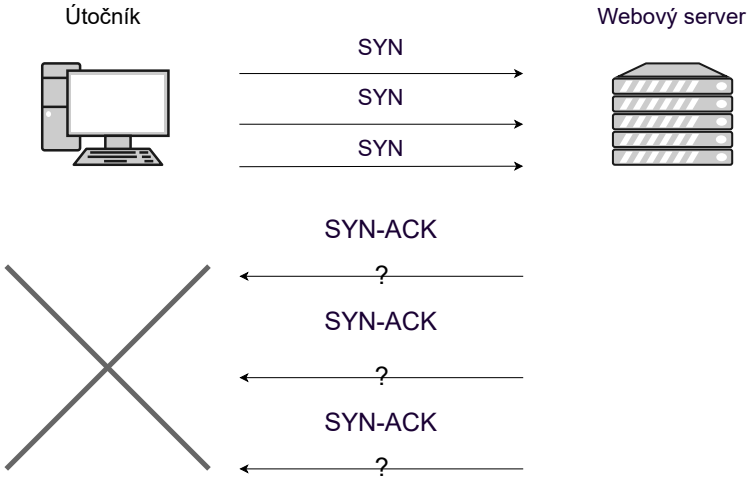
munikační a paměťové kapacity koncových zařízení (např. webový server). Z toho důvodu nejsou koncová zařízení již schopna poskytovat své služby. Útoky jsou zaměřené nejčastěji na protokoly TCP (Transmission Control Protocol), UDP (User Datagram Protocol), HTTP (Hypertext Transfer Protocol), ICMP (Internet Control Message Protocol) [8].

SYN flood

Před každým TCP/IP spojením mezi serverem a koncovým zařízením je nutné provést třícestný handshake. Klient jako první zasílá SYN paket za účelem navázání spojení a synchronizace se serverem. Ten mu potvrdí požadavek v podobě paketu SYN-ACK popřípadě ještě doplní, které konkrétní pořadí klient dostane. V posledním kroku klient zasílá finální odpověď ACK, po kterém se komunikační kanál otevře a data mohou oboustranně proudit.

SYN flood útok spočívá v částečném spojení, viz obr. 2.2, kdy útočník neustále vytváří nová a nová spojení, ale nikdy neodpoví na paket SYN-ACK, který je pro zahájení komunikace stěžejní. Tímto jednoduchým způsobem se velmi rychle zahltí kapacity koncového zařízení, neboť na každý požadavek musí zvlášť odpovědět.

Mezi metody obrany lze zařadit SYN cookies, webový server si do odpovědi zaznamená tzv. initial sequence number, který si vypočítá z parametrů, které získá v SYN paketu (IP adresa, porty atd.). Následně SYN paket od uživatele odebere z fronty, aby uvolnil místo dalším novým požadavkům. Když odpověď ACK dorazí, webový server zhodnotí tzv. initial sequence number, pokud mu přijde povědomé, spojení se vytvoří. Další metodou je snížení času na polootevřené spojení, může se ale stát, že zahodí i legitimního uživatele [9].



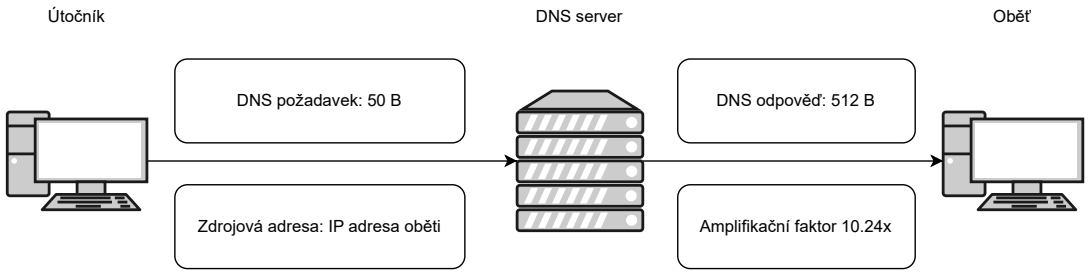
Obr. 2.2: Popis útoku SYN flood

2.2 Logické útoky

Logické útoky (Logical attacks) se zaměřují na nedostatky v samotném systému, softwaru nebo protokolu. Některá zařízení jsou ohrožena už jen tím, že mají stále nastavené výchozí konfigurace například původní přístupové údaje. Útočník získá přístup k informaci jako je například konkrétní verze operačního systému, na kterém webový server běží. Příkladem může být útok Slowloris, který využívá protokol HTTP. Požadavek na webový server je zcela legitimní, útok spočívá v nedostatečném nastavení systému, který umožňuje zbytečné a zdlouhavé udržování spojení.

Domain Name System reflection attack

Aby byl útok DNS (Domain Name System) reflection attack úspěšný, je potřeba spolupráce s více zařízeními například botnetů (skupina počítačů infikovaných virem). Tyto zařízení zasílají malé požadavky na DNS server. Hlavním úkolem DNS serveru je najít k doméně odpovídající IP adresu. Útočník sestaví svůj požadavek nejčastěji na předem připravenou doménu, viz obr. 2.3, která bude generovat odpovědi objemných velikostí, pomocí EDNS (Extension mechanisms for Domain Name System) lze i dosáhnout velikosti 4 KB [10]. Vše je zasláno na zdrojovou IP adresu. K požadavku je zdrojová adresa podvržena a místo ní použita IP adresa oběti. Amplifikační faktor udává míru zesílení, v tomto konkrétním případě dokáže pouhý jeden útočník vygenerovat z poměrně malého požadavku čítající 50 B na odpověď, která má hodnotu 512 B, zvětšení je tedy o 10,24 krát. Většinou se však jedná o celou armádu botnetů. Výsledkem je fatální selhání linky na straně příjemce. Přesné číslo zvětšení lze vypočítat následujícím způsobem (2.1) [11].

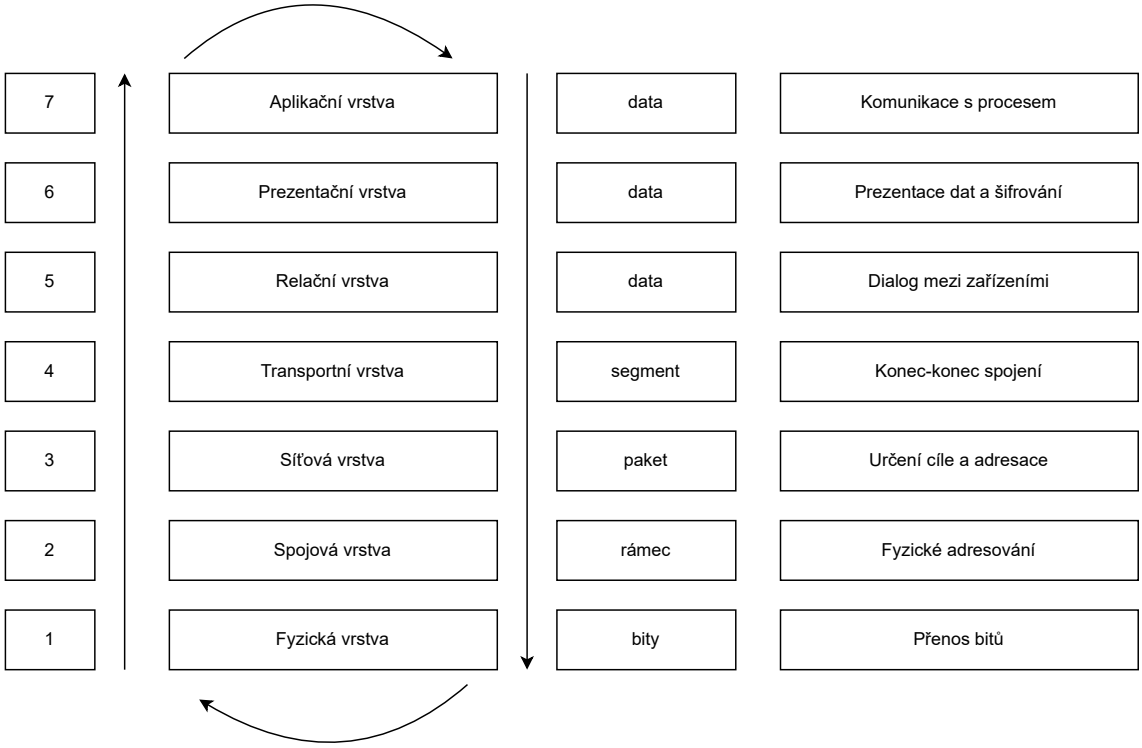


Obr. 2.3: Princip DNS reflection útoku

$$Amplifikační\ faktor = \frac{Velikost\ odpovědi}{Velikost\ požadavku} \tag{2.1}$$

2.3 Model ISO/OSI

Mezinárodní organizace ISO (International Organization for Standardization) je zakladatelem modelu OSI (Open System Interconnection Reference Model), jedná se o stěžejní jádro síťové komunikace. Skládá se ze sedmi vrstev, viz obr. 2.4, a zároveň všechny na sebe navazují. Definuje vzájemnou spolupráci nejen mezi počítači, ale i mezi dalšími periferními zařízení.



Obr. 2.4: Model ISO/OSI

2.3.1 Útoky na relační vrstvě

Relační vrstva především řídí dialog mezi nižší transportní vrstvou a vyšší aplikační vrstvou, tato vrstva musí mít neustálý přehled mezi dvěma koncovými zařízeními. Úkolem je sledovat datový tok, který musí být bezchybný. V případě výkyvu přenosu dat, lze využít službu synchronizace, která dokáže například udržet dialog, i když dojde ke ztrátě dat na některých nižších úrovní [12].

Útok na webový server pomocí Telnetu

Telnet byl první protokol na světě, který umožnil vzdálené připojení pomocí uživatelského rozhraní. Pomocí této služby lze ovládat počítače, směrovače, přepínače

nebo i drobné zařízení jako jsou IP kamery. K využití této funkce je zapotřebí zapnout port 23, na kterém oboustranné spojení probíhá. Bezpečnost je nulová, veškerá komunikace probíhá v otevřeném textu. To znamená, že i přihlašovací údaje včetně hesel jsou nezašifrované. Z toho důvodu se preferuje zabezpečená varianta SSH (Secure Shell). I když se Telnet z bezpečnostního hlediska již moc nevyužívá, je stále aktivní v mnoha zařízeních [13]. Vysvětlením může být výchozí nastavení koncových zařízení, kde například firewall může být nastavený metodou co není zakázáno, je povoleno. Jelikož se jedná o neúmyslné nastavení této služby, bývají přihlašovací údaje v podobě „admin/admin“, „root/root“. V nejhorších případech je nastaveno pouze uživatelské jméno bez hesla, například v cisco zařízeních [14].

Útočníkovi stačí oskenovat síť se zaměřením na otevřené porty. Pokud oběť nalezne, začne s konáním útoku. K úspěšnému prolomení přístupu do Telnetu existuje několik variant. Tou první je útok hrubou silou, kde například pomocí předem definovaného slovníku jde otestovat různá pořadí znaků, písmen nebo čísel. Další možností je odposlouchávání celé sítě, zda například správce webového serveru nevyužívá Telnet ke vzdálenému přístupu, pokud ano, útočník zachytí přihlašovací údaje v otevřeném textu. Třetí způsob je založen na zranitelnosti samotného protokolu nebo jiných aplikací a služeb na straně oběti. Také lze použít předem nadefinované nástroje, které dokáží prolamovat Telnet. Po úspěšnému přihlášení může útočník provádět útoky lokálně. Se získanými znalostmi půjde jednoduše odepřít službu na konkrétním serveru nebo mazat data [14].

Telnet flood

Cílem útočníka je znepřístupnit tuto službu na všech zařízeních, které může správce sítě vzdáleně ovládat. Většinou to jsou starší přepínače nebo směrovače. Realizace může být opřena o spolupráci botnetů nebo v zaslání objemného požadavku na port 23, na kterém Telnet operuje.

Telnet útoky – obranné opatření

Zakázat veškeré nevyužívané a otevřené porty. Ke vzdálenému připojení využívat bezpečnější variantu SSH. Vytvořit seznam důvěrných IP adres, ze kterých bude vzdálený přístup povolen, v opačném případě dojde k zamítnutí. Dále nastavit dostatečně silné heslo a neustále aktualizovat programy, které mají přístup k otevřeným portům. Alternativním řešením může být využití VPN (Virtual Private Network), který vytvoří zašifrovaný komunikační tunel.

2.3.2 Útoky na prezentační vrstvě

Hlavním úkolem prezentační vrstvy je příprava dat pro aplikační vrstvu a naopak. Díky možnosti transformace lze různé kódy a abecedy převést do přenosové podoby, proto lze říci, že aplikační vrstva může použít jakoukoliv syntaxi. Další významnou výhodou je fakt, že jako jediná vrstva ze všech sedmi vrstev dokáže manipulovat s uživatelskými daty, díky tomu lze komprimovat, dekomprimovat, šifrovat a dešifrovat komunikaci [15].

Secure Sockets Layer útok

SSL (Secure Sockets Layer) je protokol, který zaručuje šifrovanou komunikaci mezi počítači nebo aplikacemi (poštovní zprávy, bankovnictví). Výsledkem je zabezpečená stránka HTTPS, která je v dnešní době značně rozšířená, tím pádem roste i počet SSL handshake flood útoků [16].

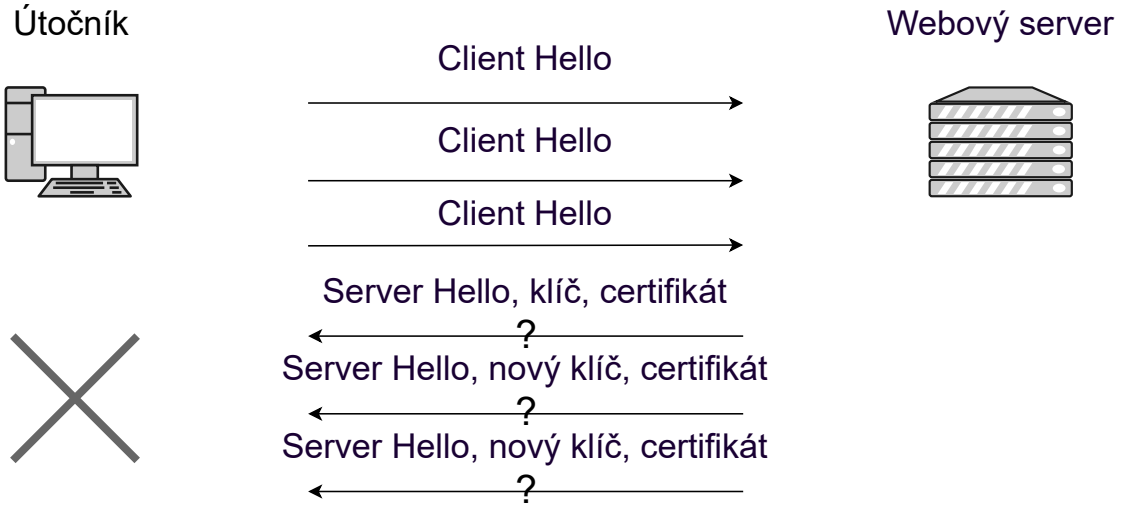
Tento útok cílí na proces zahájení šifrování tedy vyjednávání klíčů a zaslání ověřeného certifikátu. Tento úkon je pro webový server velmi náročný. Skládá se z několika částí. Nejdříve útočník sestaví legitimní požadavek na webový server, ve kterém uvede podrobnosti a důležité informace k zahájení šifrování. Ve zprávě `Client Hello` je uvedena verze protokolu, náhodné prvočíslo (Random), hashovací funkce (Secure Hash Algorithm 256), podpisové algoritmy s využitím eliptických křivek (Elliptic Curve Digital Signature Algorithm) a Cipher suite. Cipher suite je předem nadefinovaná kombinace autentizačních a šifrovacích algoritmů, účelem je vytvořit jednotný formát, který zpřehlední tvorbu zabezpečeného spojení. Server na základě těchto získaných informací, musí vytvořit parametry, které dokáží vyhovět oběma stranám. Útočník však úspěšně vytvořené spojení ignoruje a trvá na znovu vyjednávaní (renegotiate). Tímto způsobem výpočetně zatíží procesory daného webového serveru [17].

Secure Sockets Layer útoky – obranné opatření

Prvním řešením je blokovat všechny uživatele, kteří ve velmi krátkém časovém sledu opakovaně požádají o nové parametry klíče k ověření certifikátu. Druhé řešení je v ignorování těchto žádostí, neboť komunikace funguje a není důvod ke znovu generování nových parametrů. Nejlepší řešení bude v analyzování celkového provozu tedy nejlépe zachycovat možné útoky v uzlech již před webovým serverem. Tuto funkci plní například Honeypot.

Secure Sockets Layer traffic flood

Jedná se o záplavový útok, který využívá protokol SSL (Secure Sockets Layer). K úspěšnému spojení dvou koncových zařízení je potřeba úmluva, která sjednotí parametry tak, aby komunikace mohla probíhat zabezpečeně. Cílem útoku je zahltit volnou kapacitu možných otevřených spojení pro daný webový server. Princip spočívá v nekonečném otevírání spojení, která však nikdy nejsou dokončena, viz obr. 2.5. Legitimní uživatelé jsou zcela ignorováni, proto je vhodnější při podezření na útok si tyto uživatele udržet, aby se snížila volná kapacita pro útočníky.



Obr. 2.5: Záplavový útok na protokol SSL

Secure Sockets Layer traffic flood – obranné opatření

Nejúčinnějším řešením je omezit počet spojení z jedné konkrétní IP adresy. Nepředpokládá se, že by běžný uživatel měl v prohlížeči otevřeno 30 aktivních oken v jeden okamžik na stejnou webovou stránku, kterou pravidelně aktualizuje. U legitimních uživatelů může dojít k omezení, proto je nutné využívat automatizované detekce, které tyto útoky založené na polootevřených spojení zahodí.

2.3.3 Útoky na aplikační vrstvě

Aplikační vrstva umožňuje širokou škálu funkcí, tou nejdůležitější je umožnění komunikace procesů s celým ISO/OSI modelem. Díky tomu lze komunikovat mezi různými aplikacemi v operačním systému [18].

Významné protokoly na aplikační vrstvě:

- FTP (File Transfer Protocol) slouží k přenosu souborů,
- HTTP (Hypertext Transfer Protocol) základní přenosový protokol webových stránek,
- DHCP (Dynamic Host Configuration Protocol) přiřazení adres na lokálních sítích,
- SMTP (Simple Mail Transfer Protocol) přenos elektronické pošty,
- DNS (Domain Name System) protokol pro správu doménových jmen v celém internetu.

Slowloris

Slowloris je útok na aplikační vrstvě, který opakovaně odesílá neúplné HTTP požadavky. Tyto nedokončené GET requesty účelně nutí protistranu čekat. Při větší zátěži dochází k vyčerpání kapacity koncového zařízení [19]. Příklad nedokončeného HTTP requestu je na výpise 2.1, kde na konci chybí znak uzavření `\r\n`.

Výpis 2.1: Hlavička nedokončeného GET požadavku

```
1 GET / HTTP1.1\r\n
2 HOST: 192.168.43.130\r\n
3 User-Agent: Mozilla/5.0 Firefox/68.0\r\n
4 Accept-Encoding: gzip, deflate\r\n
```

Slow Post Attack

Slow Post Attack je jednoduchý útok, který je založený na HTTP požadavku POST. Ten má za úkol předávat informace od uživatele k serveru. Nejčastěji se jedná o registraci, přihlášení nebo zápis. Nejdůležitějším parametrem je zde Conten-Lenght, který určuje délku přenášených dat. Tuto skutečnost však útočník zneužije a zadá nadměrně vysokou hodnotu. Ta způsobí čekání na straně koncového zařízení. Příklad takové požadavku je na výpise 2.2.

Výpis 2.2: Hlavička modifikovaného POST požadavku

```
1 POST / registrace-formular.html HTTP1.1\r\n
2 HOST: 192.168.43.130\r\n
3 User-Agent: Mozilla/5.0 Firefox/68.0\r\n
4 Content-Lenght: 50000000000000\r\n
5 \r\n
```

Slowloris a Slow Post Attack útoky – obranné opatření

Snížení čekací doby na dokončení požadavku. Průběžná analýza obsahu příchozích hlaviček, kontrola velikosti a uzavírání nečinných komunikací. Omezení počtu spojení z jedné IP adresy je účinné pouze tehdy, pokud je útok veden z jednoho zdroje.

3 Příprava experimentálního testování

3.1 Scénáře experimentálního testování

Testovací část této práce je zaměřena na útok aplikační vrstvy Slowloris. Dalším stěžejním útokem je útok SSL handshake flood, který operuje na prezentační vrstvě. Útok Slowloris je detekován pomocí signatur a strojového učení. Řešení je dále rozšířeno o metodu anomálie, která má za cíl zjistit, jestli jsou logické útoky obecně detekovatelné na základě odchýlení zátěže sítě od běžného stavu. Útok SSL flood má podobné metody detekce, kde detekce pomocí anomálie je založena na analýze počtu otevíraných zdrojových portů. Dále jsou zde také zpracovány útoky transportní vrstvy TCP SYN flood a UDP flood, které slouží k otestování variabilnosti jednotlivých detekčních metod, viz tab. 3.1. Primárním cílem je zachytit bezpečnostní incident bez ohledu na principu útoku a vrstvy konání útoku.

Tab. 3.1: Přehled útoků simulovaných v experimentální části

Název útoku	Síťová vrstva	Princip útoku	Metoda detekce
Slowloris	aplikační	vyčerpání kapacit nových spojení	signatura/iptable anomálie strojové učení
SSL flood	prezentační	vytížení procesorů	signatura/iptable anomálie strojové učení
TCP SYN flood UDP flood	transportní	vyčerpání kapacit nových spojení	strojové učení

Testování má za cíl zjistit mezní únosnost webového serveru, který běží na operačním systému `Ubuntu`. Dále reakce takového stroje na útok a seznámení se se situací, která není přívětivá. Například práce s terminálem a otevírání logů, zda jsou nefunkční po celou dobu nebo v jen určitých intervalech.

Ověření zda opravdu došlo k odepření služby, je velice jednoduché. DoS a DDoS útoky mají vždy stejné dopady, které se projevují zpomalením nebo v horším případě výpadkem celé služby. Je těžké rozeznat, o jaký typ útoku se jedná, neboť je výsledek vždy stejný. Dokonce velký nápor legitimních uživatelů může způsobit zahlcení linky. Cílem bude rozeznat chování jednotlivých útoků, jakým způsobem se projeví a kolik času je k tomu potřeba. Na základě zjištěných informací lze vytvořit detekční mechanismy, které budou situaci dále a neustále vyhodnocovat.

První způsob detekce využívá samotnou znalost signatur. Tato metoda spoléhá na to, že většina dnešních útoků mají specifickou strukturu nebo formu, podle kte-

rých je lze lehce rozeznat. Například útok Ping of death je založený na principu zasílání překročení limitních velikostí IP paketu, výsledkem je selhání počítače nebo útok Slowloris, který záměrně neukončuje HTTP GET hlavičky. Znalosti je potřeba neustále aktualizovat, neboť útočníci cíleně modifikují již existující útoky tak, aby zmátli detekční mechanismy. Iptables (`Linux`) nebo také označení firewall (u operačních systémů `Windows`) umožní kontrolu nad síťovým provozem. Logické řešení je nastavení těchto bran u vstupu do vnitřní sítě, kde může být webový server. Hlavní výhodou je pak skutečnost, že veškerý provoz je zachytáván, a pokud není nastaveno žádné limitující pravidlo, jsou pakety vpuštěny dále. Vytvořený seznam pravidel dokáže spolehlivě odfiltrovat podezřelé nebo přímé útoky. Dále je vhodné si část provozu uložit k pozdější analýze. Při neobvyklé aktivitě v síti lze pak jednoduše rozlišit útok nebo zvýšený provoz. Informace o zdroji a cíli usnadní blokaci nebezpečného paketu. Pokud vznikne podezření, že webový server je již pod nátlakem útoků, tuto situaci je nutné podrobněji prozkoumat pomocí jednoduchých nástrojů jako je například `netstat`, který funguje v terminálu pro Linux nebo v příkazovém řádku pro Windows. Umí zobrazit všechna aktivní spojení. Pomocí filtrů lze nastavit kolik spojení je tvořeno jednou konkrétní IP adresou [20].

Druhá metoda detekce (anomálie) bude hlavně využívat data z programu Zeek, který zachytává a zaznamenává veškerý síťový provoz. Hlavním cílem je vytvořit na základě strojového učení univerzální detekční mechanismus, který by dokázal detekovat co nejvíce variací DDoS útoků. Je proto nutné se zaměřit na znaky dopadu, které mají všechny tyto útoky společné. Analýza dat bude zpracována na základě předem nadefinovaných podmínek vytvořených v jazyce `Python`, tyto podmínky budou navrženy tak, aby vyhovovaly parametrům a rozsahu dané konkrétní sítě a v konkrétním čase. Výstupní hodnoty budou velmi významné pro strojové učení, je tedy potřeba porovnat jednotlivé algoritmy strojového učení vzhledem ke vstupním parametrům. Mezi zkoumané algoritmy patří Algoritmus k-nejbližších sousedů, Rozhodovací stromy a Metoda podpůrných vektorů. U Algoritmu k-NN (k-nejbližších) sousedů bude sledován vliv počtu sousedů na konečný výsledek, u Rozhodovacího stromu vliv množství dat na výsledek a u Metody podpůrných vektorů rozdíly mezi lineární funkcí, polynomiální funkcí a sigmoidní funkcí.

Třetí metoda detekce se opírá o stejná data jako v detekci pomocí strojového učení, ale liší se způsobem zpracování dat, kde výstupní parametry síťové provozu budou využity jako proměnné hodnoty do vzorců, kde výstupem bude rozhodnutí, zda se jedná o legitimní provoz nebo o útok.

K úspěšnému otestování navržených metod je důležité zvolit simulátory útoku, které vykonají přesné a charakteristické znaky tak, aby navržené metody opravdu fungovaly. Příkladem je simulátor útoku Slowloris [21], který vytváří nedokončené hlavičky HTTP GET, některé nástroje pod stejným názvem však fungují s odliš-

ným principem generování útoku například nikoliv pomocí HTTP hlaviček, ale pomocí TCP záplavového útoku. Pro rozmanitost útoků byl zvolen program Pentmenu [22], který v sobě obsahuje hned několik. K zachycení komunikace v síti byl využit program Zeek [23], který zaznamenává parametry provozu. Pro simulování chování legitimního uživatele byl využit virtuální server Penterepmail [24], který má webovou aplikaci, kde na základě proklikávání různými sekci lze zaznamenat potřebné parametry pro analýzu metod určené pro aplikační vrstvu. Využití síťových generátorů umožnilo vytvořit provoz, který plní znaky reálného prostředí.

3.2 Příprava experimentálního prostředí

K vytvoření experimentální sítě byl využit počítačový software GNS3. Tento počítačový software GNS3 je program s otevřeným kódem, který umožňuje simulovat síťové prostředí na operačních systémech Windows, MacOS a Linux. Pro možnost využití všech funkcí, které program nabízí, je doporučeno od vývojářů také nainstalovat virtuální server GNS3, který umožní lepší dosáhnutí výsledků simulací. Díky této virtualizaci lze rozšířit virtuální síťovou topologii o další drobná zařízení jako jsou routery nebo switche, která jsou klíčová pro zapojení dalších virtuálních strojů.

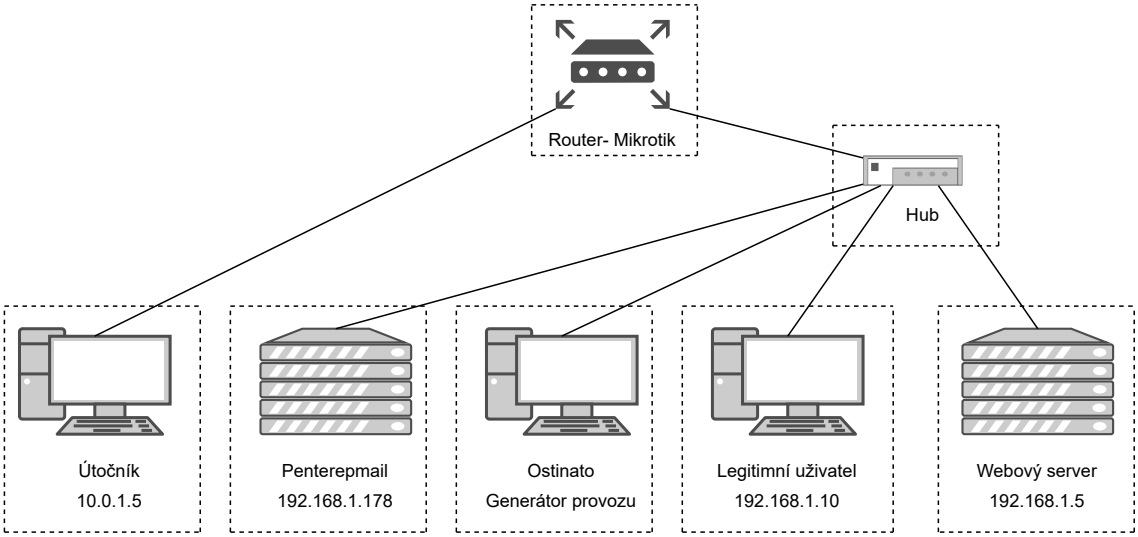
Instalace se skládá ze dvou částí. První část obsahuje pouze import serveru GNS3 do virtuální stroje VMware pomocí souboru .ova a následné spuštění. Žádné další nastavení není vyžadováno. Druhá část je instalace klienta, tento klient je uživatelsky přívětivý a zajišťuje veškerou konfiguraci nebo podobu sítě, dále zde lze nastavit síťové rozhraní nebo importování nových zařízení. Po úspěšné instalaci je nutno nastavit propojení klienta se serverem, který běží ve virtuálním stroji VMware [25].

Do topologie lze umístit základní zařízení typu směrovač, přepínač nebo rozbočovač. Pro účely testování byl využit volně dostupný směrovač MikroTik, který je určen k virtualizaci v prostředí programu GNS3. Největší výhodou lze považovat možnost importování virtuálních strojů běžících v programu VMware, virtuální stroje postačí vložit do příslušného adresáře, ze kterého program automaticky rozpozná a vyobrazí v nabídce pro uživatele možnost přidání. Dále je nutno mít nastavený síťový adaptér pro každý virtuální stroj zvlášť, který bude spojen s virtualizací GNS3.

Nástroje použité v topologii:

- **operační systémy**: Kali Linux [26], Ubuntu Linux [27],
- **webový server**: Apache2 ve verzi 2.4.41, 4GB RAM (Random Access Memory), 40GB kapacita úložiště,
- **simulátory útoku**: Pentmenu, Slowloris,
- **IDS**: Zeek,
- **virtuální server**: Penterepmail,
- **generátory**: PackETH [28], Nping, D-ITG [29], Ostinato [30].

Testovací topologie se skládá ze čtyř virtuálních strojů a jednoho generátoru provozu sítě, viz obr. 3.1. Síť má dvě části. V té první se nachází útočník s operačním systémem `Kali Linux`. Ve druhé části je operační systém `Ubuntu Linux`, na kterém je nainstalovaný webový server `Apache2`, protože je k útoku typu Slowloris nejvíce náchylný [31]. Dle základní konfigurace u `Apache2` činí čekací lhůta až 300 sekund [32]. Nachází se zde také legitimní uživatel, na kterém jsou nainstalovány softwarové generátory kompatibilní s operačním systémem `Linux`. A generátor Ostinato, který je nezávislý a má své vlastní uživatelské rozhraní a porty. Všechny zařízení v druhé části jsou připojeny ke směrovači pomocí rozbočovače. Celá síť je z bezpečnostního důvodu bez přístupu k internetu.



Obr. 3.1: Topologie sítě

Instalace webového serveru

Instalace webového serveru ve verzi Apache2 na operačním systému `Ubuntu` vyžaduje několik úkonů. Tou první je aktualizování seznamu všech balíčků příkazem `sudo apt-get update`. Dále instalování webového serveru `sudo apt install apache2`. Služba je po instalování ihned aktivní, k ověření může posloužit příkaz `sudo service apache2 status` nebo navštívení služby pomocí prohlížeče. Zobrazení IP adresy lze pomocí `ifconfig` z balíčku `net-tools`. Obecné řízení provozu Apache2, viz výpis 3.1.

Výpis 3.1: Příkazy k řízení obecného provozu webového serveru

```
# service apache2 start          //spuštění
# service apache2 stop           //zastavení
# service apache2 status         //zobrazení stavu
# service apache2 restart        //restartování
```

Zabezpečení webového serveru

Záruka bezpečné komunikace mezi webovým serverem a klientem je podmíněna certifikátem. Self-signed certificate (certifikát podepsaný sám se sebou) je specifická forma digitálního certifikátu, který slouží pro testování a experimentální účely. Některé prohlížeče výstražně informují uživatele, že spojení není stále zabezpečené, neboť nepovažují tento způsob zabezpečení za věrohodné. Tento certifikát však nadále umožňuje komunikovat přes protokol HTTPS, který operuje na portu 443.

Nejprve je nutné v Apache povolit modul `mod_ssl`, který šifrování umožní. K generování nového certifikátu je potřeba několik parametrů. První `openssl` vytváří a spravuje certifikáty i klíče. Druhý `req -x509` specifikuje formát certifikátu. Třetí `-nodes` předá webovému serveru pokyn, aby nevyžadoval heslo po každém restartování. Čtvrtý `-days` nastavuje platnost certifikátu, doporučeno je do jednoho roku, protože mnoho moderních prohlížečů zamítá s delší platností. Pátá `-newkey rsa` založení klíče a určení konkrétního šifrovacího algoritmu s konkrétní délkou, minimum je 2048 bitů. O uložení privátního klíče a certifikátu poslouží tyto příkazy `-keyout` /definice cesty/ klíče a `-out` /definice cesty/ certifikátu. Finální podoba příkazu je zobrazena na výpise 3.2. Dalším obsahem pro certifikát jsou samotné osobní údaje, které jsou vhodné vyplnit, viz výpis 3.3. Důležité přesměrování nezabezpečeného provozu z HTTP na HTTPS je nastaveno v souboru `000-default.conf` v adresáři `/etc/apache2/sites-available/`, viz výpis 3.4. Původní nastavení konfiguračního souboru `192.168.1.5.conf` v adresáři `/etc/apache2/sites-available/` je nutno přepsat na nově vytvořený certifikát a klíč, viz výpis 3.5 [33].

Výpis 3.2: Parametry k vytvoření certifikátu SSL

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048
```

Výpis 3.3: Doplnění osobních údajů

```
Country Name (2 letter code): CZ
State or Province Name (full name): Jihomoravsky kraj
Locality Name (eg, city) : Brno
Common Name: 192.168.1.5
```

Výpis 3.4: Přesměrování z portu 80 na port 443

```
<VirtualHost *:80>
    ServerName 192.168.1.5
    Redirect /https://192.168.1.5/
</VirtualHost>
```

Výpis 3.5: Nastavení konfiguračního souboru

```
<VirtualHost *:443>
   ServerName 192.168.1.5
   DocumentRoot /var/www/192.168.1.5

   SSLEngine on
   SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
   SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key
</VirtualHost>
```

3.3 Instalace systému Suricata

Suricata je nástroj, který monitoruje a detekuje provozní výkyvy v sítích, na základě předem určených pravidel vytřídí pakety na nebezpečné nebo na legitimní. Při detekci hrozby je vše okamžitě zaznamenáno do souboru s příponou `.log`, dále je zde zaznamenána posloupnost všech událostí tak, aby byla pozdější analýza jednodušší. Společně s webovým serverem tvoří účinný bezpečný zautomatizovaný systém.

Před instalací je vyžadováno aktualizování seznamu balíčku v operačním systému `Ubuntu` příkazem `apt-get update`. Dále je nutné mít dostupné příslušenství ke kompilaci nástroje Suricata, viz výpis 3.6. Následně je potřeba uvést do provozu všechny příkazy zobrazených na výpise 3.7.

Výpis 3.6: Souhrnné příslušenství ke kompilaci nástroje Suricata

```
apt-get install rustc cargo make libpcre3 libpcre3-dbg libpcre3-dev build-essential
     autoconf automake libtool libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev
    zlib1g zlib1g-dev libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev
    libjansson4 pkg-config -y
```

Výpis 3.7: Uvedení příkazů do provozu

```
apt install python3
apt install python3-pip
pip3 install --upgrade suricata-update
ln -s /usr/local/bin/suricata-update /usr/bin/suricata-update
```

Samotná instalace nejnovější verze Suricaty je na výpise 3.8. Tento průběh zahrnuje i kompilaci celého nástroje, který vyžaduje delší časovou kapacitu. Po úspěšném dokončení celého procesu, je písemně indikováno na konci terminálu `You can now start suricata by running as root` [34].

Výpis 3.8: Kompletní instalace nástroje Suricata

```
wget https://www.openinfosecfoundation.org/download/suricata-6.0.0.tar.gz //
    Ostatní verze jsou stále ke stažení na adrese:https://www.
    openinfosecfoundation.org/downloads/
tar -xvzf suricata-6.0.0.tar.gz
cd suricata-6.0.0
./configure --enable-nfqueue --prefix=/usr --sysconfdir=/etc --localstatedir=/var
make && make install-full
```

Je potřeba nastavit správné parametry ve výchozím nastavení tak, aby detekční systém mohl korektně monitorovat síť. V souboru `suricata.yaml`, který je umístěn v adresáři `/etc/suricata/`, je nastaven rozsah sítě, ve kterém webový server operuje, dále je přiložen soubor `webserver.rules`, ve kterém jsou aplikovaná pravidla pro zachytávání hrozeb na webový server [34]. Oba postupy, viz výpis 3.9.

Výpis 3.9: Nastavení parametrů pro nástroj Suricata

```
vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[192.168.1.0/24]"
-
-
-
default-rule-path: /var/lib/suricata/rules
rule-files:
  - suricata.rules
  - webserver.rules
```

Finální spuštění detekčního nástroje musí probíhat na správném rozhraní webového serveru, podrobněji zobrazeno na výpise 3.10. Po každém vložení, úpravě či smazání pravidla v souboru `webserver.rules` je nutný restart systému, neboť díky tomu lze zabránit chybnému spuštění.

Výpis 3.10: Úspěšné spuštění nástroje Suricata

```
root@ubuntu:~/suricata-6.0.0# suricata -c /etc/suricata/suricata.yaml -i ens33
29/11/2020 -- 05:29:01 - <Notice> - This is Suricata version 6.0.0 RELEASE running
    in SYSTEM mode
29/11/2020 -- 05:29:18 - <Notice> - all 2 packet processing threads, 4 management
    threads initialized, engine started.
```

3.4 Instalace systému Zeek

Před samotnou instalací programu Zeek je nutné mít v operačním systému `Ubuntu` nainstalované požadované knihovny a nástroje, které zaručí správný chod celého systému. Tento úkon vyžaduje následující příkaz, viz výpis 3.11.

Výpis 3.11: Příkaz obsahující potřebné knihovny a nástroje

```
sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python3
    python3-dev swig zlib1g-dev
```

Klonování umožní přenést celý zdrojový adresář ze stránek vývojářů na potřebné místo úložiště v síti, není tedy potřeba pracovat se soubory s příponou `.tar.gz`. Samotná instalace je složena ze tří příkazů, první `./configure` se stará o kontrolu splněných požadavků potřebné ke kompilaci a následující příkazy `make` a `make install` celý instalační proces dokončí, viz výpis 3.12.

Výpis 3.12: Příkazy potřebné k instalaci systému Zeek

```
./configure
make
make install
```

Po úspěšné instalaci je nutné nastavit prvotní parametry v souborech umístěné v adresáři `/usr/local/Zeek/etc/`. V souboru `networks.cfg` lze nastavit vnitřní síť, ve kterém se nachází webový server, a také venkovní síť. Z důvodu laboratorního prostředí byly rozsahy IP adres zvoleny z privátní sítě, viz výpis 3.13.

Výpis 3.13: Nastavení sítě v systému Zeek

```
10.0.1.0/24         Public IP space
192.168.1.0/24      Private IP space
```

Dále v souboru `node.cfg` je nutno určit konkrétní síťové rozhraní, na kterém bude systém zachytávat pakety ze síťového provozu, viz výpis 3.14.

Výpis 3.14: Konkretizace síťových rozhraní

```
[Zeek]
type=standalone
host=localhost
interface=ens33
```

Základní znalosti nutné k orientaci v systému Zeek, viz výpis 3.15. Konkrétně se jedná o spuštění programu, umístění logů nebo adresář určený pro vkládání vlastních signatur podle předem definovaných pravidel.

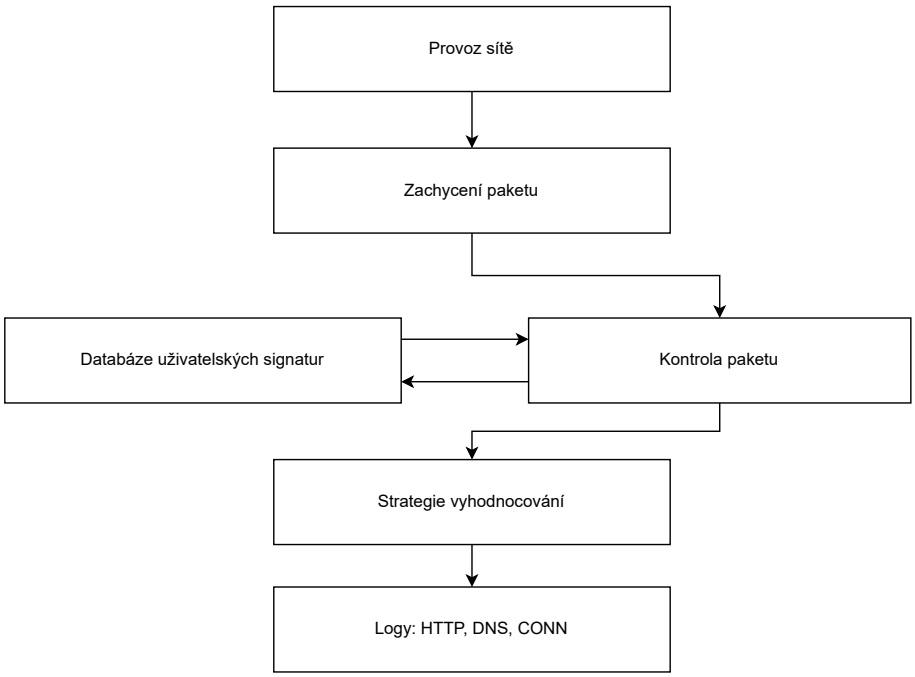
Výpis 3.15: Základní funkce v Zeeku

```
/usr/local/Zeek/bin/Zeekctl                                  // spuštění Zeeku
/usr/local/Zeek/logs/current                                 // umístění logů
/usr/local/Zeek/share/Zeek/base/frameworks/signatures        // vkládání signatur
```

3.5 Detekování pomocí programu Zeek

Jedná se o obranný systém, který pouze zaznamenává aktivity a detaily v síti. Tento systém vyhodnocuje v reálném čase tři základní způsoby provedení útoku v provozu. První případ je úmyslné zahlcení paketů, které zpomalí rychlost celé sítě nebo dojde k výpadku konkrétní služby. Druhý případ je útok, který vyvolá chybu v programu,

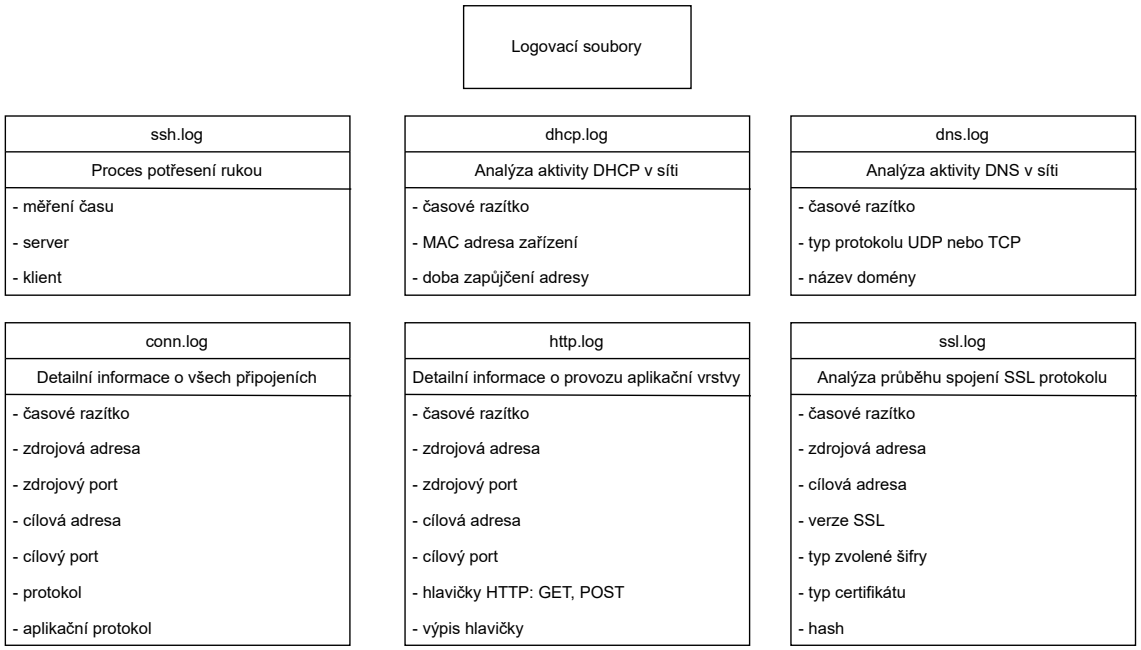
a tím ho vyřadí. A třetí případ jsou sofistikované útoky, které jsou nenápadné a nezanechávají za sebou žádné stopy. Na obr. 3.2 lze vidět schéma fungování sytému Zeek, program začíná proces zachycením paketu pomocí nástroje `libpcap`, následně jsou odchycené pakety vyhodnocovány a kontrolovány, zda splňují kontrolní součet IP hlavičky. Pokud ano, jsou dále zkoumána pomocí vlastních přidaných signatur, tento krok však není povinný, neboť Zeek disponuje vlastními skripty, které detekují různé útoky nebo výkyvy v síti zvlášť. Ty následně ukládá do záznamů.



Obr. 3.2: Princip fungování programu Zeek

Z důvodu přehlednosti se detekované události nebo zaznamenané provozy sítě rozdělí do různých logovacích souborů. Každý záznam popisuje svoji určitou část, ze kterého lze vyčíst podrobnější informace v dané oblasti [35]. Na obr. 3.3 lze vidět část příkladů logovacích souborů a jejich klíčově parametry pro analýzu síťového provozu. Detailnějším příkladem využití může být detekování bezpečnostních incidentů na aplikační vrstvě, kdy je potřeba využít logovací soubor `http.log`, ve kterém se nachází mnoho užitečných parametrů. Pro zvýšení efektivity je nutno vyjmout klíčové parametry typu časové razítko, IP adresa zdroje, zdrojový port, IP adresa cíle, cílový port, typ hlavičky HTTP (GET, POST, ...) a také výpisu hlavičky.

Podstatná část informací je v HTTP hlavičce, kde je obsažen tzv. user-agent, který identifikuje uživatele na základě několika informací jako jsou platforma prohlížeče, operační systém, jejich verze a další údaje, viz výpis 3.16, které hlavně slouží pro optimalizování webové stránky uživatelům. Tyto nastavení jsou užitečná serve-



Obr. 3.3: Část logovacích souborů v programu Zeek

rům. Pro detekci DDoS útoků jsou také velmi přínosná, neboť lze rychle vydedukovat, že provoz pochází neustále z jednoho zdroje, nikoliv od legitimních uživatelů, u kterých se pochopitelně user-agent řetězce mění, naopak od možných útočníků budou stále identická.

Výpis 3.16: Příklad řetězce v hlavičce

```
1 Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6),
2 AppleWebKit/537.36 (KHTML, like Gecko)
3 //převzato ze simulátoru útoku programu Slowloris
```

3.6 Průzkum generátorů

K simulaci legitimních klientů je vhodné využít generátory síťového provozu, které lze nainstalovat na libovolný operační systém. Pro účely testování byl přidán samostatný virtuální stroj s operačním systémem `Ubuntu Linux`, na kterém budou jednotlivé generátory otestovány. Samotné prostředí GNS3 nabízí několik předem nakonfigurovaných zařízení, které postačí stáhnout a naimportovat do topologie.

PackETH je generátor síťového provozu s uživatelským rozhraním, kde mezi povinnými parametry patří MAC adresa zdrojového a cílového virtuálního stroje, definování IPv4 nebo IPv6, porty, dále zvolení typu protokolu mezi TCP, UDP nebo ICMP. Na konec lze zvolit i payload. Nejpodstatnější nastavení, který tento generátor nabízí, je množství paketu a časový sled mezi nimi [36].

Nping je součástí programu Nmap, který je předem nainstalovaný na operačním systému `Kali Linux`. Samotné generování probíhá na základě příkazu, kde je potřeba uvést typ protokolu, množství paketu, port a IP adresu cílové destinace, viz výpis 3.17. Není zde však možnost zadat zpoždění mezi jednotlivými pakety.

Výpis 3.17: Příklad příkazu nping

```
1 nping -tcp -c 20 -p 443 <cílová adresa>
```

D-ITG je generátor síťového provozu, který vyžaduje spuštění i na cílovém klientovi. Díky tomu lze v síti rozšířit o další protokoly typu VoIP a Telnet. Nevýhodou je absence uživatelského rozhraní, kvůli kterému může docházet ke zhoršení škálovatelnosti jednotlivých parametrů [36].

Ostinato je program na generování síťového provozu, který je kompatibilní s nástrojem GNS3. Tento generátor lze do topologie naimportovat na přímo jako samostatný a nezávislý stroj. Dále lze snímat provoz sítě, pomocí připojeného portu k dané topologii. Mezi vedené statistiky patří počet přijatých rámců, počet odeslaných rámců, rychlost odesílání, rychlost přijetí, množství přijatých rámců v bajtech a množství odeslaných rámců v bajtech. Mezi nejužitečnější funkci patří možnost přidání počtu proudů s odlišnými parametry jako je IP adresa nebo protokol. Díky tomu lze částečně simulovat reálné prostředí provozu [36].

Porovnání generátorů

Mezi kritéria srovnání byly zařazeny uživatelské rozhraní, protokoly a latence mezi jednotlivými pakety. Díky tomuto průzkumu lze odvodit efektivnost a uživatelskou přívětivost každého generátoru, viz tab. 3.2.

Tab. 3.2: Srovnání generátorů

Název generátoru	Uživatelské rozhraní	Internetové protokoly	Latence
PackETH	ANO	TCP, UDP, ICMP	ANO
Nping	NE	TCP, UDP, ICMP a ARP	NE
D-ITG	NE	TCP, UDP, Telnet, VoIP	NE
Ostinato	ANO	TCP, UDP, ICMP	ANO

Efektivita generátorů záleží na umístění v síti. Pro účely testování provozu na webový server bylo vhodné umístit generátor a webový server co nejblíže, nejlépe do jedné sítě. V topologii testování se jedná o síť 192.168.1.0. V druhé síti 10.0.1.0 dokázal vytvořit provoz pouze program Nping běžící na operačním systému `Kali Linux`, u ostatních generátorů nebylo možno zachytit žádný provoz. Systém na zachytávání

komunikace byl umístěn na webovém serveru. Program Ostinato disponuje jednou velmi zajímavou funkcí, jedná se o možnost nastavit více proudů na jednom jediném portu. Díky tomu lze nasimulovat legitimní provoz, neboť lze modifikovat IP adresa a protokoly tak, aby se neustále neopakovaly.

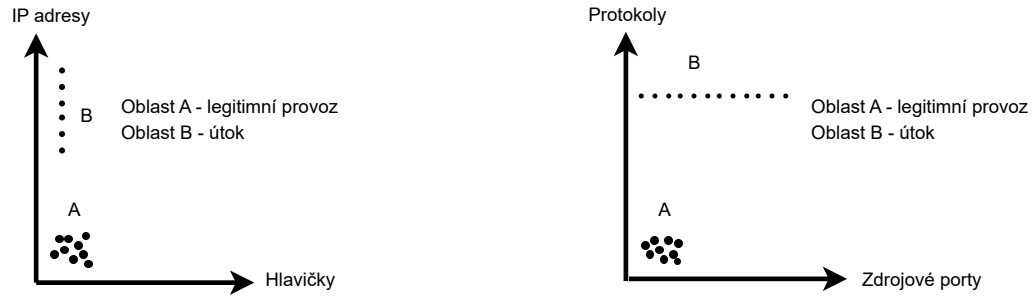
U všech generátorů převažují hlavně protokoly transportní vrstvy, lze tedy usoudit, že generování aplikačního provozu je složité, a proto tuto funkci žádný generátor nenabízí. Jako alternativní řešení byl využit webový server Penterepmail a manuální interakce, díky tomu se v provozu objevily provozy aplikační vrstvy.

3.7 Navržené metody detekce

Síťové provozy zachycené pomocí záznamových souborů budou dále zpracovány pomocí vytvořených podmínek v programovacím jazyce `Python`. Cílem je rozdělit komunikaci na útoky a na legitimní část. Základním kritériem pro určení bezpečnostního incidentu bude časové razítko jednotlivých paketů, dále může být rozšířeno o další parametry, která dokáží účinněji operovat na určité vrstvě. Příkladem může být aplikační vrstva, ve které se přidá hodnota user-agentu obsahující v hlavičce.

Program načte logovací soubor, ze kterého vyjme následující klíčové parametry: čas, IP adresa zdroje, IP adresa cíle, port cíle a user-agent. Následně všechny vybrané hodnoty vloží do nového souboru a doplní o hlavičky, které usnadní následnou práci s údaji. Hlavní část programu je tvořena cyklem FOR, který opakuje posloupnosti vložených podmínek. Nejpodstatnější část navrženého programu spočívá v porovnání časového razítka prvního a následujícího paketu, pokud je časový rozdíl nižší než 10 sekund, lze považovat provoz za podezřelý, v opačném případě se jedná o provoz legitimní. Výhodou takového řešení je přesná identifikace bezpečnostního incidentu na konkrétním indexu. Nelegitimní provoz musí být dále podroben rozboru, k tomuto účelu byly vytvořeny sady map. Každá jednotlivá mapa zaznamenává svůj sledovaný parametr (IP adresa, port nebo protokol) pomocí unikátního klíče a hodnoty. Konkrétním příkladem je způsob detekování útočníka, který neustále modifikuje svoji IP adresu, v tomto případě IP adresa reprezentuje unikátní klíč a počet opakování reprezentuje hodnotu klíče.

Datasety pro strojové učení budou tvořeny z nejpočetnějších hodnot z každé skupiny IP adresa, port, protokol a hlavička. Modely vysvětlují míru důležitosti těchto vybraných parametrů v posouzení charakteru síťového provozu, viz obr. 3.4.



Obr. 3.4: Model klíčových hodnot pro strojové učení

3.8 Strojové učení

3.8.1 Rozhodovací strom

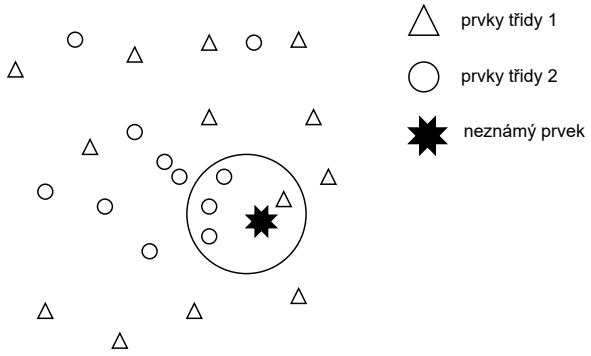
Rozhodovací stromy tvoří uzly, které reprezentují podmínky. Tyto podmínky musí být věcné a srozumitelné. Jedná se o tzv. transparentnost, kdy je potřeba, aby to bylo pochopitelné jak pro stroje, tak i pro člověka. Cesta začíná od kořene stromu a dále pomocí rozhodnutí true nebo false se přes jednotlivé uzly stromu, dá dostat až na konec, který reprezentuje list stromu, odtud vzejde konečný výsledek. Na obr. 3.5 lze vidět příklad rozhodovací stromu při analýze síťového provozu.

Obr. 3.5: Příklad analýzy provozu dle rozhodovacího stromu

3.8.2 K Nearest Neighbors

Jedná se o algoritmus strojového učení, který je využíván v mnoha oblastech například v průmyslu, ve zdravotnictví nebo hlavně v informačních technologiích. Mezi hlavními úkoly, které tento model řeší, je předpovídání analýz, rozpoznávání řeči a tváře nebo také v dolování dat [37].

Algoritmus k-nejbližších sousedů (k-NN) klasifikuje vícerozměrné vzorky podle jejich vzdálenosti od známých množin vzorků, které jsou již natrénované. Po vložení neznámého prvku do trénovací množiny je nutno zajistit všechny vzdálenosti od všech natrénovaných prvků vůči neznámému. Následně ten nejbližší známý (nadefinovaný) prvek klasifikuje neznámý prvek do stejné třídy, tento způsob využívá pouze jednoho souseda. Další varianta spočívá ve využití více sousedů, kdy kolem neznámého prvku se vytvoří kruh, velikost takové oblasti závisí na počtu zvolených sousedů. Neznámý prvek se klasifikuje do třídy podle nejpočetnějších a nejbližších sousedů. Je vhodné zvolit lichý počet sousedů z důvodu jednoznačného rozhodování. Na obr. 3.6 je vidět případ, kde prvek třídy 1 je velmi blízko neznámého prvku, ale není nejpočetnější, a proto jsou výsledkem prvky třída 2, která je druhá nejbližší, ale první nejpočetnější [37].



Obr. 3.6: Popis principu Algoritmu k-nejbližších sousedů

3.8.3 Metoda podpůrných vektorů

Metoda podpůrných vektorů je technikou strojového učení, která slouží pro klasifikaci dat na základě trénovací množiny. Cílem Support Vector Machines (SVM) je vyhledat nadrovinu, ve které lze již rozdělit prvky lineárním způsobem. Pokud existuje problém, který by nebyl řešitelný lineárně, lze využít jádrovou transformaci Sigmoidní nebo Polynomickou. Díky tomu je algoritmus flexibilnější a variabilnější k vyřešení jakéhokoliv problémů [38].

3.8.4 Používané knihovny

Základem každého strojového učení je příprava dat. Vstupní parametry musí být k jednotlivým metodám kompatibilní. Proto byla vybrána knihovna `pandas` [39], která dokáže zpracovat data z různých formátů jako je například CSV (Comma-separated values) nebo SQL (Structured Query Language) a umožní oddělit hodnoty

od čárek. Dále disponuje statickými výpočty, základní vizualizací a také velmi dobře zpracovanou dokumentací [40].

Adaptace souborů s příponou `.log`, které jsou výstupem snímaného sítového provozu z programu Zeek, bude řešeno pomocí parsování. Princip parsování spočívá v rozkladu logovací zprávy na jednotlivé dílky, které jsou již přizpůsobené k přemístění do určitých polí. Existuje přímo knihovna `ParseZeekLogs` [41], která je určená pro práci s výstupními soubory programu Zeek.

Ke zpracování strojového učení byla zvolena knihovna `Scikit-learn` [42], která dle průzkumu z roku 2021 patří mezi devět nejlepších knihoven určené pro programovací jazyk `Python` [43]. Další důvodem pro zvolení je skutečnost, že je volně přístupná všem zájemcům. `Scikit-learn` vychází ze dvou velmi úspěšných knihoven, a to `SciPy` a `NumPy`. Díky tomu obsahuje mnoho různých algoritmů a druhů učení, které jsou v porovnání s ostatními knihovny bezkonkurenční [40]. Navíc jsou zde k dispozici také metriky, které mají za cíl určit úspěšnost jednotlivých modelů strojového učení. V tab. 3.3 jsou shrnuty všechny knihovny a balíčky, které budou využity na zpracování strojového učení.

Tab. 3.3: Knihovny a balíčky zpracovávající strojové učení

Funkce	Knihovny	Balíčky
Příprava dat	pandas	pd
Parsování	parsezeeklogs	ParseZeekLogs
K-NN model	sklearn, sklearn.neighbors	KNeighborsClassifier
Decision tree model	sklearn.tree	DecisionTreeClassifier
SVM model	sklearn	svm
Přesnost	sklearn.metrics	accuracy_score
Citlivost	sklearn.metrics	recall_score
F-skóre	sklearn.metrics	f1_score

4 Experimentální testování

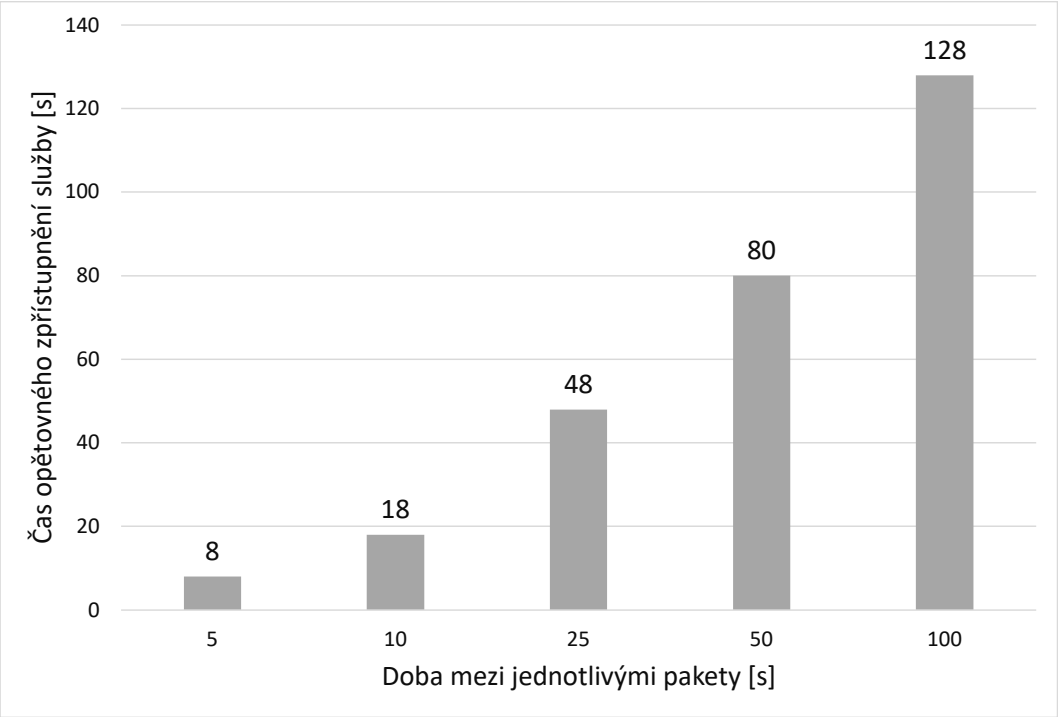
4.1 Simulace a detekce útoku Slowloris

Simulace útoku Slowloris

Při testování, pomocí nástroje Pentmenu, viz výpis 4.1, se vycházelo z hodnot, které byly pro danou službu hraniční. První hranice při které došlo k odepření služby bylo v 800 nedokončených spojení HTTP. Naopak webový server na základě testování dokázal v jednu chvíli obsloužit až 700 „uživatelů“ a to bez větších problému. Pro experimentální testovaní byla zvolena hodnota konstantní 750 paketů z důvodu vhodného testování služby na hranici své maximální možné kapacity. Jediný parametr, který zde bylo nutné opravit, byl interval mezi jednotlivými pakety nastavené v rozmezí 5, 10, 25, 50 a 100 sekund. Výsledkem je graf, viz obr. 4.1, který popisuje časové odstupy mezi jednotlivými spojení a času znovu zpřístupnění dané služby. Na základě nových poznatků lze konstatovat, že čím delší rozestup mezi jednotlivými pakety je, tím je zdlouhavější regenerování stránky. Vysvětlením je, že veškeré požadavky se webový server snaží vyřešit, a vytrvale vyčkává na vyhotovení úplného požadavku. Je důležité, aby správce sítě měl přehled o dění a mohl rychle zasáhnout, v tom případě by legitimní uživatelé nepocítili žádný výpadek.

Výpis 4.1: Simulace útoku Slowloris pomocí nástroje Pentmenu

```
1     Using netcat for Slowloris attack....
2  Enter target:
3  192.168.1.5
4  Target is set to 192.168.1.5
5  Enter target port (defaults to 80):
6  443
7  Using Port 443
8  Enter number of connections to open (default 2000):
9  750
10 Choose interval between sending headers.
11 Default is [r]andom, between 5 and 15 seconds, or enter interval in seconds:
12 5
13 use SSL/TLS? [y]es or [n]o (default):
14 no
15 Launching Slowloris...
16 .
17 .
18 .
19 Slowloris attack ongoing...this is connection 748, interval is 5 seconds
20 Slowloris attack ongoing...this is connection 749, interval is 5 seconds
21 Slowloris attack ongoing...this is connection 750, interval is 5 seconds
22 Opened 750 connections....returning to menu
23 Pentmenu>
```



Obr. 4.1: Výsledek simulace protahování doby spojení

`Apache2` umožňuje zobrazit veškerý provoz v jeden daný okamžik, který se každou sekundu aktualizuje. Na výpise 4.2 je zaznamenán začátek a také přímo průběh útoku. Důležitost analýzy se zde potvrdila, lze odvodit jaký styl útoku je právě veden. Webový server je zahlcený požadavky „R“, které nestíhá odbavovat.

Výpis 4.2: Status webového serveru

```
1  _____________R_____R_________R_____________C_________RRRRRRRRRRRRWR
2  RRRCRCRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR_______RRRRRRRRRRRRRRRRRR__R_
3  __RRRRRRRRR_____________________________________RRRRRRRRRRRRRRRRRRRRRR
4  RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
5
6  Scoreboard Key:
7  "_" Waiting for Connection, "S" Starting up, "R" Reading Request,
8  "W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
9  "C" Closing connection, "L" Logging, "G" Gracefully finishing,
10 "I" Idle cleanup of worker, "." Open slot with no current process
```

Detekce a ochranné opatření proti útoku Slowloris

Metoda detekce pomocí signatur a iptables

K detekci byl využit jednoduchý příkaz `netstat`, který byl doplněn o parametry, viz výpis 4.3, a který zobrazuje aktuální počet vytvořených spojení. Je zde vypsána

konkrétní IP, která zanechala v jeden okamžik až 750 otevřených spojení. Tuto IP adresu je nutno zablokovat pomocí firewallu, viz výpis 4.4.

Výpis 4.3: Detekce útoku Slowloris

```
root@ubuntu:~# netstat -ntu -4 -6 |  awk '/^tcp/{print$5}' |
sed -r 's/:[0-9]+$//' |  sort | uniq -c | sort -n
      1 10.0.1.1
    750 10.0.1.5
```

Výpis 4.4: Zablokování útočníka pomocí pravidel iptables

```
iptables -A INPUT -s 10.0.1.5 -j DROP
```

Existuje zde několik způsobů jak eliminovat Slowloris útok. Ve výchozím nastavení `Apache2` serveru (lze vidět na výpise 4.5) je nutné snížit dobu čekání (z původně 300 sekund na 30 sekund) na příchozí a odchozí požadavky. Parametry lze upravit v souboru, který je umístěn v `/etc/apache2/apache2.conf`.

Výpis 4.5: Výchozí hodnota nastavení

```
#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300
```

Dále je vhodné omezit počet otevřených spojení z každé konkrétní IP adresy na 10, tato hodnota je pro testovací účely dostačující. To však může být limitující pro legitimní uživatele, kteří mohou mít maximálně 10 otevřených oken v prohlížeči. Na výpise 4.6 je příkaz, který nastaví pravidlo ve firewallu. Pokud dojde k přesahu, komunikace se ukončí.

Výpis 4.6: Přidání pravidla filtru

```
iptables -I INPUT -p tcp --dport 443 -m connlimit --connlimit-above 10 -j DROP
```

Detekce pomocí systému IDS (Intrusion Detection System) spočívá v nastavení pravidel v souboru `webserver.rules`, které kopírují signatury útoku tedy počet spojení a hlavně trvání spojení, konkrétní znění pravidla je na výpise 4.7.

Výpis 4.7: Pravidlo pro detekci útoku Slowloris

```
alert tcp any any -> any 443 (msg:"Pozor! Byl detekován možný útok Slowloris";flow:
    stateless; threshold: type both, track by_src, count 50, seconds 10; sid
    :1005041;rev:1; classtype:bad-unknown;)
```

Detekce pomocí anomálie

Princip detekování pomocí anomálie spočívá hlavně ve výkyvu od běžného stavu, díky tomu je tento jev zřejmý i pouhým grafickým znázorněním testovaných hodnot. Z testování je zřejmé, že průběh útoků DDOS je velmi objemný i při sledování

poměrně krátkého časového úseku síťového provozu. Poměr mezi útokem a běžným provozem je velmi zřetelný.

Dalším způsobem, jak detekovat bezpečnostní incident pomocí anomálie, je využití rovnic. Fáze učení je založena na datech, které vznikly jako výstupní hodnoty z navrženého programu. Skládá ze dvou částí, první část hodnotí pouze incidenty, které způsobí úplné zahlcení webového serveru. Naopak druhá část se věnuje legitimnímu provozu. Vzorec pro první část zpracovávající data ze síťového provozu pracuje s argumentem U, který definuje nejčastější IP adresy spojené s útokem, a s parametrem C, který udává počet celkových adres v síti, podoba vzorce (4.1).

$$Z_1 = \frac{U}{C} \tag{4.1}$$

Druhý vzorec se liší pouze v čitateli, kde je proměnná L, která formuluje nejpočetnější IP adresy běžného provozu, podoba vzorce dle (4.2).

$$Z_2 = \frac{L}{C} \tag{4.2}$$

V první části (incidenty) je vybrán klíč s nejnižší hodnotou Z_1, neboť je to první nejnižší hodnota, která může zahltit webový server, viz tab. 4.1.

Tab. 4.1: Výsledky první fáze učení

	U	C	Výsledek Z1
1.	298	340	0,876
2.	2547	4972	0,512
3.	2571	2951	0,871
4.	714	1870	0,382
5.	1023	1030	0,993
6.	990	2409	0,411
7.	1141	1150	0,992
8.	821	1323	0,621
9.	754	1429	0,528
10.	812	831	0,977
Nejnižší hodnota	714	1870	0,382

Hodnotícím klíčem druhé části (legitimní provoz) je střední hodnota medián, na základě vzorce (4.3). Tato metodika byla vybrána z důvodu flexibility. Výsledkem běžného provozu by mělo být co nejvíce konstantní a jednotné, proto je nutné eliminovat extrémními výkyvy, které mohou negativně ovlivnit výsledek rozhodnutí.

$$Medián = \begin{cases} X_{\frac{n+1}{2}}, & \text{pokud je n liché.} \\ \\ \frac{X_{\frac{n}{2}} + X_{\frac{n+1}{2}}}{2}, & \text{pokud je n sudé.} \end{cases} \tag{4.3}$$

Nejrychlejší způsob, jak vypočítat hodnotu mediánu, je seřazení hodnot Z_2 vzestupně, viz tab 4.2. Všechny ostatní výsledky se pohybují v rozmezí od 0 až po 1, kde hodnoty blížící se k 0 značí legitimní provoz, naopak hodnoty blížící se k 1 značí útok. Konkrétní výstupy obou vzorců na základě testování vytvoří tzv. mantinely, které ohraničí tři pásma. Nulté pásmo je definováno intervalem $\leq$ 0,005, tedy hodnota mediánu a znamená legitimní provoz, viz tab. 4.2. První pásmo je definováno intervalem $\geq$ 0,382, tedy nejnižší hodnota Z_1 a znamená útok, viz tab. 4.1. Střední pásmo vzejde ze dvou krajních ohraničení, cílem je, aby zachycený provoz v tomto pásmu byl nadále podroben analýze, sníží se tím riziko omezení legitimního uživatele. Postup řešení a operací v jednotlivých pásmech je vidět na tab. 4.3.

Tab. 4.2: Výsledky druhé fáze učení

	L	C	Výsledek Z1	Seřazení
1.	1	340	0,003	0,003
2.	11	1323	0,008	0,003
3.	14	2409	0,006	0,004
4.	15	4972	0,003	0,004
5.	7	1150	0,006	0,005
6.	8	1870	0,004	0,006
7.	14	1429	0,010	0,006
8.	17	831	0,020	0,008
9.	12	2951	0,004	0,010
10.	5	1030	0,005	0,020
Medián				0,005

Tab. 4.3: Detekce dle pásma a následná operace

	Nulté pásmo	Střední pásmo	První pásmo
Stav	Legitimní provoz	Potenciálně podezřelý provoz	Útok
Operace	Žádná	Podrobnější analýza	Blokace

Testování navržených detekcí

Webový server je s nástrojem Suricata již nastavený tak, aby dokázal odolat útoku Slowloris. Z tohoto důvodu bude realizován útok, který bude silnější. V jeden okamžik se otevře 10 000 spojení, viz výpis 4.8, Po spuštění simulace útoku bylo zjištěno, že přibližně v polovině otevřených spojení tedy 5000 dochází ke kolabování samotného útočícího algoritmu, který se sám zacyklí. Došlo ke kritické chybě programu, bylo nutné stroj restartovat. Zachycení útoku je na výpise 4.9. Zatímco webový server, viz výpis 4.10, je stále dostupný pro legitimní uživatele, potvrdila se zde prevence, předvídanost a včasné zareagování, které dokázaly předejít komplikacím.

Výpis 4.8: Simulace útoku Slowloris pomocí nástroje Pentmenu

```
    Using netcat for Slowloris attack....
Enter target:
192.168.1.5
Target is set to 192.168.1.5
Enter target port (defaults to 80):
443
Using Port 443
Enter number of connections to open (default 2000):
10000
Choose interval between sending headers.
Default is [r]andom, between 5 and 15 seconds, or enter interval in seconds:
5
use SSL/TLS? [y]es or [n]o (default):
no
Launching Slowloris...
.
.
Slowloris attack ongoing...this is connection 4918, interval is 5 seconds
Slowloris attack ongoing...this is connection 4919, interval is 5 seconds
Slowloris attack ongoing...this is connection 4920, interval is 5 seconds
./pentmenu: fork: retry: Resource temporarily unavailable
Slowloris attack ongoing...this is connection 4921, interval is 5 seconds
./pentmenu: fork: retry: Resource temporarily unavailable
./pentmenu: fork: retry: Resource temporarily unavailable
./pentmenu: fork: Resource temporarily unavailable
```

Výpis 4.9: Zachycení útoku Slowloris v nástroji Suricata

```
11/29/2020-06:28:13.917512  [**] [1:1005041:1] Pozor! Byl detekován možný útok
    Slowloris [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP}
    10.0.1.5:34676 -> 192.168.1.5:443
```

Výpis 4.10: Status webového serveru

```
_____W_____________________________________________________
__________________________________________________________
Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"C" Closing connection, "L" Logging, "G" Gracefully finishing,
"I" Idle cleanup of worker, "." Open slot with no current process
```

K otestování detekce pomocí rovnice anomálie bylo nutné změnit testovací program, z důvodu kritické chyby nástroje Pentmenu při vyšším počtu otevíraných spojení. Nahrazen byl programem Slowloris, který dokáže zakládat libovolné množství nových hlaviček HTTP, řešením se stal jednoduchý princip, který cílovou hranici 10 000 otevřených spojení rozdělí na menší celky, viz výpis 4.11.

Výpis 4.11: Princip rozložení cílové hranice otevřených spojení

```
root@kali:~/slowloris# python3 slowloris.py 192.168.1.5 -s 10000
[19-05-2021 02:59:27] Attacking 192.168.1.5 with 10000 sockets.
[19-05-2021 02:59:27] Creating sockets...
[19-05-2021 02:59:35] Sending keep-alive headers... Socket count: 1021
[19-05-2021 02:59:50] Sending keep-alive headers... Socket count: 1021
[19-05-2021 03:00:05] Sending keep-alive headers... Socket count: 1021
[19-05-2021 03:00:27] Sending keep-alive headers... Socket count: 1021
[19-05-2021 03:00:42] Sending keep-alive headers... Socket count: 1021
[19-05-2021 03:01:04] Sending keep-alive headers... Socket count: 1021
[19-05-2021 03:01:19] Sending keep-alive headers... Socket count: 1021
[19-05-2021 03:01:41] Sending keep-alive headers... Socket count: 1021
[19-05-2021 03:01:56] Sending keep-alive headers... Socket count: 1021
[19-05-2021 03:02:18] Sending keep-alive headers... Socket count: 1021
```

Dále bylo zjištěno, že detekční systém Zeek, který je nainstalován na webovém serveru, nebyl schopný zachytit přesný počet vyslaných paketů z virtuálního stroje `Kali Linux`, který charakterizuje útočníka. Vysvětlení může být několik, prvním významným odůvodněním může být skutečnost, že během útoku dochází k vyčerpání kapacit webového serveru, které způsobí výpadek stroje. Z toho lze vyvodit, že je velmi důležité zvolení umístění systémů detekcí v síti. Dalším vysvětlením může být to, že pakety, které jsou modifikované k útoku, mohou být nestabilní a k cíli vůbec nedorazí, pravděpodobně se ztratí nebo zacyklí v síti. Podrobné výsledky zachycení a vyhodnocení pomocí rovnic anomálie, viz tab. 4.4.

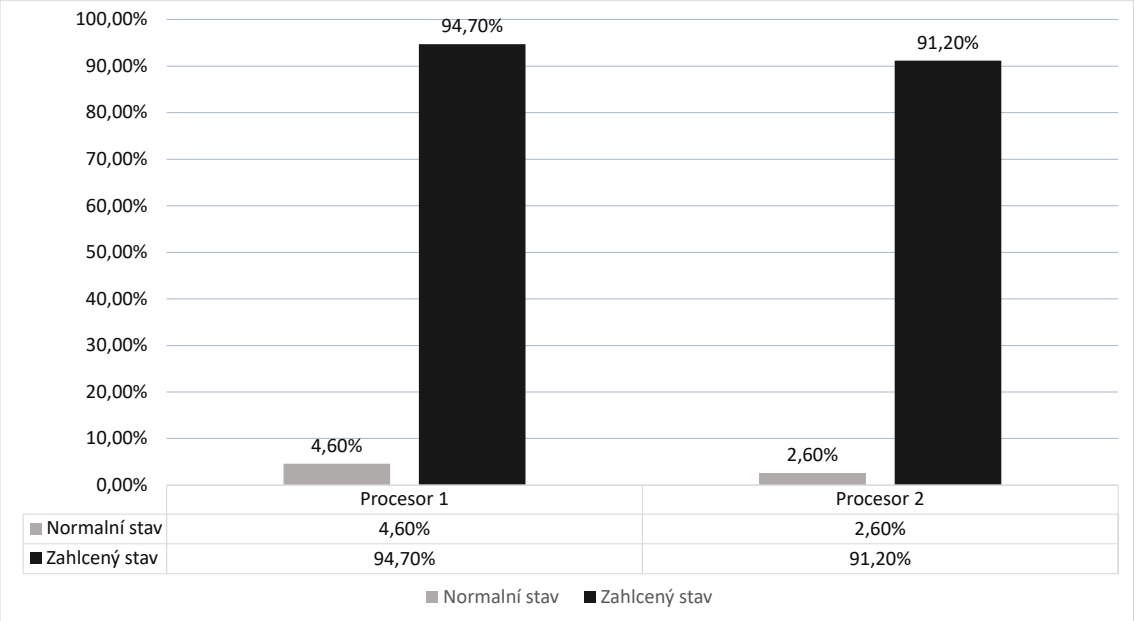
Tab. 4.4: Vyhodnocení výsledků detekcí pomocí metody rovnic anomálie

	Výsledky
Doba trvání útoku	3 s
Hodnota C, celkový počet zachycených paketů	4097
Hodnota U, pakety označené jako útok	4093
Pásmo první	0,999
Vyhodnocení	Útok

4.2 Simulace a detekce útoku SSL handshake flood

Tento útok se zaměřuje na nedokonalost protokolu SSL, který zabezpečuje komunikaci mezi uživatelem a webovou stránkou. Cílem útoku je zneužití náročnosti ve

vytváření certifikátu, které musí server neustále znovu formovat. Tento úkon je velmi náročný pro procesory. Útočník vytvoří několik žádostí `Client Hello`, který neustále obnovuje. V průběhu testování bylo zjištěno, že stroj během simulovaného útoku byl těžko ovladatelný, reakce v terminálu byly zpomalené. V obr. 4.2 lze vidět závažnost toho útoku, neboť zde může dojít i k hardwarovému poškození, pokud by tento negativní jev probíhal déle.



Obr. 4.2: Vytížení procesoru během útoku

Detekce a ochranné opatření proti SSL handshake flood útoku

K identifikaci takového útoku lze využít nástroj Suricata, ve kterém je vloženo pravidlo, viz výpis 4.12, které rozezná pakety s hlavičkou SSL nebo `Client Hello`. Běžné sestavení spojení s webovým serverem trvá v řádech milisekund až sekund, je zde však mnoho okolností, které mohou dobu trvání prodloužit. Aby nedošlo k omezení legitimních uživatelů, je nutno nastavit pravidlo tak, aby se hlášení zobrazovalo až ve chvíli, kdy jsou otevřená spojení v počtu 10 a v intervalu 15 sekund. V tomto intervalu byl identifikován simulátor útoku, viz výpis 4.13.

Výpis 4.12: Pravidlo pro detekci útoku SSL handshake flood

```
1 alert tls $HOME_NET any -> any 443 (msg:"Pozor! Byl detekován možný útok SSL";flow:
      stateless; threshold: type both, track by_src, count 10, seconds 15; priority
    :1; sid:1005041;rev:1; classtype:bad-unknown;)
```

Další možnosti detekce nabízí samotný operační systém, `Linux` má několik nástrojů, které dokáží monitorovat zátěž procesoru, disku, dočasné paměti a také sítě.

Například přímo v `Ubuntu` je program s grafickým rozhraním, který zobrazuje barevně grafy v časovém horizontu. Další možnost je využít terminál, kde pomocí nástroje `htop`, viz výpis 4.14, je možno zobrazit správce procesů. Graf lze přímo zobrazit v terminálu pomocí příkazu `ttyload`, viz výpis 4.15.

Výpis 4.13: Zachycení útoku SSL handshake flood v nástroji Suricata

```
11/29/2020-06:21:06.993484  [**] [1:1:1] Pozor! Byl detekován možný útok SSL [**] [
    Classification: (null)] [Priority: 1] {TCP} 10.0.1.5:58524 -> 192.168.1.5:443
11/29/2020-06:21:06.998275  [**] [1:1:1] Pozor! Byl detekován možný útok SSL [**] [
    Classification: (null)] [Priority: 1] {TCP} 10.0.1.5:58530 -> 192.168.1.5:443
```

Výpis 4.14: Detekování útoku na základě monitorování procesorů

```
  1  [||||||||||||||||||||||||||||||||||||||||||||94.7%]   Tasks: 117, 464 thr; 2 running
  2  [||||||||||||||||||||||||||||||||||||||||||||91.2%]   Load average: 13.19 10.53 7.57
     Uptime: 00:05:00
```

Výpis 4.15: Vykreslování grafu v terminálu během útoku SSL handshake flood

```
ubuntu                  28.08, 16.99, 7.25   14:40:15       ttyload, v0.5.3

 42.00                            *
 39.38                           * *
 36.75                        ***   ***
 34.12                                * **                          *
 31.50                     **           *  *                         *
 28.88                                      **             * ****   *  ****
 26.25                   **                   ****          *    ***
 23.62                                            ***
 21.00                  **                           **   *
 18.38                                                ***          *******
 15.75                                                    *********
 13.12                *       ****************************
 10.50              **     ***
  7.88                 ****                                  *****************
  5.25          ********    ******************************
  2.62    ******************
  0.00
         ^14:35 ^14:36   ^14:37          ^14:38          ^14:39          ^14:40
  Legend:
     1 min: *, 5 min: *, 15 min: *
     1&5 same: *, 1&15: *, 5&15: *, all: *
```

Detekování útoku SSL handshake flood pomocí systému Zeek bylo snadnější, neboť je zde vyhrazen konkrétní soubor `.log` pro detekování protokolů SSL, který vyhodnocuje všechny potřebné parametry. Během testování bylo zjištěno, že testovací nástroj po celou dobu nemění IP adresu, ale modifikuje svůj port. Vysvětlení je logické, za normálních okolností postačí běžnému legitimnímu uživateli několik portů, například při otevírání webové stránky ve více oknech v prohlížeči. Útočníci však otevírají nové porty v řádech tisíců během několika sekund. Tato metoda detekce je také svým způsobem založena na detekování pomocí anomálie. Částečný

výpis portů a jejich počet opakování během útoku byl zpracován na základě naprogramovaných podmínek v `Pythonu`, viz výpis 4.16.

Výpis 4.16: Otevírání portů během útoku

```
{32980: 2, 32988: 1, 33006: 2, 33008: 2, 33022: 3, 33030: 2, 33038: 1, 33040: 1,
    33042: 2, 34118: 2, 34366: 3, 34380: 2, 33656: 2, 34434: 2, 33194: 1, 34512: 2,
     32880: 1, 33028: 1, 33520: 1, 34514: 2, 33208: 1, 34544: 2, 33168: 1...}
```

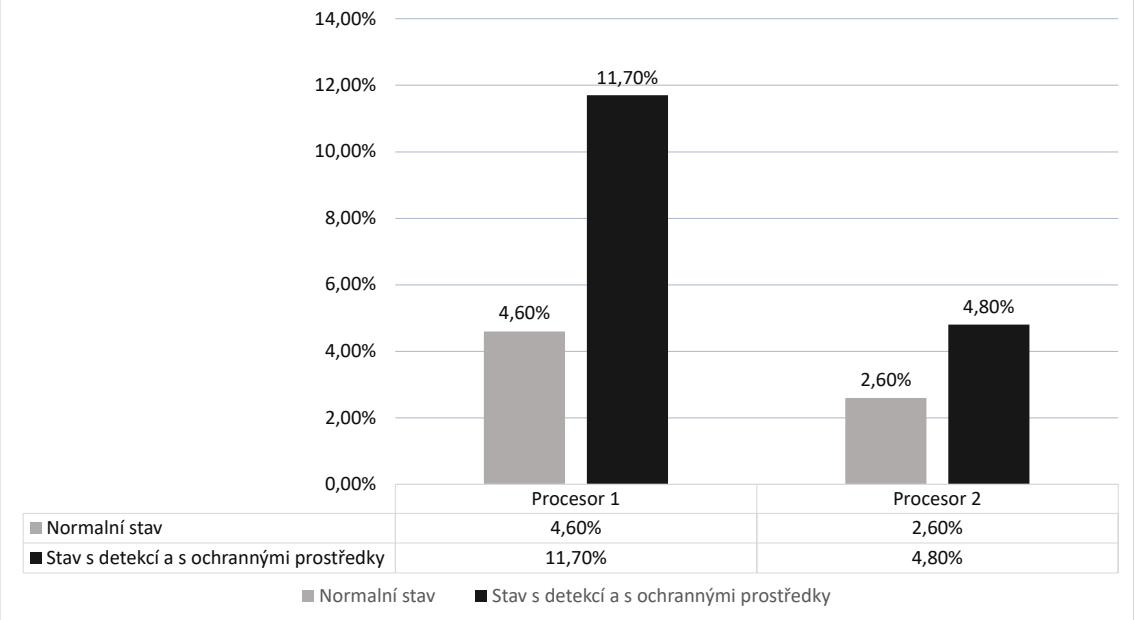
Díky detekčním metodám lze eliminovat chybné označení legitimního uživatele. Nejefektivnější řešení je zablokování IP adresy útočníka, viz na výpise 4.17.

Výpis 4.17: Zablokování útočníka pomocí pravidel iptables

```
iptables -A INPUT -s 10.0.1.5 -j DROP
```

Testování navržených detekcí

Proces sestavování certifikátu je velmi složitý, skládá se z mnoha části. Zde velmi hrozilo, že by se pravidlo zahození takového útoku mohlo přímo dotknout i legitimních uživatelů. Jediným řešením bylo zablokovat konkrétního útočníka v pravidlech firewallu. Výsledkem testování s navrženou detekcí je zobrazen na obr. 4.3. Je zde vidět mírné zatížení procesoru oproti normálnímu stavu bez zavedených detekcí. Tento jev byl způsoben procesy detekčních nástrojů, které souběžně běžely s webovým serverem. Na výpise 4.18 lze ověřit, že během konání útoku na webový server nedošlo k žádným výkyvům.



Obr. 4.3: Vytížení procesoru během testování útoku na zařízení

Výpis 4.18: Monitorování procesorů

```
1   1  [|||||||||                                  11.7%]   Tasks: 124, 464 thr; 2 running
2   2  [|||                                         4.8%]   Load average: 0.33 0.08 0.26
3   Uptime: 00:05:00
```

Je důležité implementovat detekční systémy možného průniku již před samotným koncovým zařízením. Takové řešení by zaručilo včasné varování hrozícího útoku, a také by ušetřilo kapacitu procesorů webového serveru.

4.3 Simulace a detekce kombinace útoků

Do testovací fáze byly zahrnuty 4 útoky s různými intenzitami a množstvím. Legitimní provoz byl proveden pomocí generátoru Ostinato, který umožní jak modifikaci IP adres, tak nastavení zpoždění mezi jednotlivými pakety. Běžný provoz byl generován pravidelně v počtu 150 paketů každých 15 sekund. Zastoupení útoků s běžným provozem je zcela náhodné a bylo prováděno ručně bez předem nadefinovaného algoritmu. Orientačně je v tab. 4.5 znázorněno, jak mohou být jednotlivé útoky v datasetu zastoupené vůči legitimnímu provozu. Cílem této metodiky generování bylo otestování detekčního mechanismu, zda je schopný detekovat i menší podíl zastoupení útoku v síti. Veškerý provoz byl ukládán do souboru s příponou `.log`, ze kterého se následně vyjmuly klíčové parametry, a ve formátu `.csv` byly datasety předány strojovému učení.

Tab. 4.5: Příklad metodiky generování provozu

Útoky	Podíl 90%	Podíl 75%	Podíl 60%	Podíl 30%
Slowloris	x			
SSL flood		x		
TCP SYN flood			x	
UDP flood				x

Zobrazení a také vyhodnocení výsledků probíhalo na základě ukazatelů úspěšnosti, díku tomu lze rychle zjistit, který algoritmus strojového učení je pro řešení nejefektivnější. Všechny výsledky z přesnosti, z citlivosti a z F-skóre mají škálu hodnocení od 0 do 1, kde hodnoty blížící se k 0 jsou nejhorší, naopak hodnoty blížící se k 1 nejlepší. Prvním ukazatelem úspěšnosti je přesnost, kde v čitateli je počet všech správně označených subjektů a v jmenovateli jsou všechny označené subjekty, hodnota přesnosti se vypočítá dle vzorce (4.4) [44].

$$Přesnost = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.4}$$

Citlivost je definována poměrem mezi správně klasifikovanými subjekty a všech pozitivně klasifikovaných subjektů, hodnota citlivosti se vypočítá dle vzorce (4.5) [44].

$$Citlivost = \frac{TP}{TP + FN} \tag{4.5}$$

F-skóre vychází z harmonického průměru dvou hodnot, a to z preciznosti a z citlivosti. Konečný výsledek se vypočítá podle vzorce (4.6) [44].

$$F\text{-}skóre = 2 * \frac{\frac{TP}{TP + FN} * \frac{TP}{TP + FP}}{\frac{TP}{TP + FN} * \frac{TP}{TP + FP}} \tag{4.6}$$

Provedené měření mělo za cíl zjistit vliv počtu sousedů (k-NN) na finální F-skóre výsledek, který zahrnuje dva důležité ukazatele, a to přesnost a citlivost. K účelu testování byl dataset rozdělen na dvě části. První část byla trénovací a zabírala 80 % kapacity dat, zbylých 20 % bylo nutno využít k testování. Bylo zjištěno, že nejlepší a nejoptimálnější výsledek je při nastavení počtu sousedů na tři. Hodnota přesnosti zde dosahovala až 99,5 %, viz tab. 4.6. U vyššího počtu sousedů je riziko rozptylu.

Tab. 4.6: Výsledky měření vlivu počtu sousedů v modelu k-NN

Strojové učení: k-NN	čas výpočtu [s]	Přesnost	Citlivost	F-skóre
k-NN - 1 n_neighbors	0,012	0,980	0,974	0,983
k-NN - 2 n_neighbors	0,018	0,985	0,984	0,988
k-NN - 3 n_neighbors	0,014	0,995	0,992	0,996
k-NN - 4 n_neighbors	0,018	0,975	0,971	0,981
k-NN - 5 n_neighbors	0,020	0,980	0,991	0,983
k-NN - 6 n_neighbors	0,020	0,970	0,975	0,975
k-NN - 7 n_neighbors	0,013	0,960	0,992	0,969
k-NN - 8 n_neighbors	0,019	0,965	0,957	0,970
k-NN - 9 n_neighbors	0,012	0,955	0,944	0,963

Algoritmus Rozhodovacího stromu se hlavně opírá o podmínky, ze kterých koná definitivní rozhodnutí. Není proto nutné vynaložit velké množství datasetů. V této části se porovnávaly různé podíly trénovací části a testovací části, viz tab. 4.7.

Bylo zjištěno, že mezi prvním zkoumaným prvkem (80 % učení a 20 % testování) a posledním zkoumaným prvkem (20 % učení a 80 % testování) je rozdíl přesnosti pouhé 1 %. Takové řešení je vhodné pro krátkodobé využití s jasným cílem. Při delším využití může docházet k odchylce výsledku z důvodu nárůstu nepodstatných podmínek, které ovlivní finální výsledek.

Tab. 4.7: Výsledky testování podílu dat v datasetu v modelu Decision tree

Strojové učení: Decision tree	čas výpočtu [s]	Přesnost	Citlivost	F-skóre
80% učení 20% testování	0,012	0,990	0,984	0,992
60% učení 40% testování	0,018	0,985	0,991	0,986
40% učení 60% testování	0,014	0,975	0,969	0,980
20% učení 80% testování	0,018	0,980	0,985	0,985

Efektivnost jádrových transformací (polynomiální a sigmoidní) se velmi liší od základního lineárního jádra. V tab. 4.8 je vidět, že doba výpočtu trvání lineárního jádra je v řádech stovek sekund, naopak nejkratší doba výpočtu je v desítkách milisekund. Rozdíl je velmi značný. U F-skóre, který průměruje přesnost a citlivost, je rozdíl mezi jednotlivými transformacemi kolem 10 %, nejpřesnější byl model lineární s výsledkem 0,984.

Tab. 4.8: Výsledky porovnání jádrových transformací modelu SVM

Strojové učení: SVM	čas výpočtu [s]	Přesnost	Citlivost	F-skóre
SVM - Linear	459,050	0,979	0,969	0,984
SVM - Poly	0,028	0,722	0,992	0,822
SVM - sigmoid	0,073	0,557	0,856	0,701

4.4 Vyhodnocení dosažených výsledků

Výsledek experimentálního testování především poukázal na rozdílnost obou útoků. V prvním případě útoku Slowloris došlo k projevům odepření služby až při konkrétním počtu vytvořených spojení, ale nikdy nedokončených. Zatímco SSL handshake flood útok měl projev okamžitý. Toto zjištění mělo velkou váhu při návrhu detekčního mechanismu, který dokázal zabránit náporům útoků. Tyto návrhy částečně vyžadují zásah správce sítě, který situaci zanalyzuje a pak provede úkony. Výhodou tohoto kombinovaného způsobu může být skutečnost, že nedojde k zablokování legitimních uživatelů. Nevýhoda takového řešení je však větší časová náročnost. Dále bylo zjištěno, že systémový nástroj Suricata, dokáže snímat provoz na aplikační vrstvě a na prezentační vrstvě referenčního modelu ISO/OSI.

Detekční návrh prvního útoku Slowloris vychází z nástroje Suricata, který na základě vložených signatur dokázal včas upozornit na hrozící se nebezpečí. Dále bylo zjištěno, že výchozí nastavení webového serveru `Apache2` je tomuto útoku ještě více náchylnější, neboť dovoluje delší časovou lhůtu na příjem a odesílání žádosti.

Tento výchozí soubor byl opraven a doplněn o filtry, které zabraňují vzniku incidentů s příznaky nedokončených spojení. Testování ukázalo velkou účinnost a spolehlivost, neboť byl každý podobný útok odrazen. Velkou nevýhodou navrženého systému je ta, že může omezit legitimní uživatele. Je proto důležité si také pravidelně zpracovávat údaje o provozu sítě, tedy počet návštěvníků a jejich průměrný strávený čas na webovém serveru. Na základě těchto informací lze odladit systém detekcí tak, aby nedocházelo ke střetům s běžnými uživateli a potenciálními útočníky. Druhá metoda detekce byla navržena na základě rovnic anomálie, které umožnily rozdělit síťový provoz na tři pásma na nulté, střední a první. Z výsledků testování byl zjištěn významný rozdíl mezi hodnotou definující legitimní provoz a hodnotou definující útok. K účelu testování byl proveden útok o síle 10 000 otevřených spojení. Metoda detekce pomocí anomálie vyhodnotila stav zcela přesně konkrétně vypočítala hodnotu 0,999, která jednoznačně definuje pásmo útoku. Tato skutečnost potvrzuje hlavní znak DDoS útoků tedy objemné zahlcení koncové služby. Naopak střední pásmo sloužilo k zachytávání slabých výkyvů způsobené nejčastěji zvýšeným zájmem uživatelů o službu, a proto bylo nutno zajistit detailnější průzkum vzniklých incidentů v této skupině, aby nedocházelo k chybné blokaci legitimních uživatelů. Příkladem může být kontrola hodnot parametrů popisující aktuální stav hardwarových komponentů.

Další detekční návrh byl navržen na útok probíhající na prezentační vrstvě, i zde nástroj Suricata dokázal detekovat parametry paketů SSL. V průběhu testování a návrhu detekce se ukázalo, že je náročné najít rozumné východisko, které by neomezovalo legitimního uživatele. Proces, který vzniká při sestavení a ověření certifikátu s webovým serverem, je složitý a vyžaduje mnoho postupů. Během testování byla zprovozněna i varianta zásahu detekčního nástroje Suricata, bylo zde nastaveno pravidlo drop, které má podobné útoky ihned zahodit. Navzdory důkazu provedení v logovacím souboru, se útoku nepodařilo zabránit. Modifikace pravidla zahození úspěch nepřinesla. Lze odvodit, že detekování je jednodušší proces než-li mitigování. Detekování pomocí anomálie bylo částečně umožněno pomocí systému Zeek, který ve svém logu zaznamenává různé informace o protokolu SSL v síti. Mezi sledované hodnoty byl i zdrojový port. Během testování bylo zjištěno, že útočník modifikuje a otevírá zdrojové porty několikanásobně více než běžný uživatel v síti. Tato skutečnost umožnila identifikovat bezpečnostní incident.

Nejúčinnějším řešením bylo zablokování IP adresy útočníka tedy až při projevu nežádoucích účinků, rychlost zde je však klíčová, neboť zatížení procesorů atakují hranici 100 %, pokud průběh bude mít trvání několika hodin, může dojít k selhání hardwarových komponent například z důvodu přehřátí.

Poslední detekční návrh byl navržen za účelem detekování několika kombinací útoků v jednom síťovém provozu. Otestovány byly tři základní metody strojového

učení. Z výsledku testování bylo zjištěno, že nejefektivnější algoritmus strojového učení na detekci bezpečnostních incidentů, je Algoritmus k-nejbližších sousedů, mezi sledované parametry byla doba trvání výpočtu a F-skóre. Již z podkladů teorie popsané v kap. 3.8.2 bylo zřejmé, že princip klasifikace k-NN je velice vhodný pro detekování útoků DDoS. Alternativou může posloužit Rozhodovací strom, který je vhodný v případech, kdy ze síťového provozu není dostatek dat, i v takovém případě dokáže algoritmus určit dostačující výsledek. U Metod podpůrných vektorů (SVM) bylo pro řešení incidentů DDoS nevhodné, lze předpokládat, že pokud řešený problém není lineárního charakteru nelze ani spoléhat na jádrové transformace, které tuto skutečnost během testování nenapravily. Konkrétní porovnání je znázorněno v tab. 4.9, kde nejlepší dosažený výsledek z každé skupiny je seřazen podle času výpočtu a podle F-skóre. Z výsledků je dále patrné, že se podařilo zahrnout do datasetů efektivní parametry ze síťového provozu, které přispěly k určení útoku nebo běžného provozu až s přesností přes 95 %. Navíc se jedná o návrh, který je odolný vůči variabilním útokům.

Tab. 4.9: Porovnání nejlepších výsledků z každé skupiny

Dosažený výsledek:	Dle času výpočtu modelu [s]	F-skóre
Nejlepší varianta	Decision tree	k-NN
Průměrná varianta	k-NN	Decision tree
Nejhorší varianta	SVM	SVM

Závěr

Cílem bakalářské práce bylo provést analýzu na DoS a DDoS útoky zaměřené na webový server a navrhnout detekční metody tak, aby bylo pokud možno i během útoku zachována komunikace mezi klientem a webovým serverem. Dalším úkolem bylo vytvořit experimentální síť, kde bude zprovozněný webový server, na kterém bude provedeno testování. Výsledky byly zpracovány a porovnány se stavem zapnutého a posléze vypnutého systému detekčního mechanismu.

V teoretické části byly rozebrány útoky dle částečného referenčního modelu ISO/OSI tedy relační vrstva, prezentační vrstva a aplikační vrstva. Tento způsob popisu umožní pochopení problematiky útoků vedených na konkrétních síťových vrstvách. Dále byly útoky rozděleny na obecnější pohled s cílem zdůraznit jejich princip fungování. Všechny útoky byly popsány tak, aby vynikly jejich jedinečné signatury. Postupem času je však nutné tyto signatury aktualizovat.

Praktická část zahrnuje postup instalace webového serveru a jeho zabezpečení. Dále realizace útoku bez aplikovaných detekčních mechanismů a s aplikovanými ochrannými opatřeními, díky tomu se otestovala funkčnost. Cílem bylo také sledovat chování zahlceného serveru a jeho ovladatelnost a reakce. Výsledek byl neuspokojivý, manipulace s terminálem byla nekomfortní. Po implementování navrženého detekčního mechanismu na webový server bylo zjištěno, že na straně útočníka dochází k zacyklení programu. Funkčnost mitigačních mechanismů nebyla v nástroji Suricata vysoká, důvodem mohlo být komplexnější řešení útoku, který se skládá z více procesů. Proto bylo vhodné navrhnout více detekčních metod, například detekce na základě rovnic anomálie nebo strojového učení, oba mechanismy jsou opřeny o klíčové parametry ze síťového provozu, které se ukládají formou datasetů. Z důvodu využití strojového učení k detekci útoku a běžného provozu bylo nutné otestovat různé možnosti vyžití knihoven zabývajícími se strojovým učením nebo umělou inteligencí. Strojové učení se opíralo o tři nejzákladnější modely, a to o Algoritmus k-NN nejbližších sousedů, Rozhodovací strom a Metody podpůrných vektorů.

Výstupem bakalářské práce jsou navržené detekční mechanismy, které na základě získaných dat detekují útoky. První metoda detekce se opírá o signatury, pokud je však útok mírně modifikován, může se stát, že detekční mechanismus selže. Druhá metoda detekce je odolnější, neboť zaznamená každý nestandardní výkyv v sítí pomocí vzorců anomálie, každá odchylka od běžného stavu však nemusí znamenat útok, může se jednat o zvýšený zájem legitimních uživatelů o službu. Třetí metoda detekce je založena na modelech strojového učení a je zároveň nejefektivnější, neboť eliminuje nedostatky metody detekce pomocí signatur a rovnic anomálie. Navíc úspěšnost této metody dosahuje velmi dobrých výsledků, kde hodnota přesnosti určení správného výsledku činí až 99,5 %.

Literatura

[1] *Empty DDoS Threats: Meet the Armada Collective* [online]. 2016 [cit. 2020-10-30]. Dostupné z: https://blog.cloudflare.com/empty-ddos-threats-meet-the-armada-collective/

[2] *Number of connected devices reached 22 billion, where is the revenue?* [online]. 2019 [cit. 2020-10-30]. Dostupné z: https://www.helpnetsecurity.com/2019/05/23/connected-devices-growth/

[3] *Effective Botnet Detection ThroughNeural Networks on Convolutional Features* [online]. 2018, , 7 [cit. 2020-10-21]. ISSN 2324-9013. Dostupné z: doi:10.1109/TrustCom/BigDataSE.2018.00062

[4] *What is a botnet?* [online]. 2017 [cit. 2020-10-31]. Dostupné z: https://www.pandasecurity.com/en/mediacenter/security/what-is-a-botnet/

[5] *C&C server* [online]. 2015 [cit. 2020-11-07]. Dostupné z: http://timehosting.cz/cc-server/

[6] 503 Service Unavailable. *503 Service Unavailable* [online]. 2020 [cit. 2020-12-06]. Dostupné z: https://www.lifewire.com/503-service-unavailable-explained-2622940

[7] *Jak funguje DDoS, hlavní zbraň kyberválky* [online]. 2010 [cit. 2021-5-27]. Dostupné z: https://www.zive.cz/clanky/jak-funguje-ddos-hlavni-zbran-kybervalky/sc-3-a-155080/default.aspx

[8] *Types of DDoS Attacks* [online]. 2017 [cit. 2020-10-22]. Dostupné z: https://www.esecurityplanet.com/network-security/types-of-ddos-attacks.html

[9] *Noční můra jménem SYN flooding* [online]. 1999 [cit. 2020-10-22]. Dostupné z: https://www.root.cz/clanky/nocni-mura-jmenem-syn-flooding/

[10] *Denial of Service útoky: reflektivní a zesilující typy* [online]. 2006 [cit. 2020-10-22]. Dostupné z: https://www.lupa.cz/clanky/denial-of-service-utoky-reflektivni-a-zesilujici-typy/

[11] *DNS Amplification — Protecting Unrestricted (Open) DNS Resolvers* [online]. 2016 [cit. 2021-5-27]. Dostupné z: https://www.tripwire.com/state-of-security/security-data-protection/cyber-security/dns-amplification-protecting-unrestricted-open-dns-resolvers/

[12] Relační vrstva: Co je čím ... v počítačových sítích. *Computerworld č. 26/92* [online]. 1992, (37) [cit. 2020-10-31]. Dostupné z: https://www.earchiv.cz/a92/a225c110.php3

[13] *Telnet stále žije — alespoň na -chytrých" zařízeních* [online]. 2016 [cit. 2020-11-08]. Dostupné z: https://www.root.cz/clanky/telnet-stale-zije-alespon-na-chytrych-zarizenich/

[14] *Network attack and Countermeasures Based on telnet connection in the era of Internet of Things* [online]. 2020 [cit. 2020-11-08]. Dostupné z: doi:10.1109/ICUEMS50872.2020.00155

[15] Prezentační vrstva: Co je čím ... v počítačových sítích. *Computerworld č.26/92* [online]. 1992, **1992**(37) [cit. 2020-10-31]. Dostupné z: https://www.earchiv.cz/a92/a226c110.php3

[16] *SSL Attacks - SSL DDoS Attacks* [online]. 2016 [cit. 2020-11-08]. Dostupné z: https://security.radware.com/ddos-threats-attacks/ddos-attack-types/ssl-based-ddos-attacks/

[17] *Co je SSL?* [online]. 2019 [cit. 2020-11-08]. Dostupné z: https://www.ssl.com/cs/Nej%C4%8Dast%C4%9Bj%C5%A1%C3%AD-dotazy/faq-co-je-ssl/

[18] Aplikační vrstva. *Computerworld č.27/92: Co je čím ... v počítačových sítích* [online]. **1992**(38) [cit. 2020-10-31]. Dostupné z: https://www.earchiv.cz/a92/a227c110.php3

[19] *Performance Comparison and Analysis of Slowloris,GoldenEye and Xerxes DDoS Attack Tools* [online]. 2018 [cit. 2020-10-23]. Dostupné z: doi:10.1109/ICACCI.2018.8554590

[20] *Effectively Using and Detecting The Slowloris HTTP DoS Tool* [online]. 2015 [cit. 2020-11-15]. Dostupné z: https://ma.ttias.be/effectively-using-detecting-the-slowloris-http-dos-tool/

[21] YALTIRAKLI, Gokberk. *Slowloris* [online]. 2015 [cit. 2021-5-22]. Dostupné z: https://github.com/gkbrk/slowloris

[22] SPILLANE, Chris. *Nástroj Pentmenu* [online]. [cit. 2021-5-22]. Dostupné z: https://github.com/GinjaChris/pentmenu

[23] *Zeek: An Open Source Network Security Monitoring Tool* [online]. The Zeek Project [cit. 2021-5-22]. Dostupné z: https://zeek.org/get-zeek/

[24] *PENTEREPMAIL: Virtuální stroj pro výuku hackingu a IT bezpečnosti* [online]. HACKER Consulting, 2020 [cit. 2021-5-22]. Dostupné z: https://www.penterep.com/penterepmail

[25] *Learn How to install GNS3 VM and Link With Latest GNS3 2.x* [online]. [cit. 2021-04-11]. Dostupné z: https://luminisindia.com/it-networking-blog/167-learn-how-to-install-gns3-vm-and-link-with-latest-gns3-2-0

[26] *Kali Linux Downloads* [online]. [cit. 2021-5-22]. Dostupné z: https://www.kali.org/downloads/

[27] *Download Ubuntu Desktop* [online]. [cit. 2021-5-22]. Dostupné z: https://ubuntu.com/download/desktop

[28] JEMEC, Miha. *PackETH* [online]. 2003. [cit. 2021-5-22]. Dostupné z: https://github.com/jemcek/packETH

[29] BOTTA, Alessio, Alberto DAINOTTI a Antonio PESCAP. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*. 56. **2012**(15), 3531-3547. Dostupné z: doi:https://doi.org/10.1016/j.comnet.2012.02.019

[30] EHLERS, Bernhard. *Ostinato for GNS3* [online]. [cit. 2021-5-22]. Dostupné z: https://www.b-ehlers.de/projects/ostinato4gns3/index.html

[31] *Útok Slowloris aneb plíživé nebezpečí pro web servery* [online]. 2011, 2011 [cit. 2020-11-01]. Dostupné z: https://www.root.cz/clanky/utok-slowloris-aneb-plizive-nebezpeci-pro-web-servery/

[32] *Mitigate Slow HTTP GET/POST Vulnerabilities in the Apache HTTP Server* [online]. 2019, 6. 6. 2019 [cit. 2020-12-06]. Dostupné z: https://www.acunetix.com/blog/articles/slow-http-dos-attacks-mitigate-apache-http-server/

[33] *How To Create a Self-Signed SSL Certificate for Apache in Ubuntu 20.04* [online]. 6. 7. 2020 [cit. 2020-11-29]. Dostupné z: https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-20-04

[34] *How to Install And Setup Suricata IDS on Ubuntu 20.04* [online]. 28. 9. 2020 [cit. 2020-11-29]. Dostupné z: https://www.atlantic.net/vps-hosting/how-to-install-and-setup-suricata-ids-on-ubuntu-20-04/

[35] *Quality of Service (QoS) Comparison Analysis of Snort IDS and Bro IDS Application in Software Define Network (SDN) Architecture* [online]. 2019, , 7 [cit. 2021-04-18]. Dostupné z: doi:10.1109/ICoICT.2019.8835211

[36] *Evaluation of traffic generators over a 40Gbps link* [online]. 2014, , 5 [cit. 2021-04-11]. Dostupné z: doi:10.1109/APCASE.2014.6924469

[37] *KNN-STUFF: kNN STreaming Unit for Fpgas* [online]. 2019, , 14 [cit. 2021-4-25]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2019.2955864

[38] *Malware detection using linear SVM* [online]. 2013, , 3 [cit. 2021-4-25]. Dostupné z: doi:10.1109/IFOST.2013.6616872

[39] *Pandas Installation* [online]. [cit. 2021-5-22]. Dostupné z: https://pandas.pydata.org/docs/getting_started/install.html

[40] *An overview and comparison of free Python libraries for data mining and big data analysis* [online]. 2019 [cit. 2021-5-22]. Dostupné z: doi:10.23919/MIPRO.2019.8757088

[41] DGUNTER. *ParseZeekLogs* [online]. [cit. 2021-5-22]. Dostupné z: https://github.com/dgunter/ParseZeekLogs

[42] *Installing scikit-learn* [online]. scikit-learn developers, 2007 [cit. 2021-5-22]. Dostupné z: https://scikit-learn.org/stable/install.html

[43] *Top 9 Python Libraries for Machine Learning in 2021* [online]. 2021 [cit. 2021-5-2]. Dostupné z: https://www.upgrad.com/blog/top-python-libraries-for-machine-learning/

[44] *Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on?* [online]. 2019 [cit. 2021-5-29]. Dostupné z: https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124

Seznam zkratek

C&C	Command-and-control
DDoS	Distributed Denied of Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoS	Denial of Service
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention Systems
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
k-NN	k-nearest neighbors algorithm
OSI	Open System Interconnection Reference Model
RAM	Random Access Memory
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
SVM	Support vector machines
TCP	Transmission Control Protocol

TN	True Negative
Tor	The Onion Router
TP	True Positive
UDP	User Datagram Protocol
VPN	Virtual Private Network

A Obsah přiloženého CD

Zdrojové kódy byly vytvářeny a testovány v integrovaném vývojovém prostředí `Pycharm Community Edition` 2020.3.4 s `Pythonem` ve verzi 3.8.

- `/` ..... kořenový adresář přiloženého CD
  - `DataFiles_Input` ..... vstupní soubory
    - `Datasety` ..... síťové podklady z celého provozu
  - `Filtrace_provozu`
    - `FiltraceProvozu.py` ..... zdrojový kód
    - `FiltraceProvozuAplikacniVrstvy.py` ..... zdrojový kód
  - `Pomocné_soubory`
  - `DataFiles_Output` ..... výstupní soubory z filtrace
    - `PodkladyProStrojoveUceni.csv` ..... klíčové parametry pro strojové učení
  - `Metody_detekce` ..... metody detekce strojové učení
    - `K-NN.py` ..... zdrojový kód
    - `DecisionTree.py` ..... zdrojový kód
    - `SVM.py` ..... zdrojový kód