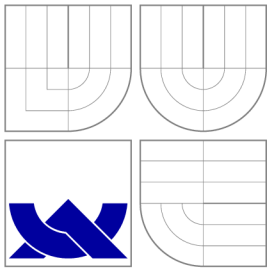
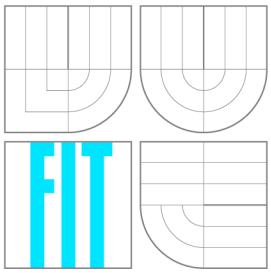


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EXTRAKCE DOSTUPNÝCH INFORMACÍ Z PROTOKOLU SSH

EXTRACTION OF AVAILABLE INFORMATION FROM SSH PROTOCOL HEADERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — NORBERT ĎURČANSKÝ
AUTHOR

VEDOUCÍ PRÁCE — KOŘENEK JAN, Ing.,Ph.D.
SUPERVISOR

BRNO 2016

Zadání bakalářské práce/18046/2015/xdurca01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Ďurčanský Norbert**
Obor: Informační technologie
Téma: **Extrakce dostupných informací z protokolu SSH**
Extraction of Available Information from SSH Protocol Headers
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s monitorovacími nástroji IPFIX kolektor a IPFIX exportér.
2. Nastudujte protokol SSH a analyzujte možnosti extrakce informací bez nutnosti dešifrovat provoz.
3. Navrhněte a implementujte modul pro IPFIX expotér, který umožní extrakci informací z protokolu SSH.
4. Implementovaný modul ověřte z pohledu stability, výkonnosti a přesnosti.
5. Proveďte měření statistik SSH provozu na reálně síti (například v produkční síti sdružení CESNET).
6. V závěru diskutujte dosažené výsledky.

Literatura:

- Dle pokynu vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kořenek Jan, Ing., Ph.D.**, UPSY FIT VUT
Konzultant: Velan Petr, CESNET
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Práca analyzuje problematiku extrakcie dostupných informácií z protokolu SSH. K riešeniu boli využité znalosti o protokole SSH, pomocou ktorých bol implementovaný modul pre FlowMon exportér. Pri skúšobných meraniach bol otestovaný na reálnej sieti a overený z hľadiska stability, výkonnosti a presnosti. Výsledný modul nám umožňuje extrakciu informácií z protokolu SSH a analýzu komunikácie bez nutnosti dešifrovať prevádzku.

Abstract

This paper analyzes issue regarding to extraction of available information from SSH protocol. To achieve this aim, knowledge about SSH protocol were used to implement plugin for FlowMon exporter. During the testing plugin was tested on real network and validated in terms of stability, efficiency and accuracy. The result plugin allows us to extract information from SSH protocol and futher analysis without decryption of traffic.

Kľúčové slová

SSH, NetFlow, FlowMon, IPFIX, IP toky, sonda, exportér, kolektor

Keywords

SHH, NetFlow, FlowMon, IPFIX, IP flows, probe, exporter, collector

Citácia

ĎURČANSKÝ, Norbert. *Extrakce dostupných informací z protokolu SSH.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kořenek Jan.

Extrakce dostupných informací z protokolu SSH

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Jana Kořeneka, Ing., Ph.D., ďalšie informácie k implementácií mi poskytol RNDr. Petr Velan. Všetky použité zdroje, ktoré som použil alebo z nich čerpal, v práci citujem s uvedením odkazu na príslušný zdroj.

.....
Norbert Ďurčanský
14. mája 2016

Poďakovanie

Veľmi rád by som sa poďakoval vedúcemu práce Janovi Kořenekovi, Ing., Ph.D. za jeho podporu a cenné rady pri vypracovaní tejto práce. Taktiež by som chcel vyjadriť vďaku RNDr. Petrovi Velanovi, ktorý mi poskytol znalosti, rady a materiály, vďaka ktorým bolo možné realizovať túto prácu.

Obsah

Kapitola 1

Úvod

Žijeme v digitálnom svete, v ktorom sa informačné technológie dostávajú medzi najrýchlejšie rozvíjajúce sa odvetvia našej spoločnosti. Komunikácia a informovanosť sú jeho súčasťou a vďaka internetu, ktorý v roku 2014 prekonal 3 miliardy pripojených uživateľov má k informáciam prístup skoro polovica ľudskej populácie [1]. Každým rokom sa tento počet zvyšuje, čím vytvára rozsiahlu sieť pripojení nazývanú Internet. Internet nám prináša technológie so zámerom uspokojiť požiadavky spoločnosti ako aj využiť túto spoločnosť k rozvoju týchto technológií. Sprístupnenie internetu užívateľom nám prinieslo okrem vývoja konkurenčných, súkromných sieti aj vývoj internetových technológií [18]. Dnes okrem bežných, užívateľských činností ako sú prehľadávanie webu, elektronická pošta, komunikácia s priateľmi, je možné prevádzkovať širokú škálu služieb, ktoré zefektívňujú prácu ľudí a strojov. Príkladom môže byť komunikácia mobilného telefónu a domu, kedy nám jednoduchá aplikácia umožňuje zistiť teplotu, spotrebu energie, ovládať svetlá, klimatizáciu, bezpečnostný alarm a všetky spotrebiče zapojené do siete. Získané dáta následne spracováva a umožňuje nám ich zobraziť pomocou webového rozhrania.

S použitím podobných technológií vzniká množstvo hrozieb, ktoré môžu spôsobiť stratu dát, ich nedovolené odpočúvanie alebo šírenie. Často sa jedna o pokusy narušenia správnej činnosti služieb, hľadania bezpečnostných dier alebo odpočúvania. S vývojom nových technológií vznikajú čoraz komplexnejšie hrozby, ktorých cieľom je poškodiť užívateľa a preto je nutné zdokonaľovať ochranu.

Sieťová bezpečnosť zahŕňa zabránenie a monitorovanie nedovoleného prístupu, zmeny dát alebo využitia sieťových prostriedkov. Taktiež, popisuje koncept, ktorý obsahuje autorizáciu a prístup k dátam na sieti, kontrolovaný a riadený sieťovým administrátorom. Štruktúra konceptu začína autentifikáciou, zvyčajne použitím prihlasovacieho mena a hesla. Po tom, ako je užívateľ autentifikovaný, firewall umožní sprístupnenie služieb, ku ktorým má prístup. Pretože firewally môžu vo svojej činnosti zlyhať a umožniť prijatie škodlivého obsahu, ako sú počítačové vírusy, červy, trójany, používajú sa antivírusové programy. Tie sú zvyčajne súčasťou operačného systému. Ich hlavnou úlohou je monitorovať a odhaľovať potenciálne nebezpečenstvá.

Pokročilé techniky komunikácie sú základom zaistenia sieťovej bezpečnosti. Medzi často používané autentizačné metódy patrí aj protokol SSH[1](secure shell). Tento protokol vznikol ako náhrada málo bezpečných protokolov ako sú `telnet` alebo `rsh`, čoho výsledkom je bezpečná kryptovaná komunikácia medzi dvoma neautentifikovanými strojmi v nezabezpečenej sieti. Implementácie tohto protokolu sa nachádzajú na väčšine moderných platforiem, či už

[1] https://www.ietf.org/rfc/rfc4251.txt

v komerčnej alebo voľne dostupnej verzií.

Cieľom tejto práce je analyzovať možnosť extrakcie informácií z autentizačného protokolu SSH bez nutnosti dešifrovať komunikáciu a navrhnúť nástroj, ktorý umožňuje monitorovanie tohto protokolu na princípe IP tokov.

Pred začiatkom implementácie, bolo potrebné zoznámiť sa s fungovaním protokolu SSH, ktorý rieši vytvorenie bezpečného komunikačného tunela. Medzi ďalšie potrebné nástroje patrí aj technológia IPFIX[2] (*IP Flow Information Export*), ktorá umožňuje monitorovanie sieťovej prevádzky na báze IP tokov. Táto technológia je využitá v sonde FlowMon[3], v ktorej bol náš modul vyvíjaný. Pre získanie informácií o priebehu komunikácie protokolu SSH bol využitý monitorovací nástroj Wireshark[14], ktorý nám umožnil rozsiahlu analýzu sieťovej prevádzky. Pomocou neho a informácií ziskaných z RFC[4] (*Request for Comments*) stránok boli určené položky, o ktoré môžu byť rozšírené IP toky pre možnosť monitorovania protokolu SSH. Výsledné riešenie bolo implementované ako zásuvný modul pre FlowMon exportér.

Prácu je možné rozdeliť do 5 častí. V 2. kapitole sú popísané rôzne spôsoby monitorovania sietí, pričom vačšia časť je venovaná monitorovaniu na báze IP tokov. V tejto časti je aj popis 2 najznámejších protokolov, ktoré na tomto princípe pracujú. Dôležitou súčasťou je aj popis princípov fungovania IPFIX exportéru, môžnosť spracovávania a analýzy dát.

V 3. kapitole sú popísané základné vlastnosti protokolu SSH, spôsob komunikácie, verzie tohto protokolu a údaje, ktoré boli použité na rožšírenie IP tokov. V ďalšej kapitole som sa venoval aj iným nástrojom, pomocou ktorých môžme sledovať SSH prevádzku. V predposlednej kapitole je špecifikovaný návrh extrakčného modulu s využitím užívateľského aplikačného rozhrania, jeho základná štruktúra, požiadavky na jeho vykonnosť a výkonnostné optimalizácie. Na záver práce sú prezentované výsledky analýzy sieťovej prevádzky protokolu SSH, grafy výkonností, popis testovacieho prostredia a overenie funkčnosti implementovaného modulu.

[2] https://tools.ietf.org/html/rfc5101
[3] https://www.flowmon.com/en
[4] https://www.ietf.org/rfc.html

Kapitola 2

Monitorovanie sieťovej prevádzky

S pokračujúcim exponencionálnym rastom množstva sieťovej prevádzky, je potrebné neustále sledovať jej stav za účelom získania užitočných informácií[19]. Tieto informácie nám slúžia na monitorovanie, zber štatistík, šetrenie sieťových prostriedkov alebo na odhaľovanie podozrivých udalostí, či už vo forme útokov alebo nestability spôsobenej preťažením siete.

Dáta a štatistiky sú často použité na plánovanie a návrh sieťových prepojení. K zisteniu aktuálneho stavu siete je možné použiť sledovanie obsahu paketov[16]. Samotný spôsob získania týchto informácií sa delí na aktívny alebo pasívny. U **aktívneho**, využitím nástrojov ako sú `ping` alebo `traceroute` dochádza k testovaniu dostupnosti zapojených prvkov, k zisťovaniu priepustnosti bez dlhodobého ukladania a tvorby štatistík. **Pasívny prístup** sleduje komunikáciu, ktorá prechádza monitorovacím bodov, pričom samotné zariadenia samy komunikujú a zasielajú dáta.

Jednou z nich je aj zachytávanie paketov. Táto metóda nám umožňuje pohľad do vnútra sieťovej komunikácie, zachytávaním celých paketov a ich ďalšej analýze. Hlavnou nevýhodou je, že pri použití vysokorýchlostných liniek, zachytávanie vyžaduje výkonný hardware a infraštruktúru na ukladanie a analýzu.

Ďalším spôsobom je metóda DPI(*anglicky deep packet inspection*), ktorá pracuje na L4 až L7 vrstve ISO/OSI[21] modelu. Touto metódou je možné získať detailné informácie o sieťovom pripojení a je vo veľkej miere použitá pri odhaľovaní nedovolených prienikov do siete. Hlavnou nevýhodou je, že pri veľkom počte prenesených paketov je jej časová náročnosť pomerne vysoká.

2.1 Monitorovanie siete na báze IP tokov

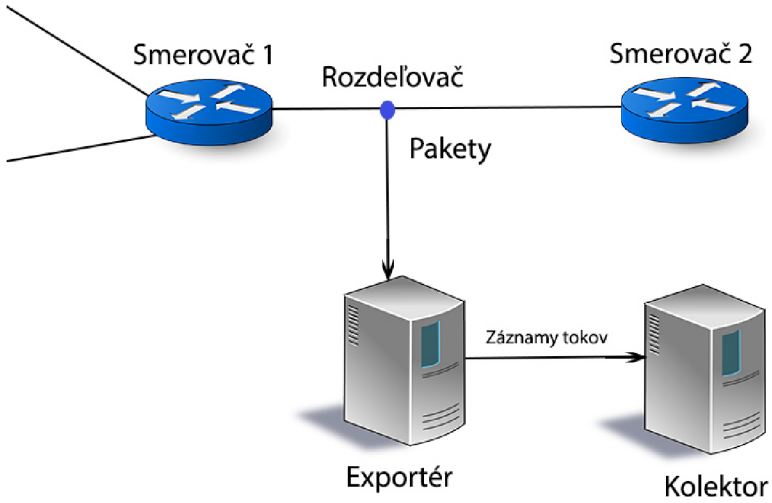
Odlišný prístup pasívneho monitorovania, ktorý umožňuje väčšiu škálovateľnosť pri použití vo vysokorýchlostných sieťach je zameraný na IP toky[11]. Pakety sú agregované do tokov (*anglicky flow record*). **Tok** je definovaný ako množina IP paketov s rovnakými vlastnosťami, ktoré prejdu monitorovacím bodom za určitý časový interval. Metóda umožňuje získať dôležité údaje o komunikácií, ako sú napríklad zdrojová, cieľová IP adresa, použité porty, transportný protokol a pod.

Okrem použitia vo vysokorýchlostných sieťach nám protokoly na tomto princípe poskytujú ďalšie výhody oproti metóde zachytávania paketov. Medzi ne patrí aj rozsiahla nasaditelnosť z dôvodu integrácie do smerovačov, prepínačov alebo firewallu. To je možné použiť na bezpečnostnú analýzu, rozbor zaťaženia, účtovanie alebo profilovanie prevádzky.

Konceptuálna architektúra monitorovania na báze IP tokov je zobrazená na obrázku 2.1

a tvorená z niekoľkých častí:

- Sonda - Odchytávanie paketov, býva súčasťou exportéra.
- Exporter - Predspracovanie, agregovanie do tokov a export do kolektoru.
- Kolektor - Príjem dát a ich ukladanie.
- Analyzátor - Analýza dát, vyhľadávanie a profilovanie, býva integrovaný do kolektoru.
- Protokol - Komunikačný protokol na prenos spracovaných dát do kolektoru.



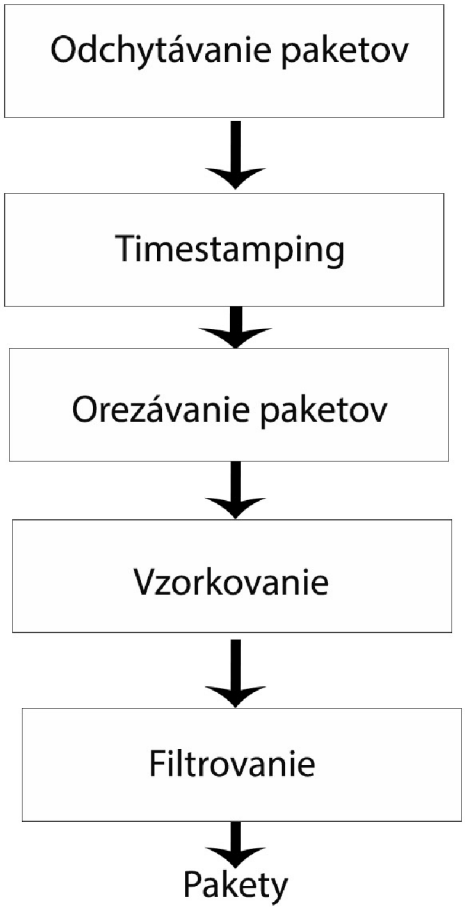
Obr. 2.1: Konceptuálna architektúra monitorovania na báze IP tokov

V 1.časti modelu sa pakety zachytávajú a dochádza k ich predspracovaniu. Táto časť modelu patrí medzi najdôležitejšie a preto je kľúčová pre monitorovanie na báze IP tokov. Spolu s odchytením paketu dochádza k `timestamping` a k ďalším voliteľným krokom zobrazených na obrázku 2.2.

Prvým z krokov je skracovanie paketu, ktoré vyberá len takú veľkú časť, ktorá sedí s predkonfigurovanou dĺžkou snímku. Vďaka tomu sa redukuje množstvo dát prijatých a spracovaných aplikáciou a taktiež počet výpočtových cyklov, veľkosť zbernice a pamäti potrebnej na spracovanie sieťovej prevádzky. Do tejto skupiny patrí aj `flow exportér`, ktorý sleduje len hlavičky paketov.

Posledným krokom je vzorkovanie a filtrovanie paketu. Aplikácie môžu definovať pravidlá tak, že len niektoré pakety su vybrané. Úlohou vzorkovania je vybrať časť paketu tak, aby bolo stále možné zistiť vlastnosti celého toku. Tento výber môže byť definovaný podľa poradia času, veľkosti alebo použitím *Flow filtra*. Ten vzorkuje dáta podľa prevádzky, čím nie je tak zaťažená cache. Vzorkovanie môže byť použité tak ako na exportéri, tak aj na kolektore.

Filtrácia na druhú stranu odstraňuje všetky pakety, ktoré nie sú v záujme spracovávania. Hlavným použitím je klasifikácia prevádzky na základe hodnôt v hlavičke paketov, pričom filtrovanie dát nezávisí na poradí, čase a pod.



Obr. 2.2: Proces zachytávania paketov.

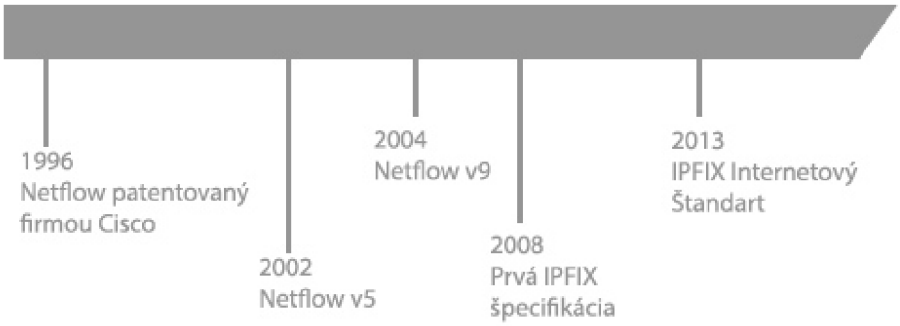
Sonda môže byť zariadenie, rozhranie alebo software bežiaci na aktívnych prvkoch siete, zvyčajne na smerovacích zariadeniach. Zachytené pakety sú predané exportéru, ktorý tieto pakety analyzuje a agreguje do tokov[15].

Predtým než sa dokončí analýza celého toku a jeho nasledný export sú tieto informácie uložené v pamäti tokov (*anglicky flow cache*). Väčšinou je sonda priamo integrovaná do exportéru, ktorý po stanovení toku za ukončený, exportuje záznam. Tento proces sa udeje vložením záznamu (*anglicky flow record*) do datagramu exportovacieho protokolu, ktorý je popísaný v kapitole 2.2, 2.3. Záznam je definovaný, ako špecifická informácia o toku rozšírená o merané položky.

Protokol komunikuje s kolektorom, do ktorého sú tieto dáta exportované. Dokáže prijímať dáta od viacerých exportérov a ukladá ich do svojho úložiska. Často sa robí aj spracovanie týchto dát pre účely agregácie, filtrovania a kompresie.

Poslednou časťou modelu je dátová analýza, ktorá slúži na profilovanie prevádzky, klasifikáciu, detekciu anomálií alebo k vyhladávaniu dát. Tieto operácie su často integrované do kolektoru, čím dostávame jednoduchší tvar architektúry pre meranie toku. Je dôležité spomenúť, že sa jedná len o základný model, ktorý je často rozšírený a kombinovaný rôznymi spôsobmi pre optimálne riešenie.

Tento spôsob monitorovania vytvoril základ pre moderné protokoly, ako sú NetFlow a IPFIX. História vývoja týchto protokolov je znázornená na obrázku 2.3.



Obr. 2.3: História vývoja architektúry IPFIX a NetFlow

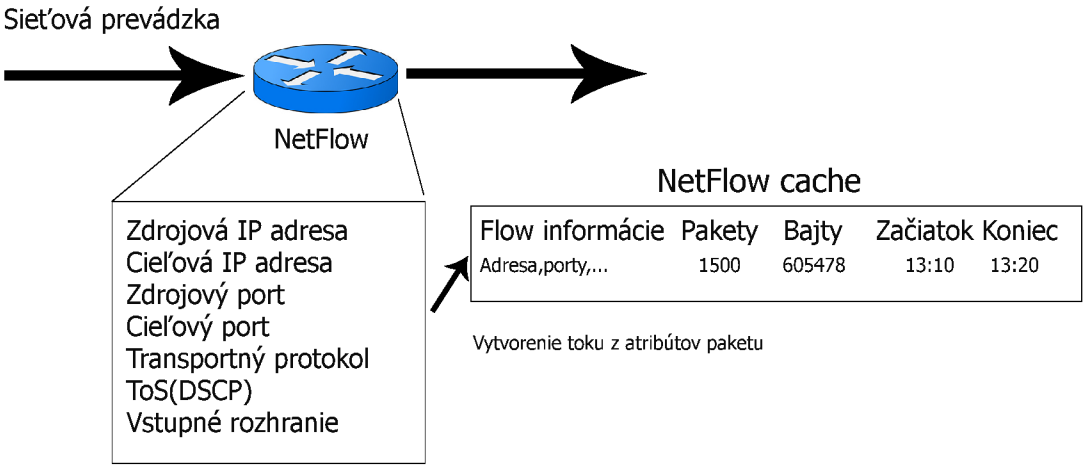
2.2 NetFlow

Viditeľnosť do vnútra sieťovej komunikácie je nevyhnutným nástrojom pre IT špecialistov. Ako odpoveď na nové požiadavky a na tlak zo strany sieťových operátorov je dôležité rozumieť ako sa sieť správa. Netflow bol vyvinutý firmou Cisco pre účely monitorovania sieťovej prevádzky[2]. Pôvodne proprietárny štandart napĺňuje tieto kritické potreby a vytvára prostredie, kde sieťový administrátori majú nástroj na monitorovanie kto, kde, čo, kedy a ako sieťová prevádzka funguje.

Architektúra Netflow definuje exportér ako sondu/smerovač pre získanie štatistík o tokoch, ktoré sú následne pomocou tohto protokolu odoslané na kolektor. Tento spôsob, kde smerovač okrem svojej primárnej činnosti zbiera aj informácie o komunikácií je výpočtovo náročný. Pre zvýšenie výpočtového výkonu je preto lepšie využiť samostatné sondy umiestnené na vstupe do siete, ktoré komunikujú s kolektorom. Obrázok 2.4 popisuje vytvorenie Netflow záznamum, analýzou paketu a vytvorenie Netflow záznamu o toku. Bolo vytvorených viacero verzií protokolu, ale najviac sa využívajú protokoly Netflow verzia 5 a Netflow verzia 9.

Netflow v5 nám poskytuje získanie základných informácií o sieťovej komunikácií. Prenesený záznam obsahuje zdrojovú, cieľovú IP adresu, porty, počet prenesených bajtov pre účtovanie a pod. Hlavička paketu zapúzdruje informácie o verzií protokolu, indentifikátor zdroju dát a ďalšie znázornené v tabuľke 2.1.

Trochu iný prístup prináša Netflow v9, ktorý rozširuje protokol o šablóny, ktoré popisujú ako vyzerajú štruktúry záznamov. Týmto je možné využitie širších možností Neflow, pretože umožňuje sondám zvoliť si, aké dáta zbierajú a odosielajú. Vďaka šablónam je možné rozlišovať typy záznamov a zhlukovať ich do dátových množín. Výsledná množina sa skladá z dátových záznamov rovnakého typu a identifikátoru šablóny, ktorý tento typ definuje. Netflow v9 paket obsahuje hlavičku a dátovú alebo vzorovú množinu. Vzorová množina obsahuje záznamy, ktoré definujú položky v dátovom zázname. Medzi ďalsie rozšírenia patrí aj pridanie informácií o linkovej vrstve.



Obr. 2.4: Spôsob vytvorenia záznamu o toku

Bity	Obsah	Popis
0-1	version	Číslo identifikuje použitý Netflow protokol
2-3	count	Počet tokov v aktuálnom pakete (1-30)
4-7	SysUptime	Čas od spustenia exportéru (milisekundy)
8-11	unix_secs	Čas v sekundách od 1.1.1970
12-15	unix_nsecs	Čas v nanosekundách od 1.1.1970
16-19	flow_sequence	Sekvenčné číslo určujúce celkový počet flow záznamov
20	engine_type	Typ exportujúceho procesu
21	engine_id	Identifikátor exportéru dát
22-23	sampling_interval	Prvé 2 bity udávajú spôsob vzorkovania dát, ostatné udávajú hodnotu vzorkovacieho intervalu

Tabuľka 2.1: Hlavička Netflow paketu.

Exportovaný paket obsahuje hlavičku 2.1, za ktorou nasleduje jedna alebo viac množín (*anglicky flowset*). Tieto množiny môžu byť jedným z týchto typov:

- vzorová množina
- dátová množina
- vzorová množina možností

Pre zaistenie efektivity v exportéri pri spracovavaní veľkého objemu paketov, sú pakety zapúzdrené do UDP datagramu. Hoci verzia 9 je navrhnutá, aby bola nezávislá na protokole, UDP transport nezaisťuje doručenie paketov. Stratou dát môže dojsť k ovplyvneniu analýzy a preto pri využití tohto protokolu v prostredí citlivom na stratu paketov je možné na prenos použiť protokol SCTP[1], ktorý zaistí ochranu dát, za cenu zníženia výpočtového výkonu obidvoch strán.

[1] https://tools.ietf.org/html/rfc2960

2.3 IPFIX

Požiadavky na vytvorenie jednotného štandartu a na väčšiu voľnosť v odosielaní získaných dát umožnili vytvoriť architektúru IPFIX[6]. IPFIX architektúra bola špecifikovaná a vytvorená organizáciou IETF[2] (*Internet Engineering Task Force*) a je podobná Netflow v9 zložená z exportéru, kolektoru, sondy a protokolu. Okrem toho špecifikuje aj protokol na prenos informácii IP tokov z exportéru do kolektoru.

Protokol IPFIX sa používa k popisu štruktúry šablón a slúži k zapúzdreniu informácií a ich prenosu. Zásadným prínosom je, že sa nejedná o proprietárny protokol, ale zavádza jednotný štandart pre export informácií IP tokov. Tento prenos je možné zaistiť okrem UDP a SCTP aj cez TCP protokol. Ďalším odlíšením od Netflow v9 je aj počet položiek, ktoré sú dostupné na prenos informácií. Oproti Netflow ktorý poskytuje do 100 rôznych elementov, ich pre protokol IPFIX definuje organizácia IANA[3] viac než 400. Jednou z rozšírení a výhod je aj možnosť definície vlastných položiek v rámci organizácie, ktoré sa nazývaju `EN` (*enterprise number*). Jedná sa o identifikátor organizácie určujúcej význam položky. Každá organizácia má svoj špecifický identifikátor, o ktorý si môže zažiadať. Týmto mechanizmom je možné zabrániť kolíziam v číslovaní a dosiahnúť veľkého množstva informácií, ktoré môžme v protokole prenášať. Takto je možné vytvoriť proprietárne elementy, špecifické pre danú oblasť a obohacovať tento protokol o informácie z aplikačnej vrstvy. V tabuľke 2.2 je popísaný formát hlavičky paketu, ktorý je odlišný od hlavičky Netflow.

Verzia protokolu	Dĺžka paketu
Čas odoslania z exportéru	
Sekvenčné číslo	
Identifikátor zdroja dát	

Tabuľka 2.2: Hlavička IPFIX paketu.

Dĺžka paketu obsahuje celkovú veľkosť paketu v bajtoch vrátane hlavičky a všetkých množín za ňou.

Čas odoslania je 32 bitové čislo v sekundách nastavené exportérom.

Sekvenčné číslo slúži k detekcií straty paketu alebo chybného odoslania z exportéru. Na strane kolektoru sa následne vykonáva kontrola týchto čísiel a to tak, že nasledujúci paket musí mať túto hodnotu rovnú súčtu sekvečného čísla aktuálneho paketu a počtu dátových záznamov, ktoré obsahuje.

Identifikátor je 32 bitové číslo identifikujúce exportér, z ktorého bol paket odoslaný.

Táto architektúra sa často používa na profilovanie prevádzky, správu účtovania a pod. IPFIX používa štandartný formát, ktorý môžu smerovače použiť pre export sieťovej prevádzky na kolektor. Tento postup prebieha v niekoľkých krokoch. Na začiatku smerovač alebo prepínač obsahuje pamäť (*flow cache*), v ktorej zhromažduje informácie o účtovaní prichádzajúcej prevádzky. Keď pamäť expiruje, je poslaná na kolektor. Exportér kolektoru

[2] https://www.ietf.org/
[3] http://www.iana.org/

definuje veľkosť polí, ktoré má prijať a ďalšie informácie, ktoré má očakavať z prichádzajúcej prevádzky. Kolektor po prijatí agreguje sieťové štatistiky a prepošle ich aplikácií na zobrazenie. Dáta je možné filtrovať a prezerať pomocou grafických aplikácií.

Pri použití UDP prenosu medzi sondou a kolektorom je potrebné na zaistenie správnosti dát overovať platnosť šablon. Tá je vyjadrená počtom sekúnd alebo počtom paketov. Po skončení platnosti musí byť šablona znovu odoslaná. Pri neodoslaní šablony, kolektor dáta príjme, ale nebude schopný tieto dáta správne spracovať.

Medzi výhody IPFIX štandartu patrí aj možnosť uloženia dát do súboru. Tieto dáta sú ukladané rovnako ako aj posielané po sieti, čo je možné využiť hlavne pri testovaní kolektoru, kedy pracuje s nemennou vzorkou dát.

Kapitola 3

SSH

SSH (*secure Shell*)[17], patrí medzi populárne a výkonné softwarové riešenie sieťového zabezpečenia. Dáta, ktoré počítač pošle sú automaticky zašifrované a po prijatí druhou stranou sa dáta rozšifrujú (dekódujú). Výsledkom tejto metódy je transparetné šifrovanie - užívateľ môže normálne pracovať bez toho, aby sa musel starať o šifrovanie. Naviac, používa moderné a bezpečné šifrovacie algoritmy a je natoľko efektívna, že ju môžme nájsť v aplikáciach, ktoré su kritické pre chod tých najvýznamnejších spoločností.

Názov SSH popisuje spoločne kryprografický sieťový protokol a sadu nástrojov na implementáciu tohto protokolu. SSH používa klient-server model komunikácie, ktorého architektúra je zobrazená na obrázku 3.1. Dochádza k spojeniu zabezpečenej klientskej aplikácie (koniec, na ktorom je relácia zobrazená) s SSH serverom (koniec, na ktorom relácia beží).

Program nazvaný *SSH server*, ktorý sa inštaluje a je v správe systémového administrátora, prijíma alebo odmieta požiadavky na sieťové pripojenia. Užívatelia si spúšťajú programy nazývane ako *SSH klient*, ktoré posielajú požiadavky na server. Server po obdržaní požiadavky, autentizovaného klienta obslúži.

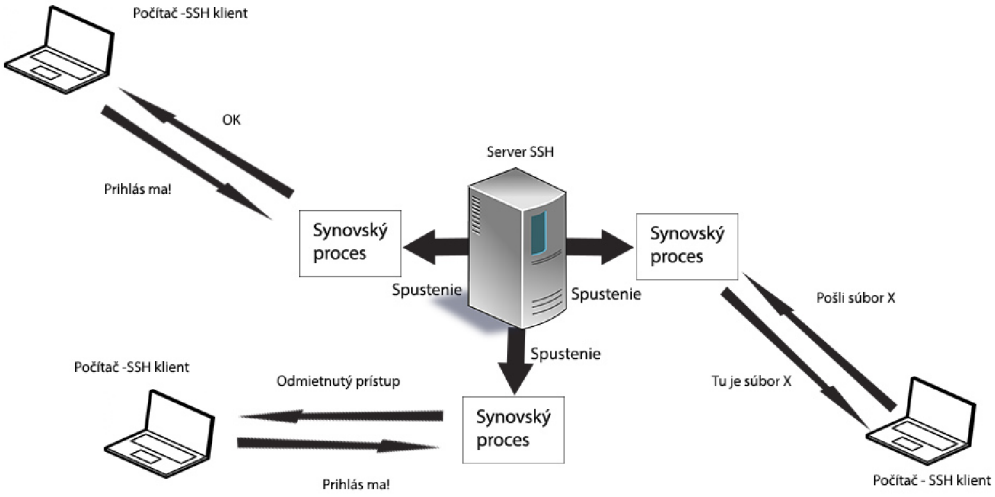
Okrem operačného systému Windows je SSH software zahrnutý vo väčšine operačných systémov. SSH taktiež podporuje tunelovanie, preposielanie TCP portov a X11 pripojenia[9]. Pre bezpečný prenos súborov je možné použiť zabezpečený prenos alebo zabezpečenu kópiu (SCP). SSH server implicitne beží na TCP porte 22.

Táto architetúra predstavuje jednoduchý koncept, ktorý sa v skutočnosti može odlišovať. Skutočné aplikácie môžu obsahovať buď len klienta, server alebo oboje.

3.1 Protokol SSH

SSH protokol pokrýva autentizáciu, šifrovanie a zaistenie integrity dát posielaných po sieti. **Autentizácia** je spoľahlivé určenie identity. SSH klient posiela digitálny dôkaz užívateľovej identity a keď sa podarí táto identita overiť, klient sa môže prihlásiť. **Šifrovanie** popisuje zakódovanie dát tak, že sú tieto dáta nečitateľné pre všetkých, ale len pre určených príjemcov.**Integrita** zaisťuje, že dáta dorazia na miesto určenia bez zmeny. Spojením týchto vlastností dostávame bezpečný komunikačný protokol.

Protokol SSH patrí medzi základné autentizačné protokoly[7]. Je určený na prevádzku cez transportný protokol `SSH-TRANS`. Tento protokol predpokladá, že všetky vrstvy pod transportnou zaisťujú integritu a dostatočnú bezpečnosť.



Obr. 3.1: Konceptuálna architekúra protokolu SSH

3.1.1 Autentizačná žiadosť

Na ustanovenie zabezpečeného kanálu je potrebné, aby klient zaslal požiadavku. Všetky autentizačné požiadavky musia použiť nasledujúci formát:

- byte - SSH_MSG_USERAUTH_REQUEST
- string - užívateľské meno v ISO-10646 UTF-8 kódovaní [5]
- string - meno služby v US-ASCII, ktorá sa spustí po pripojení
- string - meno metódy v US-ASCII
- polia pre špecifickú metódu

Len niekoľko prvých polí je pevne stanovených. Ostatné sú definové podľa vybranej metódy. Užívateľské meno a meno služby sa opakujú pri každom autentifikačnom pokuse a môžu sa meniť. Implementácia serveru musí zaistiť ich overenie pri každej správe a musí prepísať autentifikačný stav ak dôjde k jeho zmene. Ak sa mu to nepodarí musí klienta odpojiť.

Autentizácia može viesť k ďalším výmenám správ. Všetky tieto správy závisia od mena metódy, ktorú klient použil. Klient môže poslať v ľubovoľnom čase novú autentizačnú požiadavku a server musí pokračovať v autentifikácií s novo prijatou.

3.1.2 Autentizačná odpoveď

Ak server akceptuje autentifikačnú požiadavku odpovie nasledujúcou správou:

- byte - SSH_MSG_USERAUTH_SUCCESS
- name-list - autentifikácie, ktoré môžu pokračovať
- boolean - úspešne spracovaná požiadavka (true/false)

Parameter `name-list` obsahuje zoznam mien metód, ktoré podporuje server a môžu pokračovať pri autentifikácií.

3.1.3 SSH verzia 1

V tejto časti je popísaný protokol SSH verzie 1, ktorý prešiel viacerými revíziami, z nich je najviac rozšírená verzia 1.3 a 1.5. Táto verzia popisuje architektúru, ktorú je možné rozdeliť do častí:

- Ako sa ustanovuje zabezpečené spojenie.
- Autentizácia podľa hesla, verejného kľúča alebo dôveryhodný hostiteľ.
- Kontrola integrity
- Kompresia dát za podpory algoritmu deflate z utility GNU gzip[1] .

Predtým, než môže začať vlastná komunikácia, musí klient a server SSH ustanoviť zabezpečené pripojenie, obrazok 3.2 . Ustanoviť takéto spojenie má niekoľko krokov, v ktorých sa klient a server SSH v1 od začiatku dohodnú na šifrovacom algoritme a vygenerujú zdieľaný tajný kľúč relácie, čím vytvoria takéto spojenie. Postup ustanovenia spojenia má niekoľko krokov:

1. Klient kontaktuje server.
2. Klient a server si vzájomne predajú, aké verzie protokolu SSH podporujú.
3. Klient a server sa prepnú na paketový protokol.
4. Server sa identifikuje klientovi a dodá mu potrebné parametre relácie.
5. Klient odošle serveru tajný kľúč.
6. Obe strany zapnú šifrovanie a dokončí sa autentizácia serveru.
7. Zabezpečenie spojenia je týmto dokončené.

Obr. 3.2: Zabezpečené spojenie

V tejto verzií sa používa kontrola integrity 32 bitovou cyklickou redudantnou kontrolou CRC-32[2] . Tento typ kontroly je vhodný pre detekciu náhodných zmien v dátach, ale na druhú stranu je málo efektívny pri zameraní sa na poruchu dát. Túto slabosť využíval aj tzv. vkladací útok, ktorý popísal pán Futoranský a Kargieman[12]. Použitie CRC-32 je natoľko závažný nedostatok SSH-1, že znamenal urýchlený vývoj SSH verzie 2. Táto verzia už používa kryptograficky silnú kontrolu integrity, ktorá je proti tomuto útoku odolná. Pre identifikáciu komunikácie implementovaný modul umožňuje detekovať verziu protokolu SSH v1. Ostatné merané položky sú analyzované len pre verziu 2.

[1] https://ftp.gnu.org/pub/gnu/gzip/
[2] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=37010

3.1.4 SSH verzia 2

Protokol je definovaný niekoľkými návrhmi štandardizačných dokumentov pracovnej skupiny IETF a patrí medzi najviac používané v dnešnej dobe. Na rozdiel od predchádzajúcej verzie SSH-1 , ktorá popisovala monolitický protokol, kde v jednom celku je zakomponovaných mnoho rôznych funkcií. Verzia protokolu SSH-2 je rozdelená do modulov a skladá sa celkom z troch protokolov, ktoré vzájomne spolupracujú:

- Protokol transportnej vrstvy SSH (*ssh-trans*)
- Autentizačný protokol SSH (*ssh-auth*)
- Spojovací protokol ssh (*ssh-conn*)

Základným stavebným kameňom je SSH-TRANS, ktorá zaisťuje spojenie, rozdeľovanie dát do paketov, autentizáciu serveru a základné služby spojené s kontrolou integrity a šifrovania. Po dokončení spojenia cez SSH-TRANS aplikácia vytvára jediný, zabezpečený a úplne duplexný bajtový prúd vedúci k autentizovanej protistrane.

Cez toto spojenie môže klient ďalej použiť SSH-AUTH, ktorý sám autentizuje server, pričom definuje tri autentizačné metódy:

- S verejným kľúčom
- Na základe hostiteľa
- Heslom

Autentizácia verejným kľúčom využíva algoritmy, kedy klient používa zašifrovaný verejný kľúč, pomocou ktorého sa autentizuje. Ak chce pristúpiť k nejakému účtu na serverovom počítači, serveru dokáže, že je držiteľom, konkrétne súkromnej časti autorizovaného verejného kľúča. Autentizácia na základe hostiteľa sa prevádza na základe dôvery medzi hostiteľmi spolu s kryptografickým overením klienta. Posledná autentizácia heslom, je podobná ako u SSH v1, kedy klient cez zabezpečené pripojenie pošle heslo na server, kde sa overí. Verzia 2 naviac prináša zmenu užívateľského hesla.

Posledným modulom tejto verzie je spojovací protokol SSH-CONN poskytujúci široké spektrum zložitejších služieb, ktoré využívajú jediný prenosový kanál, zaistený pomocou SSH-TRANS. Sem patrí všetko pre podporu väčšieho množstva interaktívnych, tak neinteraktívnych relácií. To je napríklad multiplexovanie niekoľkých prúdov do jedného kanálu, správa smerovania X11 a pod.

Verzia 2 nám na rozdiel od predchádzajúcej verzie prináša rozšírené dojednávanie algoritmov medzi klientom a serverom, viacero metód pre výmenu kľúčov, certifikáty pre verejné kľúče, vačšiu flexibilitu pri autentizácií, silnejšiu integritu, pravidelnú výmenu kľúčov relácie a ďalšie rozdiely popísané v tabuľke 3.1.

Pri ustanovení zabezpečeného pripojenia a výmene dôležitých parametrov spojenia sa vymieňajú viacere položky popisane v RFC 4253[8]. Zpráva, ktorú pošlu obe strany a je dôležitá pre nastavenie celej relácie sa volá *Key Exchange Init*. Telo tejto správy obsahuje zoznam najdôležitejších položiek, ktoré boli použité pre analýzu komunikácie a sú zobrazené v tabuľke 3.2. Sú použité alias názvy na odlíšenie skupiny položiek zoznamu.

Po doručení paketu, sa obe strany dohodnú na parametroch relácie. Táto dohoda je založená na algoritme, kedy sa vyberá prvá metóda klienta, ktorú podporuje server.

SSH verzia 2	SSH verzia 1
oddelené trasportné, autentizačné a spojovacie protokoly	Jediný monolitický protokol
šifrovanie, mac a kompresia sa dojednávajú pre každý smer zvlášť a za použitia nezávislých kľúčov	Slabá kontrola integrity CRC-32
vďaka výmene kľúču pomocou Diffie-hellman nie je kľúč serveru potreba	Dojednáva len hromadnú šifru
podporuje certifikáty verejných kľúčov	Kľúč servera sa používa pre dosiahnutie dopredného zabezpečenia kľúča relácie

Tabuľka 3.1: Rozdiely medzi SSH-1 a SSH-2.

Názov položky	Popis
`kex_algorithms`	Zoznam podporovaných algoritmov výmeny kľúčov (napr. diffie-helman-group1-sha1)
`server_host_key_algorithms`	Zoznam podporovaných algoritmov pre autentizáciu (napr. ssh-rsa)
`encryption_algorithms client_to_server`	Zoznam podporovaných kryptovacích algoritmov z klienta na server (napr. aes128-ctr)
`encryption_algorithms server_to_client`	Zoznam podporovaných kryptovacích algoritmov zo servera na klienta
`mac_algorithms client_to_server`	Zoznam podporovaných Mac algoritmov na zaistenie integrity z klienta na server(napr. hmac-md5)
`mac_algorithms server_to_client`	Zoznam podporovaných Mac algoritmov na zaistenie integrity zo servera na klienta
`compression_algorithms client_to_server`	Zoznam podporovaných kompresných algoritmov na zaistenie integrity z klienta na server
`compression_algorithms server_to_client`	Zoznam podporovaných kompresných algoritmov na zaistenie integrity zo servera na klienta

Tabuľka 3.2: Položky SSH-2 v zpráve *Key Exchange Init*.

Pre implementáciu modulu som využil vyššie definované položky, ktoré špecifikujú komunikáciu. Tím že SSH-2 podporuje rôzne typy šifrovania, Mac a kompresie pre každý smer, je táto položka významná z hľadiska sledovania komunikácie. Položka pre šifrovanie určuje, aký algoritmus sa použije v danom smere, mac slúži na definovanie algoritmu, ktorý zvyšuje integritu dát a kompresia udáva typ algoritmu, ktorý je použitý. Zatiaľ je možné použiť kompresiu typu zlib popísanú v RFC 1950[3] a v RFC 1951[4].

Kapitola 4

Ďalšie metódy analýzy sieťovej komunikácie

V tejto kapitole sú popísané metódy, ktoré nepoužívajú ako doposiaľ popísané systémy spracovávanie na základe tokov, ale iné metódy slúžiace k získaniu užitočných informácií.

4.1 Wireshark

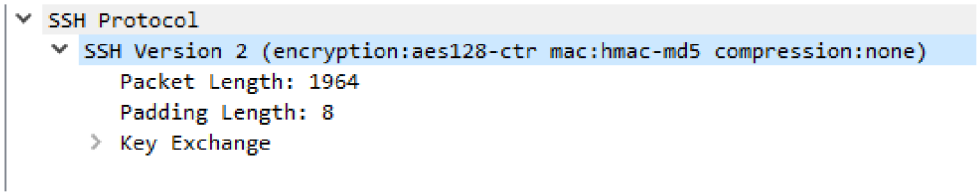
Medzi jeden z nich patrí aj nástroj `Wireshark`. Ten využíva metódu na princípe analýzy paketov. Tento nástroj bol použitý na testovacie účely a na overenie funkčnosti zásuvného modulu.

Wireshark je podobný nástroj ako `tcpdump` s grafickým rozhraním, ktorý umožnuje užívateľovi vidieť prevádzku na danom rozhraní alebo ju nahrať zo súboru. Sieťovú prevádzku je možné jednoducho analyzovať s využitím nástrojov na filtrovanie komunikácie alebo s možnosťou zobrazenia prebiehu. Naviac, Wireshark poskytuje možnosť syntaktickej analýzy paketu, čo umožňuje získať význam jednotlivých políčok. Tento nástroj je dostupný pre väčšinu operačných systémov. Na ukážku fungovania je použitý súbor `src/examples/SSH.pcap`, ktorý obsahuje jednoduchú komunikáciu klienta so serverom a je zobrazený na obrázku 4.1.

Source	Destination	Protocol	Length	Info
10.0.2.15	147.229.176.19	SSHv2	97	Client: Protocol (SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.3)
147.229.176.19	10.0.2.15	SSHv2	75	Server: Protocol (SSH-2.0-OpenSSH_5.3)
147.229.176.19	10.0.2.15	SSHv2	894	Server: Key Exchange Init
10.0.2.15	147.229.176.19	SSHv2	2022	Client: Key Exchange Init
10.0.2.15	147.229.176.19	SSHv2	78	Client: Diffie-Hellman Group Exchange Request
147.229.176.19	10.0.2.15	SSHv2	462	Server: Diffie-Hellman Group Exchange Group
10.0.2.15	147.229.176.19	SSHv2	454	Client: Diffie-Hellman Group Exchange Init
147.229.176.19	10.0.2.15	SSHv2	1030	Server: Diffie-Hellman Group Exchange Reply, New Keys
10.0.2.15	147.229.176.19	SSHv2	70	Client: New Keys

Obr. 4.1: Súbor `SSH.pcap` v programe Wireshark

Tmavou farbou sú zobrazené pakety prichádzajúce od klienta. Na analýzu komunikácie sú dôležité správy `Key Exchange Init`. Program wireshark umožňuje dekódovať použitú verziu protokolu, výsledný algoritmus šifrovania kľúčov, mac a kompresiu. Výsledky sú zobrazené na obrázku 4.2.



Obr. 4.2: Výsledok analýzy `SSH.pcap` v programe Wireshark

4.2 Využitie logovacích súborov

Ďalšou možnosťou monitorovania, bez možnosti ukladať dáta, je použitie logovacích súborov. Výhodou je, že nie je potrebný žiaden špeciálny nástroj. Na operačnom systéme Linux sú logovacie informácie o prihlásení uživateľov dostupné v súbore `/var/log/auth.log`(výpis 4.1) a taktiež v súbore `/etc/log/lastlog`(výpis 4.2). Tieto informácie môžu slúžiť na jednoduché monitorovanie prihlásení uživateľov, čas udalostí, použité porty a pod.

Výpis 4.1: Príklad výpisu súboru `/var/log/auth.log`

```
18:20:45 localhost sshd[585]: Server listening on 0.0.0.0 port 22
18:20:45 localhost sshd[585]: Server listening on :: port 22
18:23:56 localhost login[673]: pam_unix(login:session):
        session opened for user root by LOGIN(uid=0)
18:23:56 localhost login[714]: ROOT LOGIN  on '/dev/tty1'
```

Výpis 4.2: Príklad výpisu súboru `/etc/log/lastlog`

```
demoer   pts/1        rrcs-72-43-115-1 Thu Sep  5 19:37   still
        logged in
root     pts/1        rrcs-72-43-115-1 Thu Sep  5 19:37 - 19:37
root     pts/0        rrcs-72-43-115-1 Thu Sep  5 19:15   still
        logged in
root     pts/0        rrcs-72-43-115-1 Thu Sep  5 18:35 - 18:44
root     pts/0        rrcs-72-43-115-1 Thu Sep  5 18:20 - 18:20
demoer   pts/0        rrcs-72-43-115-1 Thu Sep  5 18:19 - 18:19
```

V mojej práci sa ale budem venovať problematike analýzy sieťovej komunikácie na báze tokov a vytvoreniu zásuvneho modulu, ktorý na tomto princípe funguje.

Kapitola 5

Extrakcia dát z protokolu SSH

Výsledkom praktickej časti bakalárskej práce je vytvoriť zásuvný modul pre Flowmon exportér, ktorý analyzuje SSH komunikáciu. Hlavnou úlohou FlowMon exportéru je podpora rôznych vstupných a výstupných formátov ako aj spracovanie dát. Exportér dokáže prijímať pakety z vysokorýchlostných sietí, ktoré agreguje do tokov a vytvára záznamy. Použitie protokolov Netflow v9 a IPFIX narozdiel od Netflow v5 je flexibilné. Netflow v9 definuje 104 políčok a IPFIX viac než 400 a navyše umožňuje definovať nové vlastné políčka v rámci organizácie.

5.1 Architektúra FlowMon exportéru

Pre vytvorenie zásuvného modulu bolo potrebné naštudovať ako FlowMon exportér pracuje. Celý nástroj si môžme predstaviť ako zreťazenú linku, ktorá postupne spracováva pakety prichádzajúce na vstup. Architektúra tohto nástroja je zobrazená na obrázku 5.1 a jej spracovávanie môžme rozdeliť na niekoľko častí.

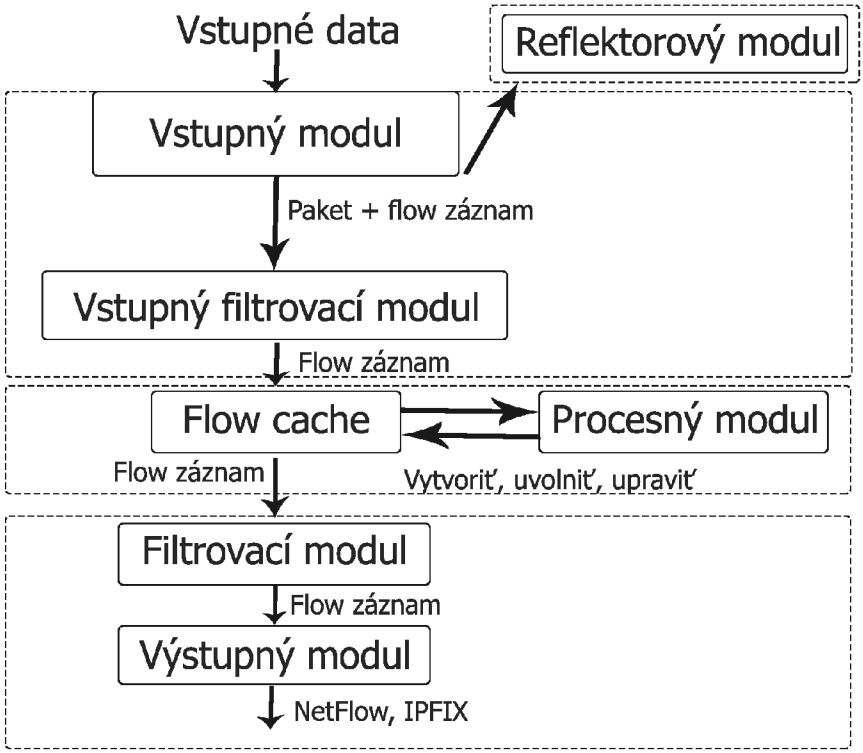
Na začiatku máme sieťový prúd paketov, ktorý príchádza do nášho exportéru. Tu sa dostávame do prvej časti, kde sa nachádza **vstupný** modul. Pakety sú spracované, vytvorí sa im záznam toku a uloží sa do pamäte(*anglicky flow cache*)[20].

Pri príchode ďalších paketov, pre ktoré už záznam o toku existuje, je existujúci záznam aktualizovaný. Zatiaľ, čo pakety prechádzajú spracovávaním, flow cache periodicky skenuje svoje záznamy a kontroluje časové razítka, ktoré detekujú prerušenie. Pri detekcií prerušenia je záznam predaný **exportovaciemu** modulu, voliteľne cez **filtrovací** modul.

Filtrovací modul rozhoduje, či daný paket bude predaný exportovaciemu modulu. Slúži na vybranie len takého toku, ktorý spĺňuje určité požiadavky, ako napríklad špecifický protokol alebo porty. Výstupný modul vytvára NetFlow alebo IPFIX správy a posiela ich na kolektor. Taktiež je možné použiť aj iné výstupné formáty ako sú csv, SQL databáza a pod.

Ďalším typom modulu je **reflektor**. Prijíma dáta zo vstupného modulu, ale jeho výstup nie je použitý v exportéri. Používa sa ako špeciálna časť exportéru na spracovanie alebo ukladanie vybraných paketov.

Exportér pre svoju prevádzku potrebuje práve jeden vstupný modul a jeden alebo viac výstupných. Je možné pre každý exportný modul definovať jeho filtrovací modul, ktorý sa môže použiť aj globálne pre každý výstupný modul. Je možné použiť ľubovoľný počet vstupných a procesných modulov, ale len jeden reflektor.



Obr. 5.1: Spôsob vytvorenia záznamu o toku

Zásuvné Moduly implementujú rozšírujúce funkcie exportéra. Poznáme 5 základných druhov:

Input zabezpečuje prijímanie dát zo vstupného rozhrania.

Reflektor prijíma data zo vstupného modulu, jeho výstup je použitý interne pre účely ukladania a spracovania paketov.

Process zabezpečuje dodatočné spracovanie paketov.

Filter zabezpečuje filtrovanie záznamov tokov.

Export zabezpečuje export záznamov tokov v rôznych formátoch.

Každý modul obsahuje 3 základné funkcie, ktoré musia byť implementované: `DESC`,`INIT` a `SHUTDOWN`.

DESC funkcia sa zavolá pri načítaní modulu do exportéra. Návratová hodnota je statický ukazateľ na štruktúru *plugin_desc_t*. Štruktúra obsahuje základné informácie o module, ako je meno alebo správa pre nápovedu `flowmonexp`. Celá štruktúra je zobrazená na výpise 5.1.

Výpis 5.1: Štruktúra plugin_desc_t

```
typedef struct {
        char *name;
        char *help;
        uint16_t data_size;
        int need_packet_input;
} plugin_desc_t;
```

Parameter `name` obsahuje meno modulu, parameter `help` obsahuje krátku nápovedu, ktorá sa zobrazí v nápovede exportéru. Parameter `data_size` udáva veľkosť dát v zázname a posledný parameter `need_packet_input` znamená, že procesný modul vyžaduje vstupný paket.

INIT funkcia je zavolaná práve jeden krát pri štarte exportéra a jej hlavnou úlohou je inicializovať potrebné štruktúry, registrovať *callback* funkcie a vykonať potrebné úlohy. INIT funkcia vracia ukazovateľ na pamäť modulu. Táto pamäť je použitá na ukladanie dát a adries. Pretože niektoré moduly sú načítané viac krát, je potrebné rozlíšiť ich pamäť.

SHUTDOWN funkcia sa zavolá pred ukončením exportéra. Slúži na ukončenie spojenia, uvoľnenie pamäte, výpis štatistík a pod. Implementácia tejto funkcie je voliteľná.

Každý modul musí na začiatku zavolať *SET_PLUGIN_TYPE()* makro, v ktorom nastaví hodnotu podľa typu modulu, ktorý implementuje. Tieto hodnoty sú definované v hlavičkovom súbore *plugin.h* a sú zobrazené vo výpise kódu 5.2

Výpis 5.2: SET_PLUGIN_TYPE

```
#define PLUGIN_TYPE_INPUT 0x01
#define PLUGIN_TYPE_EXPORT 0x02
#define PLUGIN_TYPE_PROCESS 0x04
#define PLUGIN_TYPE_FILTER 0x08
#define PLUGIN_TYPE_INPUT_FILTER 0x10
#define PLUGIN_TYPE_REFLECTOR 0x20
```

Je dôležité volať funkcie použitím definovaných makier. To zaistí, že všetky problémy spôsobené zmenou rozhrania sú obslužené počas prekladu.

5.2 Procesný zásuvný modul ssh

Hlavnou úlohou tejto práce bolo vytvoriť procesný zásuvný modul pre FlowMon exportér. Po naštudovaní možností monitorovania sietí na báze IP tokov a protokolu SSH som spracoval návrh implementácie procesného zásuvneho modulu `shh`. Tento modul príjíma dáta zo vstupného modulu a rozpoznáva nešifrované zprávy protokolu SSH. Získané záznamy tokov sú rozšírené o informácie získané z tohto protokolu. Výsledné toky sú za použitia výstupného zásuvného modulu `ipfix-ng` exportované protokolom IPFIX na kolektor. Výpis informácii je možný aj vo formáte csv, ktorý som použil najmä na testovacie účely.

Zdrojový kód procesného zásuvného modulu je rozdelený do 2 súborov: `ssh.c` a `ssh.h`, ktoré sa nachádzajú v prílohe v priečinku `src/flowmonexp/process-ssh`. Hlavičkový súbor `ssh.h` definuje konštanty, hlavičky funkcií a taktiež štruktúru ssh_record_t, v ktorej sú uložené získané dáta. Zdrojový subor `ssh.c` obsahuje implementáciu funkcií rozhrania a taktiež funkcie, ktoré sa starajú o spracovanie komunikácie protokolu SSH a pridávanie získaných informácií do tokov. Súčasťou prílohy je aj `Makefile`, ktorý preloží zdro-

jový kód a vytvorí zdieľaný objekt `plugin-process-ssh.so`. Tento objekt je možné použiť na spustenie exportéra s prepínačom `-X` a s cestou k tomuto objektu.

Procesný zásuvný modul `ssh` získava nešifrované informácie o komunikácií pri výmene kľúčov medzi klientom a serverom. Tieto informácie su ukladané do polí *IPFIX Information Elements*[10]. Zoznam exportovaných polí je zobrazený v tabuľke 5.1. Všetky exportované elementy majú pridelení identifikátor *enterprise number* s hodnotou 8057, ktorý je pridelený spoločnosti CESNET na definíciu vlastných polí.

Názov elementu	**ID**	**Popis**
SSH_VERSION	900	Verzia protokolu
SSH_KEX_ALGORITHM	901	Algoritmus použitý na výmenu kľúčov
SSH_HOST_KEY	902	Algoritmus podpisovania
SSH_ENCRYPTION_CLIENT_TO_SERVER	903	Šifrovanie klient-server
SSH_ENCRYPTION_SERVER_TO_CLIENT	904	Šifrovanie server-klient
SSH_COMPRESSION_CLIENT_TO_SERVER	905	Kompresia dát klient-server
SSH_COMPRESSION_SERVER_TO_CLIENT	906	Kompresia dát server-klient
SSH_MAC_CLIENT_TO_SERVER	907	Mac Algoritmus klient-server
SSH_MAC_SERVER_TO_CLIENT	908	Mac Algoritmus server-client

Tabuľka 5.1: Polia exportované zásuvným modulom `ssh`

5.3 Implementácia zásuvného modulu

Po analýze prevádzky a komunikácie SSH protokolu, som zistil, že pre získanie užitočných informácií, budú potrebné oba toky (klient-server, server-klient). Tým vznikol problém ako riešiť spracovávanie týchto paketov. Po prezretí riešení iných nástrojov som prišiel k názoru riešiť to pomocou párovania tokov. Tento princíp patrí medzi všeobecné riešenia a používa sa v rade nástrojov, napríklad vo Wireshark.

Pakety, ktoré vyžadujú tento postup nesú informáciu, aké algoritmy pre vytvorenie spojenia podporujú. Sú to algoritmy výmeny kľúčov, použitia kompresie, šifrovania a pod. Touto metódou je potrebné zistiť, aké algoritmy prenosu a vytvorenia spojenia budú použité. Taktiež, musím uvažovať, že pakety nemusia chodiť v rovnakom poradí alebo nemusia prísť vôbec.

Po príchode paketov a overení, že sa jedná o pakety SSH protokolu (zdrojový alebo cieľový port 22 a komunikujú na TCP) sú odoslané do funkcie na ďalšie spracovanie. Ak sa jedná o počiatočné pakety, zistí sa verzia protokolu a paket sa zahodí. Ak sa jedná o správu typu *Key Exchange Init*, paket je rozparsovaný a údaje uložené. Tu sa dostávame do miesta, kde je potrebné aplikovať metódu párovania tokov.

Pretože, pakety nemusia prísť v rovnakom poradí, rozhodol som sa pre uloženie informácií použiť jednosmerne viazaný zoznam `rArray` (výpis 5.3).

Výpis 5.3: Štruktúra rArray

```
struct rArray{
        struct rElement* Act;
    struct rElement* First;
} rArray;
```

Prvky tejto štruktúry sú štruktúry `rElement`, ktorá je zobrazená na výpise 5.4. Tieto štruktúry sú uložené v privátnej pamäti modulu a obsahujú sadu funkcií pre vkladanie, mazanie, hľadanie a spracovanie prvkov.

Výpis 5.4: Štruktúra rElement

```
struct rElement {
 struct rElement *ptr;
 unsigned int timestamp;
 bool valid;
 uint64_t ip_cert;
 uint8_t ssh_host_key[SSH_MERGE_FIELD_LENGTH];
 uint8_t ssh_kex_algorithm[SSH_MERGE_FIELD_LENGTH];
 uint8_t ssh_encryption_client_to_server[ENCRYPTION_SIZE];
 uint8_t ssh_encryption_server_to_client[ENCRYPTION_SIZE];
 uint8_t ssh_compression_client_to_server[COMPRESSION_SIZE];
 uint8_t ssh_compression_server_to_client[COMPRESSION_SIZE];
 uint8_t ssh_mac_client_to_server[MAC_SIZE];
 uint8_t ssh_mac_server_to_client[MAC_SIZE];
} rElement;
```

Premenná `timestamp` určuje čas vytvorenia záznamu, `valid` značí, že daný záznam je možné exportovať. Medzi ďalšie premenné patria polia, do ktorých sú ukladané všetky algoritmy, ktoré daná strana podporuje. Tieto položky spolu s verziou protokolu sú monitorované a analyzované pre protokol SSH verzie 2. U protokolu SSH verzie 1 je monitorovaná len verzia protokolu.

Získanie oboch strán komunikácie prebieha na základe funkcie `record_crc64()`, ktorá slúži k výpočtu CRC. Deklarácia funkcie je zobrazená na výpise 5.5 a nachádza sa v hlavičkovom súbore `record.h`. Parameter `record` označuje záznam, pre ktorý sa má vytvoriť CRC. Druhý parameter umožňuje vymeniť zdrojovú IP adresu s cieľovou a tak isto aj porty. To zaistí výpočet rovnakého CRC pre toky opačného smeru. Táto vlastnosť umožňuje toky najsť a zistiť výsledné použité algoritmy.

Výpis 5.5: Hlavička funkcie record_crc64()

```
uint64_t  record_crc64(flow_record_t * record , int
        invert_addr);
```

Použitím jednosmerne viazaného zoznamu, som vyriešil problém ak pakety nechodia v rovnakom poradí a vďaka CRC je možné stalé nájsť ich odpovedajúci tok. Niekedy sa ale môže stať, že paket druhej strany nedorazí. Vtedy je potrebné zaistiť, aby nedošlo k preplneniu zoznamu. To som vyriešil nastavením primeraného časového razítka, ktoré určuje dokedy je záznam potrebné uchovávať. Prechádzaním poľa záznamov, sa počas testovania dosahoval v priemere veľkosti 50 záznamov. Táto veľkosť však závisí od vyťaženia sieťovej prevádzky ssh komunikáciou.

V správach *Key Exchange Init* sa algoritmy prenášajú v textovom formáte. Z hľadiska výkonnosti, je lepšie pracovať s číselnými hodnotami. Preto pre výkonnostné optimalizácie sú algoritmom pridelené číselné hodnoty. Tieto hodnoty sú prenášane protokolom IPFIX na kolektor. Na kolektor sú posielané 8 bitové hodnoty, čím sa redukuje veľkosť dát, ktorú je potrebné odoslať. Samotné prekódovanie reťazcov na čísla zabezpečuje funkcia `StringToNumber()`, ktorej hlavička je zobrazená na výpise 5.6.

Výpis 5.6: Hlavička funkcia StringToNumber()

```
int StringToNumber(char **string,int criteria)
```

Parameter `**string` nesie ukazovateľ na pole znakov, kde je uložený názov algoritmu, ktorý chceme uložiť do poľa. Parameter `criteria` obsahuje názov políčka, ku ktorému táto správa patrí. Túto informáciu využíva pri neúspešnom dekódovaní algoritmu. Neúspešné dekódovanie je zaznamenané do súboru spolu s textovým reťazcom, ktorý sa nepodarilo premapovať.

Dekódovanie týchto informácií je umožnené pomocou štruktúry, ktorá obsahuje mapovanie algoritmov na číselné hodnoty. Ak sa nepodarí správu dekódovať v súbore `process-ssh.log` sa náchadza názov políčka a názov algoritmu, ktorý sa nepodarilo nájsť. V hlavičkovom súbore `ssh_codes.h` sú definované algoritmy pre polia:

- **Verzia protokolu** - napr. `SSH v1, SSH v2`
- **Host key** - napr. `rsa`
- **Algoritmus výmeny kľúčov** - napr. `diffie-hellman-group1-sha1`
- **Šifrovanie** - napr. `blowfish`
- **Mac** - napr. `hmac-md5-etm`
- **Kompresia** - napr. `none, zlib`

Všetky nové algoritmy, ktoré nie sú uložené sú zaznamenané a je možnosť ich jednoducho pridať do hlavičkového súboru `ssh_codes.h`.

Pakety vieme prijímať, uložiť do štruktúry v číselnych hodnotách a potom spracovať. Proces spracovania som použil podľa definovaného štandartu v RFC 4253 [8]. Funkcia `ParseRecord` po získaní správ z oboch strán komunikácie prechádza všetky záznamy klienta a hľadá prvý, ktorý podporuje server. Pri neúspešnom výbere výsledného algoritmu je návratová hodnota `UNDEFINED`, ktorá znamená že sa nepodarilo dohodnúť medzi stranami na použitých metódach. Okrem neúspešného výberu algoritmu funkcia umožňuje vrátiť hodnotu `TRUNCATED`, ktorá znamená, že paket bol rozdelený na časti a ostatné dáta neboli získané a spracované. Je možné detekovať aj problémy nenájdenia odpovedajúceho toku, čo spôsobí nemožnosť zistiť použité algoritmy. V tomto prípade sú dáta označené príznakom `UNRESOLVED`.

Dáta sú spracované a zistili sme výsledné použité algoritmy prenosu. Pre ukladanie výsledných informácií do IPFIX elementov slúži funkcia `setKex()`. Hlavička funkcie je zobrazená na výpise 5.7. Funkcia má ako parameter výslednú štruktúru funkcie `ParseRecord()`, ktorá obsahuje výsledné použité algoritmy komunikácie.

Výpis 5.7: Hlavička funkcie setKex()

```
static inline void setKex(flow_record_t * packet,
        ssh_record_t *record,plugin_private_t *p)
```

Táto funkcia overuje, či pre daný tok neexistuje už vytvorený záznam, ktorý je validný, dekódovaný. Po nájdení validného záznamu je tok rozšírený o novo získané informácie a následne exportovaný.

Kapitola 6

Výsledky meraní

V tejto kapitole je popísaný spôsob merania implementovaného zásuvného modulu `ssh`. Funkčnosť a výkonnosť modulu bola overená na reálnej sieti. V prvej fáze som modul testoval na `PCAP` súboroch obsahujúcich SSH prevádzku. Najprv som používal malé 7kb PCAP súbory obsahujúce len jeden tok SSH prevádzky. Jedným z nich je aj súbor `SSH.pcap` (príloha `/src/examples`), ktorý obsahuje 9 paketov SSH protokolu. Neskôr som testoval modul na PCAP súboroch s prevádzkou iných protokolov, ako sú http, dns, dhcp a pod. Tu som sa zameral, hlavne na použitie viacerých tokov, použitia rôznych druhov algoritmov a typov prístupu.

Sondou zachytené toky boli spracovávané a zbierané na kolektore. Pred spustením je potrebné na kolektore definovať položky, ktoré sú prenášané. Tieto informácie sú uvedené v priečinku `src/conf/ipfixcol` v súbore `ipfix-elements.xml`.

Sieťová komunikácia je spracovávaná programom flowmonexp, ktorý využíva vytvorený `ssh` modul. Získané dáta sú agregované do tokov a následne posielané na kolektor nástrojom (*ipfixcol*).Pre správne namapovanie exportovaných polí IPFIX elementov bolo potrebné exportované položky definovať v konfiguračných súboroch `ipfix-ng_fields.txt` a `ipfix-ng_template.txt`. Súbory sa nachádzajú v priečinku `src/conf/ipfix-ng`. Po úspešnom prijatí dát na kolektor je na analýzu a výpis týchto dát použitý nástroj *fbitdump*, ktorý je súčasťou *ipfixcol*. Tento nástroj umožňuje pracovať s dátami v databáze, dotazovať sa na konkrétne hodnoty a taktiež agregovať zaznamenané toky a zpracovávať ich obsah. Detailný popis možností využitia funkcií nástroja *fbitdump* je uvedený na jeho manuálových stránkach (*man fbitdump*). Súčasťou týchto nástrojov sú aj konfiguračné súbory, ktoré sú potrebné na správnu prevádzku nástroja. Pri spustení nástroja *fbitdump* je potrebné zadať cestu k súboru `fbitdump.xml`, ktorý obsahuje nastavenie pre formátovanie na textový výstup polí. Všetky potrebné konfiguračné súbory sa nachádzajú v priečinku `src/conf/fbitdump`.

Dáta na kolektor prichádzajú v číselných hodnotách a preto bolo potrebné implementovať modul `ssh_codes` pre nástroj fbitdump. Tento modul prekladá číselné hodnoty na odpovedajúce názvy algoritmov. Tento súbor je uložený v prílohe v priečinku `src/fbitdump`. Plugin využíva rovnaký hlavičkový súbor ako zásuvný modul, z dôvodu využívania rovnakého zoznamu podporovaných algoritmov.

6.1 Overenie funkčnosti modulu ssh

K overeniu implementovaného modulu `ssh` bol z hľadiska funkčnosti otestovaný na reálnej sieti a taktiež na nižšie uvedených PCAP súboroch `FullTraffic.cap`[1] a `SSHTraffic.pcap`. Obsah týchto PCAP súborov je popísaný nižšie pred jeho výpisom. V tejto časti popisujem aj príklady použitia nástrojov `flowmonexp,ipfixcol` a `fbitdump` na overenie zásuvneho modulu.

Pred samotným spustením nástroja `flowmonexp` je potrebné spustiť kolektor spolu s konfiguračnými súbormi, ktorý bude zasielané dáta prijímať (výpis 6.1).

Výpis 6.1: Spustenie kolektoru - nástroja `ipfixcol`

```
ipfixcol  -v 1
        -c src/conf/ipfixcol/startupUDP4799.xml
        -e src/conf/ipfixcol/ipfix-elements.xml
```

Po úspešnom spustení kolektoru je možné spustiť FlowMon exportér (výpis 6.2). Exportér analyzuje získané dáta, agreguje ich do tokov a posiela ich prostredníctvom správ IPFIX protokolu na kolektor. Na jeho vstup je vložený testovací PCAP súbor `FullTraffic.cap`, ktorý obsahuje sieťovú prevádzku protokolu SSH a HTTP s využitím honey pot[2]. Všetky použité `PCAP` súbory sa nachádzajú v prílohe v priečinku `src/examples`.

Výpis 6.2: Spustenie exportéra - nástroja `flowmonexp`

```
flowmonexp \
        -I pcap -replay:file=src/examples/FullTraffic.cap \
        -X ssh/plugin-process-ssh.so \
        -P ssh \
        -E ipfix-ng:port =4799, protocol=udp ,host=localhost ,\
        template -file=src/conf/ipfix-ng/ipfix-ng_template.txt ,\
        fields -file=src/conf/ipfix-ng/ipfix-ng_fields.txt
```

Dáta prijaté na kolektore je možné získať využitím nástroja `fbitdump`. Pri spustení je potrebné definovať cestu ku konfiguračnému súboru, ktorý obsahuje formát výpisu správ. Prepínač `-R` určuje priečinok alebo súbor získaných dát.(výpis 6.3).

Výpis 6.3: Príklad získania dát z kolektoru pomocou nástroja `fbitdump`

```
fbitdump \
        -C src/conf/fbitdump/fbitdump.xml \
        -R src/examples/ic20160508125500/ \
        -o ssh-all
```

Získané dáta analýzou súboru `FullTraffic.pcap` sú uvedené vo výpisoch 6.4, 6.5 a 6.6.

Výpis 6.4: časť záznamu `FullTraffic.pcap`

```
SSH Version      SSH Kex Algorithm                SSH Host Key
SSH-2          diffie-hellman-group1-sha1           ssh-rsa
SSH-2    diffie-hellman-group-exchange-sha256       ssh-rsa
```

[1] http://2009.hack.lu/index.php/InfoVisContest
[2] http://tools.ietf.org/html/rfc4949

Výpis 6.5: časť záznamu `FullTraffic.pcap`

```
SSH Encryption CTS      SSH Encryption STC    SSH Compression CTS
aes128-cbc                    aes128-cbc              none
aes128-cbc                    aes128-cbc              none
```

Výpis 6.6: časť záznamu `FullTraffic.pcap`

```
SSH Compression STC     SSH Mac CTS     SSH Mac STC
none                      hmac-sha1       hmac-sha1
none                      hmac-sha1       hmac-sha1
```

SSH Version udáva verziu použitého SSH protokolu.

SSH Kex Algorithm udáva algoritmus výmeny kľúčov.

SSH Encryption znamená použité šifrovanie (CTS - klient-server, STC - server-klient).

SSH Compression udáva použitú kompresiu (CTS - klient-server, STC - server-klient).

SSH Mac znamená použitú MAC (CTS - klient-server, STC - server-klient).

Na výpisoch je možné vidieť dva toky protokolu SSH, s použitím rôznych algoritmov výmeny kľúčov. Aj keď SSH protokol umožňuje použitie rôznych algoritmov pre každý smer, pri testovaní na reálnej prevádzke tieto algoritmy boli identické s opačnou stranou.

Ďalším z testovaných modulov je PCAP súbor `SSHTraffic.pcap`, ktorý obsahuje komunikáciu protokolom SSH s využitím rôznych typov autentifikácie a algoritmov. Súbor bol tvorený simulovaním prístupu klienta na server `merlin.fit.vutbr.cz`. Získané dáta sú uvedené vo výpisoch 6.7, 6.8 a 6.9.

Výpis 6.7: časť záznamu `SSHTraffic.pcap`

```
SSH Version      SSH Kex Algorithm                SSH Host Key
SSH-2    diffie-hellman-group-exchange-sha256    ssh-rsa
SSH-2    diffie-hellman-group-exchange-sha256    ssh-rsa
```

Výpis 6.8: časť záznamu `SSHTraffic.pcap`

```
SSH Encryption CTS      SSH Encryption STC    SSH Compression CTS
aes128-ctr                    aes128-ctr              zlib
aes128-cbc                    aes128-cbc              none
```

Výpis 6.9: časť záznamu `SSHTraffic.pcap`

```
SSH Compression STC     SSH Mac CTS     SSH Mac STC
zlib                      hmac-sha1-96    hmac-sha1-96
none                      hmac-sha1       hmac-sha1
```

Využil som možnosti SSH klienta na definovanie algoritmov, ktoré chcem použiť. Na výpisoch je možné vidieť rôzne typy šifrovania, mac a kompresie. Ak server nepodporoval nejaký algoritmus, klientovi nepovolil prístup.

6.2 Meranie výkonu

Po dokončení implementácie boli prevedené výkonnostné merania modulu. Testami bol zásuvný modul overený na dátach zo siete spoločnosti CESNET. Hlavným cieľom meraní som chcel zistiť zaťaženie exportéra pri použití implementovaného modulu a rozdiely v použití iných modulov. Testy boli zamerané na priepustnosť paketov a rýchlosť ich spracovávania. Dôležitou súčasťou bola aj analýza využitia pamäte a CPU. Všetky merania prebehli na stroji Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz so 64GB operačnej pamäti.

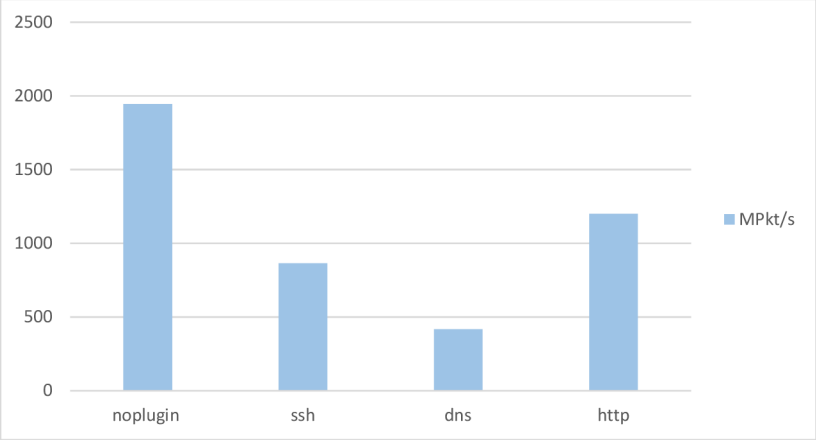
K testovaniu a analýze vlastností modulu bol použitý modul `benchmark` od CESNETu. Tento nástroj sa nachádza v prílohe v priečinku src/flowmonexp/benchmark. Ukážka testovania s použitím tohto zásuvného modulu je zobrazená na výpise 6.10.

Výpis 6.10: Ukážka testovania zásuvného modulu `ssh`

```
flowmonexp  -v0 -I benchmark:file=src/examples/SSHTraffic.pcap
        -P ssh   -E null
```

Meranie som rozdelil do dvoch častí. Pre porovnanie výsledkov boli použité moduly `http`, `dns` a exportér bol spustený aj bez zásuvneho procesného modulu (v grafe označený ako `noplugin`).

Na obrázku 6.1 je zobrazený graf spustenia exportéra využitím zásuvného modulu `benchmark` a PCAP súboru, ktorý obsahoval veľké množstvo SSH prevádzky. Okrem toho obsahoval aj pakety iných protokolov, napríklad `http` a `dns`. Zásuvný modul `noplugin` definuje maximálnu priepustnosť, ktorú je možné dosiahnuť použitím exportéra.

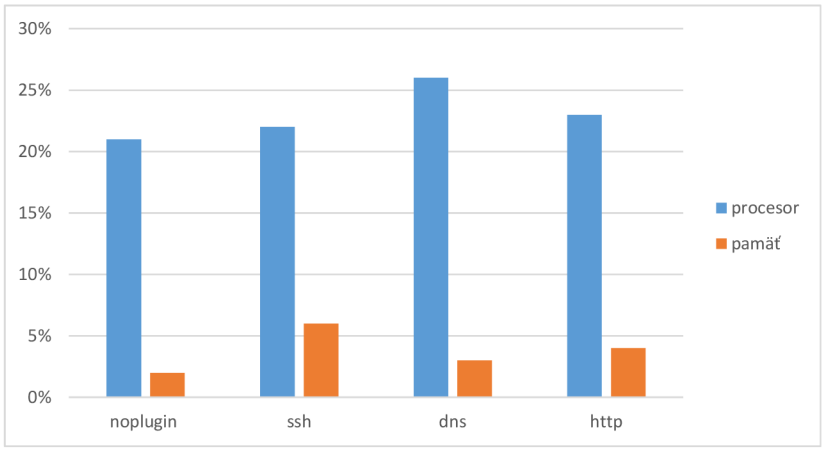


Obr. 6.1: Rýchlosť spracovávania paketov pri SSH komunikácií

Zníženie priepustnosti pre modul `ssh` je zapríčinené hlavne tým, že jednotlivé toky je potrebné párovať. Napriek tomu, je priepustnosť môjho modul pomerne vysoká v porovnaní s inými modulmi a nezaťažuje exportér pri jeho činnosti. Tieto výsledky su dané hlavne štruktúrou prevádzky, pretože len malá časť sieťovej komunikácie je tvorená SSH prevádzkou. Tým som docielil, že zásuvný modul `ssh` nebude ten, ktorý bude prevádzku brzdiť.

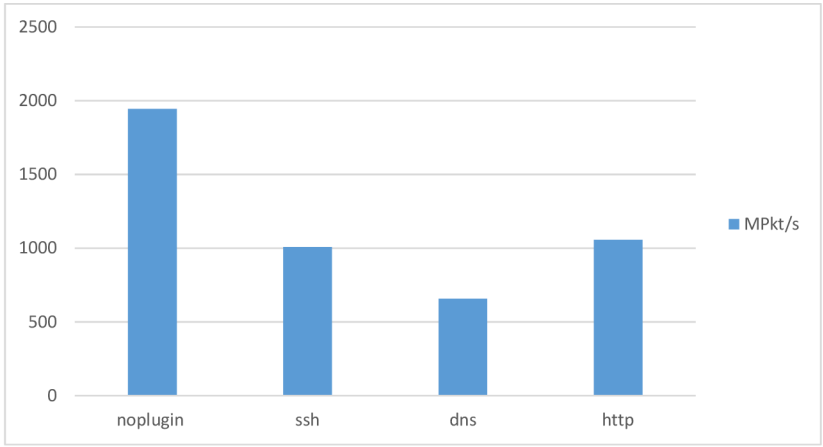
Ďalšou dôležitou vlastnosťou je zaťaženie pamäte a CPU. Na obrázku 6.2 sú zobrazené hodnoty, ktoré som nameral na testovanom PCAP súbore. Zvýšenie zaťaženia pamäte (obrázok 6.2) modulu `ssh` je zapríčinené ukladaním záznamov o tokoch. Tieto záznamy sú uložené len po určitú dobu. Zaťaženie pamäte následne po uvoľnení záznamu klesne.

Napriek tomu v porovnaní s ostatnými modulmi toto zaťaženie nie je vysoké, čo je vďaka štruktúre prevádzky a rade optimalizácií pre zaistenie vysokej efektivity spracovávania.

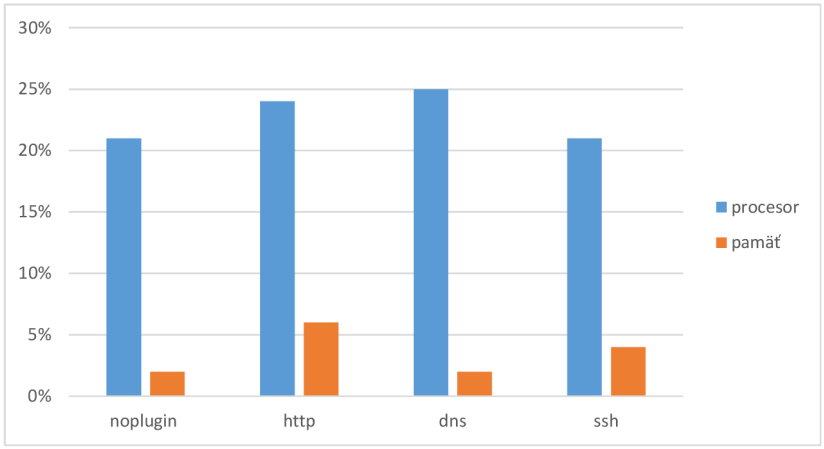


Obr. 6.2: Zaťaženie procesora a pamäte.

V ďalšom teste som použil PCAP súbor o veľkosti približne 300Mb, ktorý obsahoval veľké množstvo HTTP prevádzky, v menšom množstve SSH a DNS prevádzku. Na obrázku 6.3 si môžme všimnúť, že rýchlosť spracovávania sa zvýšila. Okrem zvýšenia rýchlosti spracovávania, zaťaženie pamäti modulu `ssh` kleslo (obrázok 6.4). Tento výsledok je spôsobený menším počtom SSH paketov, čím nie je potreba ukladať do pamäte také veľké množstvo dát.



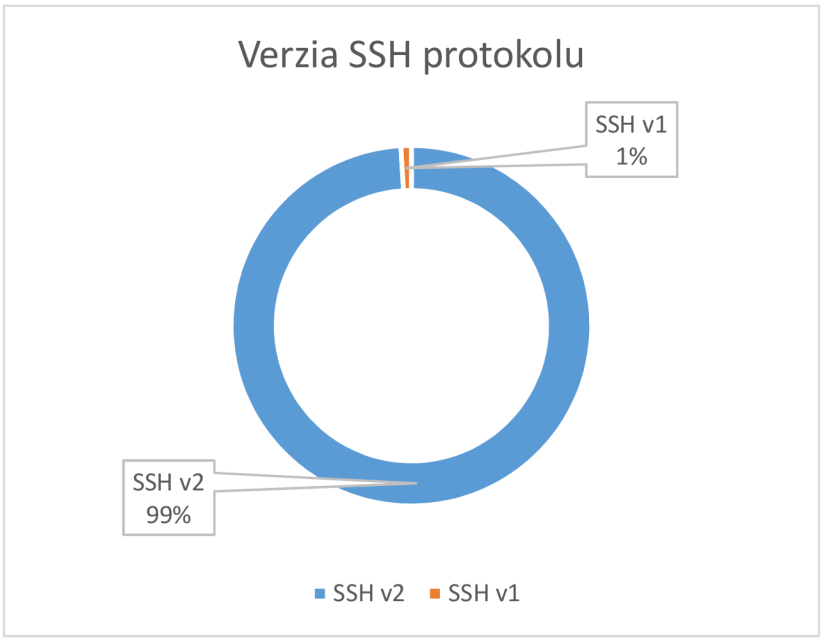
Obr. 6.3: Rýchlosť spracovávania paketov pri SSH komunikácií



Obr. 6.4: Zaťaženie procesora a pamäte.

6.3 Štatistika prevádzky

Po otestovaní procesného modulu `ssh` som ho nasadil na sieť spoločnosti CESNET. Exportér bol spúšťaný v rôzne dni počas jedného mesiaca, aby som získal najpresnejší obraz prevádzky. Podarilo sa mi zaznamenať viac než 50 000 spojení. Z nameraných dát som vypracoval štatistiku prevádzky reálnej siete. Medzi prvé položky, ktoré boli z hľadiska analýzy dôležité bola verzia protokolu.

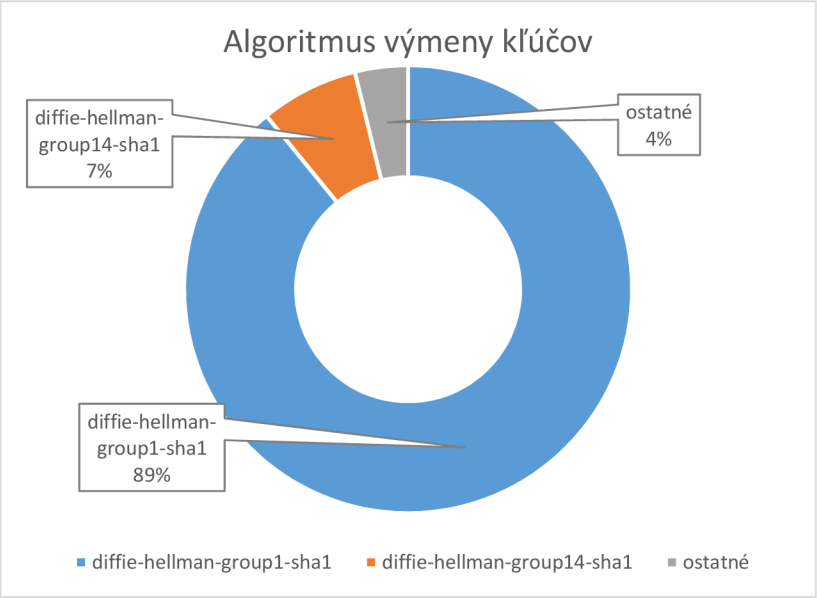


Obr. 6.5: Štatistika použitia verzie SSH protokolu.

Na obrázku 6.5 je možné vidieť, že väčšia časť klientov používa práve verziu 2, ktorá je efektívnejšia z hľadiska bezpečnosti. Ďalší dôvod môže byť aj ten, že väčšina serverov podporuje len SSH protokol verzie 2 a tak umožňuje pripojenie klientov len s touto verziou

protokolu.

Algoritmus výmeny kľúčov `Diffie-hellman` patrí medzi najčastešie používané metody, čo je aj možné vidieť na obrázku 6.6.



Obr. 6.6: Štatistika použitia algoritmu výmeny kľúčov.

Štatistické grafy pre ostatné monitorované položky sú zobrazené v prílohe A. Z nameraných dát som zistil, že väčšina klientov a serverov používa pri výtvaraní zabezpečeného spojenia tie isté typy algoritmov. Aj keď je možné na strane klienta definovať použitie iného algoritmu a servery to podporujú, väčšina klientov necháva tento výber na nastavení klientského programu.

SSH protokol patrí medzi zabezpečené spojenia a použitie stále tých istých algoritmov nevytvára také veľké riziko, aby to ohrozilo bezpečnosť klienta. Na druhej strane existujú metódy, ktoré využívajú slabosť algoritmov. Podľa článku, ktorý opisuje tzv. `logjam attack`[13] je možné prelomiť šifru až do 512 bitového šifrovania a ovládnuť pripojenie klienta (*Middle man attack*). Neoverené zdroje uvádzajú možnosť prelomenia až do 1024 bitov. Podľa štatistík sú najviac používané algoritmy `diffie-hellman-group1-sha1`, ktorý používa šifrovanie 1024 bitov a `diffie-hellman-group14-sha1`, ktorý využíva až 2048 bitovú šifru. Využitie týchto algoritmov považujem za dobré a postačujúce riešenie proti zabráneniu tohto typu útoku. Pretože medzi najdôležitejšie algoritmy patrí práve algoritmus výmeny kľúčov, ostatné nastavenia nechávajú väčšinou klientské programy podľa prednastavených hodnôt, čo je možné vidieť aj na priložených grafoch v prílohe.

Kapitola 7

Záver

Cieľom bakalárskej práce bolo zoznámiť sa s problematikou monitorovania SSH protokolu a zberu dát využitím architektúry IPFIX. Po zoznámení sa s problematikou nasledoval návrh riešenia, ktorého hlavnou požiadavkou bola možnosť výberu dát, ktoré sú relevantné pre protokol SSH.

Celý postup som rozdelil do niekoľkých častí. V prvej časti bolo potrebné zoznámiť sa s problematikou monitorovania sietí na princípe IP tokov a naštudovať dostupné riešenia. Následne mojím cieľom bolo poskytnúť všeobecný náhľad do tejto problematiky. V tejto časti boli vysvetlené najznámejšie spôsoby monitorovania sieťovej prevádzky založené na IP tokoch, ako sú protokoly NetFlow a IPFIX.

Okrem monitorovania sietí, bolo potrebné naštudovať problematiku protokolu SSH, možnosť jeho analýzy a získania dôležitých informácií. S analýzou tohto protokolu mi pomohol nástroj Wireshark, vďaka ktorému som stanovil doležité položky tohto protokolu pre monitorovanie. Po zoznamení sa s problematikou, som v tejto časti vysvetlil základné princípy komunikácie a autentizácie tohto protokolu.

Na základe štúdia a analýzy informácií som navrhol zásuvný modul `ssh` pre FlowMon exportér. Môj návrh je založený na párovaní tokov, čo je užívané riešenie v rade nástrojov, napríklad Wireshark. Mojím cieľom bolo vytvoriť modul, ktorý nebude vo veľkej miere zaťažovať exportér, čoho výsledkom bol zásuvný modul optimalizovaný na výkone pre zaistenie vysokej efektivity a rýchlosti spracovávania.

Posledná časť práce je venovaná meraniu jeho efektivity. Podľa testov, kde môj modul dosahoval rýchlosť spracovávania až 8000 MPkt/s vyplýva možnosť jeho využitia pre monitorovanie siete, tvorbu štatistík a detekciu anomálií. Nedochádza ku zníženiu výkonu kolektoru pri bežnej prevádzke a pamäťová náročnosť sa zvyšuje len minimálne.

Aj keď práca poskytuje modul, ktorý je schopný efektívne analyzovať a extrahovať položky z SSH prevádzky, je možné funkcionalitu ďalej rozširovať. Otvára priestor pre vývoj nástroja na detekciu možnosti *logjam* útoku a ďalších možnostiach detekcie slabých alebo často používaných šifrovacích algoritmov. Verím, že výsledok tejto práce prinesie širšie možnosti monitorovania sieťovej prevádzky, pomôže lepšie pochopiť problematiku, ktorú protokol SSH prináša a poskytne námet pre naväzujúce práce, či už z oblasti SSH alebo iných protokolov.

Literatúra

[1] *Internet Users.* [Online; navštívené 2016-03-03].
URL http://www.internetlivestats.com/internet-users/

[2] *Introduction to Cisco IOS NetFlow - A Technical Overview.* [Online; navštívené 2016-03-11].
URL http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

[3] *RFC 1950. ZLIB Compressed Data Format Specification version 3.3.* [Online; navštívené 2016-04-15].
URL https://www.ietf.org/rfc/rfc1950.txt

[4] *RFC 1951. DEFLATE Compressed Data Format Specification version 1.3.* [Online; navštívené 2016-05-02].
URL https://www.ietf.org/rfc/rfc1951.txt

[5] *RFC 3629. UTF-8, a transformation format of ISO 10646.* [Online; navštívené 2016-04-09].
URL https://tools.ietf.org/html/rfc3629

[6] *RFC 3917. Requirements for IP Flow Information Export (IPFIX).* [Online; navštívené 2016-04-04].
URL https://datatracker.ietf.org/doc/rfc3917/?include_text=1

[7] *RFC 4252. The Secure Shell (SSH) Authentication Protocol.* [Online; navštívené 2016-04-25].
URL https://datatracker.ietf.org/doc/rfc4252/

[8] *RFC 4253. The Secure Shell (SSH) Transport Layer Protocol.* [Online; navštívené 2016-04-05].
URL https://www.ietf.org/rfc/rfc4253.txt

[9] *RFC 4254. The Secure Shell (SSH) Connection Protocol.* [Online; navštívené 2016-04-10].
URL https://www.ietf.org/rfc/rfc4254.txt

[10] *RFC 5102. Information Model for IP Flow Information Export.* [Online; navštívené 2016-03-28].
URL https://tools.ietf.org/html/rfc5102

[11] *RFC 5470. Architecture for IP Flow Information Export.* [Online; navštívené 2016-04-05].
URL https://datatracker.ietf.org/doc/rfc5470/?include_text=1

[12] *SSH insertion attack*. [Online; navštívené 2016-04-07].
URL http://www.coresecurity.com/content/ssh-insertion-attack

[13] *Weak Diffie-Hellman and the Logjam Attack*. [Online; navštívené 2016-05-05].
URL https://weakdh.org/

[14] *Wireshark Download. Wireshark Go Deep.* [Online; navštívené 2016-03-06].
URL http://www.wireshark.org/download.html

[15] *RFC 3954. Cisco Systems NetFlow Services Export Version 9*. 2000 [cit. 2016-02-28], [Online; navštívené 2016-03-08].
URL https://www.ietf.org/rfc/rfc3954.txt

[16] Bauschert, T.: *Advances in Communication Networking*. Springer Heidelberg New York Dordrecht London, 2013, ISBN 978-3-642-40551-8.

[17] Daniel J. Barrett and Richard E. Silverman: *SSH kompletní prúvodce*. Computer Press, 2003, ISBN 80-7226-852-X.

[18] Leiner, B.; Cerf, V.; Clark, D.; aj.: *A brief history of the internet*. 2009, [Online; navštívené 2016-03-01].
URL http://dl.acm.org/citation.cfm?id=1629613

[19] Matoušek, P.: *Síťové aplikace a jejich architektura*. VUTIUM UK, 2014, ISBN 978-80-214-3766-1.

[20] Petr Velan: *FlowMon Exporter 3.05.x Documentation. 2013.*
URL https://www.invea.cz/trac/community/raw-attachment/wiki/WikiStart/flowmonexp-doc.pdf

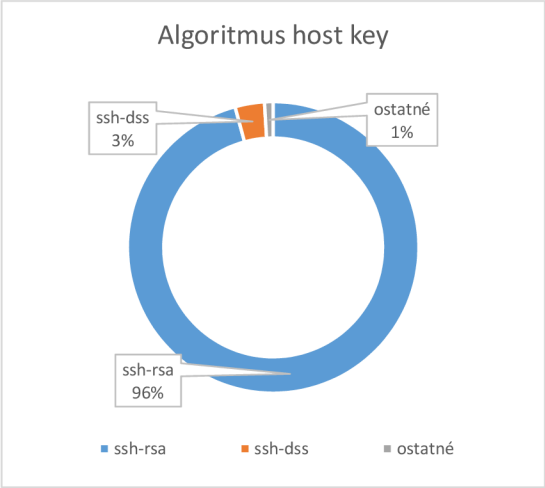
[21] Tranter, W. H.; Taylor, D. P.; Ziemer, R. E.; aj.: *OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection*. Wiley-IEEE Press, 2007, ISBN 978-0-470-54654-3.
URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5273216

Prílohy

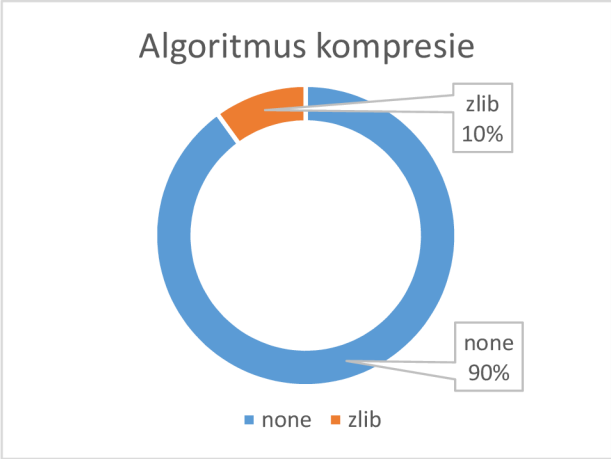
Zoznam príloh

Príloha A

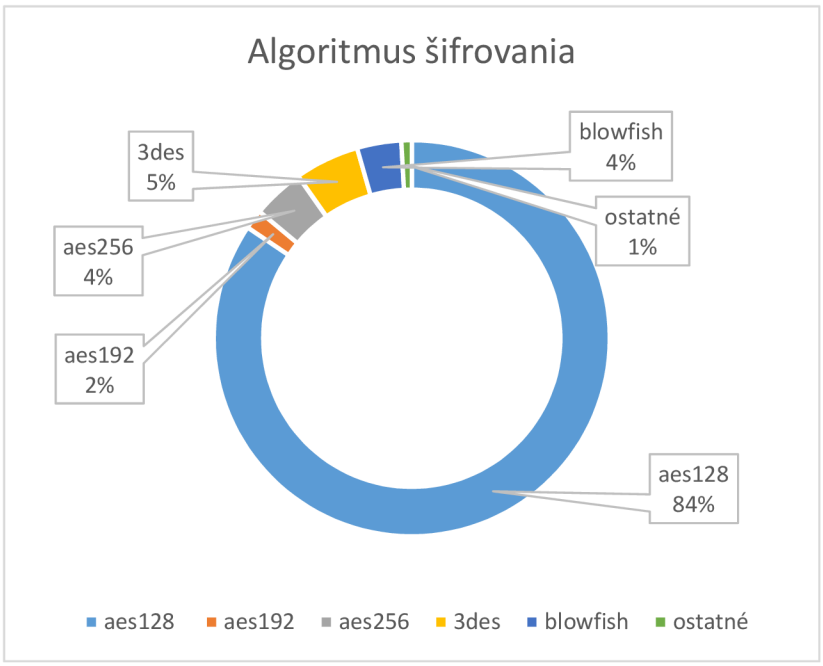
Štatistické grafy



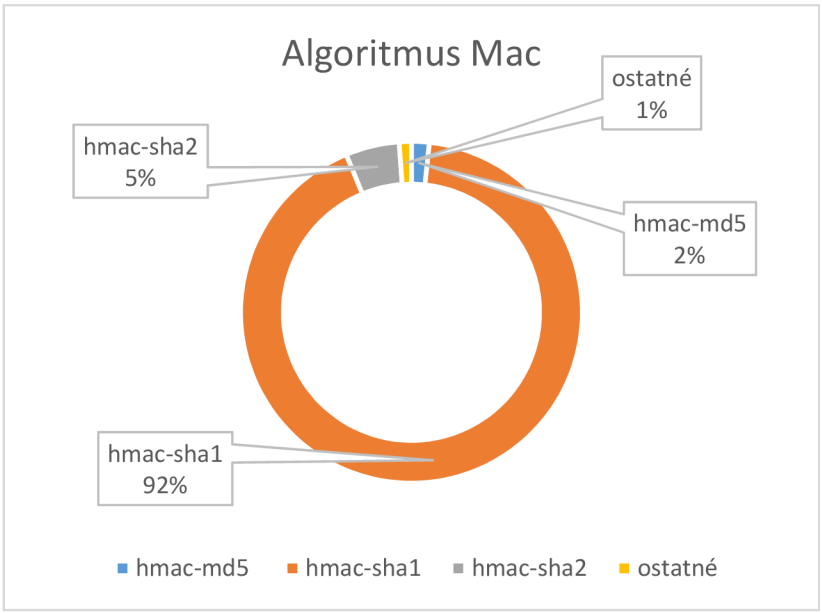
Obr. A.1: Štatistika použitia algoritmu host key.



Obr. A.2: Štatistika použitia kompresie.



Obr. A.3: Štatistika použitia šifrovania.



Obr. A.4: Štatistika použitia algoritmu mac.

Príloha B

Obsah DVD

- zdrojové kódy zásuvného modulu `ssh` a modulu pre nástroj `fbitdump`
- konfiguračné súbory pre nástroje `ipfixcol`, `fbitdump` a pre modul `ipfix-ng`
- zásuvný modul `benchmark`, ktorý slúži na testovanie výkonu modulu
- súbor `README`, ktorý obsahuje manuál k použitiu zásuvného modulu
- text práce vo formáte `PDF`