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

BEHAVIORÁLNÍ ANALÝZA SÍŤOVÉHO PROVOZU A DETEKCE ÚTOKŮ (D)DOS

NETWORK BEHAVIOR ANALYSIS AND (D)DOS ATTACK DETECTION

DIPLOMOVÁ PRÁCE

DIPLOMA THESIS

AUTOR PRÁCE
AUTHOR

Bc. David Chapčák

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Lukáš Malina, Ph.D.

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. David Chapčák ***ID:*** 154141
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Behaviorální analýza síťového provozu a detekce útoků (D)DoS

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s metodami a nástroji pro monitorování a analýzu datového provozu. Zaměřte se především na opensource nástroje. Tyto nástroje otestujte na reálném datovém provozu a zhodnoťte jejich činnost. Dále se zaměřte na behaviorální analýzu provozu a možnosti detekce anomálií a útoků (D)DoS. Navrhněte řešení umožňující monitoring a analýzu provozu, klasifikaci a detekci anomálií a útoků (D)DoS. Navržené řešení implementujte a ověřte jeho funkčnost. Otestujte a zhodnoťte implementované řešení a proveďte případné funkční rozšíření (např. dodání GUI) a optimalizace.

DOPORUČENÁ LITERATURA:

[1] Pfleeger, Charles P., and Shari Lawrence Pfleeger. Analyzing computer security: A threat/vulnerability/countermeasure approach. Prentice Hall Professional, 2012.

[2] Boyles, Tim. CCNA Security Study Guide: Exam 640 - 553. John Wiley and Sons, 2010.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 24.5.2017

Vedoucí práce: Ing. Lukáš Malina, Ph.D.

Konzultant diplomové práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Tato práce se zabývá rozborem současných open-source NIDPS nástrojů pro monitorování a analýzu datového provozu a implementováním navrhnutého řešení pro detekci útoků typu DDoS s pomocí využití jednoduché lineární regrese.

V první části se práce se zaměřuje na posouzení těchto nástrojů z hlediska umístění v síti, vyžadované funkce a detailnější rozbor mechanismů pro detekci s upozorněním na neobvyklou událost. Dále se v práci rozebírají možnosti detekce anomálií zejména z pohledu statistické analýzy, ale představuje i základy dalších přístupů, jako jsou přístupy založené na dolování dat nebo strojovém učení. V další části se práce zabývá konkrétními open-source nástroji, prakticky porovnává jejich činnosti a navrhuje řešení umožňující monitoring, analýzu provozu, klasifikaci a detekci anomálií a útoků DDoS. V poslední části implementuje navrženou metodu do nástroje Bro, kde testuje a zhodnocuje její činnost při nasazení v experimentální síti VUT v Brně.

KLÍČOVÁ SLOVA

IDS, IPS, Detekce podle anomálií, behaviorální analýza, Snort, Bro, Suricata. Lineární regresní analýza, DDoS.

ABSTRACT

The aim of this thesis is to analyze the modern open-source NIDPS tools for monitoring and analyzing the network traffic and implement proposed solution for detecting DDoS attacks using simple linear reggression.

The first part of the thesis presents these instruments in terms of their architecture and functionality. Thesis also refers more about the detailed analysis of detecting and alerting mechanisms. Then the thesis analyzes the possibilities of detection of anomalies, especially in terms of the statistical analysis and shows the basics of other approaches, such as approaches based on data mining and machine learning. The next section of the thesis presents specific open-source tools, deals with comparison of their activities and the proposal allowing monitoring and traffic analysis, classification, detection of anomalies and DDoS attacks. In the last part, the statistical method is implemented. This solution is tested in an experimental BUT network and then is evaluated it’s performance.

KEYWORDS

IDS, IPS, Anomaly detection, Behavioral analysis, Snort, Bro, Suricata, Linear regression, DDoS.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Behaviorální analýza síťového provozu a detekce útoků DDoS“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, z nichž jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonu (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.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Lukáši Malinovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czech Republic

http://www.six.feec.vutbr.cz

Experimentální část této diplomové práce byla realizována na výzkumné infrastruktuře vybudované v rámci projektu CZ.1.05/2.1.00/03.0072

Centrum senzorických, informačních a komunikačních systémů (SIX)

operačního programu Výzkum a vývoj pro inovace.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Stále rychleji se rozvíjející oblast informačních technologií a stále zvětšující se základna uživatelů využívajících jejich služeb, kromě benefitů, představuje hrozbu v podobě útoků na jejich systém a data. To, jaké následky může v současné době přinést například výpadek určité části služeb jednoho z mnoha poskytovatelů, se dá docela dobře představit. Síť a její bezpečnost představuje téměř pro každou firmu základní kámen k rozvoji a k jejímu úspěchu. Bezpečnost sítě se ovšem nevztahuje pouze na ekonomický úspěch v podnikatelském sektoru, ale i na národní bezpečnost jednotlivých států. Kvůli zvyšující se životní úrovni nebo například příchodu technologie internetu věcí, se dá očekávat ještě větší propojení a závislost na informačních technologiích a v případě nedostatečného zabezpečení hrozí nedozírné následky.

Stejně tak, jak rychle se rozvíjí technologie v této oblasti, rozvíjí se i oblast útoků, na které je potřeba reagovat. Reálné případy ukazují na nejrůznější motivy a cíle útočníků. Jejich schopnost vytvářet stále nové přístupy představuje pro danou oblast nutnost vývoje nových zabezpečovacích mechanismů.

Jedním ze způsobů zdokonalení bezpečnosti mohou být systémy IDPS (Intrusion Detection and Prevention Systems). Jejich kombinace se stávajícím zabezpečením poskytuje velký benefit pro bezpečnostní analýzu sítě a zejména schopnost těchto systémů detekovat tzv. zero-day útoky, se v dnešní době může stát klíčovým prvkem. Rozdělení systému podle funkcionality se věnuje první kapitola práce. Druhá kapitola rozebírá jejich mechanismus.

Detekce k odhalení tohoto typu útoků, může být prováděna pomocí behaviorální analýzy, která využívá velkého množství nejrůznějších přístupů, jež poskytují prostor pro optimalizaci a vývoj. Přehled těchto přístupů lze nalézt ve třetí kapitole. Na akademické půdě vzniklo již mnoho teorií, ať už se jednalo o návrhy statistického přístupu, nebo konkrétní moduly implementované do některého z dostupných systémů a zhodnocení jejich funkce. Právě dostupným open-source řešením se věnuje čtvrtá kapitola.

V budoucnu se nedá očekávat pokles počtu útoků a je jen otázkou, jaké přístupy budou pro ochranu nejvhodnější. Podle současného vývoje však můžeme soudit, že IDPS směřují tím správným směrem a jejich význam se bude nadále zvyšovat.

Další část práce se věnuje vlastnímu návrhu řešení pro detekci útoků typu DoS. Jako vhodná metoda pro tuto funkci byla zvolena lineární regresní analýza, která je implementována v podobě modulu do nástroje Bro. Výhody této metody spolu s návrhem včetně implementace jsou popsány v 7. kapitole.

Poslední část práce se věnuje testování navržené metody v experimentálním pracovišti VUT v Brně a její optimalizaci včetně zhodnocení z pohledu reakce na různé typy průběhu útoku.

1 TYPY MONITOROVACÍCH NÁSTROJŮ

Diplomová práce se zabývá nástroji, které slouží k monitorování a ovlivňování provozu na datové síti za účelem zvýšení její bezpečnosti. Na trhu je pro tuto potřebu hned řada vhodných kandidátů. Obecně se dají rozdělit na dvě skupiny, a totiž na nástroje IDS (Intrusion Detection Systems) a IPS (Intrusion Prevention Systems). Zatímco IDS slouží, jak tomu již napovídá jejich název, hlavně k detekci útoků a procesů, které jim předcházejí, ke generování upozornění a ukládání logů o proběhlých podezřelých aktivitách, IPS slouží k prevenci a aktivní ochraně. Většina dnešních nástrojů však tyto dva systémy kombinuje. Hlavní rozdíly mezi systémy jsou vidět v tabulce níže (Tab. 1.1).

Na počátku vznikly jako přídavné prvky, k již existujícím firewallům, či antivirovým programům, postupem času se však stále více rozvíjely až do té míry, že se vyvinuly ve zcela nezávislé odvětví nástrojů pomáhajících zlepšovat bezpečnost sítě.

Podle oblasti působnosti lze oba tyto systémy dělit na síťově založené a uzlově orientované, přičemž každý plní mírně odlišnou roli.

Je třeba podotknout, že systémy IPS i IDS kombinují mnoho nejrůznějších prvků, které posunují kvalitu monitorování sítě na vyšší úroveň a pro zvýšení bezpečnosti datové sítě je nejvhodnější využít oba systémy spolu s firewallem. Další informace o typech je možné dohledat v [1] a [2].

Tab. 1.1: Klíčové rozdíly IDS a IPS.

	Systémy pro odhalení průniku (IDS)	**Systémy pro prevenci průniku (IPS)**
Podpora	Podporují síťově i uzlově založené přístupy (NIDS a HIDS).	Podporují síťově i uzlově založené přístupy (HIPS a NIPS).
Činnost	Pasivně monitorují síť, její chování a detekují útoky.	Aktivně analyzují a poskytují real-time prevenci proti útokům.
Umístění	Nalézají se mimo datový proud.	Nalézají se přímo v datovém proudu.
Požadavky	Hlavním požadavkem je přesnost detekce.	Hlavním požadavkem je menší počet falešných pozitivních hlášení.

1.1 IDS

IDS je systém detekující nežádoucí průniky do sítě, který vhodně doplňuje firewall. Ale zatímco firewall kontroluje na základě daných pravidel pouze hlavičky TCP/IP a v případě škodlivých dat může tento paket propustit dále, IDS prověřuje obsah každého paketu směřujícího do monitorované sítě až na 3. vrstvu TCP/IP modelu, čímž sice zvyšuje výpočetní náročnost, ale minimalizuje riziko propuštění nežádoucí datové jednotky.

Primárním úkolem systému IDS je automatická detekce útoků, jejíž výhodou je odstranění nutnosti prozkoumávání logů zachyceného provozu, na jehož základě se dříve hrozby vyhodnocovaly.

Podle signalizace útoků a skutečné hrozby útoků lze IDS hodnotit podle tabulky níže (Tab. 1.2). Z té vyplývá, že nejdůležitějším úkolem pro IDS je maximalizovat počet pravých-pozitivních a minimalizovat počet nepravých stavů.

Tab. 1.2: Přehled možných stavů

	Útok	Upozornění
True positive (Pravý pozitivní)	✓	✓
True negative (Pravý negativní)		
False positive (Nepravý pozitivní)		✓
False negative (Nepravý negativní)	✓	

Existují dvě varianty systému IDS. Jsou to uzlově založený HIDS (Host Based Intrusion Detection Systems) a síťově založený NIDS (Network Intrusion Detection Systems).

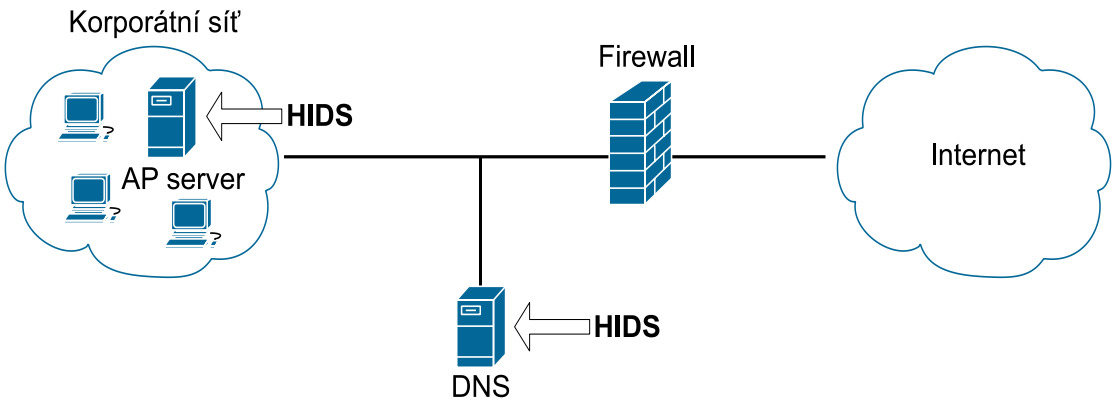
1.1.1 HIDS

Základní funkce HIDS je ochrana koncového systému. Většinou je instalován na koncovém PC nebo serverech, kde monitoruje a oznamuje aktivitu aplikací (Obr. 1.1). Má za úkol neustále monitorovat:

- Záznamy událostí.
- Záznamy systému.
- Záznamy aplikací.
- Uplatňování bezpečnostní politiky uživatele.
- Detekci rootkitu.
- Integritu souborů.

Ze záznamů si systém vytváří základní přehled o událostech. V případě útoku, který nějakým způsobem neodpovídá zaznamenaným datům, spouští HIDS upozornění nebo danou aktivitu blokuje či provádí jinou akci založenou na bezpečnostní politice konkrétního případu. [3]

Většina nástrojů HIDS poskytuje možnost předcházení útoků na bázi předchozího monitorování normálního provozu v síti, z čehož plyne závislost na záznamech generovaných systémem. Také se spoléhá na skutečnost, že útočník po sobě zanechá viditelné stopy, ať už v konfiguraci OS, registrech nebo dalších záznamech systému. Pokud je ovšem útočník schopný nezanechat po sobě žádné stopy tohoto druhu, HIDS nedokáže rozpoznat jeho aktivitu, což je jedna z největších nevýhod systému. O HIDS dále referují [1] a [2].

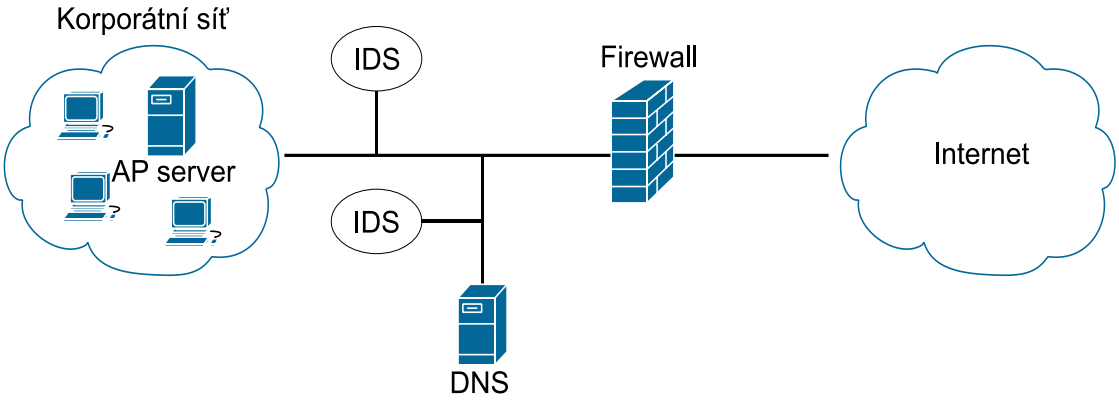


Obr. 1.1: Umístění HIDS v síti.

1.1.2 NIDS

NIDS je síťově založené hardwarové nebo softwarové řešení, jehož funkcí je monitorovat a detekovat podezřelou aktivitu na síti pomocí inspekce každého paketu, který vstupuje do monitorované sítě. Tyto pakety potom podrobuje porovnání se známými profily nebo pravidly a případně spouští upozornění na nebezpečný provoz.

Hardwarové řešení bývá umístěno v síti jako nezávislý senzor mající vlastní síťový operační systém nalézající se blízko firewallu (Obr. 1.2). Senzory využívají k monitorování a ovládání vlastní rozhraní. Více informací o NIDS lze nalézt v [1] a [2].



Obr. 1.2: Umístění NIDS v síti.

1.2 IPS

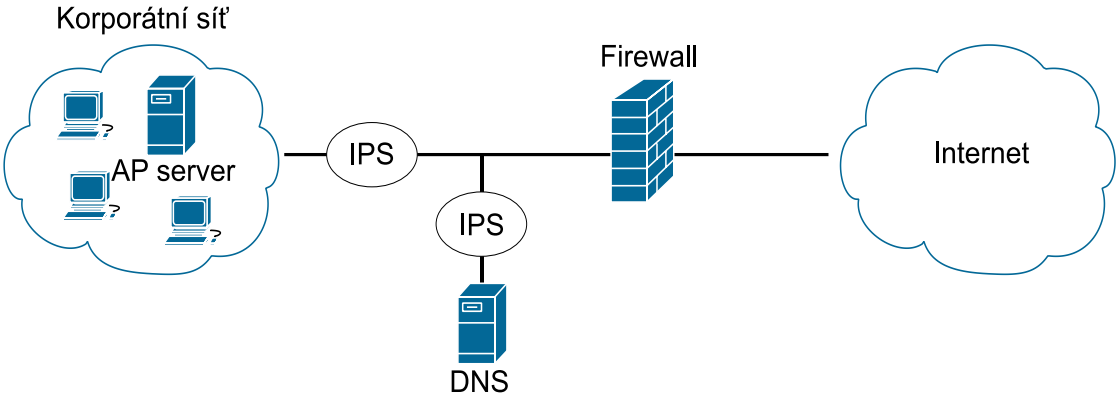
IPS jsou zařízení nebo aplikace umožňující předcházet útokům namířeným proti prvkům v komunikační síti. Jsou považovány za rozšíření již zmiňovaných systémů IDS. Ale zatímco primární funkcí IDS je detekce, IPS jsou schopny ochrany sítě díky své schopnosti zahození paketů, zamezení přístupů nebo blokací celého spojení. Dalším rozdílem je umístění v síti, kde jsou IPS umístěny „in-line“, podobně jako tomu je u firewallu. K další rozdílnosti patří schopnost zasahování do probíhající komunikace. Další informace o IPS lze nalézt v [2] a [3].

IPS lze podobně jako IDS rozdělit do dvou kategorií, na HIPS (Uzlově založené) a NIPS (Síťově založené), rozdíly mezi těmito dvěma systémy jsou v oblasti, kde vykonávají svoji činnost. Jak již z názvu vyplývá, HIPS jsou umístěny v zařízení, kde

pracují pouze s dostupnými logy aplikací a zkoumají jeho vnitřní procesy. NIPS má, na rozdíl od HIPS, svou oblast působnosti posunutou dále do sítě, takže se soustředí na analýzu dat přicházejících na vnější rozhraní.

Co se týče umístění v síti, nástroje IPS je nejvhodnější implementovat do částí mezi sítěmi, jejichž provoz chceme monitorovat. Zde mohou předávat provoz jako zařízení pracující na spojové vrstvě (Obr. 1.3).

Pro lepší představu lze u systémů IPS užít analogii s firewallem, ale zatímco firewall všechny pakety, které nesplňují alespoň jedno pravidlo, zahazuje, IPS využívá reverzního přístupu a pakety, které nesplňují žádné z předem nakonfigurovaných pravidel, propouští dále.



Obr. 1.3: Umístění IPS v síti.

2 MECHANISMY ZPRACOVÁNÍ DATOVÉHO PROVOZU U IDPS

V současné době většina síťově založených systémů pro odhalení průniků kombinuje oba dva přístupy, jak monitorovací, tak preventivní a nazývají se souhrnně IDPS (Intrusion Detection and Prevention Systems).

Mechanismy zpracování datového provozu se z velké části u každého konkrétního druhu nástroje liší. Za společného jmenovatele všech IDPS nástrojů se může považovat detekce konkrétních příznaků a akce s nimi spojené.

2.1 Detekce signatur u IDPS

Z podstaty IDPS plyne, že jsou to nástroje reagující na určitý neobvyklý jev v síťovém provozu. Aby toho byly schopny, je zapotřebí blíže specifikovat konkrétní příznak daného jevu, iniciátor dané akce, kterou na základě rozpoznání příznaku hrozby vykoná, a druh samotné akce.

Druhy detekce signatur pro další nakládání s provozem se můžou rozdělit podle jejich funkcionality do dvou základních kategorií, na atomická a stavová.

2.1.1 Atomická detekce signatur

Atomické detekce jsou základním a nejjednodušším typem definovaných detekcí, na jejichž základě se zkoumá pouze jedna událost nebo paket nezávisle na ostatních. Po každém dokončení tohoto procesu se vyhodnocuje, zda se spustí příslušná akce či nikoliv. Za jejich výhodu se dá považovat zejména úspornost výpočetních prostředků, přehlednost jednotlivých pravidel a snadná analýza. [2]

Na druhou stranu mají i atomické detekce své nevýhody. První z nich je potřeba znalosti všech událostí, které mají detekovat, zároveň s tím, že pro každou událost se musí vytvořit příslušné pravidlo detekce. To může v sofistikovanějším systému vyvolat postupem času zmatek ve spravování a značně znepřehlednit bezpečnostní politiku. Další z nevýhod je náchylnost ke generování falešných upozornění, která může znesnadnit zachycování vážnějších hrozeb.

2.1.2 Stavová detekce signatur

Stavová detekce signatur se od těch atomických liší zejména v reakci na řadu událostí, které vyžadují od IDPS uchování záznamů o předchozích stavech. Důležitou proměnnou je u těchto detekcí tzv. horizont událostí, jenž udává dobu, po kterou si IDPS uchovává v paměti předchozí stavy. Tato doba se počítá od chvíle, kdy je zaznamenán počáteční podezřelý paket. Nastavení hodnoty horizontu událostí je poměrně citlivou záležitostí, jelikož je potřeba určit vyvážený poměr vzhledem ke spotřebě systémových prostředků a dobou útoku. Pro každý typ detekce se nastavuje jiná délka horizontu událostí. Tento typ detekce využívají síťově založené systémy IDPS, jelikož v drtivé většině případů je nutné prověřit více paketů a udržovat tak jejich stavy. [2]

Potřeba detekce konkrétní události v širším kontextu zvyšuje pravděpodobnost, s jakou jsou detekovány reálné útoky, což snižuje falešné poplachy.

Mezi nevýhody patří již zmíněné nadměrné využití systémových prostředků, nicméně tento problém lze vyřešit správnou konfigurací. O detekcích je možno nalézt více v [2] a [3].

2.2 Spouštěče událostí

Další podstatnou částí IDPS je samotný mechanismus, který zapříčiní spuštění příslušné akce.

Tento mechanismus může být relativně jednoduchý, ale i komplexnější, v závislosti na potřebách konkrétní situace. Spouštěč reaguje jak na atomické, tak na stavové detekce značek. Mechanismy lze rozdělit do tří skupin podle toho, jaká událost je spouští.

2.2.1 Detekce podle vzoru

Detekce podle vzoru je elementární mechanismus pro spouštění událostí. Reaguje na určitý vzor, např. na binární nebo textovou posloupnost. V oblasti síťově založených IDPS je tato detekce využívána nejčastěji, protože mnoho příznaků útoků vyžaduje nutnost vyhledat určitý vzor v jednom (atomické detekce) nebo více (stavové detekce) paketech. Jako příklad může sloužit třeba e-mail s přílohou, která obsahuje známý malware nebo vzdálený přístup, který porušuje bezpečnostní politiku společnosti. Databázi známých vzorů je nutné pravidelně aktualizovat, nejčastěji se tak děje s pomocí

antivirů. Kvůli pravidelné aktualizaci ovšem databáze neustále narůstá a postupem času může působit relativně velkou zátěž na výpočetní prostředky systému.

Podstatnou nevýhodou atomické detekce je neschopnost hledání vzoru v zašifrovaném provozu nebo nutnost použití několika vzorů pro jediný typ útoku.

2.2.2 Detekce podle anomálie

Detekce podle anomálie, na rozdíl od předchozího mechanismu nereagují na určitý vzor v provozu, ale jak již napovídá název, reagují na určité aktivity, které se nějakým způsobem vymykají z nastaveného normálního profilu. Konfigurace normálního profilu je závislá na individuálních požadavcích sítě. Nejčastěji se posoudí pomocí časově omezeného monitorování sítě při běžném provozu. Základním problémem, kterým se musí vývojáři u této detekce potýkat, je neustálý pohyb anomálií na sítí, zároveň s tím, že by se měl celý systém vyvarovat určení na základě předem definovaného setu anomálií. Podrobněji se bude tímto typem detekce práce zabývat v další kapitole.

2.2.3 Detekce podle chování

Poslední ze tří mechanismů detekce je založena na znalosti projevu útoků, protože musí být, stejně jako detekce podle vzoru, definována před samotným použitím. Zatímco však u detekce podle vzoru definujeme vždy pouze jeden vzor, na jehož základě je následně posuzován provoz, u detekce podle chování definujeme přímo podezřelé chování, které značí neobvyklou aktivitu. Může se například jednat o nalezení jistého příkazu v historii, který používají útočníci po prolomení obrany napadeného zařízení.

Díky tomuto mechanismu jsme schopni na základě jednoho pravidla pokrýt celou řadu podezřelých událostí bez nutnosti specifikovat každou zvlášť. Nevýhodou je však hledání známek útoku až po jeho provedení. Detekcemi se podrobněji zabývá [2] a [3].

2.3 Akce

V případě detekce podezřelé aktivity se spouští příslušná akce, která se může znovu dělit na více kategorií.

2.3.1 Akce spouštějící výstrahu

Akce spouštějící výstrahu nejčastěji upozorňuje na útok pomocí vygenerované zprávy v konzoli. Může se tak stát při každém výskytu hrozby (atomické upozornění), jelikož však tato generace zpráv může v případě útoku typu flood dobře posloužit samotnému útočníkovi jako prostředek k zahlcení konzole správce, dá se využít ještě druhého typu upozornění, sumarizující tyto hlášení podle zdrojové adresy a portu.

V praxi se nejčastěji využívají oba tyto přístupy, přičemž počáteční upozornění jsou atomická, ale pokud se po nějaký čas stále opakují, IDPS přepne na sumarizaci.

2.3.2 Akce zahazující provoz

Akce zahazující provoz je jedna z nejzásadnějších akcí, kterou IDPS disponuje. Možnost zahození podezřelých paketů poskytuje IDPS silnou funkci k zastavení útoku ještě předtím, než má příležitost plně zasáhnout cílovou oblast. IDPS obecně nabízí tři způsoby zahození provozu.

Zahození podezřelého paketu je nejjednodušší formou ovlivňování provozu. Jak vyplývá z názvu, IDPS zpracuje podezřelý paket a na základě pravidel tento paket zahodí. Tato možnost není ovšem příliš výhodná, jelikož v případě cíleného útoku se IDPS zabývá vždy pouze jedním paketem, tudíž pokud se na síti objeví větší počet podezřelých datových jednotek, má to negativní dopad na výkonnost nástroje a propustnost celé sítě.

Dalším, sofistikovanějším způsobem je zahození skupiny paketů daného spojení. To může být definováno na základě zdrojové IP adresy, cílové IP adresy, cílového portu nebo případně zdrojového portu spojení. Výhodou oproti předchozímu přístupu je nižší zatížení IDPS, jelikož pakety mohou být zahazovány automaticky bez další analýzy. Na druhou stranu, i tento přístup má své negativní vlastnosti, a to zejména v útočníkově schopnosti obejít spojení na základě změny jeho typu.

Poslední z možností je zahození paketů na základě zdrojové IP adresy. V tomto případě je na základě předem definovaných pravidel zahozen paket určité zdrojové IP adresy a s ním všechny ostatní pakety všech spojení přicházející z konkrétní adresy. Výhodou přístupu je velmi nízké zatížení výpočetních prostředků IDPS, slabinou je naopak neschopnost reagovat na spoofing zdrojové IP. Více o této problematice se je možné dozvědět v [2].

2.3.3 Akce záznamu podezřelého provozu

V případě, kdy není zcela zřejmé, zda se jedná o útok, je možné zaznamenat podezřelou aktivitu nebo pakety pro detailnější prozkoumání. Díky tomuto je možné rozhodnout, zda v budoucnosti povolit, či zakázat případný další provoz tohoto druhu.

2.3.4 Akce blokování

Akce blokování umožňuje nástrojům IDPS blokovat provoz na vzdálených místech v síti díky aplikacím ACL (Access Control Lists), což poskytuje výhodu v šetření prostředků analyzátorům.

2.3.5 Akce resetování TCP spojení

Akce resetování TCP spojení je jednoduchá akce, která umožňuje generaci TCP paketu s nastavenou RST hodnotou v hlavičce. IDPS této akce využívají při náhlém ukončení TCP spojení, které vykazuje nechtěné operace.

2.3.6 Akce povolení

Akce povolení slouží k přidání výjimek do již nakonfigurovaných pravidel v případě nutnosti povolení pro nějakého uživatele či aplikaci. Nejvhodnějším nastavením je přístup zakázání všeho a následné přidání výjimek.

2.4 Zavedení IDPS

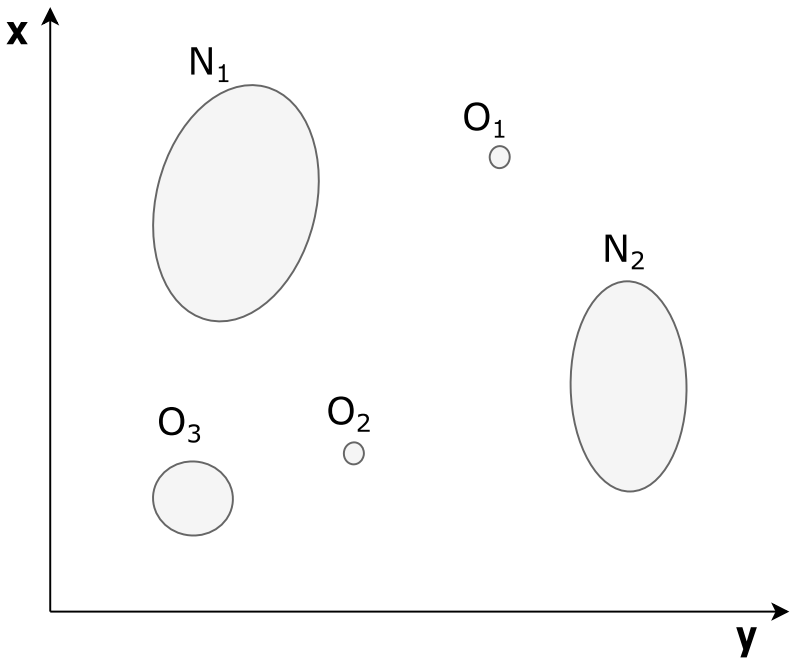
Jak již bylo zmíněno, systémy IDPS mohou pracovat ve dvou režimech. Prvním je monitorovací (pasivní), kde pouze sledují provoz na síti bez toho, aniž by ho nějak ovlivňovaly. Druhý je tzv. preventivní režim (aktivní, in-line), ve kterém již nástroje mohou do provozu zasahovat.

Samotné umístění senzorů se může lišit případ od případu podle topologie a konkrétních potřeb, které jsou od IDPS vyžadovány. V zásadě se doporučuje aplikovat systémy nejdříve v monitorovacím módu a až po zavedení přepnutí do preventivního.

3 DETEKCE PODLE ANOMÁLIE

Techniky detekce podle anomálie jsou využívány v celé řadě oborů. Můžeme se s nimi setkat ve zdravotnictví, finančním sektoru nebo právě v počítačové bezpečnosti. Pro každý případ se hodí specifický typ techniky detekce. Jelikož mechanismus generuje upozornění na neobvyklou aktivitu, je výhodou těchto systémů schopnost reagovat na předem neznámé typy útoků. Pokud se ale objeví předem známý útok, je obtížnější jej specifikovat. Systémy založené na této detekci nejčastěji ochraňují proti hrozbám, jako jsou nejrůznější změny v protokolech a aplikačních datech a také proti útokům (D)DoS.

Na obrázku (Obr. 3.1) představují oblasti O a N množiny výskytů zkoumaných dat ve dvourozměrném prostoru xy. N1 a N2 jsou považovány pro daný případ jako normální. Naopak oblasti O1, O2 a O3 se dají označit za anomální, protože se nacházejí dostatečně daleko od normálních oblastí. Úkolem detekce je odlišit anomální oblasti výskytů od normálních, nejčastěji pomocí vytvoření tzv. normálního profilu. Stanovení normálního profilu je záležitostí několika metod. Využívá se například statistického přístupu, který se snaží odhalit statistické odchylky od dříve pozorovaného provozu. Další informace o detekcích je možné dohledat v [4].



Obr. 3.1: Definice anomálií.

Jakmile je profil stanoven, přepne se systém z monitorovacího do detekčního módu a každá aktivita, která „nezapadá“ do dříve zaznamenané normy, spouští akci.

Nevýhodou mechanismu je jejich samotná podstata, a totiž nutnost stanovit, co je pro daný systém normální. V dnešní době se v sítích v relativně krátkém časovém úseku provádí mnoho změn a samotný provoz je sestaven z dat stále se vyvíjejících aplikací, zařízení a systémů, což velmi ztěžuje nastavení mechanismu detekce podle anomálie. Zároveň s tím je v mnoha případech hranice mezi „normálním“ a „nenormálním“ velmi tenká, takže může docházet k chybám. Další zápornou stránkou je nutnost zajištění provozu bez útoků a dalších neobvyklých aktivit během určování normálního profilu.

3.1 Statistická detekce anomálií

Základním předpokladem statistické detekce je fakt, že normální data se nacházejí v oblastech vysokých pravděpodobností stochaistického modelu, zatímco anomálie se objevují v oblastech s nízkou pravděpodobností.

Detekce na základě statistického přístupu má řadu výhod. Tou největší je možnost absence předchozí znalosti útoku, a tak je možné s její pomocí zachytit i zcela nové a neznámé druhy útoků (zero day útoky). Dále je statistický přístup vhodný k přesnému zachycení útoků, které trvají delší dobu (např. DDoS).

Jak to již bývá u všech přístupů zvykem, i tato detekce má své slabiny. Zkušení útočníci jsou schopni ovlivnit stanovení normálního profilu do té míry, že nejsou jejich pozdější útoky považovány za hrozbu. Dalším složitým úkolem je určení hranice detekčních mechanismů tak, aby se minimalizoval počet všech nepravých stavů (Tab. 1.2). Zároveň vyžaduje statistický přístup přesné statistické distribuce, ale ne všechny projevy útoků lze takto rozdělit. [6]

Statistická detekce anomálií používá mnoho různých algoritmů. Vzhledem k jejich široké škále bude v práci představeno pouze několik nejznámějších.

3.1.1 IDES algoritmus

Jako jeden z průkopníků IDS využívající algoritmus založený na statistickém přístupu, byl systém IDES vyvinutý na Stanfordské univerzitě.

Pro každou zaznamenanou událost generuje statistickou hodnotu T^2 nazvanou hodnotou skóre, která vyjadřuje stupeň abnormality v blízké minulosti. Vysoké hodnoty skóre znamenají podezřelé abnormální chování, zatímco hodnoty blížící se nule jsou přijaty jako normální.

Statistické T^2 je vyjádřeno jako posouzení abnormality mnoha měření. Předpokládejme, že je provedeno n měření označených jako S_i, $1 \leq i \leq n$. Korelace mezi S_i a S_k je vyjádřena C_{ik} kde $C_{ii} = 1{,}0$. Pro původní verzi IDES algoritmu bylo potom T^2 vyjádřeno jako:

$$T^2 = (S_1, S_2, \dots, S_n) C^{-1} (S_1, S_2, \dots, S_n)^t \tag{3.1}$$

Kde je C^{-1} inverzní korelační maticí vektoru $(S_1, S_2, \dots, S_n)$ a $(S_1, S_2, \dots, S_n)^t$ je jeho transpozice.

Bohužel tento algoritmus měl několik slabin. Například bylo velmi složité rozpoznat vliv jednotlivých S_i na T^2. Dalším problémem byla nemožnost zjistit, jaké S_i je nejvíce důležité, protože váhové koeficienty jsou zcela určeny korelační maticí C. Z těchto a dalších důvodů byl výpočet pro hodnotu skóre upraven do konečné podoby:

$$T^2 = \sum_{i=1}^{n} a_i S_i^2 + \sum_{i=1}^{n} a_{ij} h(S_i, S_j, C_{ij}) \tag{3.2}$$

Kde a_{ij} jsou ručně nastavené kladné koeficienty a $h(S_i, S_j, C_{ij})$ je funkce S_i, S_j a jejich korelace C_{ij}.Více o algoritmu lze nalézt v [4].

3.1.2 Detekce anomálií využívající distribuční provozní vlastnosti

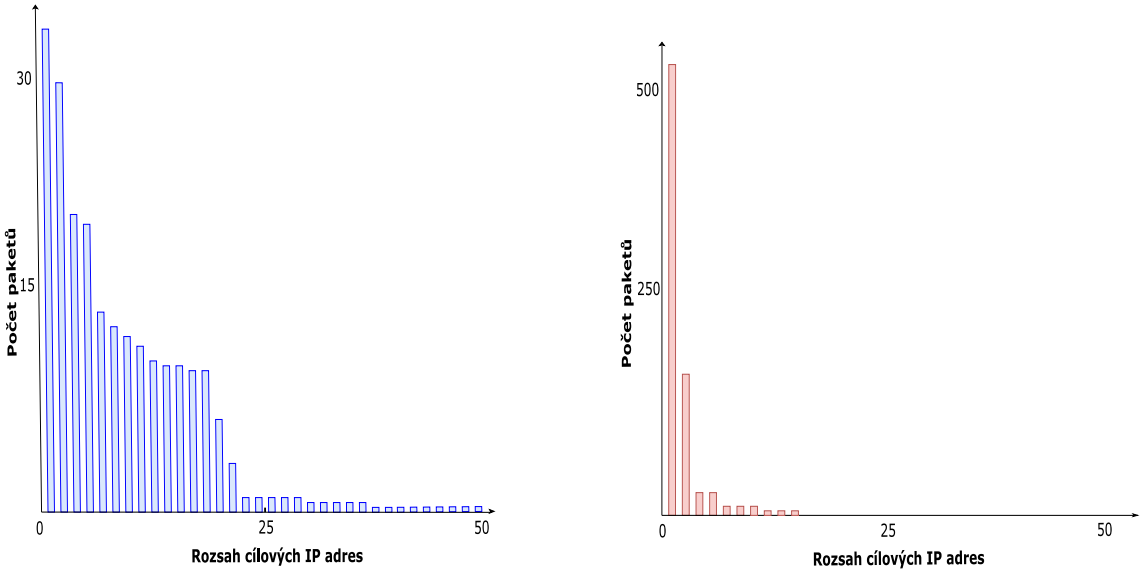
Detekce anomálií využívající distribuční provozní vlastnosti je typ detekce, který se soustředí na extrakci provozních anomálií z velkého množství dat. K tomuto účelu z nich čerpá a snaží se objevovat i interpretovat příznaky přítomné v provozu. Základem je fakt, že většina anomálií (útoků) sdílí společnou charakteristiku – vyvolávají distribuční změny v částech hlaviček paketů - cílová a zdrojová IP adresa + port. Přehled několika častých

útoků a jejich dopad na změny v distribučních provozních vlastnostech (dále jen DPV) je možné vidět v Tab. 3.1*:* Útoky a jejich vliv na DPV.. Pro stručnost se budou dále nazývat tyto části jako provozní vlastnosti. Například DoS útok působí, nezávisle na jeho objemu, velké změny v cílových adresách, které jsou soustředěny na oběť útoku. Technika se potom nezabývá přímo množstvím dat, ale zhodnocuje právě tento fakt v hlavičkách paketů. Jako příklad může posloužit graf (Obr. 3.2), na kterém první průběh (modrý) představuje přibližné distribuční rozložení IP adres při normálním provozu a druhý průběh (červený), který lze považovat za obraz rozložení při útoku.

Tab. 3.1: Útoky a jejich vliv na DPV.

Typ útoku	**Definice**	**Ovlivněné distribuční vlastnosti**
DOS	Odepření služby (distribuované nebo z jednoho zdroje).	Zdrojová IP, cílová IP a jejich porty
Skenování portů	Sondy k mnoha cílovým portům z menšího rozsahu IP adres.	Cílová IP, cílový port
Skenování sítě	Sondy k mnoha cílovým portům z menšího rozsahu IP adres.	Cílová IP, cílový port
Červi	Skenování slabých míst v síti, snaha šíření na ostatní uzly v síti a negativní činnost ovlivňující chod uzlu.	Cílová IP, cílový port

Tímto přístupem se systém odlišuje od ostatních technik, které se soustředí na množství dat. To přináší výhody v podobě schopnosti detekce takových typů anomálií, které je těžké rozpoznat ze změny množství dat na síti (např. malé DoS útoky v páteřní síti) a získání specifických informací.



Obr. 3.2: Porovnání distribuce cílových IP adres.

Základní metrikou, ohodnocující anomálie je entropie ze vzorků dat. Ta využívá rozptýlenosti v provozních vlastnostech při normálním provozu a při útoku. Jako příklad může znovu sloužit útok typu DoS. Při normálním provozu bez anomálií jsou cílové adresy v síti rozprostřeny více méně rovnoměrně po celém spektru. V případě útoku je zaznamenána velká změna v podobě radikálního zvýšení požadavků na konkrétní cílovou adresu. To způsobí v celkovém měřítku snížení rozprostřenosti provozních vlastností sítě, jež se dá vyjádřit pomocí entropie. Základem je histogram dat $X = \{n_i, i = 1, \dots, N\}$ vyjadřující množství hodnoty i v daném vzorku. Vzorec pro výpočet entropie má tvar:

$$H(X) = -\sum_{i=1}^{N} \left(\frac{n_i}{S}\right) log_2(\frac{n_i}{S}) \tag{3.3}$$

Kde $S = \sum_{i=1}^{N} n_i$ je celkový počet pozorování v histogramu. Hodnota entropie se pohybuje v rozmezí $(0;\log_2 N)$. Nabývá 0, pokud jsou všechna pozorování stejná (minimální možný rozptyl). Naopak maximální hodnoty $\log_2 N$ dosahuje v momentě, kdy jsou všechna pozorování odlišná (maximální možný rozptyl). Kompletní rozbor metody lze nalézt v [7].

Metoda subprostorů

V systému se tato metoda používá k detekci anomálií, které vznikají po výpočtu entropie. Identifikuje typické variace v souboru korelovaných dat a detekuje neobvyklé odchylky od typické variace.

Jako vstupní data slouží matice X o *t* řádcích vyjadřujících pozorování a *p* sloupcích vyjadřujících proměnné. Může se předpokládat, že sloupce vyjadřují korelaci, takže typická variace celé matice může být vyjádřena lineární kombinací méně než *p* proměnných. Použitím analýzy hlavních komponent (snížení dimenze dat s co nejmenší ztrátou), se může vybrat nový vzorek proměnných $m<<p$. který definuje nový m-dimenzionální subprostor a abnormální variace je poté definována jako jakákoliv významná odchylka dat z tohoto subprostoru.

3.1.3 Detekce na základě analýzy hlavičky paketu

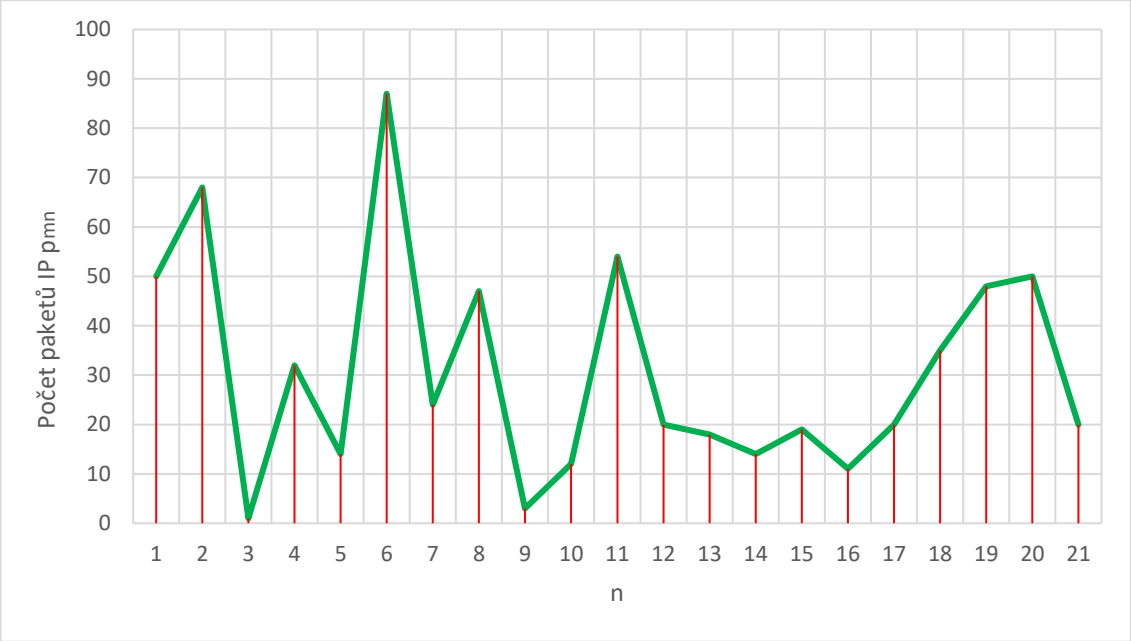
Detekce na základě analýzy hlavičky paketu je technika, která vychází z předpokladu, že v rámci delšího časového úseku lze pozorovat silné příznaky typického chování. Jako příklad může sloužit navyklé chování uživatelů, kteří navštěvují většinou své oblíbené stránky a celkově se v rámci delšího časového úseku chovají na síti stejně, takže je výsledkem jistá korelace. Jak již vyplývá z názvu, oblast zájmu je soustředěna na hlavičky paketů, konkrétně na cílovou adresu a port.

Generace signálu

Individuální pole v hlavičkách paketů nabývají diskrétních hodnot a u síťového provozu vykazují výrazné nesouvislosti. Jen těžko se dá předpokládat, že v konkrétní pozorované síti se objeví všechny adresy z rozsahu. Z tohoto důvodu se u detekce na základě analýzy hlavičky paketu převádí data hlaviček pomocí zjednodušené korelace vzorků na spojitý signál.

Pro každou adresu a_m počítá množství paketů p_{mn} poslaných v dané sekvenci. Pro výpočet výsledného signálu uvažujeme dvě vzdálené vzorkovací instance. V grafu (Obr. 3.3: Průběh vzorkování IP adresy *m*) lze vidět průběh vzorkování jedné IP adresy *m*. Konkrétní signál v místě odběru vzorku *n* má podobu:

$$C(n) = \sum_m p_{mn-1} * \frac{p_{mn}}{\sum_m p_{mn}} \quad (3.4)$$



Obr. 3.3: Průběh vzorkování IP adresy *m*.

Pro snížení výpočetní náročnosti a požadavků na prostor, se u této techniky používá datová struktura nazvaná count [i][j] pro záznam paketu adresy *j* v *i*-tém poli IP. Toto škálování poskytuje značnou úsporu v záznamech IP adres. Výpočet korelace ve dvou po sobě jdoucích vzorcích je následující:

$$C_{in} = \sum_{j=0}^{255} \frac{count[i][j][n-1]}{\sum_{j=0}^{255} count[i][j][n-1]} * \frac{count[i][j][n]}{\sum_{j=0}^{255} count[i][j][n]}, i = 1,2,3,4 \quad (3.5)$$

Pro generaci výsledného korelačního signálu S(n) na konci vzorkování se použije:

$$S(n) = A * (\alpha_1 * C_{1n} + \alpha_2 * C_{2n} + \alpha_3 * C_{3n} + \alpha_4 * C_{4n}) + B \quad (3.6)$$

Kde $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1$ jsou konstanty (scaling factors).[6]

Transformace signálu pro statistickou analýzu

K analýze signálu se dá použít FFT a diskrétní vlnková transformace. V podstatě se jedná o výpočet transformace z několika časových period signálu a následnou analýzu.

3.1.4 Detekce využívající vlnkové analýzy

Detekce využívající vlnkové analýzy je vhodný zejména pro detekování DDoS útoků. Jeho podstatou je monitorování změn v síti mezi předem definovanými časovými řadami.

Pokud je síť podrobena typem útoku DDoS, který ji zaplavuje náhodnými daty zabraňujícími ostatním datům v toku, projevují se okamžitě neobvyklé změny v množství paketů procházejícími sítí. Jak již bylo zmíněno, pro detekování tohoto typu útoků se dají v praxi aplikovat dvě metody. První je založena na stanovení „modelu“ normálního chování na síti a jeho porovnávání se současným stavem, zatímco druhá je založena na monitorování okamžitých změn, takže každá náhlá vyvolaná změna je považována za neobvyklé chování.

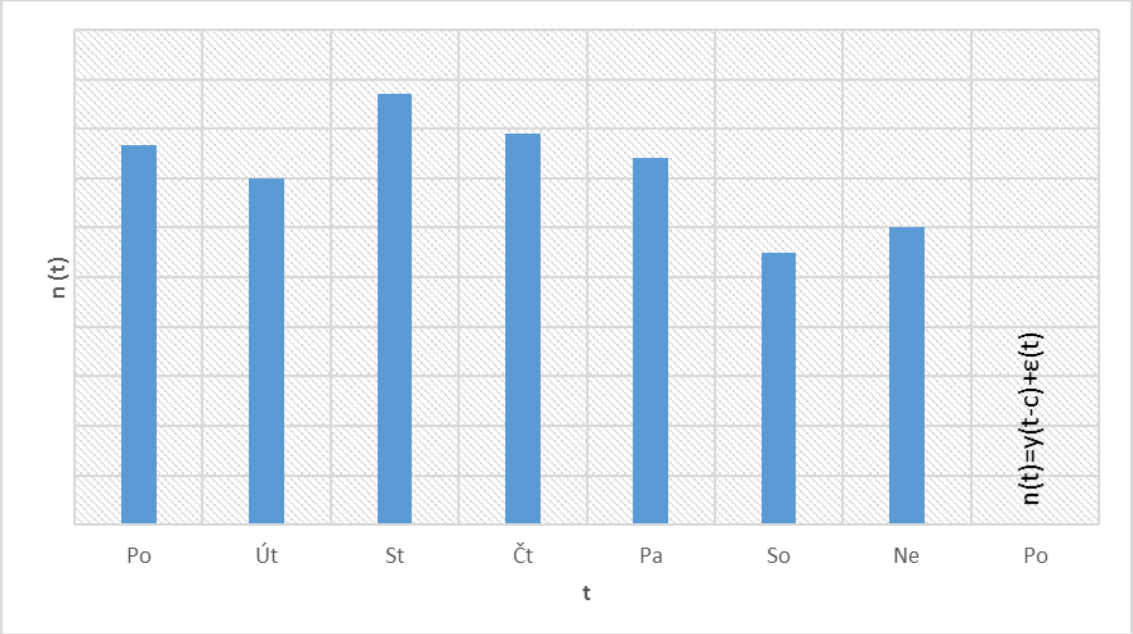
Model využívající vlnkovou transformaci vychází z předpokladu, že hustotu provozu v daném místě sítě lze v moment *t* vyjádřit jako:

$$y(t) = n(t) + a(t) \tag{3.7}$$

Kde *y(t)* značí celkový objem provozu, *n(t)* je množství dat na síti v době „normálního“ provozu a *a(t)* označuje objem dat útoku.

V praxi lze získat *y(t)* jednoduše pomocí prostého monitorování provozu na síti některým z volně dostupných nástrojů, ovšem hodnota objemu dat při „normálním“ provozu musí být stanovena některou z nepřímých metod.

Podle [20] lze celkový objem provozu na síti neobsahující útoky dělit do denních a týdenních cyklů.



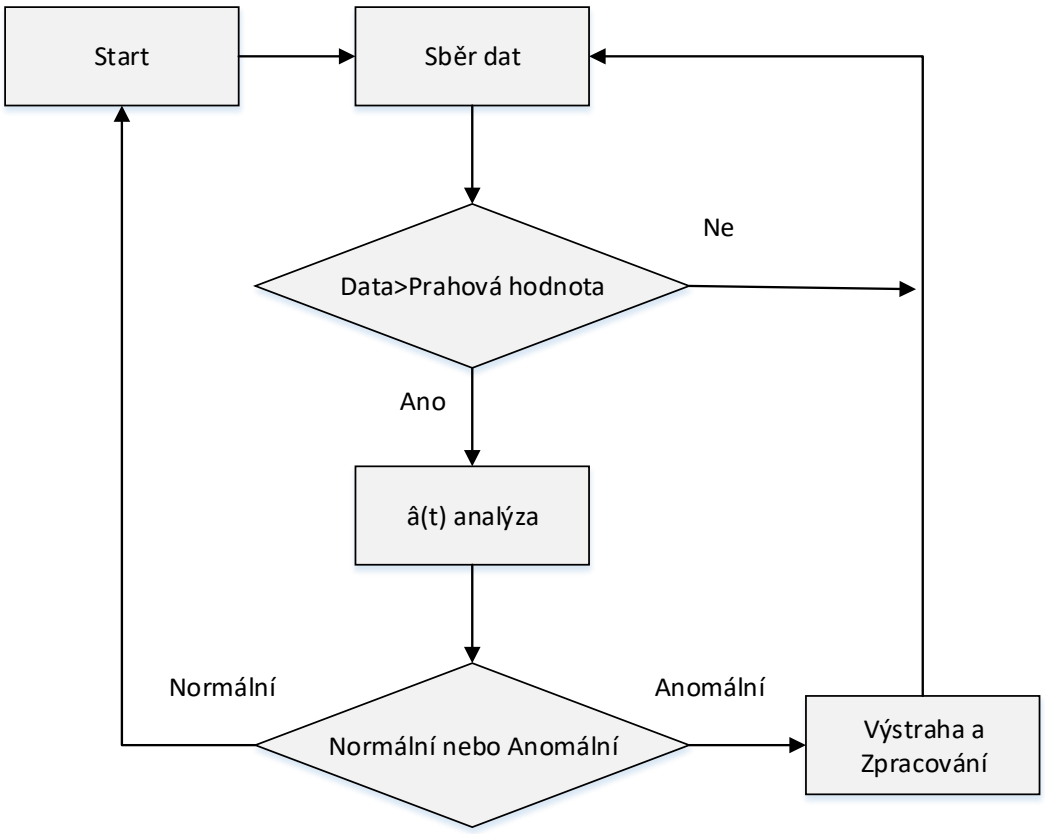
Obr. 3.4: Objem dat na síti v cyklech.

Na Obr. 3.4 je obecně znázorněn objem dat *n(t)* za každý den v týdnu. Hodnota očekávaného objemu dat pro následující cyklus *c*, se dá odvodit od předcházející hodnoty *t-c*, ke které je přičten šum *ε(t)* vyjadřující přirozenou odchylku. Ve zkratce to znamená, že provoz, který neobsahuje útok, je v rámci cyklu přibližně periodický.

Z této vlastnosti vyplývá rovnice pro množství dat útoku *a(t)*:

$$a(t) = y(t) - y(t - c) - \varepsilon(t) \tag{3.8}$$

Kde $\varepsilon(t)$ v rovnici pro množství dat útoku značí šum. Díky těmto vlastnostem se může hodnota *a(t)* vypočítat pomocí předchozí hodnoty $\hat{a}(t)$. Pokud není monitorovaný segment sítě vystaven útoku, můžeme považovat hodnotu *a(t)*= $\varepsilon(t)$. Pro zpřesnění detekce, je možné nahradit celkový objem provozu v přecházejícím cyklu za průměrnou hodnotu provozu neobsahující útok tak, že $\hat{a}(t)$ a $\hat{y}$(t) reprezentují průměrné statistiky provozu. Po splnění těchto podmínek lze analyzováním hodnoty $\hat{a}$(t) detekovat útok. Analyzování probíhá pomocí vlnkové transformace. Chod algoritmu pro detekci je na Obr. 3.5.



Obr. 3.5: Algoritmus pro detekci pomocí vlnkové analýzy.

Vlnková transformace

Bereme-li v úvahu nestacionární náhodný signál, je jeho rozložení energie vzhledem k frekvenci časově závislé, tudíž se jedná o funkci času a frekvence, která může být graficky znázorněna. Existuje několik metod, které slouží k vyšetření této závislosti. Právě mezi tyto metody patří i vlnková transformace. Ta nabízí dobré frekvenční rozlišení na nízkých frekvencích a stejně tak dobré časové rozlišení na vysokých frekvencích.

Existují dva typy transformací. Prvním z nich je tzv. spojitá vlnková transformace označovaná jako CWT, druhým je diskrétní vlnková transformace DWT. Pro odhalení útoků se používá diskrétní typ, jelikož signály obsažené na síti jsou frekvenčně omezené. Rovnice pro výpočet diskrétní vlnkové transformace je následující:

$$DMT_{m,n}(t) = \int_R f(t)\psi_{m,n}(t)dt \tag{3.8}$$

Hodnota ψ vyjadřuje vlnkovou funkci chovající se jako pásmová propust filtrující vstupní signál, který je závislý na měřítku, *m* značí měřítko a *n* posun. Více se této metodě detekce věnuje [9]

3.2 Detekce založená na strojovém učení

Strojové učení lze specifikovat jako schopnost programu nebo systému učit se a zlepšovat svůj výkon. Na rozdíl od statistického přístupu, který se zaměřuje na proces generující data, strojové učení se soustředí na zlepšování výkonu v závislosti na předchozích výsledcích.

Jako příklad může posloužit detekce založena na Bayesovských sítích. Bayesovské sítě jsou grafický model, který propojuje vztahy mezi různými zkoumanými proměnnými. Takto vzniklý model je mezi nimi schopen nalézt vzájemné závislosti, a dokonce předpovědět jejich budoucí vývoj.

Nevýhodou je nutnost označených trénovacích dat nebo možnost přeučení, která nastává v případě, pokud je model příliš přizpůsoben množině trénovacích dat a není již schopen generalizace. Další informace je možné dohledat v [10].

3.3 Detekce založená na algoritmech dolování dat

Detekce založená na algoritmech dolování dat je detekce založena na několika pokročilých technikách, jejichž podstatou je množství dat na vstupu a na jejichž základě detekuje příznaky a odchylky, které jsou jinak těžko rozpoznatelné.

3.3.1 FIRE

Jako příklad detekce může sloužit systém FIRE, který používá relativně jednoduchou formu dolování dat ke zpracování síťových informací a generování souboru pravidel, která mají za úkol detekovat útoky. FIRE nestanovuje standartní model vyjadřující stav sítě, ale spíše vytváří pravidla k detekci útoků za pomocí sběru specifických částí hlaviček paketů a jejich třídění.

Systém ke své funkci aktivně využívá tři druhy komponentů. Komplexní přehled systému je v [7].

NDC (Network Data Collector)

Plní funkci paketového snifferu, tzn. jeho primárním úkolem je zachytávání dat přicházejících na rozhraní.

NDP (Network Data Processor)

Hlavní část systému, která je schopna vykonávat procesy dolování informací. Třídí data do vybraných kategorií, porovnává je s předchozími a ukládá je na disk. Dále ze srovnávání aktuálních a minulých dat vytváří hodnotu vyjadřující míru odlišnosti a upravuje výstup do formy, která může být zpracována pomocí fuzzy logiky. Každý nový výskyt je posuzován z hlediska unikátnosti, variance a počtu.

FTA (Fuzzy Threat Analyzer)

Pomocí fuzzy logiky vyhodnocuje hrozbu a případně spouští upozornění na neobvyklé chování.

4 OPEN SOURCE IDPS NÁSTROJE

V současné době existuje řada více či méně známých open source IDPS, které kombinují několik přístupů. Většinou se dají rozdělit do dvou kategorií, na nástroje využívajících mechanismu detekce podle vzoru a na nástroje zaměřující se spíše na detekci podle anomálií.

Mezi nejznámější a nejrozšířenější patří Snort, který se dá považovat za lídra v oblasti monitorovacích nástrojů. Je založen zejména na mechanismu detekce podle vzoru.

Další nástroj, kterým se bude práce zabývat, je Suricata. Ta by se dala typově přirovnat ke Snortu, ale disponuje vyšší procesní výkonnost.

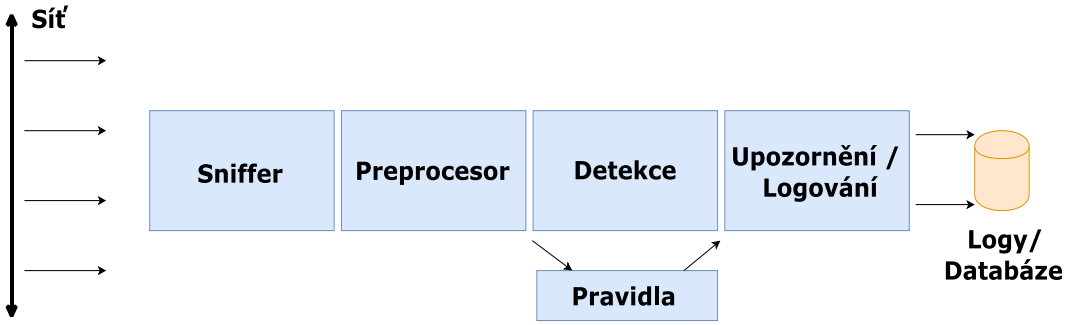
Mírně odlišným typem IDPS nástroje je Bro, který je více založen na detekci podle anomálií.

4.1 Snort

Snort je open source NIDS schopný za chodu analyzovat provoz a provádět logování paketů v IP sítích. Mezi jeho základní schopnosti patří analýza protokolů a vyhledávání škodlivého obsahu v datech paketů. Snort lze využít k detekci mnoha případů útoků a nepovoleného narušení sítě. V dnešní době patří k nejvyužívanějším open source řešením v dané oblasti.

Provoz Snortu je možný ve třech hlavních režimech. Prvním je tzv. „sniffer“ režim, ve kterém se chová podobně jako například všeobecně známý Wireshark, a totiž pouze monitoruje proud paketů a zobrazuje ho do konzole. Ve druhém režimu pracuje jako „packet logger“, takže zaznamenává zachycené pakety do logu na disk. Třetí a současně nejvíce konfigurovatelný a využívaný režim je tzv. NID (Network Intrusion Detection) režim, ve kterém dokáže provoz analyzovat, porovnávat s definovanými pravidly a na jejich základě spustit žádanou akci.

4.1.1 Architektura



Obr. 4.1: Architektura Snortu.

4.1.1.1 Sniffer

Sniffer je část Snortu přijímající provoz ze sítě. Jeho nejpodstatnější funkcí je interpretace příchozích dat, ale dalším nezanedbatelným prvkem je i schopnostukládat pakety k pozdějšímu zpracování nebo nahlédnutí.

4.1.1.2 Preprocesor

Preprocesor je důležitou součástí Snortu, která umožňuje poměrně jednoduše uživatelům přidat vlastní obsah v podobě modulárních plug-inů.

Jeho funkcí je porovnávat zachycený provoz s určenými plug-iny (např. RPC, HTTP, port scanner atd.), které mají definovanou podobu a chování žádaných paketů. V případě, že se datové jednotky shodují s požadavky určenými v plug-inu, jsou předány dále k detekci. Snort podporuje mnoho typů preprocesorů a jejich příslušných plug-inů, které umožňují velkou škálu využití. Zároveň lze tyto plug-iny modulárně povolovat a zakazovat, což nabízí velkou výhodu v optimalizaci samotného nástroje. Další podrobnosti lze nalézt v [14] a [16].

4.1.1.3 Detekce

Jakmile pakety projdou preprocesory, dostávají se do části detekce. Zde jsou data porovnávána s řetězci pravidel a následně je spouštěna požadovaná akce. Pravidla mají ve Snortu specifickou syntaxi, která může zahrnovat například typ protokolu, obsah, délku, hlavičku a jiné elementy. Zápis jednoduchého pravidla:

```
Alert tcp 192.168.1.102 21 -> 192.168.1.105 any (msg:"FTP pripojeni!");
```

4.1.1.4 Upozornění a logování

Pokud přijatá data odpovídají nastaveným pravidlům, Snort spouští upozornění. Ta mohou být poslána v podobě souboru logů skrze síť, UNIX soketů nebo mohou být například ukládána do SQL databází. Snort lze také propojit s externími nástroji, včetně plug-inů pro Perl, PHP nebo webových serverů, které umožňují zobrazení upozornění pomocí webových aplikací. V základním nastavení se ale ukládají do souboru s logy na disk. O upozornění a logování dále referuje [14].

4.1.2 Detekce anomálií ve Snortu

Snort je založen spíše na mechanismu detekce pomocí vzoru, ale díky jeho modulárním vlastnostem ho lze použít také k detekci anomálie. K tomuto účelu slouží například preprocesor zvaný SPADE.

Pro detekci anomálií Snort udržuje pravděpodobnostní tabulky obsahující počet výskytů různých typů paketů za určitý časový interval, přičemž vyšší váhu mají nedávno zachycené pakety. Pravděpodobnost výskytu P(X) je pro paket X zlogaritmována pomocí vztahu $\log_2 P(X)$. Vzniklá hodnota a(X) zvaná „Raw Packet Score“ je dále upravena do výsledné hodnoty A(X) nazývající se „Relative Packet Score“ dělením současného a(X) nejvyšším možným výskytem a(X), čímž vznikne hodnota v intervalu <0;1>, která je nejvhodnější pro porovnávání. Čím je A(X) nižší, tím značí typičtější chování.

Při nastavování preprocesoru se specifikuje mez A(X), při které se spouští akce upozornění. [13]

4.1.3 Grafická nadstavba

Jednou z doplňkových funkcí je grafická nadstavba, která dokáže přehledně sumarizovat naměřené hodnoty, jenž získá z logů. Jednou z předností balíku je možnost přístupu z vně analyzované sítě. Balík programů k tomu určených se nazývá ELK stack.

ELK stack

Skládá se ze tří programů:

- Elasticsearch
- Logstash
- Kibana

Základní funkcí Elasticsearch je skladování analyzovaných dat. Tyto data následně analyzuje Logstah a Kibana vytváří výsledné grafické rozhraní.

4.2 Suricata

Další ze zástupu open-source IDS nástrojů. Je považován za přímého konkurenta Snortu, dokonce je kompatibilní s velkou částí souborů pravidel ze Snortu. Mezi její výhody oproti předchozímu nástroji patří zejména vícevláknové analýzy a GPU zpracování, které umožňuje vyšší výkonost zachytávání paketů. Suricata zavádí tzv. proměnné relace, které umožňují ukládat určitá data přenosu do proměnné, jenž může spouštět podmínku pravidla. Co se týče architektury, je shodná s nástrojem Snort (Obr. 4.1).

Mezi nevýhody patří menší rozšíření nástroje, a tím pádem menší zásoba dokumentace a uživatelských zkušeností.

4.3 Bro

Bro je pasivní open-source síťový analyzátor provozu, který splňuje všechny požadavky na IDS software, takže je schopen monitorovat síť a na základě daných příznaků ukládat zaznamenaný provoz do logů různých formátů k pozdějšímu zpracování. Tyto logy neobsahují pouze obsáhlý přehled zaznamenaných spojení, ale obsahují i údaje z aplikační vrstvy OSI/ISO modelu (např. http spojení s URI, hlavičky klíčů, MIME typy a odpovědi serverů, DNS požadavky s odpověďmi, SSL certifikáty, klíčové části SMPT atd.).

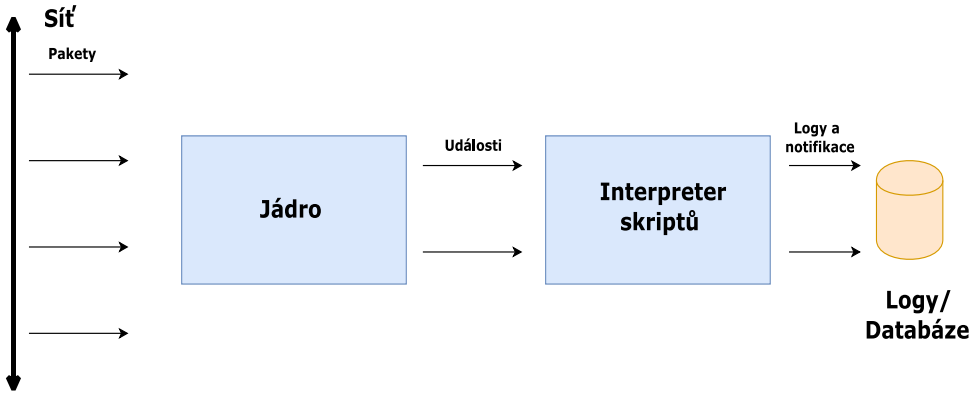
Mimo samotné ukládání do logů má Bro vlastní mechanismus sloužící k analýze a detekci. Mezi největší výhody patří jeho vlastní procedurální scriptovací jazyk, pomocí něhož je uživatel schopen plně přizpůsobit analýzu provozu jeho potřebám, a dokonce vytvořit zcela nový mechanismus detekce podezřelé aktivity. Dá se říci, že Bro samo o

sobě není pouze IDS, ale je to i skriptovací platforma, která byla vytvořena k tomu, aby pracovala se síťovým provozem, což umožňuje použít Bro ve zcela nových možnostech napsáním vlastního skriptu.

Bro je schopný pracovat ve dvou režimech. Buď jako tzv. standalone, což znamená, že operuje pouze na jednom zařízení, nebo jako tzv. „cluster mode“, ve kterém je schopný monitorovat a spravovat několik nezávislých zařízení.

Zatímco u předchozích open-source nástrojů se pracovalo s pravidly, se kterými se porovnával příchozí provoz, u Bro se vytvářejí skripty, které vykonávají potřebné akce, a i všechny předinstalované systémové akce jsou vykonávány pomocí skriptů. [17]

4.3.1 Architektura



Obr. 4.2: Architektura Bro-IDS.

4.3.1.1 Jádro

Základní funkcí jádra je redukovat příchozí proud paketů do sérií událostí srozumitelných pro Bro, které odrážejí aktivitu na síti. Je napsán v programovacím jazyce C++. Jako příklad funkce může posloužit klasický HTTP požadavek. Bro si zaznamená IP adresu, číslo portu, URI a verzi HTTP do příslušné události, ale další údaje o spojení již nezapisuje. [17]

4.3.1.2 Interpreter skriptů

Část, ve které se spouští mechanismus zpracování událostí zapsaný ve skriptovacím jazyce nad zjištěnými událostmi v jádru. Ten může posloužit jako spouštěč různé akce nebo jako nástroj zaznamenávající požadované vlastnosti a statistiky z příchozích událostí. [17]

4.3.2 Logy

Výstup analýzy Bro třídí do záznamů ve formátu log. Do těchto souborů ukládají data používané skripty. Existuje několik kategorií (Tab. 4.1).

Tab. 4.1: Přehled logovacích souborů Bro.

Název logu	Funkce
Capture_loss	Eviduje míru zahození paketů důvodem přehlcení vstupního rozhraní při monitorování provozu.
Conn	Záznam všech TCP/UDP/ICMP spojení.
Dns	Záznam DNS aktivit.
Http	Eviduje všechny HTTP požadavky a odpovědi.
Known_services	Služby běžící na hostitelském systému.
Loaded_scripts	Seznam načtených skriptů.
Packet_filter	Seznam aplikovaných paketových filtrů.
Reporter	Zprávy o upozornění a chybách.
Software	Používaný software na síti.
Stats	Statistiky paměti, událostí, paketů a zpoždění.
Weird	Nečekané události na síti.

4.4 Teoretické porovnání

Stručný přehled základních znaků každého ze zmiňovaných open-source nástrojů je shrnut do následující tabulky (Tab. 4.2).

Tab. 4.2: Porovnání open-source IDPS

Nástroj	Možnost GUI (analýza)	Primární detekce	Způsob detekce	Možnost využití více jader CPU
Snort	Ano	Signatury	Pravidla	Ne
Suricata	Ano	Signatury	Pravidla	Ano
Bro	Ano	Anomálie	Skripty	Ne

5 PRAKTICKÉ OVĚŘENÍ OPEN SOURCE IDPS NÁSTROJŮ

Primárním úkolem praktické části práce je porovnat a zhodnotit funkci třech open-source IDPS nástrojů (Snort, Suricata a Bro). Porovnávaly se z hlediska procesní rychlosti zpracování dat a detekce útoků. Testovací scénáře byly realizovány virtuálně ve virtualizačním nástroji VirtualBox v. 5.1.6.

Verze IDPS měřených nástrojů byly:

- Snort 2.9.8.3
- Suricata 3.2
- Bro 2.5

Hardwarová výbava hostujícího PC byla následující:

- CPU: Intel Core i5 2,5 GHz
- RAM: 4 Gb
- OS: Windows 10

Virtuální paměť použitá na straně generátoru provozu byla 2048 Mb, na serveru 1024 Mb.

5.1 Použité programy

K vypracování praktické části byla použita kromě samotných IDPS, řada dalších programů, např. distribuce OS nebo již zmiňovaný virtualizační nástroj.

5.1.1 VirtualBox v 5.1.6

VirtualBox je známý open-source nástroj sloužící k virtualizaci od společnosti Oracle. Má širokou škálu využití, může sloužit k virtualizaci desktopů, serverů, operačních systémů síťových prvků apod. na různých platformách. Skládá se z mnoha komponent. Může sloužit jako hypervisor pro hostující platformu, API pro ovládání hostovaných systémů a má vlastní grafické rozhraní usnadňující jejich správu. V práci je použit k virtualizaci testovaných zařízení. Více informací o nástroji lze nalézt například v [18].

5.1.2 Kali Linux 2016.2

Kali je linuxová distribuce založena na Debianu. Kali je určena pro pokročilé testování průniků a analyzování bezpečnostní sítě. Slouží ji k tomu řada nástrojů. V práci je instalována na virtualizovaném desktopu a je použita ke generaci provozu a útokům na server.

5.1.3 Ubuntu 16.04.1 server

Ubuntu je open source softwarová platforma založená na linuxovém jádře. Verze pro servery postrádá v základní instalaci grafické rozhraní, ale nabízí několik užitečných funkcí pro administrátory.

5.1.4 Apache 2

Apache 2 je open-source HTTP server s modulární architekturou podporující systémy Unix a Windows. Je velmi flexibilní, nabízí možnost vytvoření jednoho serveru podporujícího několik webových stránek jako virtuální hosty a může fungovat jako proxy. Podporuje SSL/TLS spojení, řadu způsobů autentizace a systém filtrů.

V praktické části práce byl Apache intstalován na serveru Ubuntu. V současné době je vyvíjen za přispění komunity dobrovolníků, další informace lze nalézt např. v [19].

5.1.5 Apache JMeter

Apache JMeter je open-source Java aplikace sloužící k zátěžovým testům a měření výkonu serveru. V práci slouží ke generování provozu, nicméně může být použit i pro další testování a měření. Podporované protokoly jsou např. HTTP, HTTPS, SOAP, FTP, LDAP, TCP, SMTP atp.

5.1.6 Ettercap 0.8.2

Ke generaci útoků DoS byl použit nástroj Ettercap. Je to obsáhlý soubor nástrojů sloužících k testování a penetraci sítě. Útok byl vytvářen jedním z dostupných modulů provádějící SYN flood na adresu serveru.

5.1.7 Nmap

Nmap je open-source nástroj pro mapování sítí dostupný téměř všem linuxovým distribucím. Je schopný uskutečnit řadu síťových monitorování a bezpečnostních testování. V práci se využívá ve třetím scénáři při skenování portů.

5.2 Doplňkové nástroje

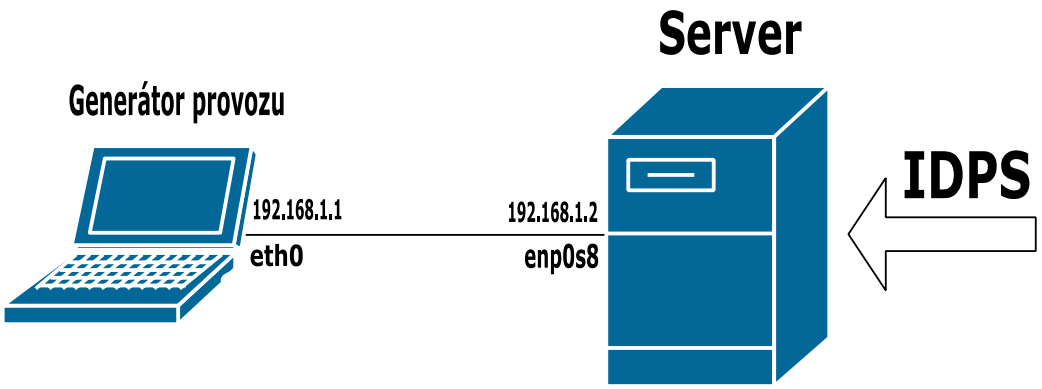
Každý z nástrojů vyžaduje ke své funkci několik dodatečných nástrojů a knihoven. Jedná se například o program sloužící k zachytávání paketů na síťovém rozhraní nebo různé překladače. Seznam základních instalovaných balíčků je v tabulce (Tab. 5.1).

Tab. 5.1: Seznam doplňkových nástrojů.

Snort	Suricata	Bro
libpcap	libpcre3	bison
libpcre3	libtool	libpcap
libdumbnet	libpcap	libgeoip
zlib1g	libnet1	libssl
liblzma	libyaml	python
openssl	zlib1g	zlib1g
libssl	libcap	libmagic
	libmagic	swig
	libjansson	libgoogle

5.3 Topologie zapojení

Testování probíhalo v jednoduchém zapojení realizovaném virtuálním spojením mezi distribucí Kali a HTTP serverem Ubuntu. Zařízení byla spojena přes vnitřní síť (Obr. 5.1).



Obr. 5.1: Zapojení praktického měření.

5.4 Metodika měření

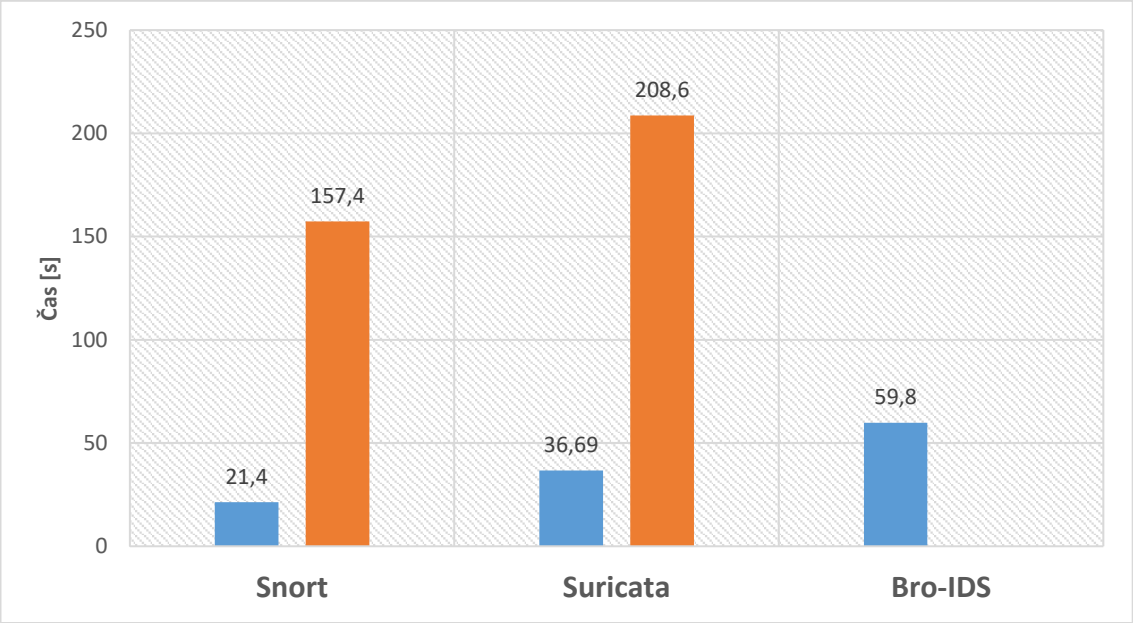
Pro ověření IDPS nástrojů byl generován provoz za pomocí nástroje JMeter a programů pro útoky ve třech různých scénářích. V prvním scénáři byly generovány pouhé HTTP požadavky a byla zkoumána zejména schopnost zpracování souboru s velkým množstvím dat. Nástroje Snort a Suricata byly testovány bez aplikovaných pravidel a následně s aplikovanými pravidly. U nástroje Bro sice existuje framework, který umožňuje detekci na základě signatur, ale vzhledem k podstatě nástroje bylo měření prováděno pouze s aplikací skriptů. Ve druhém scénáři byla ověřována detekce DoS útoku a ve třetím scénáři byla testována schopnost rozpoznání skenování portů.

Vytvořené provozy byly zachyceny do souborů typu pcap pomocí paketového analyzátoru tcpdump na síťovém rozhraní serveru. Následně byly tyto soubory podrobeny analýze IDPS nástrojů a vyhodnocen počet vzniklých událostí v záznamech. IDPS nebyly uživatelsky upravovány, použilo se jejich základní nastavení se současnými doporučenými pravidly.

5.5 První testovací scénář

V prvním testovacím scénáři byla ověřována procesní rychlost nástrojů. Provoz s HTTP požadavky byl vygenerován pomocí nástroje Jmeter. Požadavky generovaly čtyři vlákna (uživatelé) po dobu 10 minut. Výsledný soubor obsahoval 491 905 paketů a měl velikost 1,41 Gb.

Následně byla provedena detekce (Obr. 5.2) ze souboru, nejprve u každého nástroje bez aplikovaných pravidel (modrý průběh), a potom se základními pravidly (oranžový průběh). Vzhledem k odlišnému přístupu nástroje Bro, měření v tomto případě probíhalo pouze jednou s aplikovanými skripty.



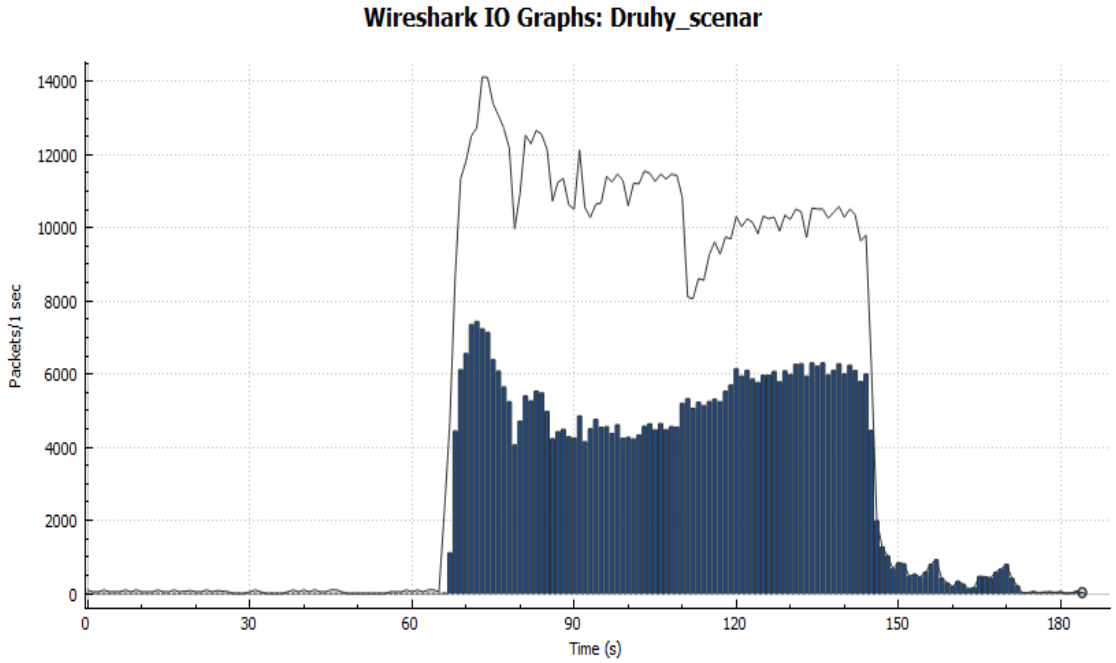
Obr. 5.2: Porovnání procesní rychlosti.

5.6 Druhý testovací scénář

Ve druhém scénáři byl spolu s HTTP požadavky generován DoS útok pomocí nástroje Ettercap (Obr. 5.3). Požadavky byly generovány z IP 192.168.1.1 a DoS útok z 172.16.23.2. Počet odeslaných paketů byl:

Tab. 5.2: Počet odeslaných paketů.

Zdroj	Počet odeslaných paketů
172.16.23.2	315164
192.168.1.1	2810



Obr. 5.3: Průběh DoS útoku ve druhém scénáři.

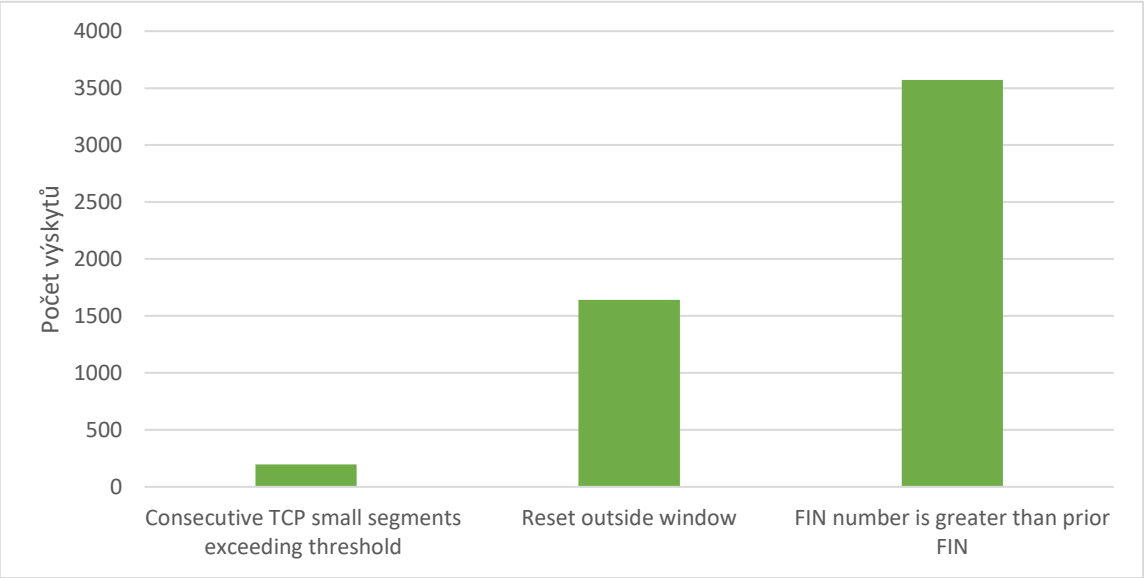
5.6.1 Mechanismy používané k detekci DDoS útoků

U nástrojů NIDS se nejčastěji využívá dvou mechanismů k detekci útoků DoS. Jako první se nejčastěji využívá přístup detekce podle anomálie, ve kterém se na základě daného klasifikačního algoritmu rozlišuje, zda je daný provoz v normě, či naopak normě neodpovídá.

Druhý přístup využívá konkrétních specifických znaků v DDoS paketech, jako může být například typ nastavených TCP hlaviček atp. v kombinaci s metodami zpracování signálu nebo použitím definovaného prahu pro dané datové jednotky.

5.6.2 Detekce nástroje Snort

Snort reagoval na přijatý soubor generací souboru „alert“, který obsahoval několik hlášení (Obr. 5.4). Počet jednotlivých hlášení je znázorněn v grafu.



Obr. 5.4: Obsah souboru s výstrahami.

První položka označuje počet překročení nastavené hranice pro malé segmenty TCP. Druhý případ upozorňuje na nedovolenou velikost okna. Největší zastoupení má hodnota upozornění na nesrovnalosti hlavičky FIN v TCP.

Po aplikaci jednoduchého pravidla Snort již DoS v plné míře detekoval.

```
alert tcp any any -> $HOME_NET 80 (flags: S; msg:"Mozny DoS utok!";flow:stateless; threshold: type both, track by_dst, count 10, seconds 5; sid:1001;rev:1;)
```

Příklad záznamu v souboru alert měl podobu:

```
[**] [1:10001:1] Mozny DoS utok! [**]
[Priority: 0] 12/11 -20:04:29.012326 172.16.23.2:44832 -> 192.168.1.2:80 TCP TTL:64 TOS 0x0 ID:32487 IpLen:20 DgmLen:40 ******S* Seq: 0xFFAADAB Ack:0x0 Win: 0x7FFF TcpLen: 20
```

5.6.3 Detekce nástroje Suricata

Nástroj Suricata v tomto případě vygeneroval soubor se statistickými informacemi, ale nezaznamenal žádné podezřelé chování.

Tab. 5.2.: Obsah stats.log.

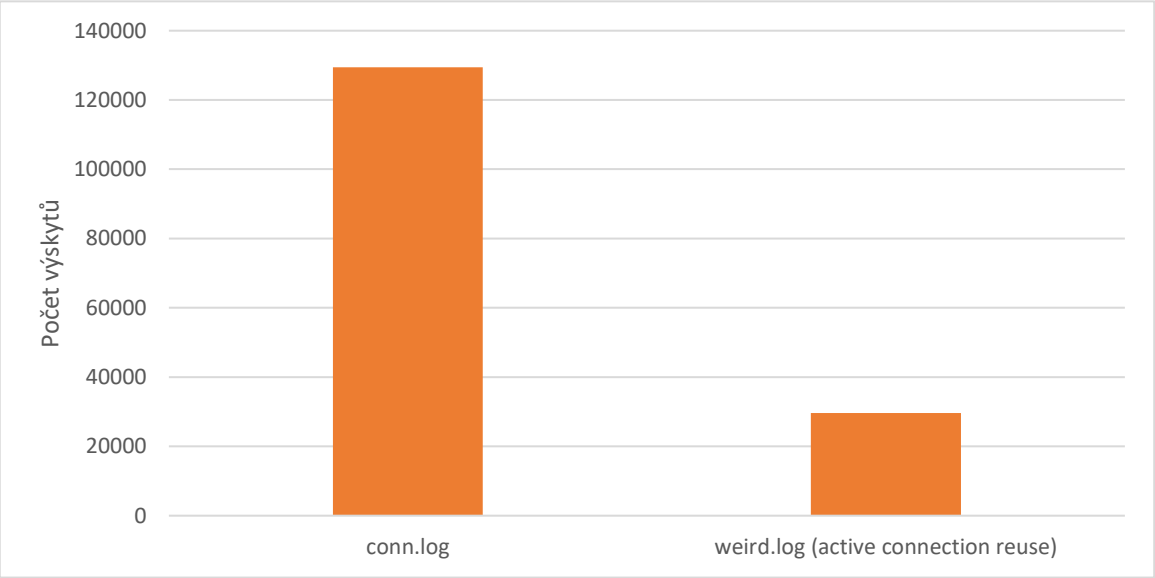
Modul	Počet [pakety]
decoder.pkts	867592
decoder.bytes	54831212
decoder.ipv4	867588
decoder.ethernet	867592
decoder.tcp	867588
decoder.avg_pkt_size	63
decoder.max_pkt_size	11650
tcp.sessions	66648
tcp.invalid_checksum	329672
tcp.syn	126943
tcp.synack	91291
tcp.rst	56728
detect.alert	1
flow_mgr.closed_pruned	1021
flow_mgr.new_pruned	72
flow.spare	10000
flow_mgr.flows_checked	1719
flow_mgr.flows_notimeout	1719
flow_mgr.rows_checked	65536
flow_mgr.rows_skipped	64562
flow_mgr.rows_maxlen	5
tcp.memuse	409600
tcp.reassembly_memuse	12320544
flow.memuse	25950112

Ve stejném případě jako u Snortu, po aplikaci pravidla ve stejném znění Suricata již na DoS útok reagovala a vygenerovala soubor fast.log se záznamy o útoku. Příklad záznamu byl:

```
12/11/2016-20:05:34.011385  [**]  [1:10001:1]  Mozny  DoS  utok!  [**]
[Classification:  (null)]  [Priority:  3]  {TCP}  172.16.23.2:35307  ->
192.168.1.2:80
```

5.6.4 Detekce nástroje Bro

Bro generovalo řadu logů reagujících na DoS útok. Největší zastoupení měla položka „active connection reuse“ v souboru weird.log, jenž slouží k záznamu podezřelých aktivit na síti. Tato položka značí opětované navazování spojení. Log zaznamenávající spojení se serverem obsahoval 129 463 položek.



Obr. 5.5: Počet výskytů v logovacích souborech Bro.

5.7 Třetí testovací scénář

Ve třetím případě byla testována schopnost nástrojů zachytit skenování portů pomocí Nmap. Topologie zůstala stejná jako v předchozích případech.

Skenování portů se používá tehdy, když chce útočník zjistit, jaké služby jsou na serveru k dispozici. Vytváří se posíláním paketů na definovaný rozsah portů a naslouchání odpovědi.

V tomto scénáři bylo skenování generováno nástrojem Nmap a zachyceno pomocí Tcpdump na síťovém rozhraní serveru. Následný soubor typu *.pcap byl podroben offline detekcí a hodnoceny výsledky. Vygenerováno bylo celkově 2011 paketů.

5.7.1 Detekce nástroje Snort

Snort při analýze vygeneroval dva soubory. Jeden byl záznam detekce a druhý byl soubor s upozorněními, který poukazoval na podezřelé skenování. V souboru se nacházelo šest záznamů o podezřelém chování. Záznam jednoho případu detekce v souboru s upozorněním byl:

```
[**] [1:2002910:5] ET SCAN Potential VNC Scan 5800-5820 [**]
[Classification: Attempted Information Leak] [Priority: 2]
12/9-11:29:14.584494 192.168.1.1:38463 -> 192.168.1.2:5802
TCP TTL:51 TOS:0x0 ID:30369 IpLen:20 DgmLen:44
******S* Seq: 0x13C4F084  Ack: 0x0  Win: 0x400  TcpLen: 24
TCP Options (1) => MSS: 1460
```

5.7.2 Detekce nástroje Suricata

Stejně jako Snort, i Suricata vygenerovala kromě souboru se záznamem detekce soubor fast.log obsahující šest záznamů. Ekvivalentní záznam k předchozí detekci pomocí nástroje Snort byl:

```
12/9/2016-11:29:14.584494  [**] [1:2002910:6] ET SCAN Potential VNC Scan
5800-5820 [**] [Classification: Attempted Information Leak] [Priority:
2] {TCP} 192.168.1.1:38463 -> 192.168.1.2:5802
```

5.7.3 Detekce nástroje Bro

Bro v tomto případě kromě logů zaznamenávajících provoz, vygenerovalo soubor notice.log, který obsahuje zprávu o proběhlém skenování:

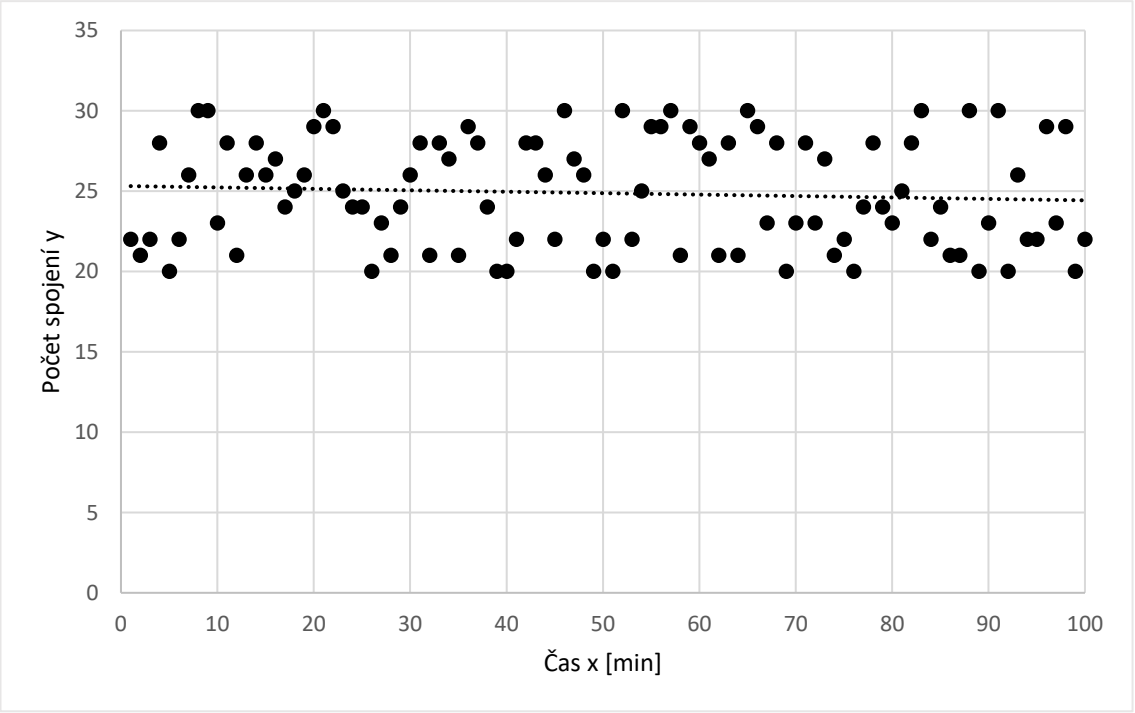
```
Scan::Port_Scan   192.168.1.1 scanned at least 15 unique ports of host
192.168.1.2 in 0m0s     remote      192.168.1.1 192.168.1.2 -
broNotice::ACTION_LOG   3600.000000 F
```

6 LINEÁRNÍ REGRESNÍ ANALÝZA

Další část práce se věnuje implementaci statistické metody v nástroji Bro a následnému ověření její funkčnosti. Pro implementaci byl vybrán nástroj Bro zejména z důvodu jeho událostmi řízenému skriptovacímu jazyku, který poskytuje vhodnou platformu pro implementaci.

Pro odhalení útoku typu DoS je implementována metoda využívající detekci pomocí lineární regresní analýzy, a to díky schopnosti predikce a vhodné podobě k implementaci do nástroje.

Lineární regresní analýza je statistická metoda sloužící k vyšetřování a modelování vztahu mezi proměnnými využívající této závislosti k predikci následující proměnné. V dnešní době je považována za jednu z nejrozšířenějších statistických technik. Analýza nalézá uplatnění v široké škále oborů od bankovnictví až po zdravotnictví. Práce aplikuje tuto metodu zejména z důvodu její schopnosti predikovat následující stav, což je vhodné i k odhalení případného útoku. Obecné modelové znázornění provozu neobsahující DoS útok, je zobrazeno na Obr. 6.1.



Obr. 6.1: Modelové zobrazení lineární regrese.

Obrázek znázorňuje počet aktuálních spojení v čase závislých na konkrétním případě nasazení sondy. Je patrné, že se hodnoty v rámci kratšího časového intervalu pohybují kolem přímky s určitou odchylkou ε (lze považovat za statistickou odchylku). Základním předpokladem lineární regrese je nalezení takové přímky, pro niž je součet druhých mocnin odchylek co nejmenší. Pokud y reprezentuje počet spojení a x zastupuje čas, rovnice přímky, kolem které se hodnoty pohybují, má tvar:

$$y = \beta_0 + \beta_1 x \tag{6.1}$$

β_0 a β_1 jsou koeficienty zvané koeficienty regrese. Koeficient β_1 ovlivňuje sklon přímky. Indikuje změnu v průměru pravděpodobnostní distribuce y za každou jednotku x. Parametr β_0 vyjadřuje průsečík přímky s osou Y. Pokud bychom tedy chtěli vyjádřit pozici každého bodu rovnicí, stačí k rovnici této přímky přičíst odchylku ε. Výsledná podoba rovnice tedy je:

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{6.2}$$

Tato rovnice se obecně nazývá model lineární regrese. Proměnná x je pojmenována jako nezávislá a proměnná y jako závislá. Vzhledem k tomu, že přechozí vztah obsahuje pouze jednu nezávislou proměnnou, jedná se o jednoduchý lineární model regrese.

6.1 Odhad koeficientů regrese

Jelikož nejsou koeficienty regrese ovlivňující podobu přímky předem známé, vzniká potřeba jejich odhadu z relevantních dat. Odhad vychází z předchozího pozorování hodnoty x a její odezvy y. Ve výsledku tedy existuje pro každé pozorování x odpovídající hodnota y. Jako jedna z metod může být pro odhad použita metoda nejmenších čtverců.

Pokud máme k dispozici data v podobě páru x a y (x_n,y_n), odhad koeficientů β_0 a $\beta_1$spočívá v co možná nejmenším součtu čtverců rozdílů mezi pozorováním y_n a přímkou. Tato metoda vyšetřuje odchylku y od její očekávané hodnoty:

$$Q_n = y_n - (\beta_0 + \beta_1 x) \tag{6.3}$$

V případě *n* pozorování metoda nejmenších čtverců počítá s tzv. kritériem *Q* vyjadřující součet čtverců všech pozorování:

$$Q = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 \tag{6.4}$$

Odhadnuté koeficienty b_1 a b_0, které splňují podmínku nejmenšího kritéria *Q* lze nalézt pomocí dvou způsobů. Buď numerickým, nebo analytickým přístupem. Podstata numerické procedury pro hledání systematicky ohodnocuje kritérium *Q* pro různé odhady b_0 a b_1, až nalezne takové koeficienty, které mají *Q* nejnižší.

Analytický přístup spočívá v nalezení právě takových odhadů, pro které je *Q* nejnižší. Tento model se hodí v případech, kdy není regresní model příliš komplexní. Rovnice pro nalezení odhadů analytickým přístupem jsou:

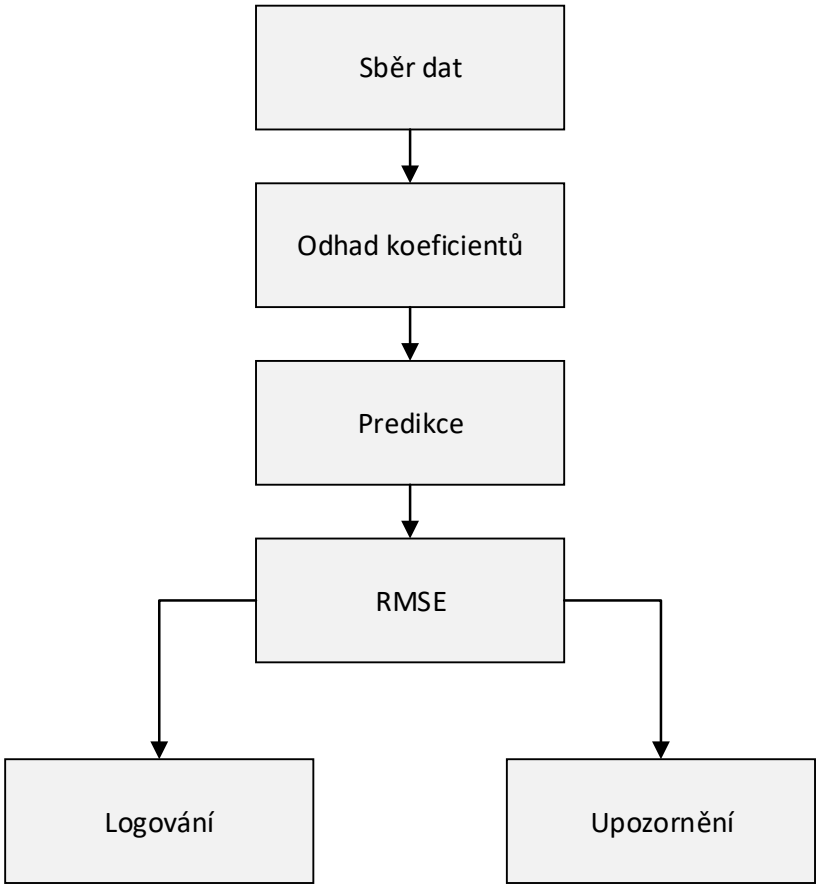
$$b_1 = \frac{\sum(x_i - \bar{x}) - (y_i - \bar{y})}{\sum(x_i - \bar{x})^2} \tag{6.5}$$

$$b_1 = \frac{1}{n}(\sum y_i - b_1 \sum x_i) = \bar{y} - b_1 \bar{x} \tag{6.6}$$

Další informace o lineární regresi lze nalézt v [21] a [22].

7 NÁVRH A IMPLEMENTACE LINEÁRNÍ REGRESE V BRO

Nástroj Bro obsahuje akcemi řízený skriptovací jazyk poskytující velmi široký prostor k úpravě jeho funkcionality. Veškerý výstup z nástroje je řízen některým z předem aplikovaných skriptů, který efektivně upozorňuje samotný nástroj na výskyt definované události a vyžaduje od něj další informace o spojení tak, aby byl schopen vykonat příslušnou činnost.

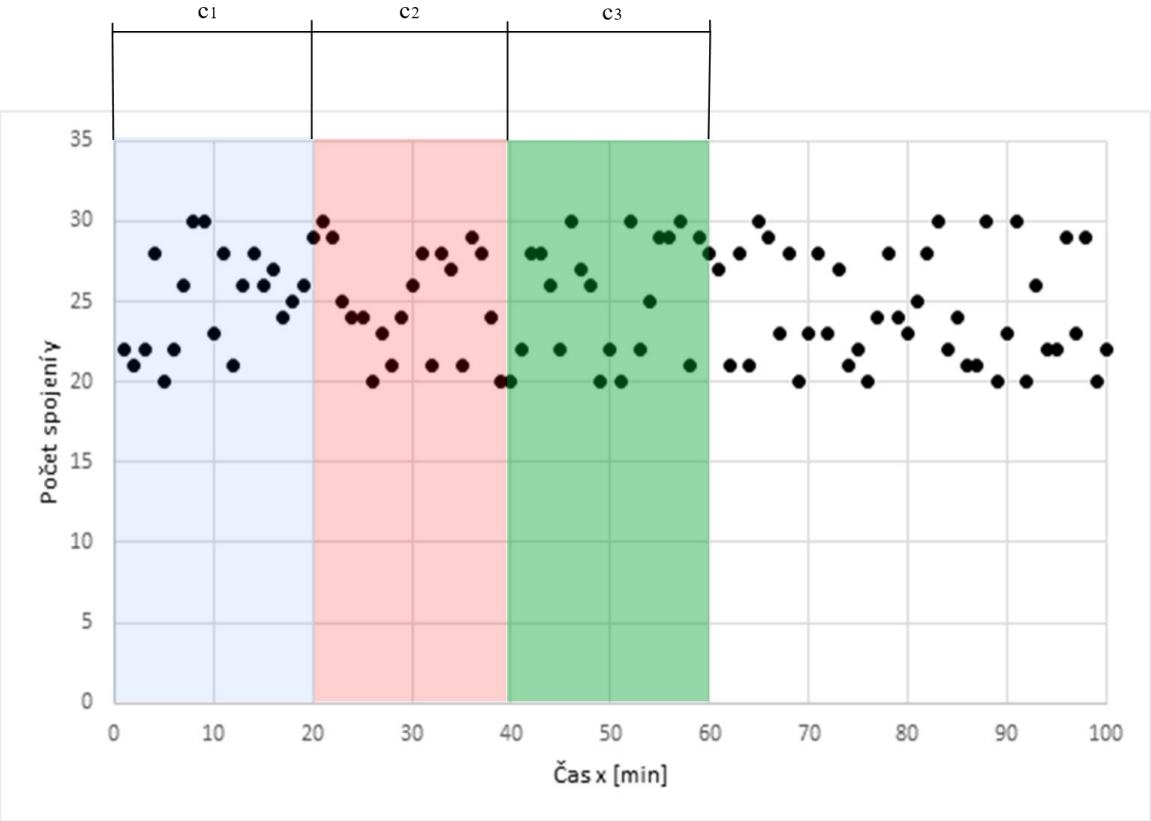


Obr. 7.1: Návrh implementace modulu.

7.1 Sběr dat

Pro samotný výpočet lineární regrese je základním předpokladem sběr dat z odposlouchávaného rozhraní. Modul pracuje s některou z událostí, které dokáže jádro Bro generovat. V daném časovém intervalu c ($c_i = \sum y_{ci}$) se zaznamenává počet výskytů

události a následně se ukládá do vhodné datové struktury *s* (s_i=[c_1;c_2;c_3;c_n…]) pro další zpracování. Grafické znázornění kolekce dat je na Obr. 7.2.



Obr. 7.2: Sběr dat v intervalech *c*.

Potřebná data pro výpočet lineární regrese se zaznamenávají v podobě numerického typu integer do datové struktury. Bro umožňuje ukládat data do čtyř základních datových struktur. Stručný přehled je uveden v tabulce (Tab. 7.1).

Tab. 7.1: Přehled datových struktur.

Typ struktury	Popis
Record	Soubor hodnot různých typů, nejsou asociovány k žádné hodnotě.
Set	Kolekce indexovaných hodnot - nepřiřazuje se k dalším hodnotám.
Table	Přiřazuje jednu hodnotu ke druhé a indexuje.
Vector	Jako Table, ale je vždy indexován typem "count" a indexuje od 0.

Při tvorbě modulu se data potřebná k výpočtu lineární regrese ukládají do datové struktury typu Vector zastupující celou datovou sadu. Tato struktura je podobná struktuře Table, ale vždy indexuje od nuly a pro indexování využívá numerického typu count (64-bit unsigned integer).

Periodizace sběru dat je zajištěna vytvořenou událostí spojeni(), která se pomocí příkazu schedule, definujícím dobu čekání, spouští každých *x* vteřin a ukládá do Vectoru globální proměnnou typu integer zaznámenávající počet navázaných spojení. Tato událost se spouští pomocí nadřazené události bro_init(), zahajující svoji činnost po inicializaci Bro.

7.2 Odhad koeficientů a predikce

Dalším krokem k úspěšné detekci je odhad koeficientů jednoduché lineární regrese. V modulu jsou vypočítávány koeficienty b_0 a b_1 pomocí analytického přístupu podle rovnic (6.5) a (6.6). Kvůli optimalizaci programu jsou implementovány funkce vypočítávající průměr, rozptyl a kovarianci, za jejichž pomocí jsou koeficienty získány. Ukázka funkcí pro odhad se nachází níže.

```
function b1(yt: vector of double):double{
local conb1: double =0;
conb1=kovariance(yt)/rozptyl(xosa);
return conb1;
}

function b0(yt: vector of double):double{
local conb0: double =0;
conb0=prumer(yt)-b1(yt)*prumer(xosa);
return conb0;
}
```

Následná predikce hodnot je získána z modelu lineární regrese (6.1). Pro výpočet jsou nutné dvě datové sady (trénovací a testovací), ze kterých se stanovují hodnoty predikce *y*. Modul pracuje ve výsledku s třemi strukturami dat, jichž využívá pro výpočet lineární regrese a chyby RMSE (střední kvadratická odchylka). Ukázka algoritmu pro výpočet lineární regrese:

```
   function linearni_regrese(trenovaci: vector of double,
testovaci: vector of double):vector of double{
   local b0koef: double = b0(trenovaci);
   local b1koef: double = b1(trenovaci);
   for(i in predikce)
   predikce[i]=b0koef+b1koef*testovaci[i];
   local_prediction=predikce;
   return local_prediction;
   }
```

7.3 RMSE

Pro zhodnocení odchylky aktuálních dat od predikovaných využívá modul pro výpočet středního kvadratického průměru rozdílu mezi predikovanými a aktuálními hodnotami tzv. střední kvadratické odchylky (dále jen RMSE). RMSE je používána jako běžný model k hodnocení výkonosti statistického modelu. Její výpočet vychází z předpokladu, že je znám počet *n* chyb *e* modelu.

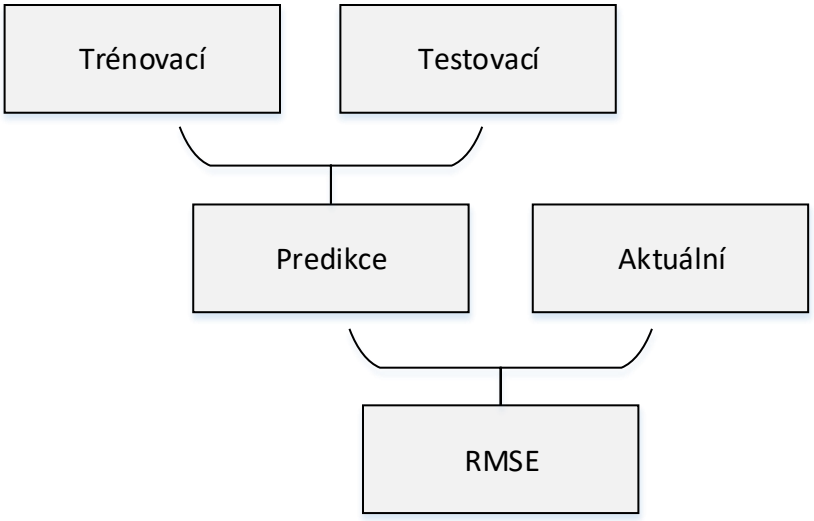
$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{7.1}$$

Kde $e = y_i - \hat{y}_i$ přičemž y_i značí aktuální a $\hat{y}_i$ predikovaná data. Více o RMSE se lze dozvědět z [23].

Výpočet byl v modulu proveden pomocí funkce chybarmse(). Algoritmus s globální proměnnou „cyklus“ značící počet intervalů má podobu:

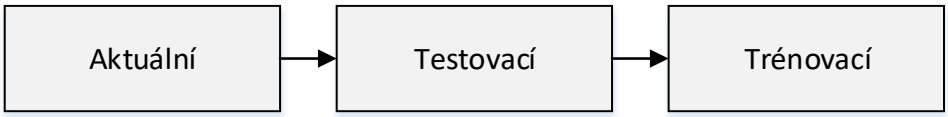
```
function chybarmse(aktualni: vector of double):double{
local chyba: double=0;
local prumer_chyba: double=0;
chyba_predikce=local_prediction-aktualni;
for(i in chyba_predikce)
     chyba=(chyba_predikce[i]*chyba_predikce[i])+chyba;
prumer_chyba=sqrt(chyba/cyklus);
return prumer_chyba;
}
```

Blokové schéma zahrnující potřebné datové sady k získání RMSE je:



Obr. 7.3: Potřebné datové sady pro výpočet RMSE.

Po stanovení chyby jsou datové sady aktualizovány, vzhledem k dalšímu cyklu. Při prvním cyklu programu je Trénovací sada definována a Testovací sada se „učí“ ze samotného provozu. Cyklus vývoje datových sad je vidět na Obr. 7.4.



Obr. 7.4: Cyklus datových sad.

7.4 Generování logů

Nedílnou součástí monitoringu v Bro je generace záznamů tzv. logů, které obsahují informace o uskutečněném spojení. O typu zaznamenaných dat rozhoduje samotný modul, jenž zároveň definuje událost spouštějící samotné logování. Pro generaci souborů se záznamy o probíhající relaci se v Bro používá tzv. „Logging Framework“.

Generaci lze rozdělit na čtyři části. V první části je nutné definovat globální ID pro identifikaci jednotlivých záznamů, dále je nutné pomocí typu record reprezentujícího soubor různých typů identifikovat všechna pole, jež budou zaznamenávána. Následuje vytvoření logovacího streamu při inicializaci (nejčastěji při události bro_init()), pomocí funkce Log::create_stream obsahující ID, samotný obsah a cestu, kam se uloží soubor obsahující logy. V poslední části se zavoláním funkce Log::write zapisuje hodnota do souboru. V modulu jsou vytvořeny dva logovací proudy. Jeden zaznamenává všechna spojení, která aktivují událost new_connection, do souboru dos.log. Druhý proud zapisuje vypočtené hodnoty RMSE včetně proměnné „pocet“ do souboru rmse.log.

7.5 Akce upozornění

Upozornění na podezřelou aktivitu je v algoritmu řešeno překročením definovaného prahu RMSE (globální konstanta prahrmse). Hodnota tohoto prahu je individuální vzhledem k charakteru sítě, v níž je modul nasazen. Detekce probíhá na základě porovnání právě vypočtené hodnoty RMSE s tímto prahem v podmínce typu if ověřované

v každém cyklu výpočtu chyby. Akce následující po splnění podmínky může nabývat několika podob. Podle potřeb je možné logovat upozornění do souboru, odeslat upozorňující email atp. Kvůli účelům testování byla vyžadována okamžitá reakce, proto byla do modulu implementována akce print, která vypíše konkrétní hlášku do konzole.

```
if(poradi==cyklus+1 && start!=0){
  local chybar: double = chybarmse(actual);
  linearni_regrese(tren,test);
  local recb: Dos::Doslog=[$rm=chybar];
  Log::write(Dos::LOGDOS,recb);    ##Zapis rmse do logu
  if(chybar>prahrmse){
       print "Mozny DOS!!!";
  }
}
```

Dalšímu testování reakce RMSE na DDoS útoky se bude věnovat nadcházející část práce.

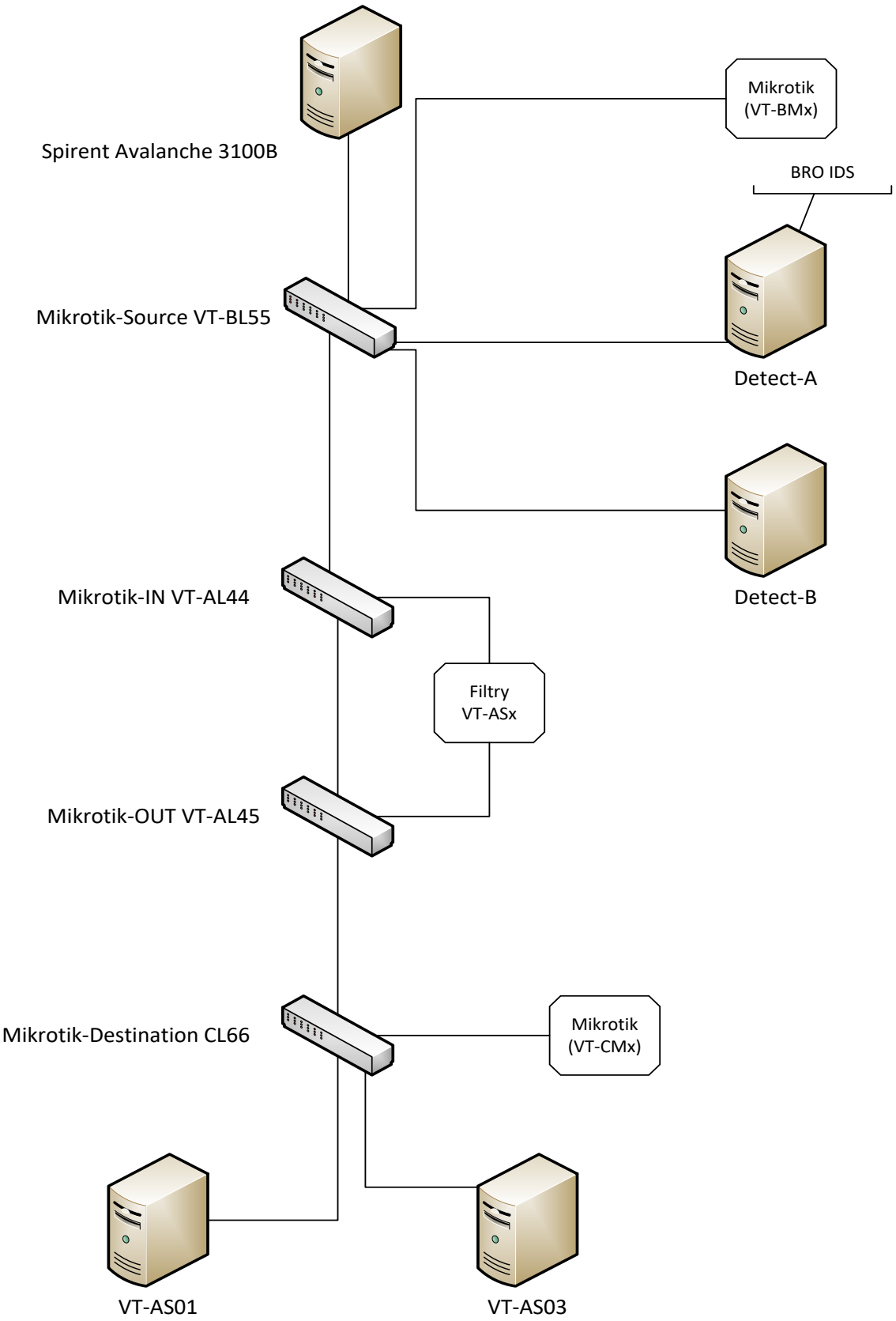
8 TESTOVÁNÍ

Poslední část práce se zabývala samotným testováním a optimalizaci implementovaného modulu využívajícího lineární regresní analýzy k detekci útoků DDoS. K testování bylo využito prostředků experimentálního síťového pracoviště FEKT VUT v Brně.

8.1 Experimentální pracoviště

Pracoviště se skládá ze soustavy serverů, filtrů a přepínačů Mikrotik (Obr. *8.1*.). Generace útoků je realizována pomocí síťového testeru Spirent Avalanche 3100B. Ten generuje provoz na servery VT-AS přes přepínač Mikrotik-Source VT-BL55, jenž zrcadlí pakety na detekční servery Detect-A (VT-AS30) a Detect-B (VT-AS29). Na detekčním serveru Detect-A je nainstalován Bro verze 2.5 s připraveným modulem používající lineární regresní analýzu. Provoz je zároveň zrcadlen i na soustavu přepínačů Mikrotik VT-BMx.

V další části jsou datové jednotky posílány skrze soustavu filtrů VT-ASx určenou pro různorodou filtraci k cíli, na servery VT-AS01 a VT-AS03, s nainstalovaným webovým serverem Avalanche. Před samotným cílem, mohou do provozu generovat šum přepínače Mikrotik VT-CMx.



Obr. 8.1: Blokové schéma zapojení experimentálního pracoviště.

8.2 Průběh testování

Útok typu HTTP flood generoval server Avalanche ve třech variantách. První varianta obsahovala klasický náběh s trváním DDoS útoku po dobu 10 minut. Ve druhém případě byl generován průběh se schodovitým náběhem a ve třetím případě útok pilovitého průběhu s cyklem sběru dat v délce 10 sekund a 2 sekund. Datové sady typu Vector obsahovaly čtyři intervaly a konstanta prahrmse byla nastavena na hodnotu 50 000 z důvodu předpokládaných rychlých změn objemů přijatých dat v testovacích scénářích, a tudíž nutné rychlé odezvy na tuto skutečnost. Testovala se reakce nástroje v podobě hodnot výpočtu RMSE, počtu aktivovaných událostí a zátěže CPU.

Tyto tři scénáře útoků byly do testovací fáze vybrány z důvodu jejich různorodé změny objemu vysílaných dat, jež umožňovala vyšetřit průběh chování výpočtu RMSE, a tak i schopnosti detekce.

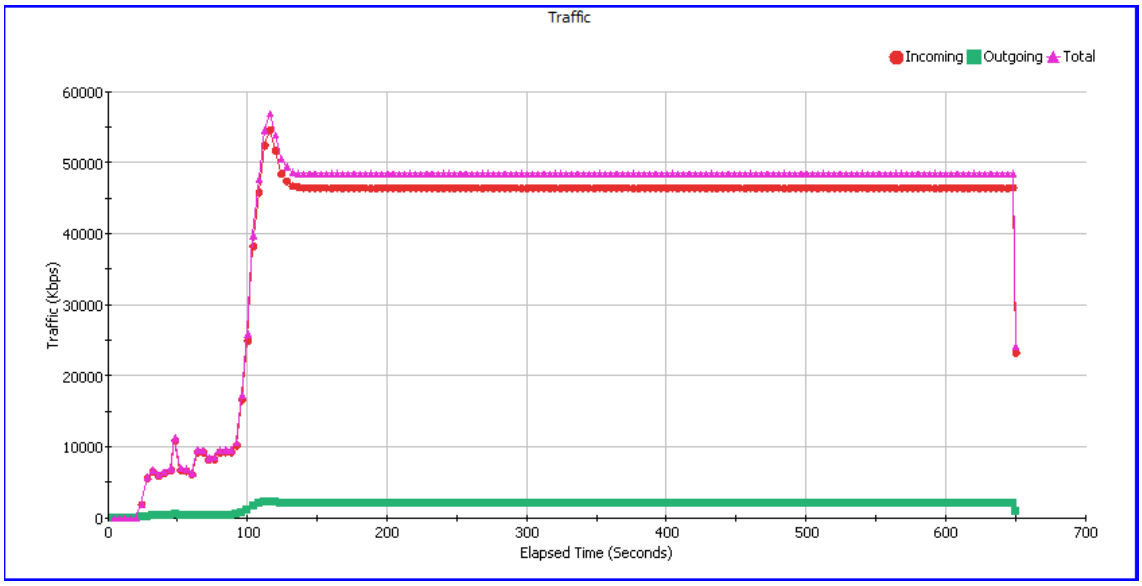
Jako aktivátor inkrementace plnila úlohu událost new_connection, která je obecně generována pro každé nové spojení. Tato událost se spouští s každým novým paketem předem neznámého spojení zahrnující TCP, UDP a ICMP toky. Více o typech událostí, které jsou v Bro implementovány, je možné zjistit v [17].

8.3 Útok s klasickým náběhem

V první fázi testování generoval server DDoS útok s klasickým náběhem (Obr. *8.2*). Podobu provozu je Avalanche schopen zaznamenat do souboru ve formátu csv. Hodnoty nastavených konstant byly:

- Čas cyklu: 10 vteřin.
- Počet cyklů: 4.

Jedna celá sada dat se tak skládala ze 40 vteřin záznamů počtu generovaných spojení. Útok dosahoval rychlosti provozu téměř 50 000 kb/s.

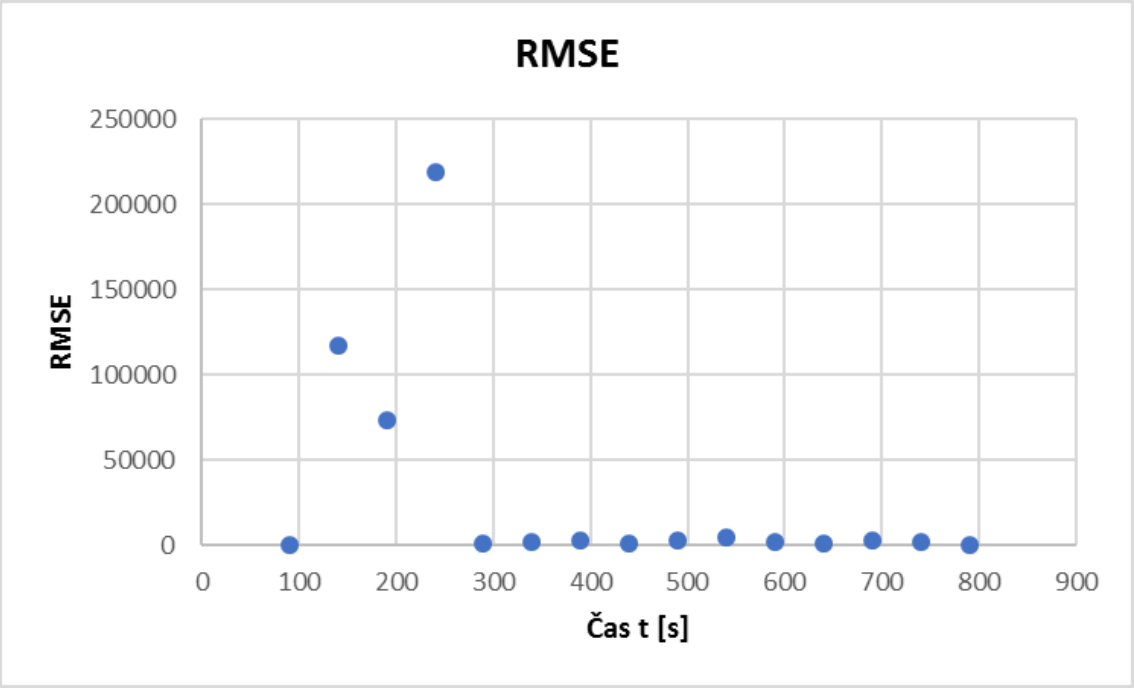


Obr. 8.2: Generace útoku s klasickým náběhem zaznamenaná na serveru Avalanche.

Nejdříve byl hodnocen počet aktivací události new_connection. Z Obr. 8.3 je patrné, že průběh aktivace události kopíruje samotný průběh útoku. V části s nejvyšší zátěží se tak událost každých deset vteřin generovala přibližně 5000 krát.

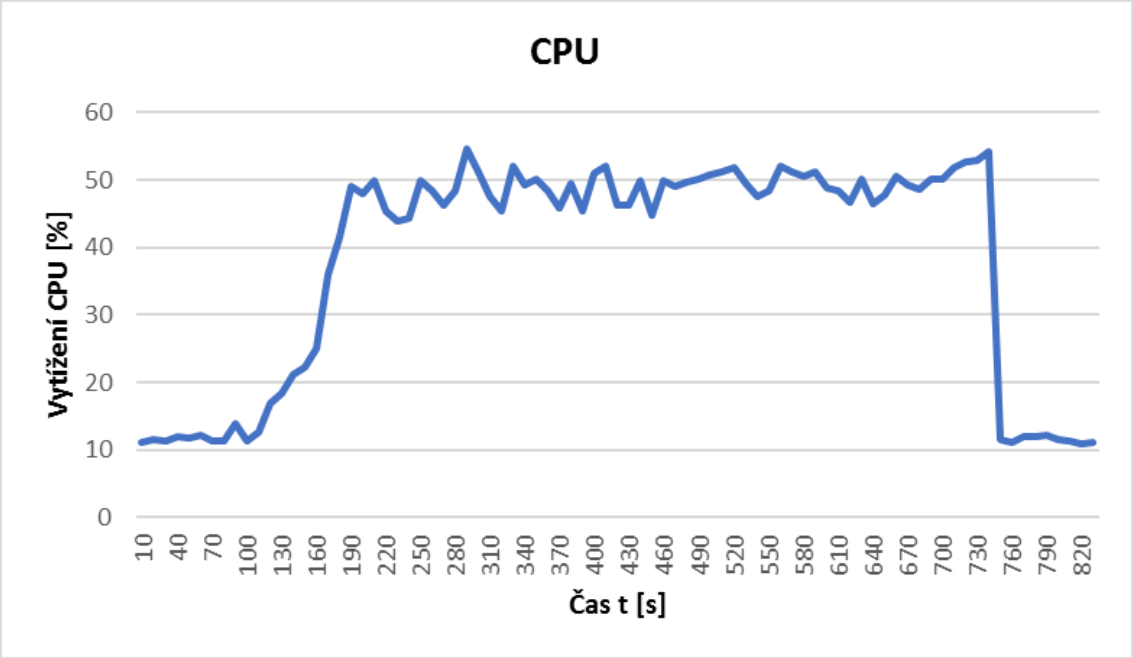
Obr. 8.3: Počet aktivací události – klasický náběh.

Dále byla pomocí logu zaznamenávána hodnota RMSE v čase. Její výsledky jsou zřejmé na Obr. 8.4. RMSE byla vypočítávána každých 40 vteřin. Podle průběhu je patrná reakce na změnu charakteru obdržených dat. Nejvyšší hodnotu má chyba v místě s největší změnou objemu provozu. To je zapříčiněno chybnou predikcí následujícího stavu pomocí odhadu koeficientů lineární regrese. Po ustálení změny je patrný pokles RMSE.



Obr. 8.4: Změna hodnoty RMSE vzhledem k průběhů útoku s klasickým náběhem.

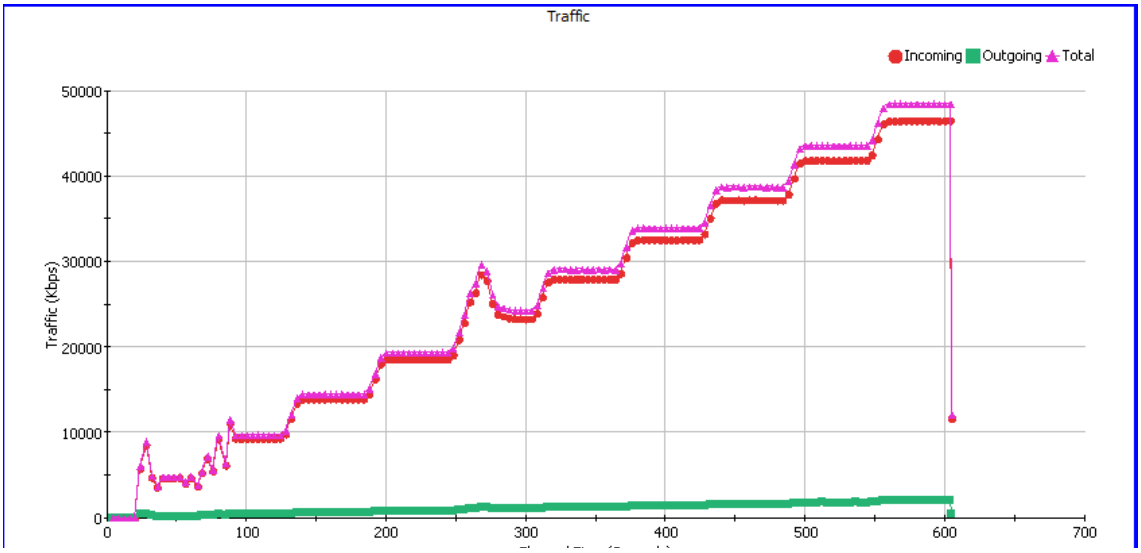
V poslední fázi měření se vyšetřovalo zatížení CPU při útoku (Obr. 8.5). V tomto bodě bylo zjištěno zatížení, které se dá považovat za velmi vysoké vzhledem k množství dat, jež útok obsahoval. Zatížení bylo v čase poměrně proměnlivé, což bylo pravděpodobně způsobeno výpočty prováděnými v modulu.



Obr. 8.5: Změna zatížení CPU vzhledem k průběhu útoku s klasickým náběhem.

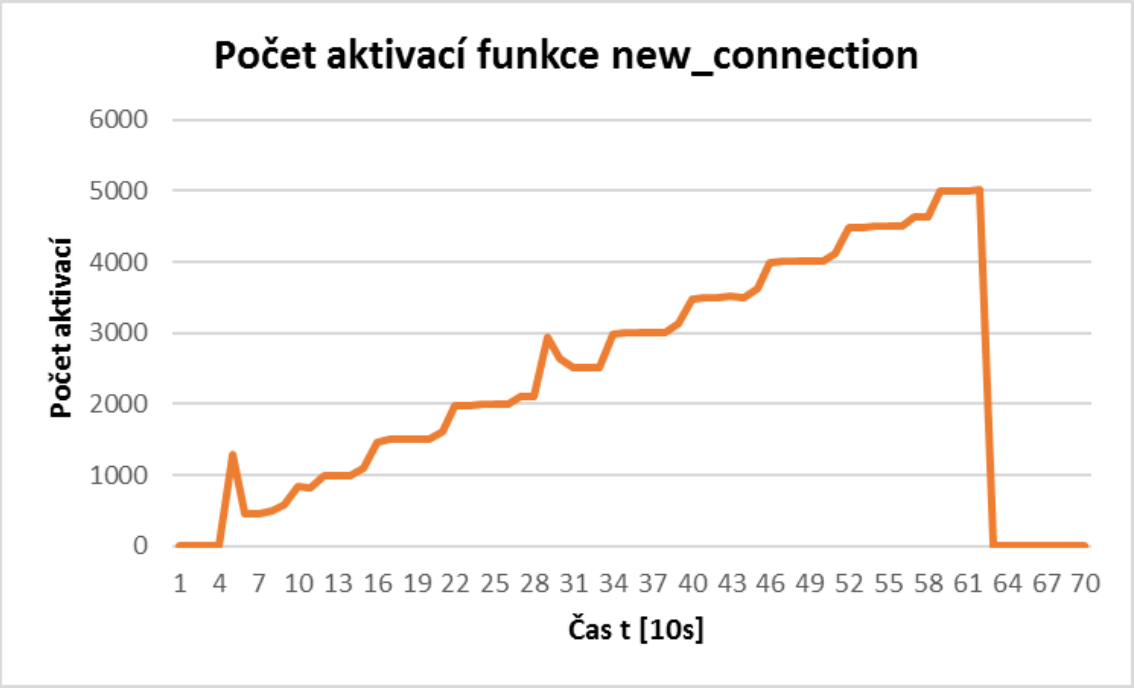
8.4 Útok se schodovitým náběhem

Ve druhém případě měření se testovala reakce nástroje s aplikovaným modulem lineární regrese na útok se schodovitým náběhem. Jeho průběh zaznamenaný na serveru Avalanche lze vidět na Obr. 8.6. Maximální objem vyslaných dat za jednotku času byl v tomto případě 48 477 kb/s.



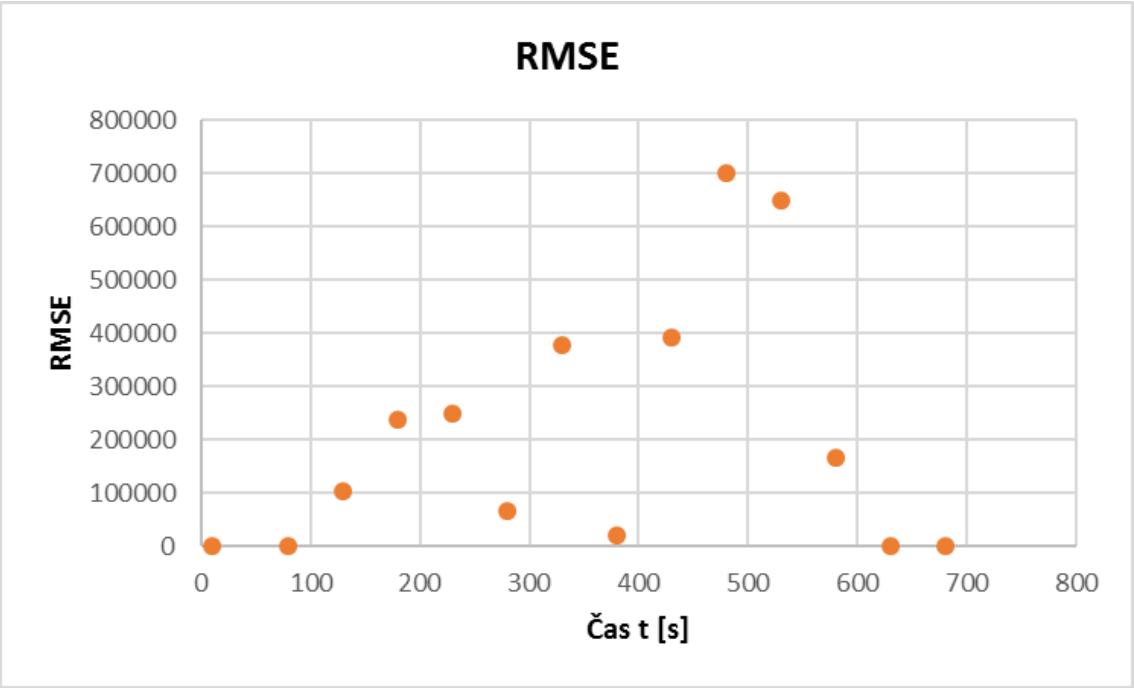
Obr. 8.6: Generace útoku se schodovitým náběhem zaznamenaná na serveru Avalanche.

Na Obr. 8.7 je opět patrná poměrně věrná rekonstrukce útoku zaznamenaná v počtu aktivací události.



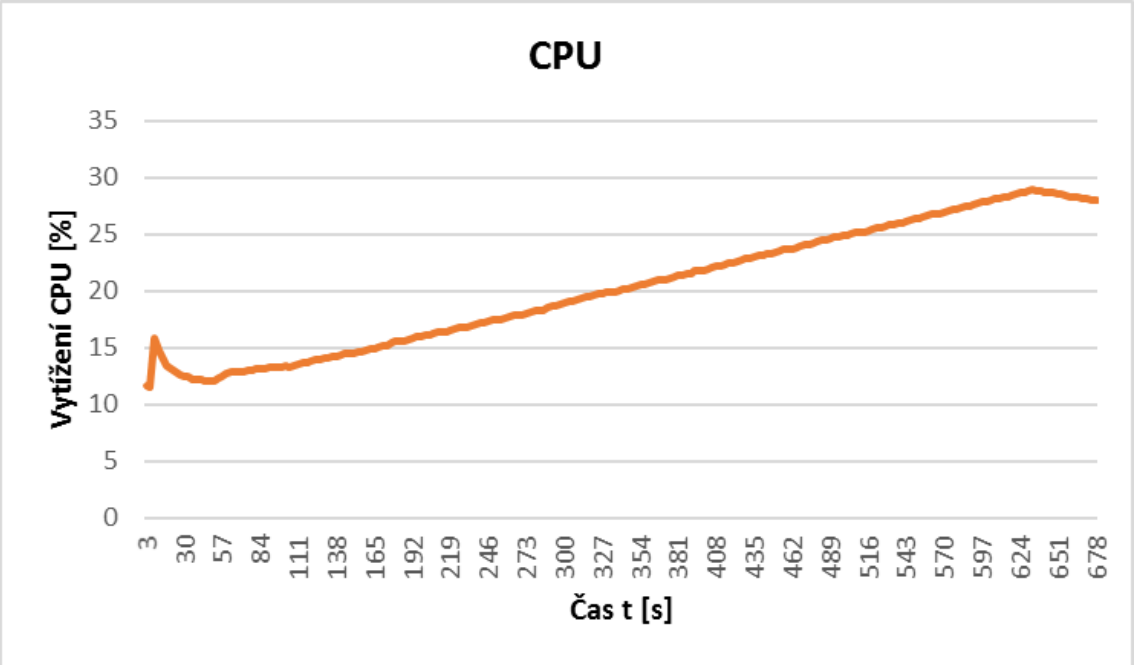
Obr. 8.7: Počet aktivací události – schodovitý náběh.

Z průběhu vývoje hodnot RMSE (Obr. 8.8) vyplývá poměrně značná závislost chyby na časovém intervalu sběru dat a vzrůstající tendenci k objemu provozu. Vzhledem k průběhu se chyba mění a nenalézá konstantní úroveň, stejně jako je tomu v prvním případě.



Obr. 8.8: Změna hodnoty RMSE vzhledem k průběhů útoku se schodovitým náběhem.

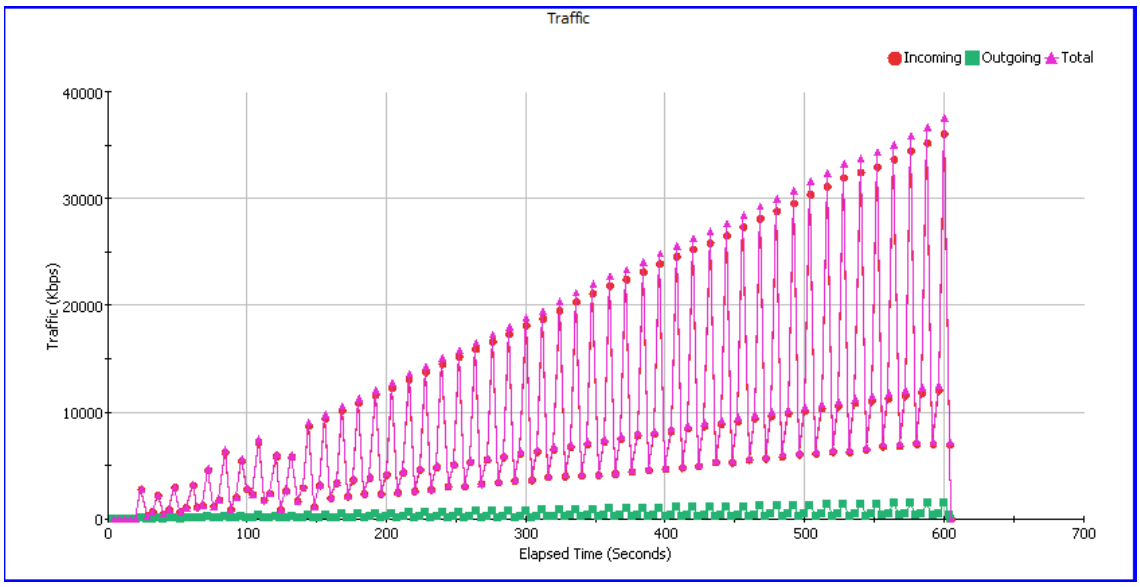
Závislost reakce zatížení CPU na útok (Obr. 8.9) je přibližně lineární. Reakce na tento typ útoku je oproti předchozímu případu mírnější, i když dosahuje v nejpozdějším stádiu stejného objemu provozu jako v prvním případě, zatížení CPU je přibližně o 20% nižší.



Obr. 8.9: Změna zatížení CPU vzhledem k průběhu útoku se schodovitým náběhem.

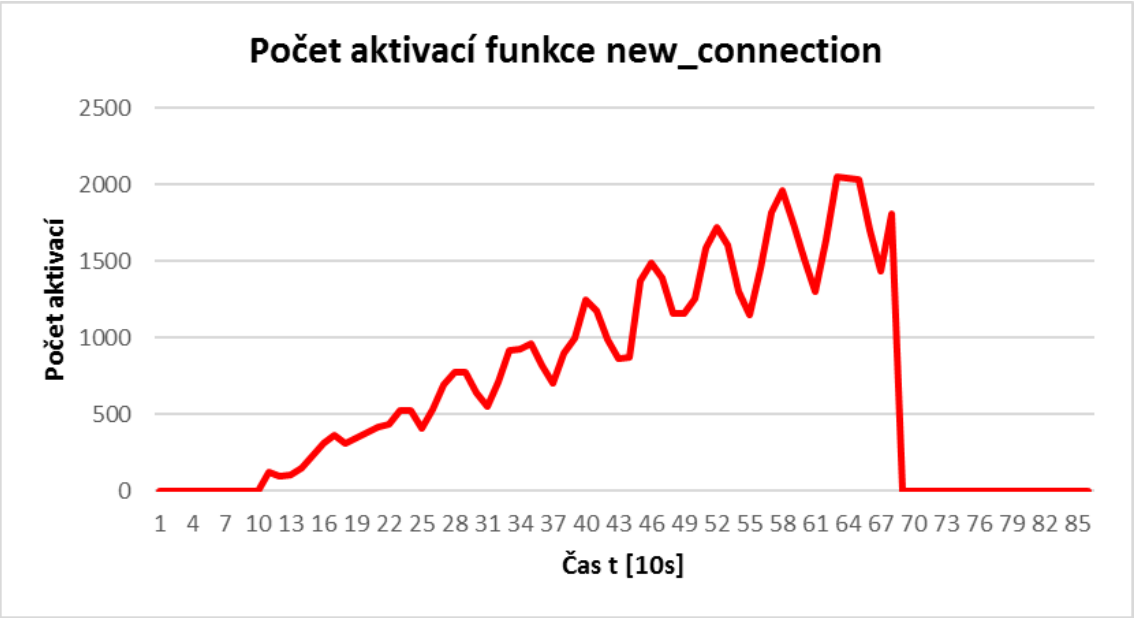
8.5 Útok se zubovitým náběhem

Poslední fáze měření testovala reakci modulu na útok se vzrůstajícím zubovitým náběhem (Obr. 8.10). Samotné měření bylo prováděno ve dvou krocích. V prvním případě bylo ponecháno nastavení konstant modulu na hodnotě známé z přechozích pokusů. Ve druhém případě byla zvolena optimalizace v podobě změny času sběru dat z 10 vteřin na 2 vteřiny (proměnná cas_zaznamu) a hodnocena reakce aktivace události a zátěže CPU.



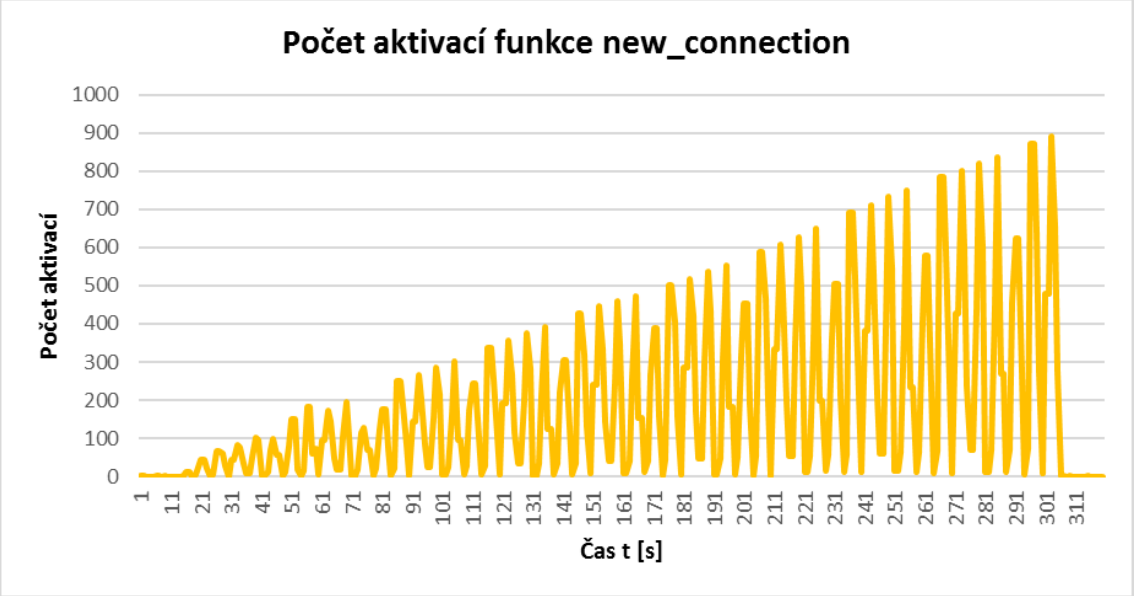
Obr. 8.10: Generace útoku se zubovitým náběhem zaznamenaná na serveru Avalanche.

Z Obr. 8.11 je patrná horší rekonstrukce průběhu útoku z důvodu jeho vysoké proměnlivosti a nastavení proměnné pro sběr dat na hodnotu 10 vteřin. Pro úspěšnou detekci není nicméně nutné mít věrohodně zrekonstruovaný signál. Naopak je nutné najít kompromis mezi zatížením CPU a úspěšnou detekcí. Vliv optimalizace doby, kdy je prováděn sběr dat a dopad její změny na zátěž, je rozebírán dále.



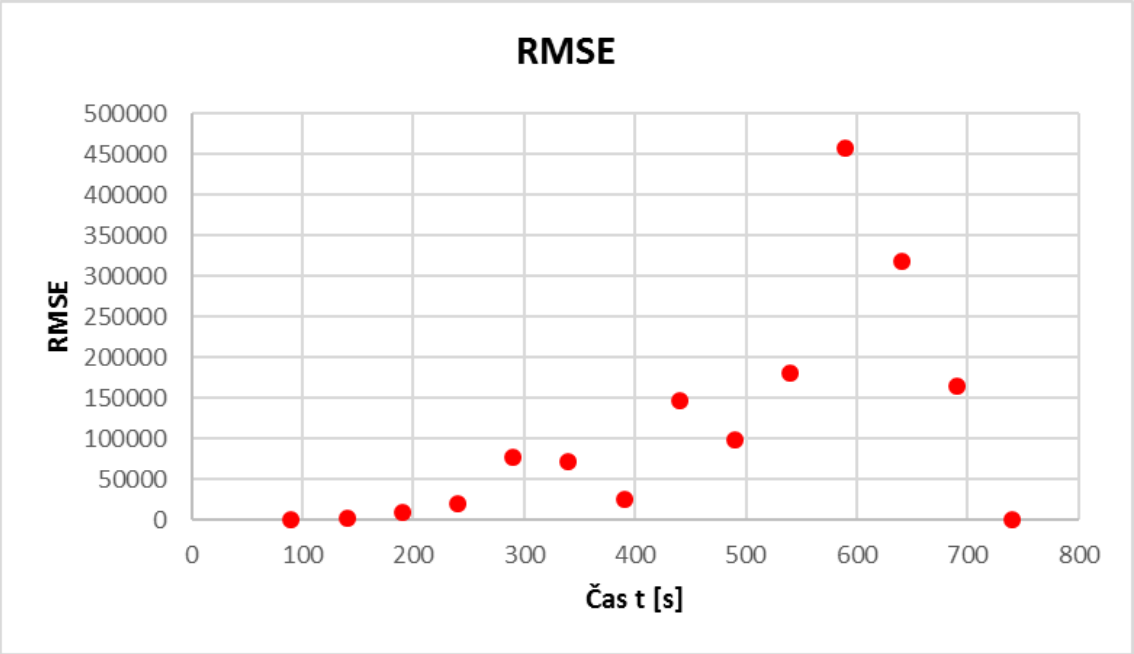
Obr. 8.11: Počet aktivací události – zubovitý náběh s dobou sběru dat 10s.

Po změně nastavení konstanty pro délku sběru dat na 2 vteřiny (datová struktura má tedy délku sběru dat 8s). Rekonstrukce útoku podle počtu spojení je zaznamenána na Obr. 8.12. Je zjevné, že úprava doby zaznamenávající spojení, má výrazný vliv na schopnost přesné rekonstrukce provozu, a to z důvodu častějšího „vzorkování“ otevřených spojení.



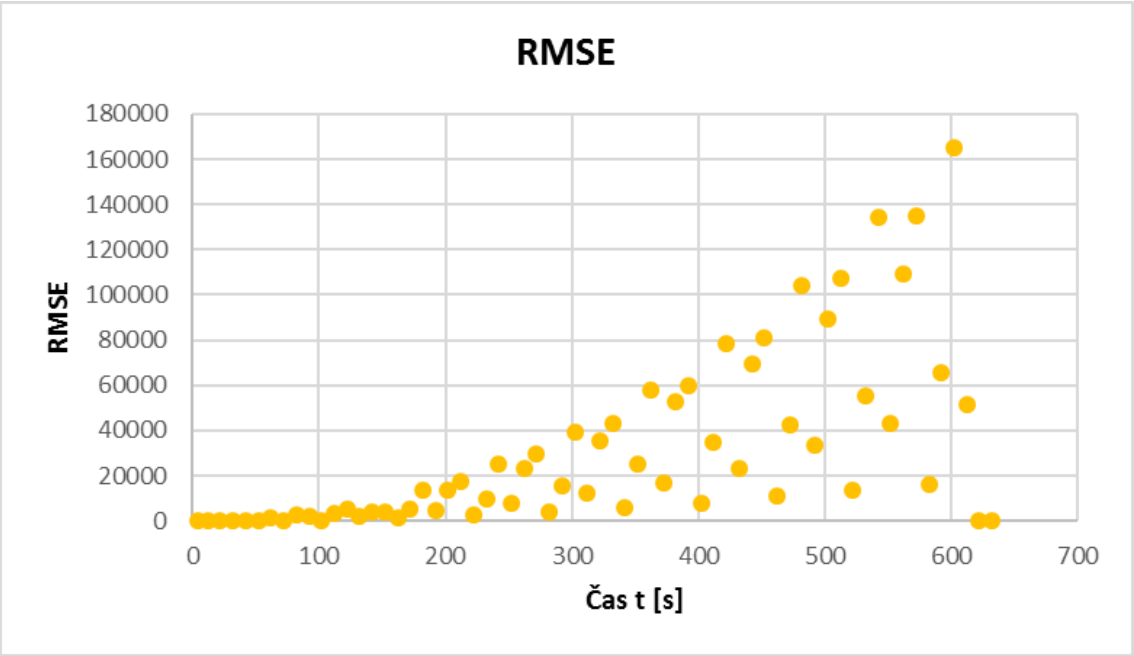
Obr. 8.12: Počet aktivací události – zubovitý náběh s dobou sběru dat 2s.

Hodnota chyby RMSE je v tomto případě znovu závislá na vzrůstajícím objemu dat provozu a opět relativně výrazně variuje vzhledem k stále se měnícímu průběhu útoku.



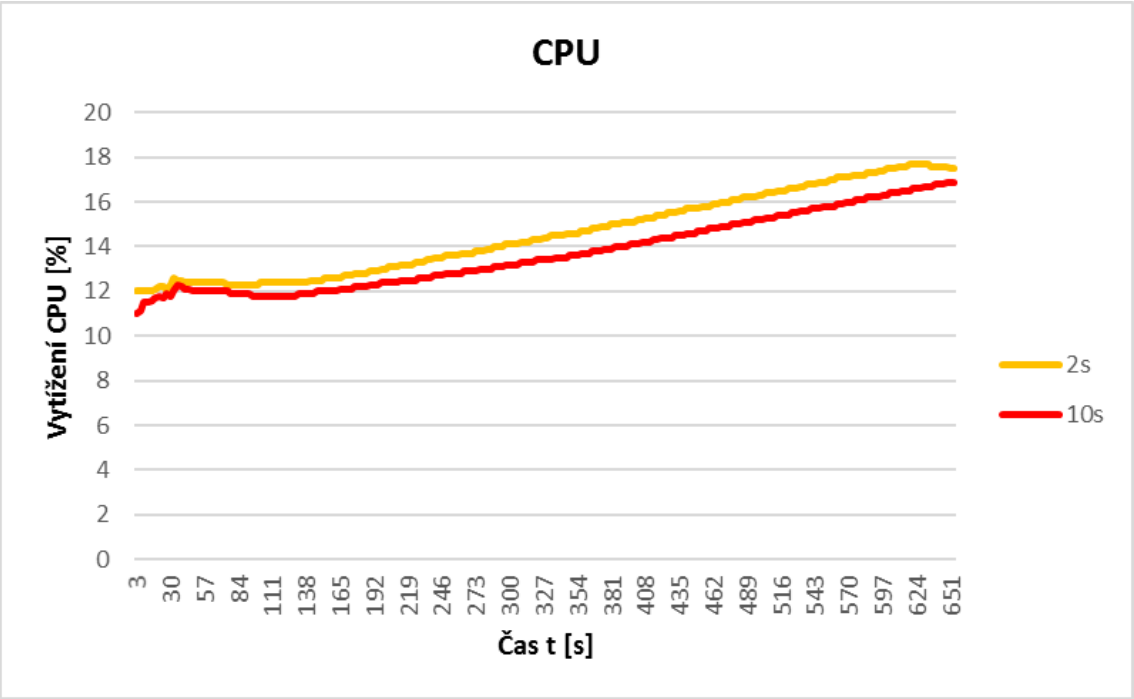
Obr. 8.13: Změna hodnoty RMSE vzhledem k průběhů útoku se zubovitým náběhem a dobou sběru dat 10s.

Při pětinásobném zrychlením výpočtu chyby RMSE je patrný velký rozptyl hodnot zapříčiněný momentem sběru aktuálních dat, které jsou porovnávány s daty predikovanými. Kvůli vysoké varianci průběhu se touto metodou nedá spolehlivě predikovat následující stav, což zapříčiňuje velký rozptyl hodnot.



Obr. 8.14: Změna hodnoty RMSE vzhledem k průběhů útoku se zubovitým náběhem a dobou sběru dat 2s.

Co se týče vlivu zvýšení frekvence provádění výpočtu RMSE na vytížení CPU, závislost lze vidět na Obr. 8.15. Při bližším zkoumání vyplývá, že pětinásobné zvýšení frekvence výpočtu u útoku tohoto typu zapříčiňuje v průměru 0,85% zvýšení vytíženosti CPU.



Obr. 8.15: Porovnání změny zatížení CPU vzhledem k průběhu útoku se zubovitým průběhem a změnou konstanty pro frekvenci sběru dat.

8.6 Zhodnocení dosažených výsledků

V práci byla otestována aplikovaná metoda jednoduché lineární regrese v nástroji Bro určená k detekci útoků typu DDos. Její schopnost predikovat následující stav byla nejlépe vyjádřena při útoku s normálním průběhem, kdy hodnotu RMSE výrazně zvýšil pouze prudký nárůst objemu dat. Ze zkoumání reakce na chybu vyplynulo, že pro úspěšnou detekci by v tomto případě měla být globální konstanta prahrmse nastavena na hodnotu pohybující se okolo 100 000. Problém pro mnohá řešení však může představovat vytížení CPU, jelikož při maximálním zatížení dosahovalo u tohoto typu útoku až 50% zátěže. Při dalším měření bez aplikovaného modulu vyšlo najevo, že toto relativně vysoké zatížení CPU generuje nástroj Bro v jeho základním nastavení i bez aplikovaného modulu lineární regrese.

Ve druhém případě se schodovitým průběhem útoku hodnota chyby RMSE výrazně kolísala spolu se vzrůstajícím objemem provozu a její hodnota byla v maximu nejvyšší ze všech tří testovaných scénářů. Zároveň se lineárně zvyšovalo vytížení CPU až na hranici bezmála 30%.

V poslední části s pilovitým náběhem útoku vykazoval rozptyl chyby RMSE podobný průběh a hodnoty jako u předchozího případu, nicméně po zvýšení frekvence výpočtů se její hodnota snížila. Toto zvýšení frekvence mělo pozitivní vliv na rekonstrukci samotného útoku podle aktivace události new_connection. Častější výpočet ovlivnil také zátěž procesoru, a to v průměru o 0,85%. Přehled průměrných naměřených hodnot je zpracován v Tab. 8.1.

Tab. 8.1: Průměrné hodnoty sledovaných parametrů.

Průběh	CPU [%]	RMSE
Klasický	38,19	30 963
Schodovitý	20,58	247 255
Zubovitý	13,72	120 658
Zubovitý 2	14,57	34 768

Testování potvrdilo schopnost této metody detekovat útoky typu DDoS. Různé nastavení parametrů modulu potom umožňuje ovlivňovat detekci tak, aby odpovídala konkrétním potřebám. Za velkou výhodu metody se dá považovat schopnost adaptace na postupný nárůst objemu provozu, který je přítomný v reálných sítích. Ovlivňování doby sběru dat a celkové velikosti cyklu dokáží dále modifikovat modul tak, aby co nejvíce vyhovoval potřebám nasazení.

Nevýhodou je poměrně vysoké využití CPU, jenž může být pro použití tohoto typu řešení (Bro + aplikovaný modul lineární regrese) limitující faktor kvůli nedostatku výpočetních prostředků vyhrazených k detekci.

9 ZÁVĚR

Diplomová práce se v první části věnovala teoretickému rozboru IDPS nástrojů včetně rozboru mechanismů, které jim slouží k detekci. V další části se zaměřila na teoretický rozbor přístupu k detekci anomálií se zaměřením na statistické metody, ale okrajově se věnovala i metodám zaměřeným na dolování dat a strojovému učení.

Ve třetí části se práce zabývala teoretickým rozborem konkrétních open-source IDPS nástrojů a jejich možnostmi.

V praktické části práce ověřovala funkci těchto nástrojů a porovnávala je. V prvním testovaném scénáři hodnotící procesní rychlost vyšlo najevo, že nástroj Snort má lepší procesní výkonnost než jeho přímý konkurent Suricata. Předmětem výzkumu by v budoucnosti mohla být otázka, jaké hodnoty by byly naměřeny, pokud by měla Suricata možnost přístupu ke druhému jádru procesoru. Bro vykazoval ve své základní konfiguraci v tomto pohledu nejlepší výkonnost.

Druhá část měření se věnovala reakcím nástrojů na simulovaný SYN flood DDoS útok. Z měření vyplynulo, že Snort generoval upozornění na neobvyklé prvky v některých hlavičkách TCP, ale celkově tento útok v dané konfiguraci nezaznamenal. To stejné se dá říct i o Suricatě, jejíž soubor s upozorněními neobsahoval žádnou položku.

Další část práce se zabývala reakcí nástrojů na skenování portů. Při tomto testování vyšlo najevo, že všechny nástroje byly schopny detekce skenování a záznamu upozornění do příslušných souborů.

Poslední část se věnovala návrhu a implementaci vybrané metody lineární regrese do nástroje Bro, který byl zvolen jako vhodná aplikace vzhledem k jeho skriptovací platformě. Do nástroje byl tento modul prakticky implementován a následně byla ověřena jeho funkčnost v experimentální sítí VUT v Brně. Z testování vyšla najevo jeho schopnost detekovat různé průběhy útoků typu DDoS pomocí vypočtené RMSE. Poměrně vysoké využití CPU však může představovat v této podobě v reálném nasazení problém.

V práci bylo ověřeno, že navržený způsob detekce využívající lineární regresní analýzy k odhalení útoků DDoS, splnil očekávání a útoky skutečně detekoval. Pro eliminaci nepravých pozitivních stavů je nutné ošetřit nastavitelné konstanty podle

specifických potřeb, nicméně zde existuje předpoklad, že se tento algoritmus, ať už implementovaný v Bro, nebo jiné podobě, může uplatnit v reálném nasazení.

SEZNAM POUŽITÝCH ZDROJŮ

[1] SNYDER, Joel. Do you need an IDS or IPS, or both? In: *SearchSecurity* [online]. USA: TechTarget, 2009 [cit. 2016-10-29]. Dostupné z: http://searchsecurity.techtarget.com/Do-you-need-an-IDS-or-IPS-or-both

[2] CARTER, Earl. a Jonathan. HOGUE. Intrusion prevention fundamentals. Indianapolis, IN: Cisco Press, c2006. ISBN 1587052393.

[3] RAO, Umesh Hodeghatta. a Umesha. NAYAK. *The InfoSec handbook: an introduction to information security*. New York, New York: Apress, 2014. Expert's voice in information security. ISBN 9781430263821.

[4] LUNT, Teresa F.; TAMARU, Ann; GILLHAM, F. A real-time intrusion-detection expert system (IDES). SRI International. Computer Science Laboratory, 1992.

[5] HOFMEYR, Steven A.; FORREST, Stephanie; SOMAYAJI, Anil. Intrusion detection using sequences of system calls. *Journal of computer security*, 1998, 6.3: 151-180.

[6] PATCHA, Animesh; PARK, Jung-Min. An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer networks, 2007, 51.12: 3448-3470.

[7] LAKHINA, Anukool; CROVELLA, Mark; DIOT, Christophe. Mining anomalies using traffic feature distributions. In: ACM SIGCOMM Computer Communication Review. ACM, 2005. s. 217-228.

[8] KIM, Seong Soo; REDDY, AL Narasimha; VANNUCCI, Marina. Detecting traffic anomalies through aggregate analysis of packet header data. In:

International Conference on Research in Networking. Springer Berlin Heidelberg, 2004. s. 1047-1059.

[9] IEEE, TIANJIN UNIVERSITY OF TECHNOLOGY a IEEE Engineering in Medicine and Biology Society. Ed.: Peihua Qiu .. EMB. Image and Signal Processing, 2009. CISP '09. 2nd IEEE, TIANJIN UNIVERSITY OF TECHNOLOGY a IEEE Engineering in Medicine and Biology Society. Ed.: Peihua Qiu .. EMB. Image and Signal Processing, 2009. CISP '09. 2nd International Congress on. Piscataway, NJ: IEEE, 2009. ISBN 9781424441297.

[10] VALDES, Alfonso; SKINNER, Keith. Adaptive, model-based monitoring for cyber attack detection. In: International Workshop on Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2000. s. 80-93.

[11] LAZAREVIC, Aleksandar, et al. A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. In: SDM. 2003. s. 25-36.

[12] BEALE, Jay, Andrew R. BAKER a Joel. ESLER. Snort: IDS and IPS toolkit. Burlington, MA: Syngress, c2007. ISBN 9781597490993.

[13] BILES, Simon. *Detecting the Unknown with Snort and the Statistical Packet Anomaly Detection Engine (SPADE)*. Computer Security Online Ltd.

[14] OREBAUGH, Angela., Simon. BILES a Jacob. BABBIN. *Snort cookbook*. Sebastopol: O'Reilly Media, 2005. ISBN 9780596007911.

[15] CHANDOLA, Varun; BANERJEE, Arindam; KUMAR, Vipin. Anomaly detection: A survey. ACM computing surveys (CSUR), 2009, 41.3: 15.

[16] ROESCH, Martin. *Writing Snort Rules* [online]. 2001 [cit. 2016-11-21]. Dostupné z: http://paginas.fe.up.pt/~mgi98020/pgr/writing_snort_rules.htm

[17] *Bro 2.5 documentation* [online]. In: , The Bro Project. 2016 [cit. 2016-11-20]. Dostupné z: https://www.bro.org/sphinx-git/intro/index.html#architecture

[18] DASH, Pradyumna a Jarek BLAMINSKY. *Getting started with oracle VM virtualbox: build your own virtual envinronment from scratch using virtualbox*. Birmingham, England: Packt Publishing, 2013. ISBN 9781782177838.

[19] FORD, Andrew. *Apache 2: pocket reference*. Sebastopol: O'Reilly, 2008. ISBN 978-0-596-51888-2.

[20] *Proceedings of the Second Internet Measurement Workshop: IMW 2002 : Marseille, France, November 6-8, 2002*. New York, N.Y.: Association for Computing Machinery, c2002, s. 71-82. ISBN 158113603x.

[21] MONTGOMERY, Douglas C., Elizabeth A. PECK a G. Geoffrey VINING. *Introduction to linear regression analysis*. 5th ed. Hoboken, NJ: Wiley, 2012. ISBN 9780470542811.

[22] KUTNER, Michael H. Applied linear statistical models. 5th ed. /. Boston: McGraw-Hill Irwin, c2005. ISBN 0-07-238688-6.

[23] CHAI, T. a R. R. DRAXLER. Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. *Geoscientific Model Development* [online]. 2014, **7**(3), 1247-1250 [cit. 2017-05-19]. DOI: 10.5194/gmd-7-1247-2014. ISSN 1991-9603. Dostupné z: http://www.geosci-model-dev.net/7/1247/2014/

SEZNAM POUŽITÝCH ZKRATEK

CPU	Cetrální Procesorová Jednotka
DoS	Denial of Service
DDoS	Distributed Denial of Service
FFT	Rychlá Fourierova Transfromace
FTP	File Transfer Protocol
HIDS	Host Intrusion Detection Systems
HIPS	Host Intrusion Prevention Systems
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection Systém
IDPS	Intrusion Detection and Prevention Systems
IP	Internet Protocol
IPS	Intrusion Prevention Systems
LDAP	Lightweight Directory Access Protocol
NIDS	Network Intrusion Detection Systems
NIPS	Host Instrusion Prevention Systems
OS	Operační systém
RMSE	Root Mean Squared Error
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SPADE	Statistical Packet Anomaly Detection Engine
SSL	Secure Sockets Layer
TCP	Transmition Control Protocol
TLS	Transport Layer Security

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

Na přiloženém CD lze najít vytvořený modul využívající jednoduché lineární regrese pro odhalení útoků typu DDoS. Dále přiložené CD obsahuje archiv s nástrojem Bro ve verzi 2.5 a složku obsahující data z proběhlého testování:

- **Bro-2.5.tar** - Archiv s nástrojem Bro ve verzi 2.5.
- **Dos.bro** - Soubor obsahující vytvořený modul využívající lineární regrese.
- **Readme.txt** - Soubor obsahující informace k instalaci Bro a spuštění modulu.
- **Testování** - Adresář, který obsahuje data z testování včetně vygenerovaných logů modulu.