VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE ÚTOKU UHÁDNUTÍ HESLA V SÍŤOVÉM PROVOZU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MAREK HURTA
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE ÚTOKU UHÁDNUTÍ HESLA V SÍŤOVÉM PROVOZU

DETECTION OF BRUTE-FORCE PASSWORD ATTACK IN NETWORK TRAFFIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MAREK HURTA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN ŽÁDNÍK, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Tato bakalářská práce se zabývá monitorováním sítí pomocí IP toků. Popisuje framework Nemea, který se využívá pro tvorbu modulů schopných detekovat síťové anomálie. Dále popisuje způsoby, jakými je možné napadnout protokoly SSH, RDP a Telnet. Následně se zaměřuje na slovníkový typ útoků a útok hrubou silou, přičemž analyzuje jejich charakteristiky. Na základě této analýzy vytváří obecné signatury, které popisují tyto útoky. Signatury útoků slouží jako základ pro detekční algoritmus, který využívá histogramovou analýzu z uložených záznamů. Výsledky detekčního algoritmu jsou porovnány vůči existujícím řešením.

Abstract

This bachelor's thesis is aimed at monitoring of computer networks using IP flows. It describes NEMEA framework which is used for creating modules. These modules are able to detect network anomalies and attacks. Next part describes a few methods how SSH, RDP and Telnet protocols could be attacked. Following chapters analyze some types of attacks such as Dictionary or Brute-Force attack and tries to find their common characteristics. Based on this analysis, signature of attack is created. Proposed detection algorithm uses these signatures for computing detection thresholds which are used in histogram analysis. Finally, results of proposed detection algorithm are compared with the results from other known methods.

Klíčová slova

Nemea, NetFlow, Detekcia uhádnutí hesla, Histogramová analýza

Keywords

Nemea, NetFlow, Detection of Brute-Force password attack, Histogram analysis

Citace

Marek Hurta: Detekce útoku uhádnutí hesla v síťovém provozu, bakalářská práce, Brno, FIT VUT v Brně, 2015

Detekce útoku uhádnutí hesla v síťovém provozu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Žádníka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Marek Hurta
11. května 2015

Poděkování

Rád by som poďakoval vedúcemu práce Ing. Martinovi Žádnikovi Ph.D. za pomoc pri vytváraní tejto práce. Ďalej by som chcel poďakovať svojej rodine za podporu počas celého štúdia.

© Marek Hurta, 2015.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

V dnešnej dobe si môžeme všimnúť narastajúci záujem o internetovú bezpečnosť z pohľadu bežných užívateľov ako aj väčších organizácií. Dôvodom tohto záujmu je čoraz väčší počet rôznych druhov útokov či pokusov o preniknutie do bežne používaných systémov. Snahou útočníkov je získať prístup k staniciam, či už za účelom získania citlivých informácii alebo k ich zneužitiu. Útočník týmto spôsobom získava prostriedky, ktoré mu umožnia vytvárať masívnejšie útoky a zároveň znižujú šancu na jeho odhalenie, pretože útok za neho vykonávajú napadnuté stanice. Útočníci používajú okrem známych metód aj čoraz sofistikovanejšie druhy útokov ako napríklad pomalé hádania hesiel na rôzne druhy serverov.

Z tohto dôvodu je kladený stále čoraz väčší dôraz na systémy pre detekciu prienikov IDS (*angl. intrusion detection system*), ktoré sú schopné zachytávať tieto bezpečnostné hrozby. Do tejto kategórie sa radí aj modulárny systém pre detekciu sieťových útokov **Nemea** (*angl. Network Measurements Analysis)*. Tento framework využíva modul, ktorého úpravou sa zaoberá táto práca. Úprava tohto modulu je založená na behaviorálnej analýze sieťových tokov a hlavným prínosom by mala byť schopnosť modulu reagovať na rôzne druhy útokov a ich modifikácii, ktoré by v pôvodnej verzií vyžadovali manuálny zásah do nastavení modulu. Navrhnutý algoritmus je okrem iného generický, z toho vyplýva možnosť jeho využitia nad akýmkoľvek cieľovým protokolom.

Kapitola 2 popisuje protokol NetFlow, jeho základnú architektúru a aktuálne používané verzie. Tak isto objasňuje pojmy ako IP tok, IPFIX a predstavuje nástroje pre prácu s dátami Nfdump a NfSen. Kapitola 3 sa zmieňuje o frameworku Nemea, ktorého súčasťou je mnou upravovaný modul. Ďalej podrobne popisuje základné stavebné časti tohto systému, spôsob akým sú medzi sebou prepojené a formát správ, ktoré si medzi sebou vymieňajú. Popis protokolov, na ktoré sa primárne zameriava táto práca obsahuje kapitola 4. Kapitola tiež pojednáva o možných spôsoboch napadnutia týchto protokolov. V kapitole 5 je načrtnutý spôsob vytvárania tzv. signatúry útoku, ktorá je kľúčovou časťou detekcie. Následne sú v kapitole ukázané referenčné signatúry pre všetky cieľové protokoly vo forme histogramov. Obsahom kapitoly 6 je podrobný popis detekčného algoritmu, ktorý je založený na histogramovej analýze. V závere kapitoly je stručné porovnanie pôvodnej a aktuálnej metódy. Kapitola 7 popisuje výsledky detekcie nad reálnymi dátami. Ďalej obsahuje rôzne porovnania oproti pôvodnej verzii modulu. Záverečná kapitola 8 popisuje zhodnotenie celej práce.

Kapitola 2

Monitorovanie siete

Nasledujúca kapitola objasňuje základné princípy a postupy, ktoré sa využívajú pri monitorovaní sietí. Taktiež bude čitateľ oboznámený s protokolom NetFlow a jeho základnou štruktúrou.

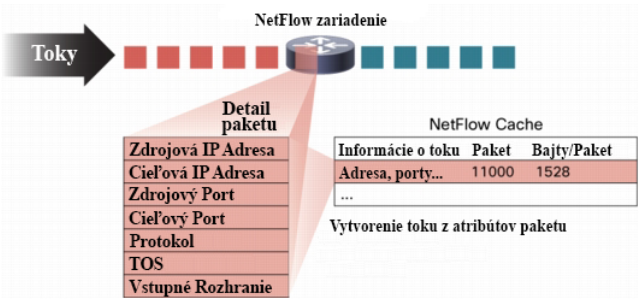
2.1 NetFlow

NetFlow je sieťový protokol vyvinutý firmou Cisco pre monitorovanie a meranie počítačových sietí pomocou IP tokov (*angl. IP flows*). Tok je definovaný ako jednosmerná postupnosť IP paketov prechádzajúcich pozorovaným bodom v sieti za určitý časový interval [11]. Pakety tvoriace jeden tok, sa vyznačujú spoločnými vlastnosťami [13]:

- Zdrojová/Cieľová IP adresa
- Zdrojový/Cieľový port
- Číslo protokolu (3.vrstva ISO/OSI modelu)

Jednotlivé vlastnosti IP toku znázorňuje obrázok 2.1. Na základe týchto položiek je možné každý paket jednoznačne priradiť konkrétnemu toku. Z pohľadu NetFlow protokolu je klasický tok rozšírený o ďalšie dve vlastnosti:

- Typ služby
- Číslo vstupného rozhrania



Obrázok 2.1: Štruktúra IP toku [3]

Pre každý tok sa ďalej udržujú štatistiky o počte prenesených bajtov, počte paketov, čase vzniku komunikácie, dobe trvania komunikácie a iné. Tieto informácie sú uložené v NetFlow Cache v každom NetFlow zariadení.

2.2 Architektúra NetFlow

NetFlow architektúra pozostáva z troch základných častí:

- Exportér
- Kolektor
- Analyzátor

Exportér a kolektor sú prepojené pomocou NetFlow protokolu, ktorý umožňuje prenos jednotlivých NetFlow záznamov. V praxi sa bežne môžeme stretnúť s prípadom prepojenia jedného kolektoru a viacerých exportérov (v prípade kedy chceme monitorovať viac liniek naraz je potrebné aby každá linka mala samostatný NetFlow exportér).

2.2.1 Exportér

NetFlow exportér je sieťové alebo SW zariadenie, ktoré získava informácie o linke na základe monitorovania paketov, ktoré skladá do spoločných tokov pričom každý unikátny paket zakladá nový tok [11]. Pre jednotlivé toky vytvára záznamy obsahujúce informácie o danom toku a štatistiky. Záznamy sú na exportéri uložené v cache pamäti. Štruktúru uloženia zobrazuje obrázok 2.2.

SrcIf	SrcIPadd	DstIf	DstIPadd	Proto	ToS	Flgs	Pkts	Src Port	Src Msk	Src AS	Dst Port	Dst Msk	Dst AS	NextHop	Bytes/ Pkt	Active
Fa1/0	173.100.21.2	Fa0/0	10.0.227.12	11	80	10	11000	162	/24	5	163	/24	15	10.0.23.2	1528	1745
Fa1/0	173.100.3.2	Fa0/0	10.0.227.12	6	40	0	2491	15	/26	196	15	/24	15	10.0.23.2	740	41,5
Fa1/0	173.100.20.2	Fa0/0	10.0.227.12	11	80	10	10000	161	/24	180	10	/24	15	10.0.23.2	1428	1145,5
Fa1/0	173.100.6.2	Fa0/0	10.0.227.12	6	40	0	2210	19	/30	180	19	/24	15	10.0.23.2	1040	24,5

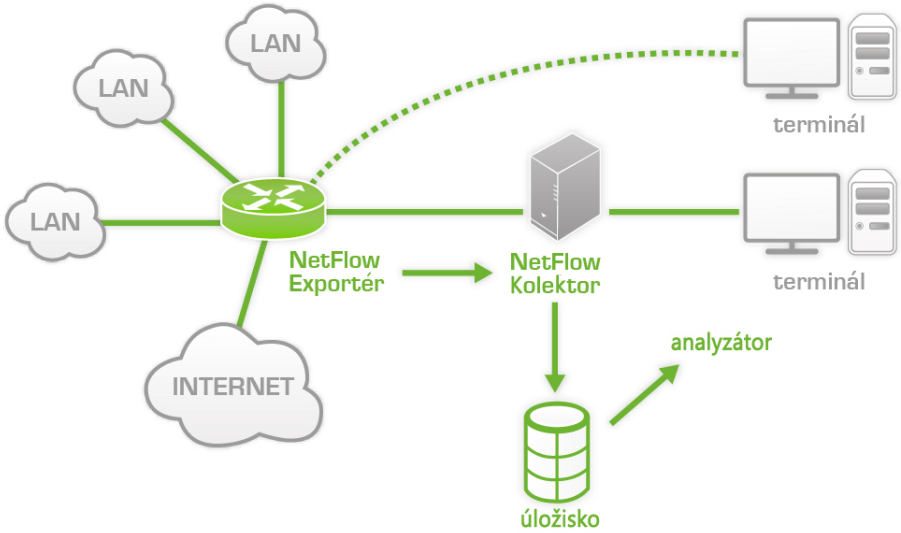
Obrázok 2.2: Položky uložené v NetFlow cache exportéru

Počas monitorovania môže dôjsť k exspirácií niektorých záznamov. V tomto prípade je exspirovaný záznam odoslaný na kolektor a uvoľnený z cache pamäte. K exspirovaniu záznamu dochádza [3]:

- V prípade plnej cache pamäte.
- Pokiaľ bol tok ukončený, tj. bol prijatý paket, ktorý obsahoval TCP príznak **FIN** indikujúci koniec spojenia alebo **RST** označujúci reset spojenia.
- V prípade vypršania aktívneho časovača, ktorý reprezentuje maximálnu dobu počas ktorej môže byť záznam v cache pamäti. Ďalší prichádzajúci paket z takéhoto toku vytvorí opäť nový záznam v cache. Bežná doba exspirácie je približne 30 minút.
- V prípade vypršania pasívneho časovača toku, tj. tok, ktorý nebol aktualizovaný novým paketom po dobu stanovenú časovačom sa stáva neaktívnym (exspiruje). Bežná hodnota časovača je 15 sekúnd.

2.2.2 Kolektor

Kolektor je zariadenie, ktoré pomocou NetFlow protokolu komunikuje s jedným alebo niekoľkými exportérmi za účelom získavania NetFlow záznamov. Na vyžiadanie sú záznamy prenesené z exportéru na kolektor kde sú následne uložené počas ľubovoľne dlhého časového obdobia [9]. Princíp prepojenia týchto prvkov ilustruje obrázok 2.3.



Obrázok 2.3: NetFlow architektúra [4]

Jednotlivé záznamy sa prenášajú pomocou UDP protokolu na čísle portu 2055, bežne sa však využívajú aj iné porty. Nakoľko je toto spojenie nespoľahlivé a spôsobuje stratu paketov, využíva sa protokol SCTP (Stream Control Transmission Protocol), ktorý garantuje bezstratové spojenie [19].

Nevýhodou použitia tohto protokolu je komunikácia všetkých kolektorov so všetkými sieťovými prvkami, ktoré dokážu exportovať NetFlow záznamy. Tento princíp môže spôsobovať nadmerné zaťaženie sieťových prvkov, ktoré sa odráža na ich celkovej výkonnosti.

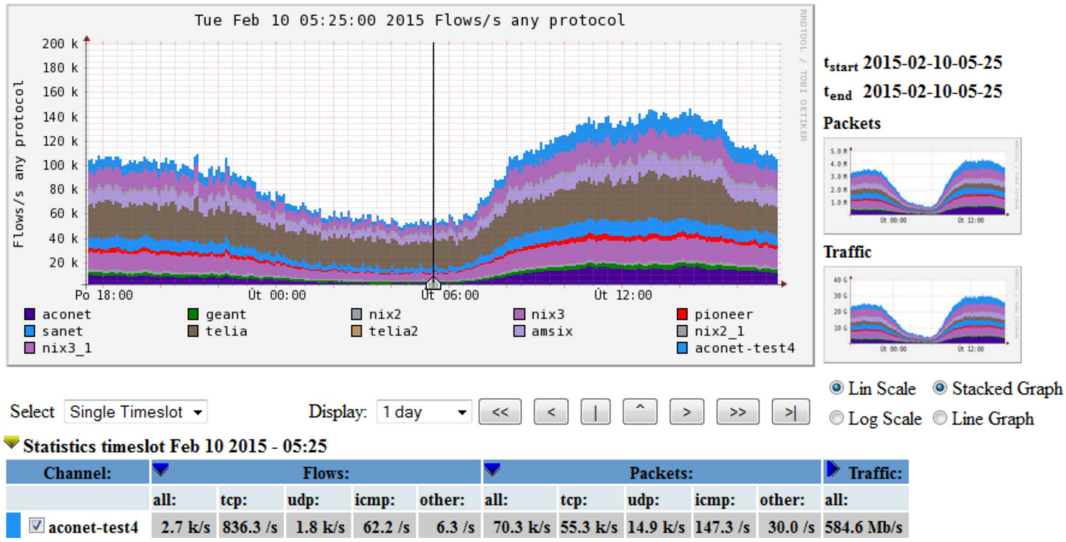
2.2.3 Analyzátor

Analyzátor je software, ktorý umožňuje definovať požiadavky alebo vytvárať profily. Nad dátami uloženými na kolektore prebieha analýza, ktorá môže byť vykonávaná v reálnom čase ale takisto môže ísť o analýzu nad ľubovoľne veľkým časovým intervalom.

Výsledkom toho sú špecifické informácie a štatistiky o monitorovanej sieti, ktoré môžu byť neskôr použité napríklad pre analýzu rôznych druhov útokov.

Príkladom takéhoto software môže byť nástroj **Nfdump**[1] s jeho webovým GUI **Nfsen**[2], ktorý je možné vidieť na obrázku 2.4:

[1] http://nfdump.sourceforge.net/
[2] http://nfsen.sourceforge.net/



Obrázok 2.4: Webové GUI nástroja Nfdump

2.3 Verzie protokolu

Firma Cisco vytvorila niekoľko verzií NetFlow protokolu. Niektoré z nich (v2, v3, v4) neboli nikdy vydané a ide len o interné verzie. Medzi najpoužívanejšie patria **verzia 5** a **verzia 9**. Práve posledná verzia 9 poslúžila ako základ pre vytvorenie nového protokolu IPFIX (Internet Protocol Flow Information Export), ktorý sa stal štandardom IETF (Internet Engineering Task Force) [2].

- **NetFlow v5 -** Najrozšírenejšia verzia protokolu, ktorá sa skladá z hlavičky (24 bajtov) a záznamu (48 bajtov). Táto verzia má fixný formát uloženia dát a nie je možné do tohto formátu pridávať nové položky [10]. Keďže v dnešnej dobe je snahou získavať čoraz väčšie množstvo informácií o tokoch je nutné tento typ protokolu rozširovať o ďalšie položky. Nakoľko má táto verzia protokolu fixnú štruktúru bolo potrebné vytvoriť verziu, ktorá je založená na dynamickom formáte uloženia dát.

- **NetFlow v9** [11] **-** Je založený na princípe šablón, do ktorých je možné pridávať ľubovoľné položky. Záznamy sa opäť skladajú z hlavičky, za ktorou je umiestnená jedna alebo viacero šablón. Po nich nasleduje dátová časť záznamu, ktorá odpovedá štruktúre šablóny. Kolektor je schopný podľa prijatej šablóny spracovať prichádzajúce záznamy. Pokiaľ však kolektor narazí na položku, ktorú nevie rozpoznať, vynechá ju a pokračuje v spracovaní ďalej. Výhodou tohto spôsobu je možnosť posielania len požadovaných položiek, čím sa redukuje objem posielaných dát a celkové zaťaženie.

Kapitola 3

Nemea

Nemea (Network Measurements Analysis) je framework slúžiaci predovšetkým na tvorbu systému, ktorý je schopný analyzovať dáta z monitorovanej siete automaticky a v reálnom čase. Základnými stavebnými časťami celého systému sú prvky nazývané moduly, o ktorých pojednáva podkapitola 3.1. Podkapitola 3.3 popisuje formát používaných správ. Na záver kapitoly bude čitateľ oboznámený s rozhraním TRAP.

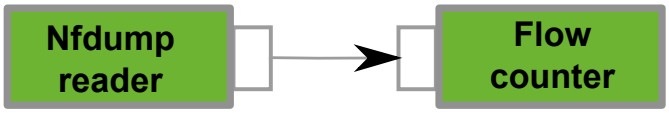
Informácie obsiahnuté v tejto kapitole boli čerpané prevažne z technickej správy popisujúcej framework NEMEA [7].

3.1 Modul

Modul je samostatná jednotka, ktorá je spustená nezávisle na ostatných častiach systému a je schopná prijímať tok dát prostredníctvom jej vstupného rozhrania. Prijaté dáta sú spracované a prebiehajú nad nimi rôzne druhy analýz, detekčné algoritmy zamerané na odhalenie bezpečnostných hrozieb alebo anomálii a iné. Výstupom môže byť tok dát, ktoré modul prijal na vstupnom rozhraní, ale aj iné informácie, ktoré slúžia ako vstupné dáta pre ostatné moduly.

Takýmto spôsobom je teoreticky možné dosiahnuť zapojenie neobmedzeného počtu modulov pričom každý z nich si zachováva svoju funkcionalitu a tým prispieva k vytvoreniu komplexného systému pracujúceho v reálnom čase.

Systém je navrhnutý tak, že dokáže pracovať nielen nad aktuálnym tokom dát, ale aj nad dátami, ktoré sú zozbierané do súborov a uložené v dátovom úložisku. Jednoduché prepojenie dvoch modulov je znázornené na obrázku 3.1.



Obrázok 3.1: Triviálne zapojenie dvoch modulov

Prvý modul číta uložené dáta zo súboru a preposiela ich cez svoje výstupné rozhranie na vstup druhého modulu. Systém môže pracovať nad veľkým množstvom rozličných dát alebo je možné modul nastaviť tak, že odosiela určitý počet záznamov opakovane za sebou čím simuluje nasadenie v reálnych podmienkach. Druhý modul má za úlohu sčítavať počet záznamov, paketov a bajtov. Štatistiky získané z tohto modulu môžu teoreticky slúžiť ako vstupné dáta pre iný modul alebo sa môžu ukladať do logu.

Jednou z hlavných výhod tohto systému je modularita, ktorá umožňuje dynamické pridávanie alebo odoberanie jednotlivých modulov počas behu systému bez nutnosti jeho zastavenia. Nový modul sa snaží pripojiť na špecifické rozhranie modulu, s ktorým by mal byť prepojený. Pokiaľ sa mu to z určitých dôvodov nepodarí, pokúša sa o pripojenie až kým nebude úspešný.

Podobný princíp je uplatnený aj pri situácii keď dôjde k výpadku spojenia kedy sa zasiahnutý modul opakovane snaží obnoviť spojenie. Tieto vlastnosti napomáhajú k rýchlej reakcii na zmenu požiadaviek spojených s monitorovaním siete.

Aj keď sa celý systém skladá zo vzájomne previazaných častí, porucha jedného modulu (napr. *segmentation fault*) nemusí nutne znamenať zastavenie činnosti celého systému. Ďalšou z výhod je možnosť využitia rozličných druhov programovacích jazykov.

3.2 Rozhranie modulu

Moduly si preposielajú dáta medzi sebou prostredníctvom rozhraní, ktoré umožňujú jednosmerný prenos. Dôležité je aby bol medzi dvomi komunikujúcimi rozhraniami dodržaný rovnaký formát posielaných dát. Jednotlivé moduly, môžu prijímať rozličné druhy vstupných formátov a na svojom výstupe opäť odosielať dáta v rôznych formátoch. Musí byť však zachovaná vyššie spomínaná kompatibilita medzi dvomi rozhraniami.

Formát dát na vstupe a výstupe modulu je možné definovať pri zapojení modulu do celého systému. Umiestnenie jednotlivých položiek v zázname a ich význam špecifikuje protokol **UniRec**, ktorý je popísaný v nasledujúcej kapitole 3.3. Princíp prepojenia jednotlivých modulov a parametre potrebné pre ustanovenie spojenia zabezpečuje platforma **TRAP**, ktorá je implementovaná knižnicou libtrap. Platforma TRAP bude rozobratá v kapitole 3.4.

3.3 UniRec formát

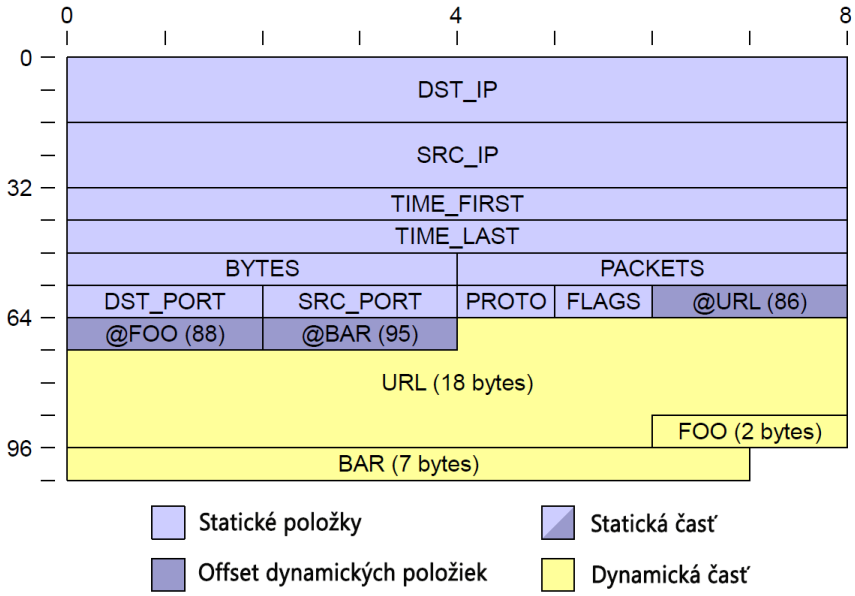
UniRec (Unified Record) je dátový formát správ posielaných prostredníctvom TRAP rozhrania. S rastúcim počtom modulov sa zvyšujú požiadavky na rozmanitosť prenášaných dát a samotný objem týchto dát. Z toho vyplývajú určité podmienky, ktoré by mal UniRec spĺňať:

- Pamäťová nenáročnosť
- Flexibilita
- Rýchla dostupnosť dát zo správy

UniRec záznam pozostáva z položiek (*angl. fields*), ktoré majú svoje meno a typ. Zoskupenie niekoľkých položiek do jedného celku sa nazýva šablóna (*angl. template*). Takáto šablóna je potom použitá v každom TRAP rozhraní. Nastavuje sa pri jeho inicializácií a zabezpečuje jednotvárnosť všetkých posielaných správ. Samotný záznam vyzerá ako jednoduchá štruktúra z programovacieho jazyka C, v ktorej sú všetky položky usporiadané za sebou. Delí sa na dve hlavné časti:

- Statická časť - prvky, z ktorých pozostáva majú vopred známu veľkosť
- Dynamická časť - veľkosť jednotlivých prvkov sa môže meniť

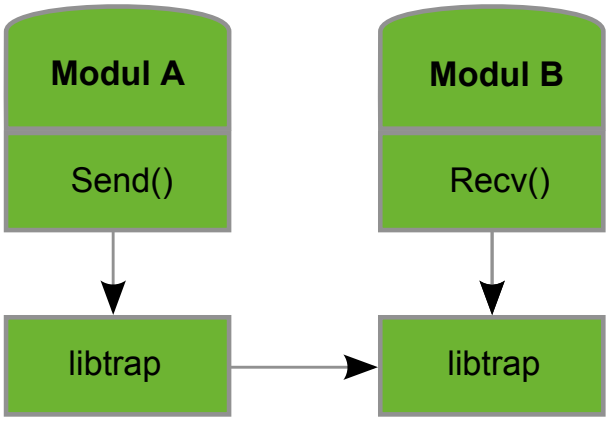
V rámci statickej časti záznamu sú za posledným prvkom uložené offsety jednotlivých položiek dynamickej časti. Hodnota offsetu vždy ukazuje na koniec príslušnej položky. Tento princíp je možné vidieť na obrázku 3.2.



Obrázok 3.2: UniRec záznam obsahujúci statickú časť spolu s offsetmi pre prvky z dynamickej časti

3.4 TRAP

Komunikácia medzi jednotlivými modulmi prebieha prostredníctvom TRAP (Traffic Analysis Platform) rozhrania, konkrétne pomocou jeho knižnice libtrap, ku ktorej majú prístup všetky moduly. Princíp komunikácie je opäť znázornený na obrázku 3.3.

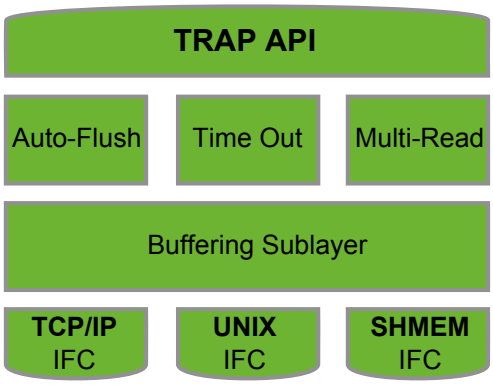


Obrázok 3.3: Prepojenie modulov s knižnicou libtrap

Základné operácie čítania a zapisovania môžu mať blokujúci alebo neblokujúci charakter v závislosti na nastavení parametrov prenosu. Modul odosiela dáta na výstup hneď ako sú

dostupné. Naopak v module, ktorý prijíma dáta, je čítanie implementované cyklom. Pokiaľ dáta prichádzajú na vstupné rozhranie, modul ich spracuje. V prípade, že žiadne dáta nie sú k dispozícií, čítanie je zablokované až do doby, kedy sú dáta opäť dostupné k čítaniu. Toto správanie sa však u niektorých modulov môže líšiť, v závislosti na konfigurácií rozhrania.

Knižnica libtrap sa skladá z niekoľkých častí, ktoré sú zakomponované do viacerých úrovní. Presné rozdelenie je zobrazené na obrázku 3.4.



Obrázok 3.4: Architektúra knižnice libtrap

Najnižšia vrstva je tvorená základnými prostriedkami, ktoré zabezpečujú komunikáciu medzi rozhraniami modulov:

- **TCP/IP** socket - využíva sa pri prepojení modulov, ktoré sú spustené na rozdielnych fyzických strojoch.
- **UNIX** socket - využíva sa pri bežnom type komunikácie dvoch modulov v rámci jedného fyzického stroja.
- **Shared memory** - využíva sa pri prepojení N modulov k jednému výstupnému rozhraniu.

Druhá vrstva zabezpečuje agregovanie požiadaviek, ktoré sú posielané medzi modulmi. Nakoľko je možné naraz preniesť väčšie množstvo požiadavkov, redukuje sa réžia, ktorá by bola potrebná pri prenose každého požiadavku samostatne. Táto možnosť je dostupná ako na vstupnom tak aj na výstupnom rozhraní. V špecifických situáciách je možné buffrovanie vypnúť, napríklad pri posielaní správ s vysokou prioritou, kedy sa hlavný dôraz kladie na rýchlosť ich doručenia.

Na ďalšej vrstve sa nachádzajú prostriedky, ktoré rôznym spôsobom rozširujú správanie nižších vrstiev:

- **Auto-Flush** - zabezpečuje vyprázdňovanie buffra na základe časovača. Z toho vyplýva, že dáta môžu byť odoslané aj pred tým než sa buffer úplne zaplní.
- **Time Out** - knižnica libtrap disponuje niekoľkými hodnotami časovačov, ktoré sú využívané vo funkciách reprezentujúcich čítanie a zápis. Na základe hodnoty časovača je možné nastaviť tieto funkcie do blokujúceho alebo neblokujúceho režimu.
- **Multi-Read** - využíva sa pri moduloch, ktoré používajú rôzny počet vstupných rozhraní. Čítanie je zabezpečené pomocou vlákien, ktoré čakajú na prichádzajúce dáta a následne ich predávajú na ďalšie spracovanie.

Kapitola 4

Cieľové protokoly, spôsob ich napadnutia a detekčné metódy

Nasledujúca kapitola popisuje protokoly, na ktoré sa moja práca zamerala. V podkapitolách 4.1, 4.2 a 4.3 budú postupne rozobrané cieľové protokoly a ich základné vlastnosti. Podkapitola 4.4 pojednáva o rôznych spôsoboch ich napadnutia a takisto o rôznych metódach útokov, ktoré sa pri napadnutiach používajú. V závere tejto kapitoly sú uvedené niektoré existujúce práce, ktoré sa zaoberajú detekciou týchto útokov.

4.1 SSH protokol

Protokol SSH (*angl.* Secure Shell) je protokol, pre zabezpečené prihlásenie na vzdialenú stanicu a pre iné sieťové služby. Takisto sa jedná o aplikáciu typu klient/server [6].

Tento protokol bol navrhnutý ako protiklad k existujúcemu protokolu Telnet, ktorý počas komunikácie neposkytuje účastníkom žiadnu formu zabezpečenia prenášaného obsahu, keďže dáta sú prenášané len v textovej podobe. Takýto typ komunikácie je možné jednoducho odchytiť a získať z neho citlivé údaje. Protokol SSH beží väčšinou prostredníctvom TCP/IP na porte 22, pričom sa skladá z troch základných častí.

4.1.1 Transport Protocol Layer

Zabezpečuje dôveryhodnosť, integritu a poskytuje autentizáciu zo strany servera(*anlg. Host-based authentication*). Tento protokol je navrhnutý tak, aby zaručil jednoduchú a rýchlu výmenu potrebných parametrov pri zahájení spojenia. Taktiež sa snaží o minimálny počet výmen parametrov (*angl. round-trips*) medzi užívateľom a hostom. Vo väčšine prípadov dochádza ku dvom takýmto výmenám a v najhoršom prípade až ku trom [24].

Počas výmeny týchto parametrov dochádza k voľbe hashovacieho algoritmu, algoritmu pre výmenu správ a výberu šifrovacích algoritmov. Server sa identifikuje užívateľovi pomocou unikátneho host-key a ten podľa neho zistí, či naozaj komunikuje so správnym serverom. Pokiaľ by užívateľ nikdy nekomunikoval s týmto serverom, nerozpozná jeho host-key a nemusí pokračovať v spojení.

4.1.2 User Authentication Protocol

Slúži k autentizácii užívateľa. Taktiež sa predpokladá, že bude zabezpečená integrita a dôveryhodnosť nižšou vrstvou protokolu. Server postupne oznamuje užívateľovi aké formy

autentizácie je možné použiť [22]. Je možné zvoliť jednu z nasledujúcich [18]:

- **Public key** - Tento spôsob je postavený na autentizácii pomocou privátneho kľúča užívateľa. Klient posiela serveru signatúru podpísanú práve jeho privátnym kľúčom. Server následne musí skontrolovať validitu prijatej signatúry pomocou verejného kľúča od daného užívateľa. Najčastejšie sa môžeme stretnúť s RSA šifrovacím algoritmom.

- **Host-based** - Táto metóda vyžaduje aby klient poslal správu podpísanú privátnym kľúčom užívateľa na server, ktorý ju následne overuje odpovedajúcim verejným kľúčom. Po overení dochádza k autorizácii prostredníctvom užívateľského mena klienta. Nakoľko je táto metóda voliteľná, neodporúča sa jej používanie vo vysoko zabezpečených sieťach.

- **Password** - Tento spôsob je založený na autorizácii pomocou hesla. Aj keď je heslo posielané ako obyčajný text s kódovaním ISO-10646 UTF-8, celý paket musí byť zašifrovaný na transportnej vrstve. Samotný server môže heslo odmietnuť v prípade, že sa nezhoduje s heslom užívateľa alebo došlo k jeho exspirovaniu.

- **None** - Pri tomto spôsobe prebieha autentizácia bez akýchkoľvek povinných prvkov. Takýto spôsob nie je doporučený.

4.1.3 Connection Protocol

Spojovacia vrstva beží nad dvoma spomínanými vrstvami a teda očakáva zabezpečenie integrity, overenie dôveryhodnosti a taktiež autentizáciu užívateľa. Samotná vrstva zabezpečuje interaktívne prihlasovanie, vzdialené spúšťanie príkazov a preposielanie TCP/IP a X11 spojení [23].

4.2 RDP

Protokol RDP slúži k vzdialenému zobrazovaniu a ovládaniu aplikácií postavených na operačnom systéme Windows prostredníctvom sieťového pripojenia. Tento protokol je navrhnutý pre podporu rozličných typov sieťových topológií a je založený na rodine protokolov ITU T.120. RDP je multi kanálový protokol, ktorý vytvára osobitné virtuálne kanály pre prenos komunikácie medzi zariadeniami a prezentáciu dát [1].

Server využíva vlastný grafický ovládač na vykresľovanie výstupu aplikácie. Tieto dáta sú rozdelené na pakety a následne preposielané prostredníctvom siete ku klientovi. Na strane klienta sa tieto pakety interpretujú a pomocou API funkcií sa transformujú na výsledné grafické užívateľské rozhranie. Signály z periférnych zariadení (myš, klávesnica) sú preposielané na server, kde sú opäť spracované.

Tento protokol používa RC4 šifru, ktorá šifruje dáta rôznej veľkosti s použitím 56 alebo 128 bitového kľúča. Tak isto sú dostupné rôzne úrovne zabezpečenia celej komunikácie [5]:

- **Low** - Dáta posielané od klienta na server sú šifrované 56 bitovou šifrou. V opačnom smere dáta nie sú šifrované.

- **Client Compatible** - Na tejto úrovni sú šifrované oba smery a na šifrovanie sa využíva maximálna dĺžka kľúča, ktorú podporuje klient.

- **High** - K šifrovaniu dochádza na oboch smeroch s využitím 128 bitového kľúča. Pokiaľ klient nepodporuje tento typ zabezpečenia, nie je mu umožnené pripojiť sa.

- **FIPS Compliant** - Na tejto úrovni je využívaný FIPS (*Federal Information Process Standard*) 140-1, ktorý validuje šifrovacie metódy. Klient, ktorý nepodporuje túto úroveň šifrovania nemôže byť pripojený.

4.3 Telnet

Telnet je aplikačný protokol, poskytujúci obojsmernú a interaktívnu formu komunikácie prostredníctvom 8 bitového textového režimu. Tento protokol zabezpečuje pripojenie na vzdialenú stanicu pomocou terminálového rozhrania. Pôvodným účelom tohto protokolu bolo konfigurovanie vzdialených zariadení [16].

Keďže bol tento protokol navrhnutý začiatkom 70.tych rokov, kedy bol počet používateľov internetu niekoľkonásobne menší ako dnes, nebolo potrebné dbať na bezpečnostné riziká až do takej miery ako v dnešnej dobe. Z dnešného pohľadu má telnet dva zásadné nedostatky:

1. **Šifrovanie** - Celá komunikácia prostredníctvom tohto protokolu je nešifrovaná. Všetky dáta vrátane hesiel sú posielané v jednoduchej textovej forme. Z toho vyplýva, že jednoduchým zachytením komunikácie napríklad programom Wireshark, je možné sa dostať k nešifrovanému obsahu.

2. **Autentizácia** - Väčšina implementácií protokolu telnet, nepodporuje možnosť autentizácie.

V dnešnej dobe je telnet nahradený bezpečnejším protokolom SSH, ktorý bol podrobne rozobratý v kapitole 4.1.

4.4 Typy útokov na uvedené protokoly

Keďže väčšina dnešných služieb na autentizáciu užívateľa používa heslo, stretávame sa s útokmi, ktoré sú primárne zamerané práve na prelomenie hesla. Po uhádnutí získava útočník plnú kontrolu nad vzdialenou stanicou, z ktorej môže napríklad získať citlivé údaje, prípadne ju môže zneužiť na ďalšiu nebezpečnú činnosť. Takisto môže PC pripojiť ku skupine už napadnutých strojov, na ktorých je nainštalovaný škodlivý software, ktorým môže ľahko útočiť na ostatné PC bez prezradenia svojej identity.

Pri tomto spôsobe útočenia sa väčšinou stretávame s dvoma typmi útokov:

- **Útok hrubou silou (*angl. brute-force attack*)**
- **Slovníkový útok (*angl. dictionary attack*)**

4.4.1 Útok hrubou silou

Tento typ útoku je nebezpečný hlavne vtedy ak si užívateľ zvolí príliš krátke heslo, ktoré je napríklad zložené len z písmen alebo číslic. Takto slabé heslo je možné postupným skúšaním rôznych kombinácií znakov zistiť bez väčších problémov. S rastúcim počtom znakov v hesle, rastie aj počet možných kombinácií a tým pádom aj čas potrebný na jeho prelomenie [17].

Možnosťou ako sa vyhnúť prelomeniu hesla pomocou tejto metódy, je zavedenie minimálneho počtu znakov, z ktorých sa musí heslo povinne skladať. Nakoľko väčšina zo služieb už podporuje túto možnosť nie je tento typ útoku až tak často vídaný.

4.4.2 Slovníkový útok

Druhým spôsobom hádania hesiel je slovníkový útok, ktorý sa zakladá na predpoklade, že útočník disponuje slovníkom obsahujúcim bežne používané heslá a s týmito heslami sa následne snaží prihlásiť na server [17]. Samotný útok je často plne automatizovaný pomocou software alebo skriptu, ktorý pracuje nad slovníkom a háda heslá na jeden alebo viacero účtov v závislosti od taktiky útočníka.

S tým je však spojený aj fakt, že takto automatizovaný útok je podstatne jednoduchšie dohľadateľný, pretože všetky spojenia, ktoré nastávajú medzi útočníkom a potenciálnou obeťou majú približne rovnaký charakter. Práve týmito spoločnými charakteristikami sa zaoberá kapitola 5.

Pokiaľ sa jedná o skúseného útočníka, stretávame sa aj so sofistikovanými postupmi ako sa vyhnúť jednotvárnemu vzoru, ktorý takýto útok nadobúda a tým pádom znemožniť samotnú detekciu. Takýmito technikami môže byť napríklad rozloženie celého útoku na dlhšie časové obdobie, prípadne obmedzenie počtu hádaných hesiel za sebou na hodnotu, ktorá sa podobá bežnému pokusu o prihlásenie.

4.5 Existujúce detekčné metódy

Táto podkapitola predstavuje niektoré práce, ktoré sa zaoberajú detekciou útokov a stručne popisuje metódy na ktorých sú tieto práce založené.

4.5.1 Brute-force detektor

Brute Force detektor [14] vznikol ako diplomová práca *Ing. Vaška Pacholíka* a práve jeho úpravou sa zaoberá táto bakalárska práca. Pôvodný detekčný algoritmus je založený na vytváraní detekčných okien, oproti ktorým sa porovnávajú prichádzajúce toky. Detekcia prebieha nad všetkými tokmi daného hosta a nedochádza k rozlišovaniu dvojice host, obeť. Intervaly, ktoré definujú jednotlivé detekčné okná obsahujú statické hodnoty paketov a bajtov.

Detekcia prebieha nad vzorkou dát tvorenou poslednými N tokmi. Veľkosť parametru N tým pádom ovplyvňuje veľkosť tzv. časového okna. Každý prichádzajúci tok je porovnaný voči detekčnému oknu a v prípade, že spadá do tohto intervalu je označený ako podozrivý. Pri vyhodnocovaní sa následne sleduje pomer bežných tokov a tokov, ktoré sú označené ako podozrivé. Pokiaľ tento pomer presiahne hodnotu detekčného prahu, dochádza k reportovaniu hosta.

4.5.2 Flow-based Brute-force Attack Detection

Táto práca [21] sa zaoberá výskumom detekcií pomocou záznamov z jednotlivých tokov a podrobne rozoberá princíp detekcie útočníka, ktorý je postavený na dvoch hlavných metódach. Provou metódou je detekcia založená na signatúre tokov (*angl. signature-based*). Táto signatúra popisuje sieťovú komunikáciu pomocou špecifických hodnôt a štatistík, ktoré sú vypočítané z jednotlivých tokov.

Druhou metódou je detekcia založená na princípe podobností (*angl. similarity-based*). Tento prístup vychádza z prvej spomínanej metódy a jeho základom je vyhľadávanie podobných skupín tokov namiesto ich porovnávania oproti známej signatúre. Metódu je možné uplatniť v automaticky generovaných útokoch, kedy sa môžu jednotlivé zhluky tokov navzájom podobať.

4.5.3 Unveiling Flat Traffic on the Internet

Nasledujúca práca [12] skúma dopad preposielania TCP paketov na tzv. plochý typ sieťovej komunikácie (*angl. Flat newtwork traffic*). Ten je spôsobený opakovaným výskytom podobných vzorov v komunikácii. Tieto vzory vznikajú napríklad pri opakovaných pokusoch o prihlásenie pri brute-force útoku. V rámci sieťovej komunikácie dochádza v určitých prípadoch k preposielaniu TCP paketov a tým pádom ku generovaniu nepravidelných vzorov v sieťovej komunikácii. Z toho vyplýva že, plochý typ sieťovej komunikácie(ktorý môže reprezentovať útok) je ovplyvnený profilom sieťovej komunikácie spôsobenej retransmisiou TCP paketov.

Kapitola 5

Analýza charakteristiky útoku

Samotná detekcia útokov tohto typu je založená na sledovaní jednotlivých tokov a počítaní štatistík, ktoré sú neskôr použité v detekčnom algoritme. Na to aby sme mohli označiť komunikáciu medzi potenciálnym útočníkom a jeho obeťou za podozrivú, je potrebné získať obecnú charakteristiku útoku, oproti ktorej by sme mohli porovnávať zozbierané štatistiky. Tvorbou obecnej charakteristiky útoku sa zaoberá podkapitola 5.1.

Obecná charakteristika (ďalej len signatúra), môže obsahovať napríklad počet paketov alebo bajtov v rámci jedného toku. Táto signatúra slúži ako vzor pre histogramovú metódu detekcie, ktorá tvorí základ celého modulu. Táto metóda bude podrobne rozobratá v podkapitole 6.1. V ďalších podkapitolách si môže čitateľ prezrieť jednotlivé signatúry pre každý typ protokolu.

5.1 Tvorba signatúry útoku

Pred vytvorením signatúry, bolo potrebné najskôr zozbierať súbor dát, ktoré obsahovali záznamy o útoku. Na tento účel som použil pôvodnú implementáciu **Brute force modulu** [14], ktorý bol vyvinutý prostredníctvom diplomovej práce *Ing. Vaška Pacholíka*, po ktorom som následne tento modul prebral.

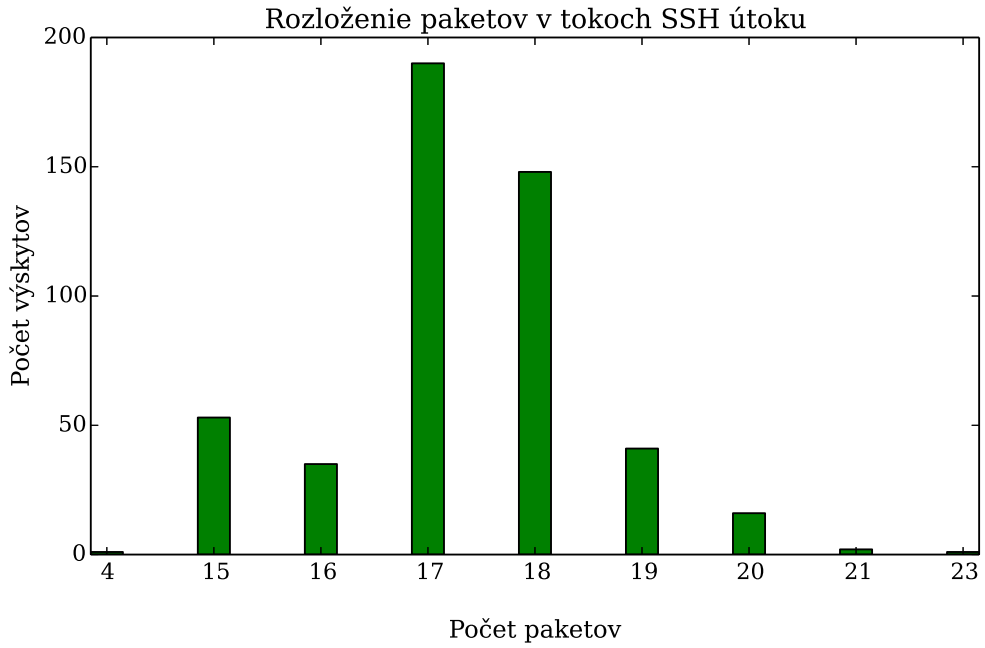
Stručnému porovnaniu rozdielov metód použitých v oboch verziách a ich dopadu na detekciu sa venuje kapitola 6.5.

Výstupom pôvodného modulu je správa vo formáte "WARDEN REPORT", ktorá okrem iného obsahuje IP adresy útočníka, obete a časovú značku kedy bol útok zachytený. Na základe týchto údajov bol pomocou programu **NfSen**, ktorý tvorí grafické webové rozhranie pre program **NfDump**, dohľadaný reportovaný útok priamo v dátach uložených v dátovom úložisku. Formát týchto dát je možné vidieť na obrázku 5.1.

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	Flows
2015-03-30 20:13:57.035	3.434	TCP	Src IP :58968	->	Dst IP :22	.APRS.	0	13	1273	1
2015-03-30 20:14:04.202	3.511	TCP	Src IP :60894	->	Dst IP :22	.APRS.	0	13	1273	1
2015-03-30 20:14:00.745	3.739	TCP	Src IP :59867	->	Dst IP :22	.APRS.	0	14	1313	1
2015-03-30 20:13:53.260	7.007	TCP	Src IP :55552	->	Dst IP :22	....S.	0	4	240	1
2015-03-30 20:14:07.402	3.458	TCP	Src IP :33676	->	Dst IP :22	.APRS.	0	13	1273	1
2015-03-30 20:14:03.696	7.016	TCP	Src IP :44134	->	Dst IP :22	....S.	0	4	240	1
2015-03-30 20:14:14.145	7.008	TCP	Src IP :53632	->	Dst IP :22	....S.	0	4	240	1
2015-03-30 20:14:18.190	3.456	TCP	Src IP :36895	->	Dst IP :22	.APRS.	0	13	1273	1
2015-03-30 20:14:10.709	3.699	TCP	Src IP :34701	->	Dst IP :22	.APRS.	0	13	1273	1
2015-03-30 20:14:14.753	3.642	TCP	Src IP :35776	->	Dst IP :22	.APRS.	0	13	1273	1

Obrázok 5.1: Záznamy zobrazujúce jednotlivé toky útoku na SSH.

Z obrázku 5.1 je možné vidieť istú podobnosť medzi jednotlivými tokmi a to hlavne v počte paketov a bajtov. Tieto podobnosti tvoria základ pre tvorbu unikátnej signatúry útoku. Po spracovaní dát o počte paketov v rámci jedného toku do histogramu je možné vidieť, výslednú charakteristiku zachyteného útoku, ktorá je ukázaná na obrázku 5.2:



Obrázok 5.2: Ukážkový príklad jednej zo vzorových signatúr.

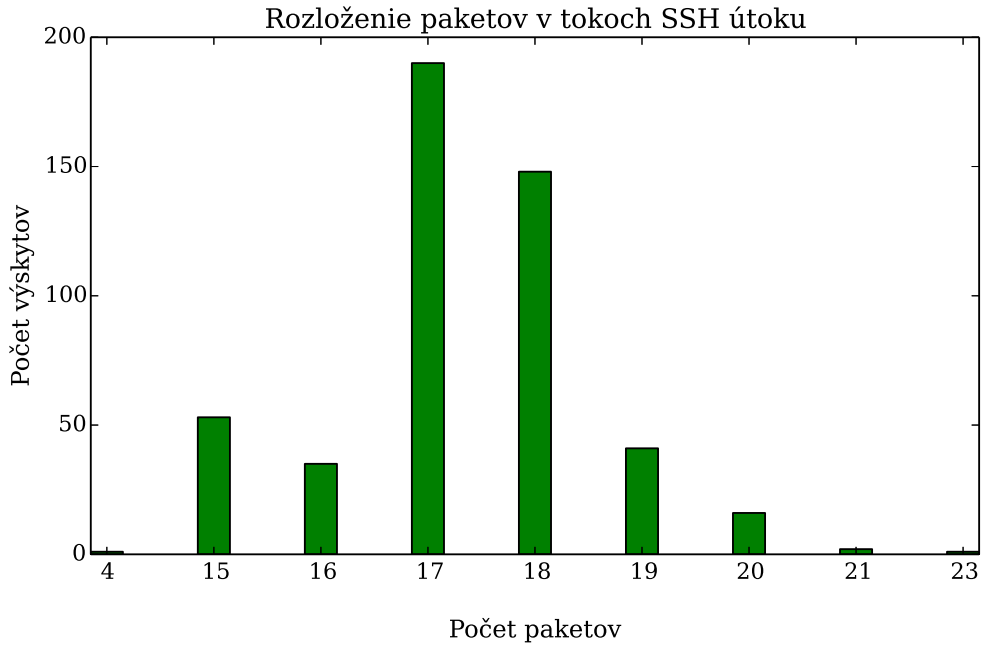
Takto vytvorený histogram slúži ako referenčný vzor, z ktorého sa pomocou matematických operácii získavajú potrebné hodnoty, ktoré slúžia pri samotnej detekcii. Podrobný postup ako aj všetky matematické operácie budú uvedené v kapitole 6.1. Pre porovnanie som vytvoril histogram, ktorý zachytáva legitímnu komunikáciu na protokol SSH. Na obrázku B.1 je možné vidieť rovnomernejšie rozloženie paketov v tokoch legitímnej komunikácie, oproti tomu na obrázku 5.2 je zreteľne viditeľná špička útoku.

5.2 Prehľad jednotlivých signatúr

V tejto práci som sa zameral na protokoly SSH, RDP a Telnet, ktorých signatúry budú ukázané v nasledujúcich podkapitolách. V prípade snahy o detekciu nad inými protokolmi je potrebné opäť vykonať analýzu týchto protokolov. V žiadnom prípade nie je možné vychádzať z hodnôt, ktoré boli namerané pre mnou zvolené protokoly. Pri prvotnom pohľade je možná istá podobnosť medzi jednotlivými signatúrami a z nich odvodenými hodnotami, avšak tento typ podobnosti je čisto náhodný. Väčšina výsledkov je úzko prepojená s dátovou sadou, z ktorej bola vytvorená a vo všetkých prípadoch bolo potrebné opakované zbieranie dát a ich následná transformácia do histogramu.

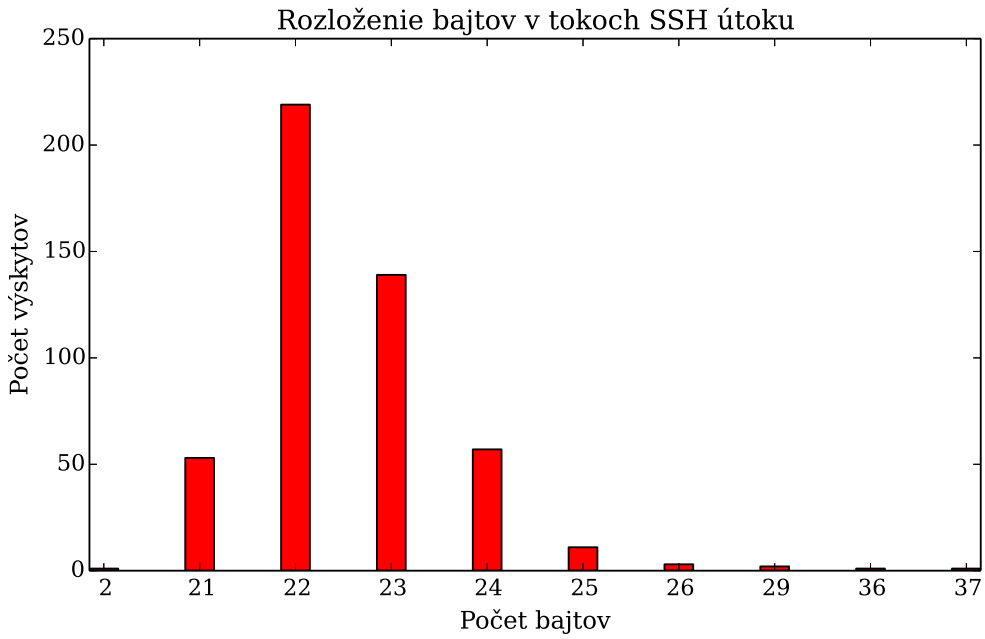
5.2.1 SSH signatúry

Signatúra popisujúca rozloženie paketov v jednotlivých tokoch útoku je zobrazená na obrázku 5.3:



Obrázok 5.3: Signatúra rozloženia peketov v útoku na SSH.

Z obrázku 5.3 je zreteľne vidieť charakteristiku väčšiny útokov na SSH protokol. Na základe tejto signatúry boli následne výpočtami určené prahové hodnoty pre detekciu. Pre zvýšenie presnosti detekcie som vykonal analýzu aj nad počtom bajtov v jednotlivých tokoch, výsledkom ktorej je histogram zobrazený na obrázku 5.4:



Obrázok 5.4: Signatúra rozloženia bajtov v útoku na SSH.

Vo všetkých histogramoch zobrazujúcich rozloženia bajtov v tokoch je počet bajtov vydelený hodnotou 100. Táto úprava bola nutná z dôvodu veľkej rozmanitosti, ktorú hodnoty bajtov v tokoch nadobúdali. Pokiaľ by bol histogram vytvorený z pôvodných hodnôt, nebolo by vôbec možné vykonávať následné výpočty. Ak by sme zobrali tok, ktorý obsahuje napríklad 2247 bajtov, v zobrazenom histograme by sa tento tok zaradil do kategórie 22.

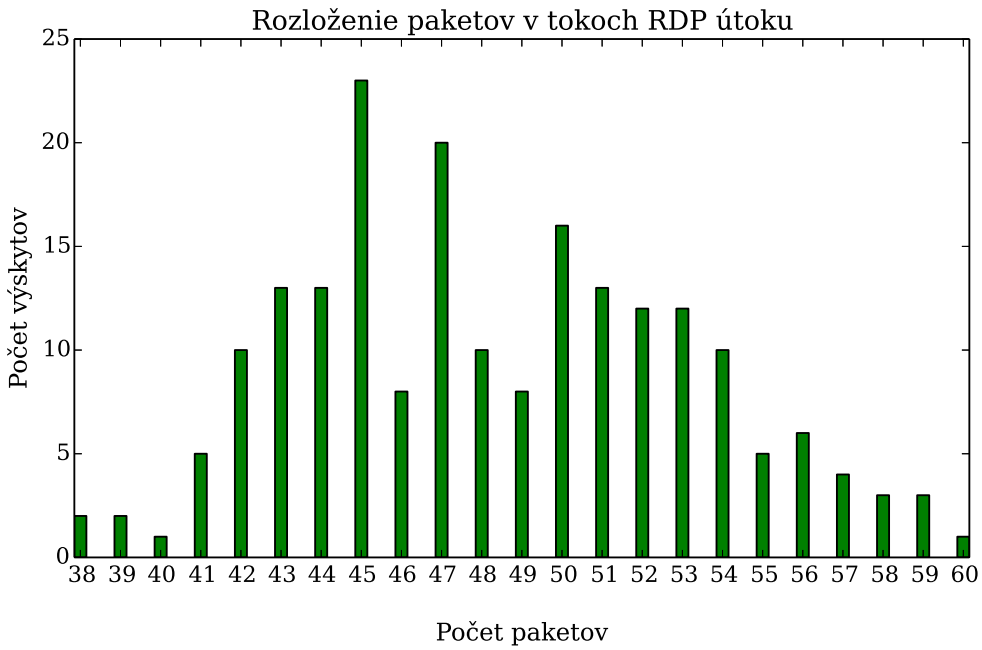
5.2.2 RDP signatúry

V prípade protokolu RDP som pri meraniach zistil, že hodnoty paketov a bajtov v tokoch sa líšia vzhľadom na smer komunikácie. Po rozdelení dát podľa smeru na prichádzajúce (*angl. incoming*) a odchádzajúce (*angl. outgoing*) a následnom zobrazení histogramov, som dospel k záveru, že hodnoty detekčných prahov by boli veľmi nepresné, pokiaľ by boli počítané z dát, v ktorých by boli oba smery zlúčené. Z toho dôvodu sú histogramy počítané pre každý smer osobitne.

V nasledujúcich prípadoch sa pri opakovaných meraniach vyskytovali približne rovnaké počty paketov v rozsahu **30 až 80** pre prichádzajúci smer a **60 až 150** paketov pre odchádzajúci smer v rámci jedného toku.

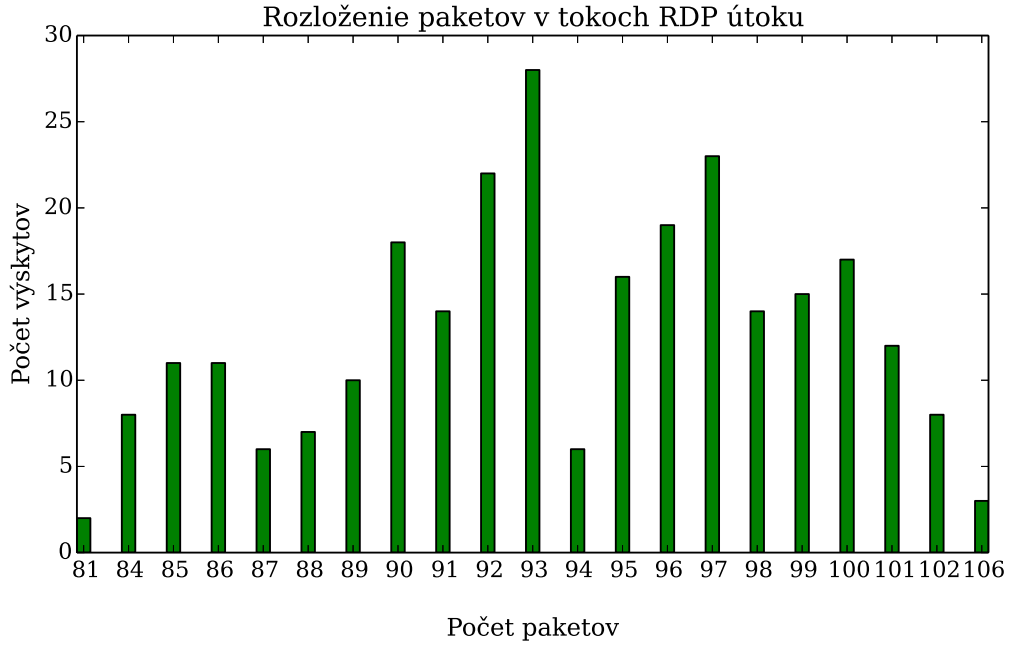
Tieto hodnoty približne odpovedajú hodnotám, ktoré sú uvedené v práci *J. Vykopala* a *M. Vizváryho* s názvom *Flow-based detection of RDP brute-force atacks* [20]. Nedosiahnutie úplnej zhody s hodnotami uvedenými v tejto práci je možné vysvetliť menším počtom meraní prípadne špecifickou vzorkou dát.

- **Histogram rozloženia paketov v tokoch v prichádzajúcom smere:**
  Hneď v prvom histograme je možné vidieť rozdielne zastúpenie paketov v jednotlivých tokoch. V porovnaní s histogramom pre SSH na obrázku 5.3 sú hodnoty mierne rozptýlené, čo bude mať vplyv pri výpočte detekčných prahov.



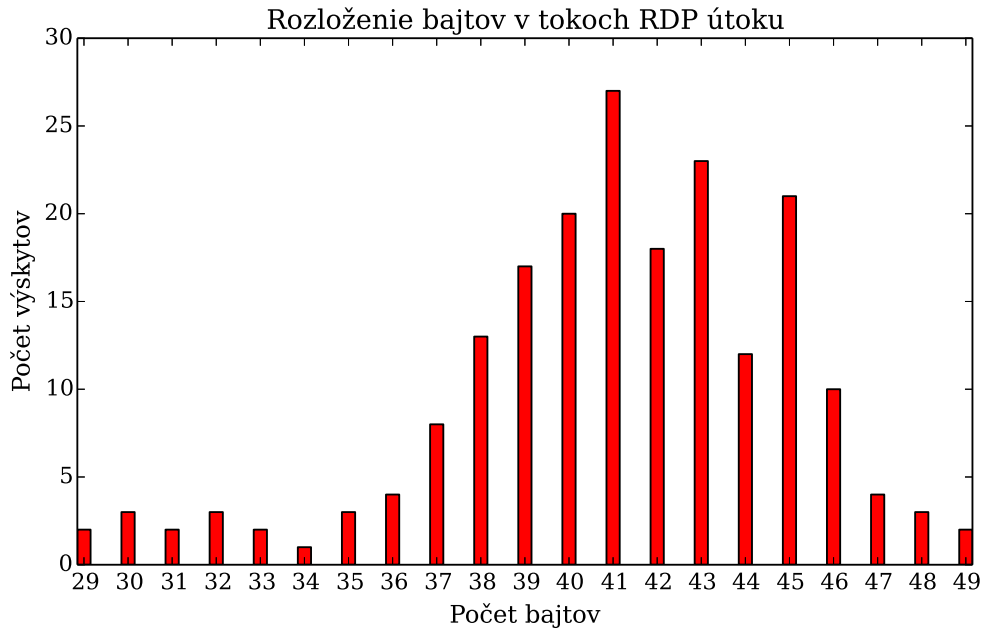
Obrázok 5.5: Signatúra rozloženia paketov v útoku na RDP v prichádzajúcom smere.

- **Histogram rozloženia paketov v tokoch v odchádzajúcom smere:**



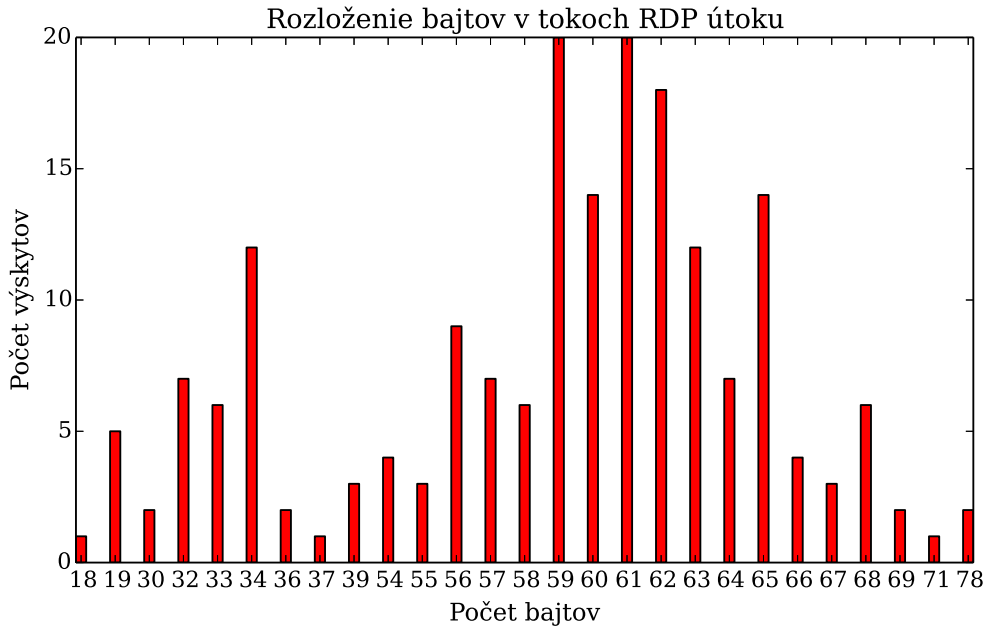
Obrázok 5.6: Signatúra rozloženia paketov v útoku na RDP v odchádzajúcom smere.

- **Histogram rozloženia bajtov v tokoch v prichádzajúcom smere:**



Obrázok 5.7: Signatúra rozloženia bajtov v útoku na RDP v prichádzajúcom smere.

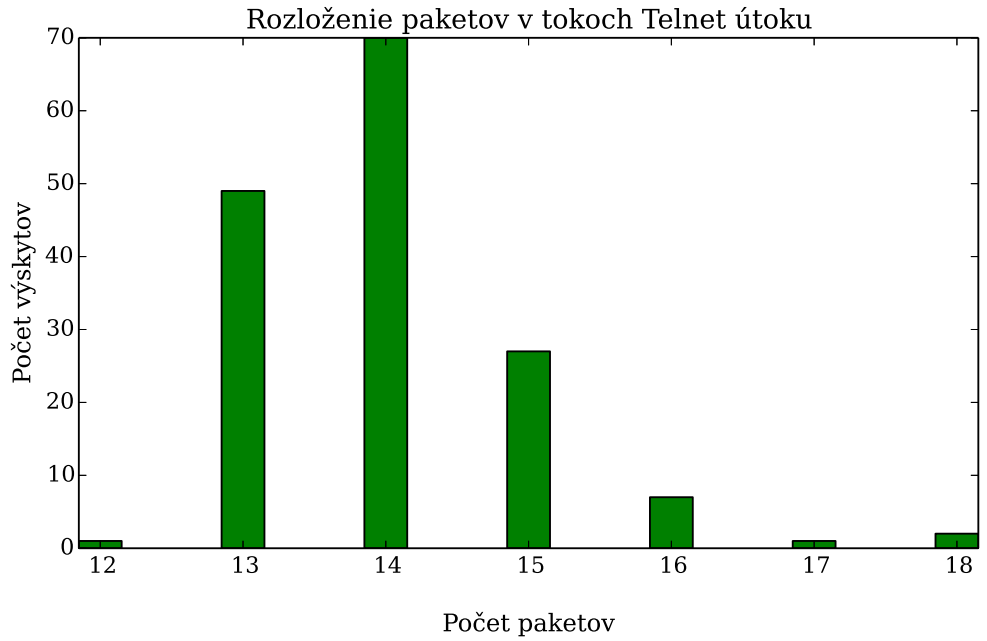
- **Histogram rozloženia bajtov v tokoch v odchádzajúcom smere:**



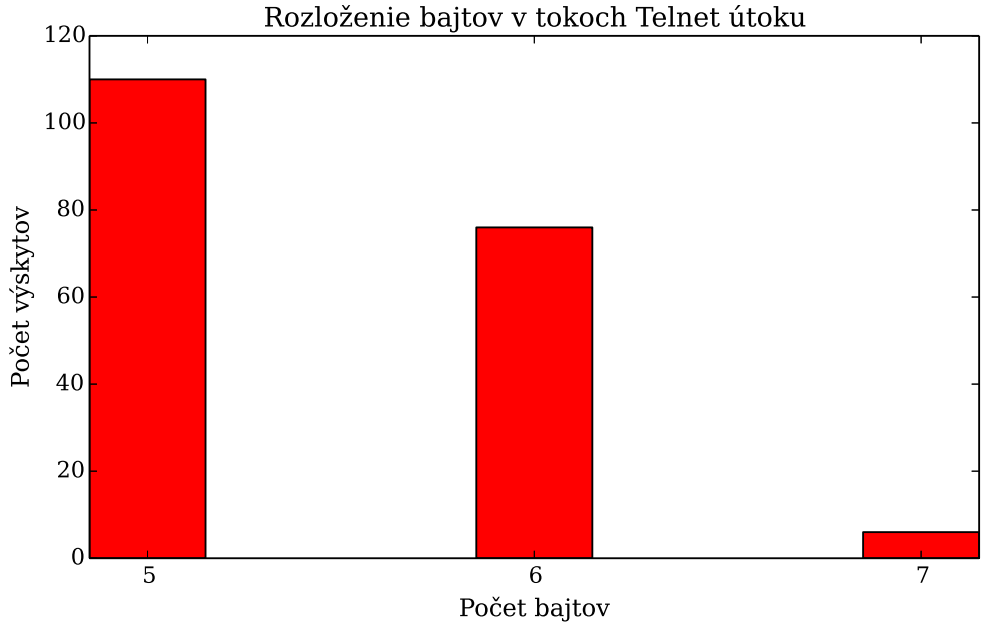
Obrázok 5.8: Signatúra rozloženia bajtov v útoku na RDP v odchádzajúcom smere.

5.2.3 Telnet signatúry

Nasledujúce dva histogramy zobrazujú rozloženia paketov a bajtov pri útoku na protokol Telnet:



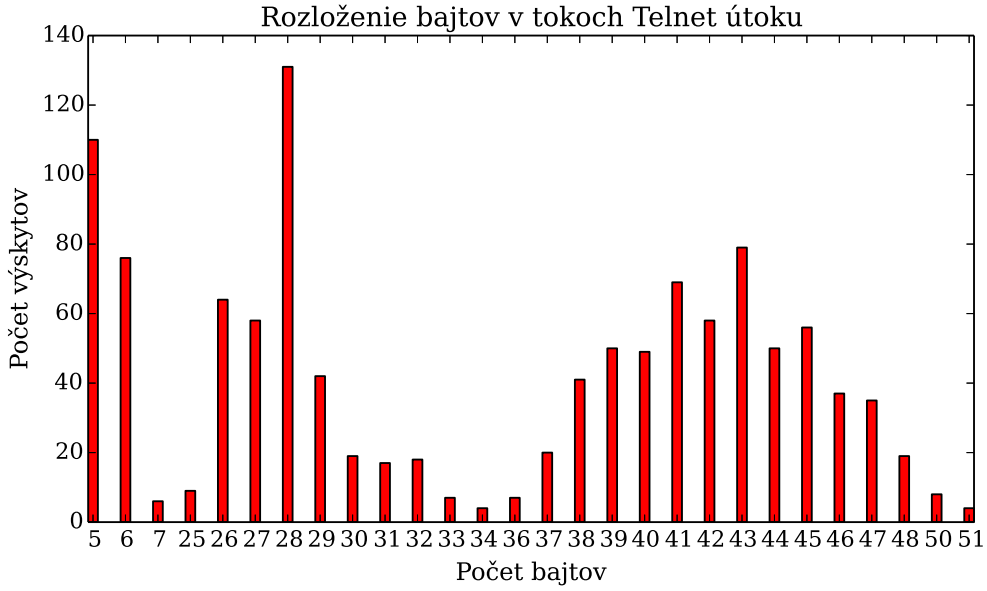
Obrázok 5.9: Signatúra rozloženia paketov v útoku na protokol Telnet.



Obrázok 5.10: Signatúra rozloženia bajtov v útoku na protokol Telnet.

Z histogramu na obrázku 5.10 je možné vidieť dopad nízkeho počtu telnet útokov na sieti. V porovnaní s ostatnými protokolmi bol počet útokov na službu Telnet výrazne nižší. Väčšina z detekovaných útokov bola označená ako skenovanie a teda sa nejednalo o plnohodnotný útok, ktorý by poskytol potrebné dáta na vyhodnotenie.

V rámci dlhodobého zbierania dát potrebných k analýze útoku, sa mi podarilo zachytiť niekoľko silných útokov, ktorých spoločná signatúra rozloženia bajtov je zobrazená na obrázku 5.11.



Obrázok 5.11: Signatúra rozloženia bajtov v útoku na protokol Telnet.

Pre porovnanie vzorka dát v tomto histograme sa pohybuje okolo hranice 2000 tokov, kým doposiaľ uvedené histogramy boli zostavené väčšinou zo vzorky tokov o veľkosti približne 200. Z histogramu je možné vidieť, tri rôzne typy rozloženia bajtov v tokoch. Prvá špička približne odpovedá vzorovej signatúre na obrázku 5.10.

Kapitola 6

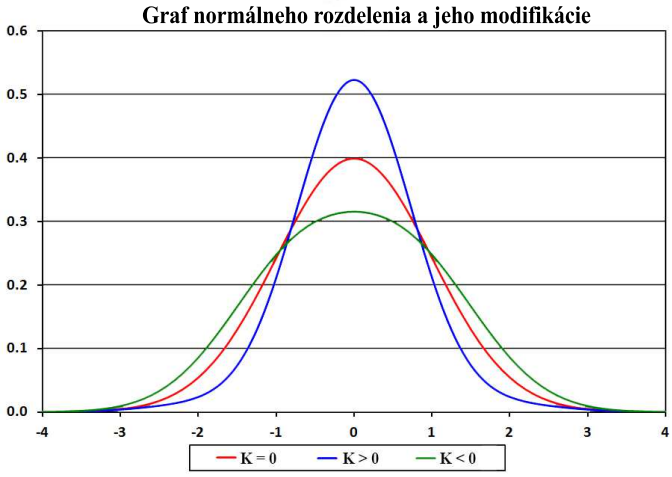
Návrh detekčného algoritmu

Algoritmus, podľa ktorého prebieha detekcia útokov sa skladá z niekoľkých častí, ktorých jadrom je histogramová analýza vykonávaná opakovane po splnení určitých podmienok. Detailný popis celého algoritmu je uvedený v podkapitole 6.1. Ďalej je do algoritmu zakomponovaná optimalizácia, ktorá pomáha pri výpočte koeficientov určených na porovnanie s detekčnými prahmi. O tejto optimalizácií pojednáva podkapitola 6.4. V prípade prekročenia detekčných prahov dochádza k ohodnoteniu vzorky dát a následne sa podľa dosiahnutého ohodnotenia táto vzorka prehlási za pozitívnu alebo negatívnu. Pri pozitívnom výsledku je IP adresa útočníka spolu s ďalšími informáciami o útoku reportovaná do systému Warden.

V závere kapitoly je krátke porovnanie pôvodnej a mnou navrhnutej metódy.

6.1 Histogramová analýza

Základom histogramovej analýzy je výpočet koeficientu špicatosti. Koeficient špicatosti je charakteristika rozdelenia náhodnej veličiny, ktorá charakterizuje mieru špicatosti rozdelenia [15]. Špicatosť rozdelenia sa porovnáva so špicatosťou normálneho rozdelenia, ktorého hodnota je 3. Kladná hodnota koeficientu hovorí, že súbor hodnôt z ktorých bol koeficient vypočítaný je špicatejší než normálne rozdelenie. Príklad rozdelení ilustruje obrázok 6.1:



Obrázok 6.1: Typy normálneho rozdelenia podľa koeficientu špicatosti [8].

Koeficient špicatosti sa vypočíta pomocou nasledovného vzorca:

$$K = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{s^4} - 3$$

Kde s je smerodatná odchýlka a $(x_i - \bar{x})$ je odchýlka od strednej hodnoty.

V kapitole 5 je možné vidieť, že útoky na jednotlivé protokoly majú vo väčšine prípadov približne rovnakú charakteristiku a teda je možné uplatňovať detekčné algoritmy, ktoré sú založené na sledovaní určitého vzoru chovania. Detekčné okno je vytvorené na základe statických hodnôt, pričom najčastejšími parametrami tohto okna sú hodnoty paketov a bajtov v jednom toku.

Problém však nastáva v prípade, že útočník zámerne upraví komunikáciu tak, aby hodnoty patekov alebo bajtov v jednotlivých tokoch nespadali do vzorového okna. Tým pádom môže byť detekcia útoku neúspešná.

Z toho vyplýva, že by bolo potrebné pri tvorbe detekčného okna vychádzať z dynamického intervalu hodnôt paketov a bajtov. Preto je v tejto metóde detekčné okno reprezentované koeficientom špicatosti spolu s ďalšími výpočtami, ktoré budú uvedené v nasledujúcej podkapitole. Po aplikovaní tejto zmeny, by mala byť detekčná metóda schopná detekovať spomínané typy útokov.

6.2 Spracovanie prichádzajúcich dát

Modul je navrhnutý tak, aby bolo možné detekovať útoky nad ľubovoľným protokolom. Prichádzajúce toky sú na začiatku porovnávané s aktuálnou konfiguráciou modulu, ktorá okrem iného obsahuje zoznam protokolov, nad ktorými bude aktuálne prebiehať detekcia. Pokiaľ sa číslo protokolu prichádzajúceho toku nezhoduje s jedným z protokolov uvedených v konfiguračnom súbore, bude tento tok zahodený. Táto úprava bola nutná z dôvodu zníženia pamäťovej náročnosti modulu.

Toky následne prechádzajú filtrom, ktorý má za úlohu odhaliť skenovanie týchto protokolov. Filtrovacie pravidlá zobrazuje tabuľka 6.1:

Číslo pravidla	**TCP príznaky**	**Počet paketov**
1.	SYN	-
2.	SYN + RST	2
3.	SYN + ACK	1
4.	SYN + ACK	2
5.	RST + ACK	1
6.	ACK	1

Tabuľka 6.1: Zoznam filtrovacích pravidiel

Samotné skenovanie modul ignoruje nakoľko k nemu dochádza vo väčšine prípadov a možným zahrnutím do funkcionality modulu by mohlo dôjsť k enormnému zvýšeniu počtu reportovaných útočníkov. Keďže tomuto typu útoku obvykle predchádza fáza skenovania, dochádzalo by tým pádom k dvojitému reportovaniu tých istých IP adries.

6.3 Detekčný algoritmus

Na základe IP adresy potenciálneho útočníka sa vyhľadá záznam, v ktorom sú uložené všetky histogramy útočníka a jeho obetí pre oba smery komunikácie. Podľa IP adresy obete a smeru komunikácie sa vyhľadá záznam pre túto dvojicu. Tento záznam obsahuje dva histogramy. Prvý histogram je tvorený hodnotami paketov a druhý histogram je tvorený hodnotami bajtov z prichádzajúcich tokov. Po nájdení tohto záznamu sú do oboch histogramov pridané konkrétne hodnoty paketov a bajtov z prichádzajúceho toku. Táto časť je zobrazená na obrázku 6.3 ako funkcia *storeFlows()*. Nasledujúci obrázok popisuje formou pseudo-kódu hlavnú časť detekčného algoritmu, ktorá prebieha nad paketovým aj bajtovým histogramom:

```
storeFlows()
if (countOfStoredFlows > 200) then
   K = countHistogramKurtosis()
   if (K > 0) then
      attackRatio = kurtosisRatio
   end if
   //compare standard deviation of histogram with detection thresholds
   if (SD <= bottomThreshold) then
      attackRatio += deviationBottomRatio
   else if (bottomThreshold < SD && SD <= topThreshold) then
      attackRatio += deviationTopRatio
   else
      oldK = K
      recomputeHistogram()
      K = countHistogramKurtosis()
      if (oldK < 0.0 && K > 0.0) then
         attackRatio += recKurtosisRatio
      end if

      if (SD <= recBottomThreshold) then
         attackRatio += recDeviationBottomRatio
      else if (recBottomThreshold < SD && SD <= recTopThreshold) then
         attackRatio += recDeviationTopRatio
      end if
   end if
end if
```

Obrázok 6.2: Pseudo-kód detekčného algoritmu, K = koeficient špičatosti, SD = smerodatná odchýlka histogramu

Pokiaľ počet tokov zaradených do histogramu danej dvojice presiahne hodnotu 200 dochádza ku spusteniu detekčného algoritmu, nad aktuálnym paketovým a bajtovým histogramom pre oba smery komunikácie. Funkcia *countHistogramKurtosis()* reprezentuje výpočet koeficientu špicatosti z aktuálneho histogramu. Princíp tohto výpočtu bol vysvetlený

v podkapitole 6.1. Na základe hodnoty koeficientu špicatosti(K) sa priradí spracovanému histogramu váha. Táto váha je reprezentovaná hodnotou *kurtosisRatio*. V prípade, že koeficient špicatosti(K) nadobudne kladnú hodnotu, histogram bude ohodnotený váhou, ktorej hodnota je uvedená v tabuľke 7.3.

Následne dochádza k sekundárnej časti algoritmu, v ktorej sa porovnávajú detekčné prahy s hodnotou smerodatnej odchýlky vypočítanej z prvkov aktuálne analyzovaného histogramu. Hodnota detekčných prahov, je veľkosť smerodatnej odchýlky, ktorá bola vypočítaná z referenčnej signatúry útoku pre daný protokol. Na základe experimentov boli od tejto hodnoty odvodené parametre *bottomThreshold* a *topThreshold*. Pre protokol SSH sú definované hodnoty týchto parametrov len pre detekciu nad paketovým histogramom, nakoľko som pri analýze charakteristiky útoku zistil, že táto charakteristika vypočítaná z paketového histogramu je dostačujúca na detekciu útoku nad týmto protokolom. Naopak v prípade protokolu RDP, boli charakteristiky útoku značne rozdielne pre paketovú a bajtovú časť a tak isto pre oba smery komunikácie. Tieto rozdiely je možné vidieť na jednotlivých histogramoch v podkapitole 5.2.2. Hodnoty detekčných prahov pre protokol SSH a Telnet sú uvedené v tabuľke 7.4 a pre protokol RDP v tabuľkách 7.5 a 7.6. Podľa výsledku porovnania týchto dvoch hodnôt je analyzovanému histogramu opäť pripočítaná váha, ktorej hodnoty sú uvedené v tabuľke 7.2. Priraďovanie váhy histogramu prebieha nasledovným spôsobom:

1. Ak je hodnota smerodatnej odchýlky histogramu(SD) menšia ako dolná hranica detekčného prahu(*bottomThreshold*), histogramu sa priradí váha odpovedajúca dolnej hranici detekčného prahu(*deviationBottomRatio*).

2. Ak je hodnota smerodatnej odchýlky histogramu v rozmedzí dolnej hranice a hornej hranice detekčného prahu(*topThreshold*), histogramu sa priradí váha odpovedajúca tomuto intervalu(*deviationTopRatio*).

3. Ak je hodnota smerodatnej odchýlky histogramu väčšia ako horná hranica detekčného prahu, dochádza k prepočítaniu histogramu tzv. optimalizačnej časti, rozobratej v kapitole 6.4.

Po prebehnutí detekčného algoritmu nad dvojicou histogramov, dochádza k záverečnej fáze, kedy sa hodnoty premennej *attackRatio* z oboch histogramov sčítavajú v určitom pomere. Tento pomer udáva prínos oboch čiastkových ohodnotení histogramov do výsledného koeficientu útoku. V niektorých prípadoch je možné týmto pomerom uprednostňovať jeden z histogramov a tým pádom zvýšiť jeho dopad na celkový výsledok detekcie. Na základe experimentov bol pomer paketového a bajtového histogramu u všetkých protokolov nastavený na pomer 65%:35% v prospech paketového histogramu.

Výsledný koeficient útoku(pozostávajúci z dvoch čiastkových koeficientov v určitom pomere) je porovnaný s detekčným prahom, ktorý vyjadruje do akej miery sú analyzované histogramy podozrivé. Hodnota tohto prahu je v základnom nastavení modulu na úrovni 85% a je možné ju manuálne špecifikovať pre každý analyzovaný protokol. V prípade že komunikácia(ktorú predstavujú analyzované histogramy) bola označená ako útok dochádza k odoslaniu reportovacej správy. Táto správa obsahuje IP adresu útočníka a obete ako aj informáciu o intenzite útoku.

V rámci konfigurácie modulu je možné nastaviť časový interval počas ktorého sa budú vzniknuté reporty agregovať. Táto doba sa vzťahuje vždy na konkrétnu dvojicu útočníka a obete, to znamená že ak dôjde k detekovaniu útoku pred uplynutím tohto intervalu, dochádza k agregácii reportu. Agregácia v praxi predstavuje zvýšenie počtu čiastkových

detekcii o jedna. Výsledný report potom obsahuje informáciu o intenzite útoku, ktorá sa rovná počtu čiastkových detekcii, ku ktorým došlo počas stanoveného časového intervalu.

Všetky hodnoty priraďovaných váh, detekčných prahov a iných koeficientov potrebných pri výpočtoch je možné dynamicky meniť prostredníctvom konfiguračného súboru. Presné hodnoty jednotlivých detekčných prahov ako aj priraďovaných váh sú uvedené v podkapitole 7.2.

6.4 Optimalizácia histogramovej analýzy

Táto optimalizácia bola navrhnutá počas skúmania jednotlivých histogramov, kedy pri niektorých špecifických útokoch dosahoval koeficient špicatosti hodnoty, ktoré boli len niekoľko stotín pod nulovou hodnotou. Po spätnom dohľadaní tejto komunikácie som zistil, že vo väčšine prípadov išlo o útok. Keďže pôvodná verzia modulu nepočítala s touto optimalizáciou a v plnom rozsahu závisela len na výpočte koeficientu špicatosti, útoky s týmto typom signatúry by nebolo možné detekovať.

Z toho dôvodu som sa rozhodol zahrnúť do detekčného algoritmu časť, v ktorej dochádza k opätovnému vyhodnoteniu histogramu, avšak vzorka dát sa pri opätovnom výpočte nemusí vždy nutne zhodovať s pôvodnou vzorkou. Pred druhým vyhodnotením na najskôr nájde prvok M_1 s najčastejším výskytom paketov alebo bajtov v závislosti od typu histogramu.

Od tohto prvku sa v ďalšom kroku vytvoria hranice intervalu $< I_d \; ; \; I_h >$. Dolná hranica I_d sa získa odčítaním smerodatnej odchýlky od prvku M_1. Horná hranica I_h sa získa pripočítaním smerodatnej odchýlky k prvku M_1. Toky, ktoré nepatria do tohto intervalu sú odstránené z pôvodnej vzorky dát. Z toho vyplýva, že druhé vyhodnotenie prebieha len nad orezanou časťou histogramu.

V prípade, že nová hodnota koeficientu špicatosti nadobudne kladnú hodnotu a pôvodná hodnota pred optimalizáciou bola záporná, priradí sa histogramu automaticky váha, ktorú reprezentuje parameter (*recKurtosisRatio*), ktorého hodnota je uvedená v tabuľke 7.3. Následne sa postupuje v analýze rovnakým spôsobom ako v prvom vyhodnocovaní. Postup je možné vidieť na obrázku 6.3 (riadky 13 až 18). Jediný rozdiel je v hodnotách detekčných prahov, ktoré sú prispôsobené pre tento typ výpočtu a boli získané z rovnako upravených referenčných signatúr útokov. Tak isto sú upravené aj hodnoty priraďovaných váh. Konkrétne hodnoty sú opäť uvedené v tabuľkách 7.4, 7.5 a 7.6.

6.5 Porovnanie s predchádzajúcim detekčným algoritmom

Jednou z najväčších slabín detekcie pomocou IP tokov je možnosť tzv. naťahovania tokov (*angl. Flow Stretching*) [21]. Touto metódou je možné pozmeniť očakávanú signatúru útoku, napríklad zvýšením počtu paketov alebo bajtov. Ak by sa útočník snažil takýmto spôsobom zamaskovať svoj útok a pozmenil by signatúru útoku natoľko, že by neodpovedala fixnej signatúre použitej v pôvodnej verzií modulu, mohlo by v tomto prípade dôjsť k oklamaniu detekčného algoritmu.

Nová verzia modulu obsahuje detekčný algoritmus, ktorý je schopný detekovať útoky bez porovnávania s fixnou signatúrou. Keďže histogramová analýza prebieha nad aktuálnymi hodnotami paketov a bajtov, ich nárast alebo pokles v jednotlivých tokoch nebude mať žiadny dopad na výpočet koeficientu špicatosti a teda ani na celkový výsledok histogramovej analýzy.

Čiastočný dopad je možné zaznamenať v prípade druhej časti analýzy, nakoľko hodnoty detekčných prahov boli získané práve po analýze spoločnej signatúry útoku, ktorú ide v tomto prípade považovať za fixnú. Práve z toho dôvodu sa detekčný algoritmus skladá z niekoľkých častí, ktoré reagujú na rôzne druhy útokov rozdielne. Tým pádom existuje možnosť, že ak jedna časť detekcie pre takto špecifický typ útoku zlyhá, druhá časť môže byť stále schopná tento útok detekovať.

Kapitola 7

Testovanie a výsledky detekcie nad reálnymi dátami

Testovanie modulu po úprave detekčného algoritmu bolo rozdelené na niekoľko častí. V rámci každej časti bol modul spúšťaný s rôznou konfiguráciou, ktorá sa odlišovala v hodnotách detekčných prahov. Po vykonaní niekoľkých testov bolo možné na základe dosiahnutých výsledkov porovnať počet detekcii nad cieľovými protokolmi. Tieto porovnania slúžili na nájdenie optimálnej konfigurácie, s ktorou by mal byť modul v budúcnosti spustený.

7.1 Vzorka dát

Keďže mojím cieľom bolo porovnanie rôznych druhov konfigurácii, spustením modulu nad dátami priamo zo siete by nebolo možné porovnať dosiahnuté výsledky jednotlivých konfigurácii objektívne. V prípade, že by počas jedného merania došlo k určitému druhu anomálie napríklad neobvykle silnému útoku a dáta v rámci ďalšieho merania by sa príliš neodlišovali od bežného priemeru, výsledky prvého merania by boli značne ovplyvnené a mohli by podávať skresľujúci obraz v porovnaní s ostatnými meraniami.

Z toho dôvodu som všetky merania vykonával nad rovnakou vzorkou dát. Táto vzorka predstavovala objem dát zachytených v rámci jedného dňa na jednej linke. Konkrétne sa jedná o dáta z linky **Telia** v časovom rozmedzí 17.3.2015 23:00 až 18.3.2015 23:00.

Tabuľka 7.1 zobrazuje niektoré základné informácie popisujúce dátovú sadu nad ktorou prebiehalo testovanie modulu.

	Počet tokov	Počet bajtov	Počet paketov	Bps	Pps
Ssh	49 980 418	81.48 GB	431 405 532	8 074 090	4 976
Rdp	176 045 015	74.53 GB	354 718 285	7 393 136	4 095
Telnet	34 165 320	3.16 GB	59 675 613	314 808	690
Celá vzorka	1 798 783 377	18848.53 GB	27 913 904 843	1 866 644 831	321 821

Tabuľka 7.1: Súhrnné informácie o dátovej sade. Bps - Bytes per second, Pps - Packets per sercond.

7.2 Hodnoty spoločných váhových ohodnotení

Tabuľka 7.2 zobrazuje hodnoty spoločných váhových ohodnotení, ktoré sú priraďované v porovnaniach so štandardnou odchýlkou:

Názov váhového ohodnotenia	Hodnota
deviationBottomRatio	0.50
deviationTopRatio	0.30
recDeviationBottomRatio	0.40
recDeviationTopRatio	0.20

Tabuľka 7.2: Prehľad váhových ohodnotení.

Váhové ohodnotenie, ktoré je viazané na hodnotu koeficientu špicatosti je špecifikované parametrami *kurtosisRatio* a *recKurtosisRatio*. Hodnoty týchto dvoch parametrov sú spoločné pre všetky protokoly.

Názov váhového ohodnotenia	Hodnota
kurtosisRatio	0.50
recKurtosisRatio	0.30

Tabuľka 7.3: Váhové ohodnotenie priraďované na základe hodnoty koeficientu špicatosti.

7.3 Hodnoty detekčných prahov pre SSH a Telnet

Tabuľka 7.4 obsahuje hodnoty detekčných prahov, ktoré sa využívajú pri detekčného algoritmu pre protokol SSH a Telnet v rámci detekcii nad paketovým histogramom v prichádzajúcom smere komunikácie.

Protokol SSH		Protokol Telnet	
Názov detekčného prahu	Hodnota prahu	Názov detekčného prahu	Hodnota prahu
bottomThreshold	1.50	bottomThreshold	5.59
topThreshold	2.50	topThreshold	7.59
recBottomThreshold	1.50	recBottomThreshold	0.71
recTopThreshold	3.00	recTopThreshold	2.71

Tabuľka 7.4: Hodnoty detekčných prahov pre protokoly SSH a Telnet nad paketovým histogramom.

7.4 Hodnoty detekčných prahov pre protokol RDP

Pri analýze signatúr útokov nad protokolom RDP som narazil na značné rozdiely v hodnotách paketov a bajtov pre prichádzajúci a odchádzajúci smer komunikácie. Tieto rozdiely je možné vidieť na histogramoch, uvedených v podkapitole 5.2.2. Z toho dôvodu bolo nutné vytvoriť dve sady detekčných prahov, ktoré vychádzajú z príslušných charakteristík útokov. V tomto prípade sú uvedené hodnoty pre paketový a bajtový histogram. Rozdiely medzi jednotlivými hodnotami je možné vidieť v tabuľkách 7.5 a 7.6.

Prichádzajúci smer		Odchádzajúci smer	
Názov detekčného prahu	Hodnota prahu	Názov detekčného prahu	Hodnota prahu
bottomThreshold	11.13	bottomThreshold	30.94
topThreshold	12.13	topThreshold	32.94
recBottomThreshold	7.85	recBottomThreshold	3.80
recTopThreshold	9.85	recTopThreshold	5.80

Tabuľka 7.5: Hodnoty detekčných prahov paketového histogramu pre oba smery.

Prichádzajúci smer		Odchádzajúci smer	
Názov detekčného prahu	Hodnota prahu	Názov detekčného prahu	Hodnota prahu
bottomThreshold	1.90	bottomThreshold	78.57
topThreshold	3.90	topThreshold	80.57
recBottomThreshold	1.01	recBottomThreshold	17.39
recTopThreshold	3.01	recTopThreshold	19.39

Tabuľka 7.6: Hodnoty detekčných prahov bajtového histogramu pre oba smery.

7.5 Porovnanie rôznych konfigurácii

Prvá skupina porovnaní sa týka hodnoty *attackRatio*, ktorá predstavuje prah s ktorým je porovnávaný výsledok histogramovej metódy. Pokiaľ výsledok presiahne hranicu tohto prahu je automaticky označený za útok. Prehľad dosiahnutých výsledkov je zobrazený v tabuľke 7.7:

Číslo testu	Attack ratio	Počet detekovaných útokov		
		SSH	RDP	TELNET
1.	45%	20957	6219	25
2.	65%	15950	5702	25
3.	85%	10483	4245	20

Tabuľka 7.7: Počet detekcii v závislosti od hodnoty prahu *attackRatio*. Parameter *hostTimeout* nastavený na hodnotu 3600 sekúnd.

Pri pohľade na tabuľku 7.7 je možné vidieť výrazný rozdiel medzi počtom detekovaných útokov na protokol SSH v rámci prvého a tretieho testu. Konfigurácia z testu č.1 je nepoužiteľná v prípade reálneho nasadenia pretože s najväčšou pravdepodobnosťou obsahuje aj falošne pozitívne útoky. To znamená, že charakteristika bežnej komunikácie sa do určitej miery podobá charakteristike útoku. V tom prípade by sa medzi reportmi mohla objavovať aj úplne legitímna komunikácia. Za povšimnutie stojí nízky počet reportovaných útokov na protokol Telnet. Po preskúmaní prichádzajúcich dát som zistil, že vo väčšine prípadov ide o sken. Nastavenie hodnoty tohto parametru je potrebné zvoliť rozumne, v závislosti na objeme dát v sieti nad ktorou bude modul nasadený.

Tabuľka 7.5 ilustruje rozdiel v počte detekcii v závislosti na veľkosti časového okna, ktoré udáva dobu, počas ktorej sú záznamy pre daného hosta uložené v pamäti. Parameter *hostTimeout* je jeden z troch parametrov, prostredníctvom ktorých sa dá regulovať množstvo alokovanej operačnej pamäti. Z tejto tabuľky je vidieť veľmi malý dopad hodnoty tohto parametru na celkový počet detekcii. V tomto prípade ide však o zníženie pamäťovej náročnosti, ktorá je podstatná z hľadiska reálneho využitia modulu. Počas skúšobného

nasadenia modulu nad reálnymi dátami po dobu jeden týždeň, sa veľkosť spotrebovanej pamäte ustálila po približne troch dňoch na hodnote 1350 MB.

Číslo testu	Host timeout [sekunda]	Počet detekovaných útokov		
		SSH	RDP	TELNET
1.	1800	15950	5729	25
2.	3600	15950	5702	25
3.	5400	15910	5640	25

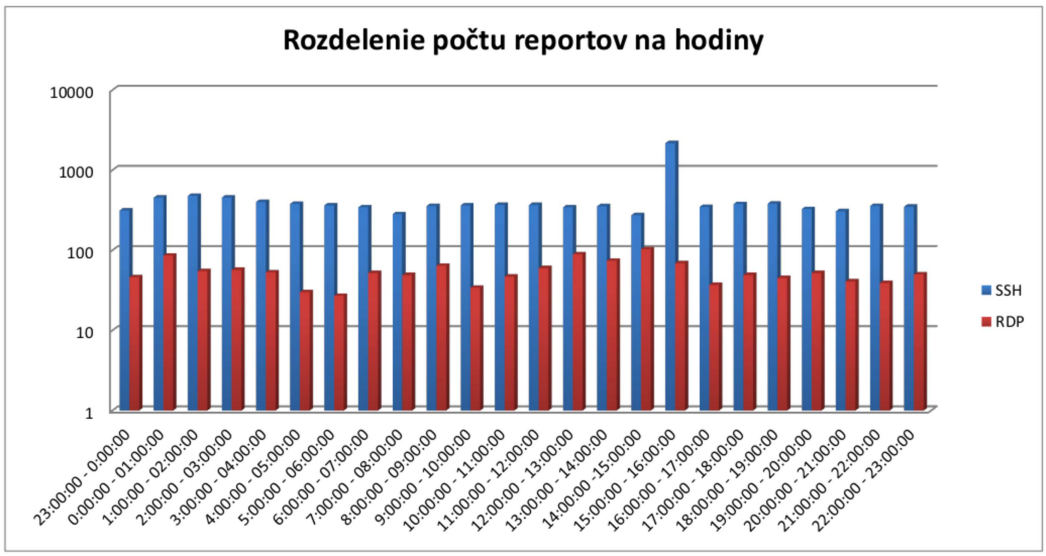
Tabuľka 7.8: Detekované útoky v závislosti od hodnoty parametru *hostTimeout*. Parameter *attackRatio* nastavený na hodnotu 65%.

Posledná tabuľka zobrazuje počet detekovaných útokov pre protokol RDP v závislosti na použitých hodnotách detekčných prahov. V teste č.1 boli použité hodnoty detekčných prahov pre paketovú časť histogramu uvedené v tabuľke 7.5 a bajtovú časť histogramu uvedené v tabuľke 7.6. Oproti tomu v teste č.2 nebola použitá časť detekčného algoritmu založená na bajtovom histograme. V oboch testoch bol parameter *attackRatio* nastavený na hodnotu 65% a parameter *hostTimeout* na hodnotu 3600 sekúnd.

Číslo testu	Počet detekovaných útokov na RDP
1.	5702
2.	4276

Tabuľka 7.9: Porovnanie odlišných konfigurácii pre protokol RDP.

7.6 Denný profil reportov



Obrázok 7.1: Počet útokov za jednu hodinu počas celého dňa.

V rámci vykonávania testov som zostavil graf, ktorý ilustruje približné rozloženie počtu reportovaných útokov počas celého skúmaného dňa. Mierka osi Y je v logaritmickom tvare.

V grafe 7.1 je možné vidieť v časovom rozmedzí od 15:00:00 do 16:00:00 pomerne silný útok na SSH protokol. Počet reportov v tomto časovom okne bol 2174, z toho až v 1885 prípadoch bol reportovaný ten istý útočník. Útok na každú z jeho obetí bol reportovaný 2 krát s približne 20 minútovým rozstupom. Po manuálnom dohľadaní dát som zistil, že útočník posielal priemerne 10 hesiel za jednu minútu. Jeden tok teda obsahoval 12 paketov, čo odpovedá bežnej signatúre útoku na SSH.

7.7 Porovnanie s pôvodnou verziou modulu

Nad rovnakou vzorkou dát bola spustená aj pôvodná verzia tohto modulu. Výsledky boli porovnané s hodnotami testu č.2 z tabuľky 7.7 pri nastavení parametru *attackRatio* na hodnotu 65% a parametru *hostTimeout* na hodnotu 3600 sekúnd. Porovnanie dosiahnutých výsledkov zobrazuje tabuľka 7.10:

Typ	Počet detekovaných útokov		
	SSH	RDP	TELNET
Pôvodná verzia	149440	71150	369
Nová verzia	15950	5702	25

Tabuľka 7.10: Porovnanie pôvodnej a novej metódy detekcie.

V porovnaní s pôvodnou verziou je vidieť značný pokles počtu reportovaných útokov. Tento rozdiel môže byť spôsobený tým, že nová metóda potrebuje na spustenie detekcie najmenej 200 tokov pre danú dvojicu. V prípade, že počet nazbieraných tokov nepresiahne túto hranicu, zozbierané toky nebudú vyhodnotené. Na druhú stranu v pôvodnej detekčnej metóde dochádzalo k vyhodnoteniu už po niekoľkých zaznamenaných tokoch, čo mohlo v konečnom dôsledku značne ovplyvňovať výsledky detekcie a tým pádom umelo navyšovať množstvo reportovaných útokov.

7.8 Porovnanie s modulom HostStats

Framework Nemea obsahuje modul HostStats, ktorý je čiastočne schopný detekovať brute force útoky. Detekcia je v tomto module založená na pod profile pre protokol SSH, ktorý porovnáva priemerný počet paketov a bajtov v rámci jedného toku a na základe nastavených prahov označuje tieto toky za podozrivé. Z tohto pohľadu sa dá povedať, že ide o rovnaký spôsob detekcie ako v pôvodnej verzií brute force modulu.

V tabuľke 7.11 je zobrazené porovnanie počtu reportov oboch modulov. V prípade brute force modulu boli opäť použité hodnoty z testu č.2, ktorého konfigurácia je zobrazená v tabuľke 7.7.

Modul	Počet detekovaných útokov
HostStats	8634
Brute force	15950

Tabuľka 7.11: Porovnanie modulu HostStats s novou metódou detekcie.

Počet reportov modulu HostStats tvorí približne polovicu z celkového počtu reportovaných útokov Brute force modulom. Tento rozdiel môže byť spôsobený tým, že modul

HostStats nie je primárne určený k detekcii brute force útokov ale poskytuje možnosť detekcie aj iných bezpečnostných anomálii. Hodnoty jednotlivých prahov na základe ktorých sa označujú toky za podozrivé sú fixne stanovené a teda v prípade výskytu útokov s odlišnými signatúrami nemusí dochádzať k ich označovaniu.

Kapitola 8

Záver

V úvodnej kapitole boli vysvetlené základné princípy, s ktorými sa môžeme stretnúť pri monitorovaní sietí. Boli objasnené pojmy ako IP tok či protokol NetFlow. Spolu s nimi bola rozobraná architektúra tohto protokolu a jeho najpoužívanejšie verzie. Nasledujúca kapitola pojednávala o frameworku Nemea, ktorý tvoril základ pre vývoj a úpravu tohto modulu. Taktiež boli popísané prvky, z ktorých sa tento framework skladá, spôsob komunikácie medzi nimi a formát v akom komunikujú. V závere kapitoly bol načrtnutý princíp rozhrania TRAP. Ďalšia kapitola popisovala protokoly, na ktoré som sa v tejto bakalárskej práci zameral a tiež spôsob ich napadnutia. V kapitole 5 bola načrtnutá analýza útoku a vysvetlený princíp tvorby signatúry, ktorý slúži k obecnému popisu útoku. Následne boli prostredníctvom histogramov demonštrované jednotlivé signatúry pre všetky cieľové protokoly. Kapitola 6 sa zaoberala popisom detekčného algoritmu na základe ktorého prebieha celá detekcia. Jeho jadrom je histogramová analýza, ktorá sa vykonáva po dosiahnutí dostatočného počtu tokov. Ďalej bola popísaná druhá časť detekčného algoritmu, ktorej výsledok závisí na porovnaní hodnôt detekčných prahov oproti hodnotám získaným z histogramu. V záverečnej kapitole sú prezentované dosiahnuté výsledky v závislosti na konfigurácii s akou bol modul spustený. Taktiež sú dosiahnuté výsledky porovnané s pôvodnou verziou modulu.

Pokiaľ je tento typ útoku vykonávaný automaticky prostredníctvom jedného z dostupných nástrojov, existuje veľká šanca, že bude tento útok odhalený, keďže väčšina z útokov má približne rovnakú charakteristiku. Problém však nastáva pokiaľ sa útočník snaží zmiasť detekčný algoritmus či už modifikáciou samotných tokov alebo vkladaním falošných tokov, ktoré majú reprezentovať legitímnu komunikáciu.

Metóda, ktorou sa zaoberá táto práca by mala byť čiastočne schopná odolávať aj takýmto pokusom o maskovanie útoku. Nakoľko je navrhnutá spôsobom, pri ktorom sa nepoužívajú fixné hodnoty parametrov, môže byť použitá pri detekcii útokov nad ľubovoľným protokolom.

Literatúra

[1] Remote Desktop Protocol. MSDN Library [online]. 2014 [cit. 2015-04-27].
URL
https://msdn.microsoft.com/en-us/library/aa383015%28v=vs.85%29.aspx

[2] NetFlow. Wikipédia [online]. Apríl 2015 [cit. 2015-04-27].
URL http://en.wikipedia.org/wiki/NetFlow#NetFlow_Versions

[3] Introduction to Cisco IOS NetFlow - A Technical Overview [online]. Máj 2012 [cit. 2015-04-27].
URL http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

[4] NetFlow. PandoraFMS [online]. Máj 2012 [cit. 2015-04-27].
URL http://wiki.pandorafms.com/index.php?title=Pandora:Documentation_en:Netflow

[5] Configure Server Authentication and Encryption Levels. TechNet Library [online]. Október 2011 [cit. 2015-04-27].
URL https://technet.microsoft.com/en-us/library/cc770833.aspx

[6] Barrett, D. J.; Silverman, R. E.: *SSH Kompletní průvodce*. Computer Press, 2003, ISBN 80-722-6852-X.

[7] Bartoš, V.; Žádník, M.; Čejka, T.: Nemea: Framework for stream-wise analysis of network traffic. CESNET technical report 9/2013, December 2013.

[8] Campbell, B.: Kurtosis. Financial Exam Help 123 [online]. Jún 2013 [cit. 2015-04-27].
URL http://financialexamhelp123.com/kurtosis/

[9] Cisco Systems: *Cisco NetFlow Collector User Guide: Overview [online]*. Február 2014 [cit. 2015-04-27].
URL http://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/6-0/tier_one/user/guide/user/overview.html

[10] Cisco Systems: *NetFlow Configuration Guide, Cisco IOS Release 12.4T [online]*. Apríl 2012 [cit. 2015-04-27], str. 44-45.
URL http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/netflow/configuration/12-4t/nf-12-4t-book.pdf

[11] Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954, RFC Editor, Október 2004.
URL http://www.rfc-editor.org/rfc/rfc3954.txt

[12] Jonker, M.; Hofstede, R.; Sperotto, A.; aj.: Unveiling Flat Traffic on the Internet: An SSH Attack Case Study. In *IFIP/IEEE International Symposium on Integrated Network Management*, Bude publikované: Máj 2015.

[13] Kretchmar, J. M.; Dostálek, L.: *Administrace a diagnostika sítí : pomocí OpenSource utilit a nástrojů*. Brno: Computer Press, 2004, ISBN 80-251-0345-5, str. 85.

[14] Pacholík, V.: *Detekce pomalých síťových útoků*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, 2014.

[15] Pacáková, V.: *Aplikovaná poistná štatistika*. Bratislava: IURA EDITION, 2004, ISBN 80-8078-004-8, str. 28.

[16] Postel, J.; Reynolds, J.: Telnet Protocol Specification. STD 8, RFC Editor, Máj 1983.
URL http://www.rfc-editor.org/rfc/rfc854.txt

[17] Shirey, R.: Internet Security Glossary, Version 2. RFC 4949, RFC Editor, August 2007.
URL http://www.rfc-editor.org/rfc/rfc4949.txt

[18] Stallings, W.: Protocol Basics: Secure Shell Protocol. *The Internet Protocol Journal*, ročník 12, č. 4, December 2009: str. 24.

[19] Stewart, R.: Stream Control Transmission Protocol. RFC 4960, RFC Editor, September 2007.
URL http://www.rfc-editor.org/rfc/rfc4960.txt

[20] Vizváry, M.; Vykopal, J.: Flow-based detection of RDP brute-force attacks. In *Proceedings of 7th International Conference on Security and Protection of Information*, 2013.

[21] Vykopal, J.; Drašar, M.; Winter, P.: *Flow-based Brute-force Attack Detection*. Garching near Muenchen: Fraunhofer Research Institution AISEC, 2013, ISBN 978-3-8396-0474-8, s. 41–51.

[22] Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Authentication Protocol. RFC 4252, RFC Editor, Január 2006.
URL http://www.rfc-editor.org/rfc/rfc4252.txt

[23] Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Connection Protocol. RFC 4254, RFC Editor, January 2006.
URL http://www.rfc-editor.org/rfc/rfc4254.txt

[24] Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253, RFC Editor, Január 2006.
URL http://www.rfc-editor.org/rfc/rfc4253.txt

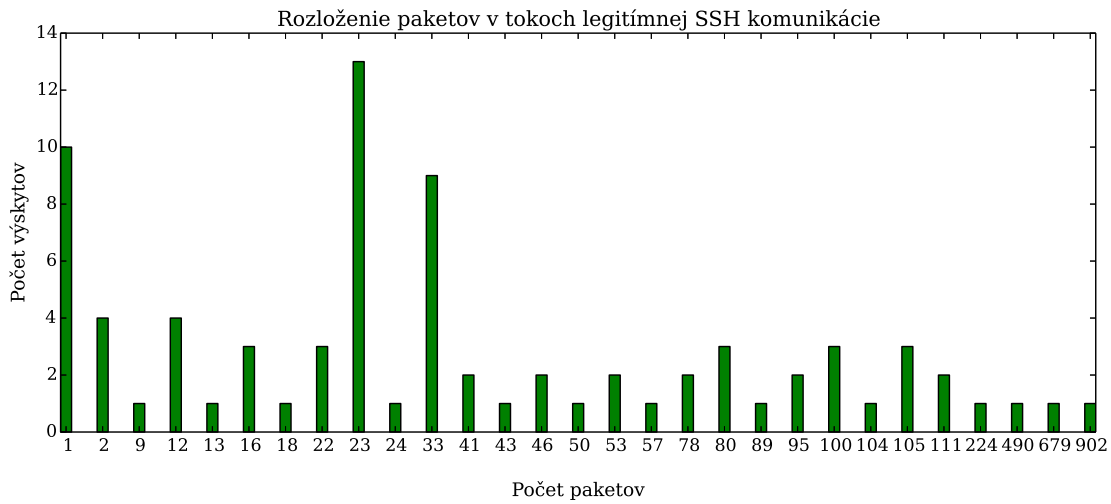
Príloha A

Obsah CD

- Zdorojové kódy Brute-Force modulu.
- Manuál pre zapojenie modulu.
- Elektronicka verzia bakalárskej práce vo formáte PDF.
- Zdrojový text bakalárskej práce pre systém LaTeX.

Príloha B

Graf legitímnej komunikácie na protokol SSH



Obrázok B.1: Graf legitímnej komunikácie na protokol SSH