Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra aplikované matematiky

Diplomová práce

Turris plugin pro analýzu síťového provozu pomocí statistických metod

Bc. Marek Krátký

Vedoucí práce: Mgr. Rudolf Bohumil Blažek, Ph.D.

4. května 2015

Poděkování

Tímto bych chtěl poděkovat panu dr. Rudolfu B. Blažkovi za vedení práce a cenné rady. Dále Robinu Obůrkovi, Michalu Vanerovi a Bedřichu Košatovi za pomoc a konzultace ohledně projektu Turris. A také bych rád poděkoval své rodině a přátelům za podporu při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. května 2015 .....

Odkaz na tuto práci

Krátký, Marek. *Turris plugin pro analýzu síťového provozu pomocí statistických metod.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato diplomová práce se zabývá návrhem a realizací zásuvného modulu pro router vyvinutý v rámci výzkumného projektu Turris sdružení CZ.NIC, správce nejvyšši české DNS domény. Výsledný modul analyzuje síťový provoz a na základě statistické metody NP-CUSUM je schopen odhalit síťové anomálie. Práce popisuje projekt Turris, typy síťových anomálií a statistické metody zaměřené na detekci bodu změny. Součástí je také analýza dat z reálného provozu a otestování implementované metody.

Klíčová slova Turris, síťové anomálie, detekce bodu změny, NP-CUSUM, statistická detekce, monitorování síťového provozu

Abstract

This master thesis deals with design and implementation of a plugin for a router developed in a research project Turris of the Czech top level DNS domain maintainer CZ.NIC. The implemented plugin analyzes network traffic uses statistical method NP-CUSUM to detect network anomalies. The thesis describes the project Turris, types of network anomalies, and statistical methods with focus on change point detection. It also includes analysis of data from real network traffic and testing of the implemented method.

Keywords Turris, network anomalies, change point detection, NP-CUSUM, statistical detection, monitoring of network traffic

Obsah

Odkaz na tuto práci viii

Úvod **1**
Motivace a cíl práce 1
Struktura práce 2

1 Analýza **3**
1.1 Projekt Turris 3
1.1.1 Turris router 4
1.1.2 Monitorování 5
1.1.3 Ucollect 7
1.1.4 Lcollect 10
1.1.5 Centrální server 11
1.2 Analýza síťového provozu 11
1.3 Síťové anomálie 13
1.3.1 Typy anomálií 13
1.3.2 Detekce síťových anomálií 18
1.4 Statistické metody 21
1.4.1 Přehled používaných statistických metod 21
1.4.2 Sekvenční detekce bodu změny 21

2 Návrh **25**
2.1 Prostředí 25
2.1.1 PostgreSQL 25
2.2 Plugin 26
2.2.1 Plugin pro sběr dat 26
2.2.2 Plugin pro zasílání upozornění 28
2.3 Možnosti detekce 30
2.3.1 Detekce záplavy pakety SYN 31
2.3.2 Detekce nedostupné služby 31

2.3.3 Detekce skenování portů . . . 32
2.3.4 Detekce resetování TCP spojení . . . 32
2.3.5 Co nelze detekovat . . . 32
2.3.6 Nastavení prahu . . . 33

3 Realizace **35**
3.1 Vývojové prostředí . . . 35
3.2 Instalace . . . 35
3.3 Klientská část . . . 36
3.3.1 Inicializace . . . 37
3.3.2 Zpracování paketů . . . 37
3.3.3 Komunikace se serverem . . . 38
3.4 Serverová část . . . 39
3.4.1 Komunikace s klientem . . . 39
3.4.2 Nastavení konfiguračního souboru . . . 40
3.4.3 Databáze . . . 40
3.5 Přehled . . . 41

4 Testování **43**
4.1 Data . . . 43
4.2 Analýza dat . . . 43
4.2.1 Obecné statistiky . . . 44
4.2.2 Poměr paketů SYN a FIN . . . 44
4.2.3 Poměr paketů SYN a SYN+ACK . . . 49
4.2.4 Určení parametrů . . . 49

Závěr **51**

Literatura **53**

A Přehled počtů TCP paketů **57**

B Seznam použitých zkratek **59**

C Obsah přiloženého CD **61**

Seznam obrázků

1.1 Schéma sítě Turris 4
1.2 Schéma moderní NetFlow architektury 12
1.3 TCP handshake 15
1.4 Komunikace pomocí TCP protokolu 18
1.5 Ilustrace detekce anomálie v symetrickém síťovém provozu 19
1.6 Ilustrace průběhu metody NP-CUSUM 24

3.1 Virtuální síť pro testování pluginu 36

4.1 Počet paketů SYN a FIN 45
4.2 Poměr paketů SYN a FIN 45
4.3 Poměr paketů SYN a FIN klienta bez anomálie 46
4.4 Poměr paketů SYN a FIN klienta bez anomálie - v hodinách . . . 46
4.5 Průběh metody NP-CUSUM s daty klienta bez anomálie 47
4.6 Poměr paketů SYN a FIN klienta s anomálií 48
4.7 Průběh metody NP-CUSUM s daty klienta s anomálií 48
4.8 Počet paketů SYN a SYN-ACK 49

A.1 Počet paketů SYN, FIN a SYN-ACK 58

Seznam tabulek

2.1 Konfigurační proměnné 28
2.2 Databázové sloupce a jejich datové typy 29
2.3 Databázové sloupce pro konfiguraci pluginu 30
2.4 Databázové sloupce a jejich datové typy 31

4.1 Statistika poměru paketů 44

Úvod

V posledních letech se stále více diskutuje o bezpečnosti počítačových sítí a o síťových útocích. Tyto útoky se snaží využít slabých míst softwaru či operačního systému, který je nainstalován na cílovém počítači.

Bezpečnost počítačové sítě ale není možné zajistit, pokud administrátor nemá podrobný přehled o tom, co se v síti v daný okamžik děje. Proto s rozvojem počítačových sítí stoupá i potřeba jejich monitorování [1]. Díky monitorování sítě je pak možné například odhalení bezpečnostní slabiny, zjištění míry využití přenosové rychlosti, zjištění nefunkčnosti nějakého prvku sítě nebo odhalení již zmíněných síťových útoků [2].

Motivace a cíl práce

Pro monitorování síťového provozu existuje mnoho populárních i málo známých nástrojů. Většina těchto nástrojů je využita především pro monitorování provozu v malých či velkých firmách, univerzitách nebo ve státních institucích [3]. Naproti tomu výzkumný projekt Turris se specializuje na koncové uživatele a jejich lokální sítě [4]. Právě zaměření na domácí sítě a fakt, že projekt Turris je distribuovaný a poměrně nový (vznikl na začátku roku 2013), vedly k vytvoření tohoto zadání a vypracování této diplomové práce. Umožní totiž nasazení statistických metod pro detekci síťových útoků v domácích sítích.

Hlavním cílem práce je vytvoření zásuvného modulu pro projekt Turris, který bude schopen pomocí statistické metody odhalit anomálie v síti. Práce by měla sloužit také jako návod, jak takový modul vytvořit, a má za úkol budoucím studentům FIT ČVUT v Praze a ostatním čtenářům přiblížit možnosti a funkce implementované v projektu Turris. Po přečtení by si tedy vývojáři dalších modulů měli utvořit představu, co vše je pro dokončení a následné spuštění zásuvného modulu potřeba. Vyvinutý modul lze tedy vnímat jako prototyp pro další vývoj a výzkum.

Struktura práce

Práce se dělí na několik kapitol, kde každá z nich popisuje určitou část procesu vývoje zásuvného modulu. První kapitola se zabývá právě analýzou projektu Turris, jeho již existujících modulů a obecně jeho možnostmi v monitorování provozu. Dále také seznamuje čtenáře s existujícími síťovými anomáliemi, které ovlivňují provoz a četnost TCP paketů. V této kapitole je také rozebrána statistická metoda užitá v modulu pro detekci anomálií.

Ve druhé kapitole je podrobně rozepsán návrh modulu a jaké části je nutné vytvořit. Jak bude vypadat výstup dat a s jakými konfiguračními proměnnými je nutné počítat. Následující kapitola pak přiblíží konkrétní implementaci modulu, vytvořené prostředí pro implementaci a seznámí čtenáře s důležitými metodami, které je při implementaci vhodné využít.

Poslední kapitola popisuje způsob testování modulu a analýzu dat. Pomocí výsledků této analýzy potom navrhne volbu konfiguračních proměnných, pro které bude statistická metoda dávat rozumná řešení.

V závěru této práce je možné nalézt shrnutí výsledků, porovnání předem vytyčených a dosažených cílů a případné návrhy na další modifikace modulů.

Kapitola 1

Analýza

Předtím, než se tato práce bude zabývat návrhem pluginu pro projekt Turris, je nutné seznámit se s technologiemi, postupy a pojmy používanými v dalších kapitolách této práce. Analytická část je věnována především seznámení se se samotným projektem Turris, jeho částmi a jeho možnostmi analýzy síťového provozu.

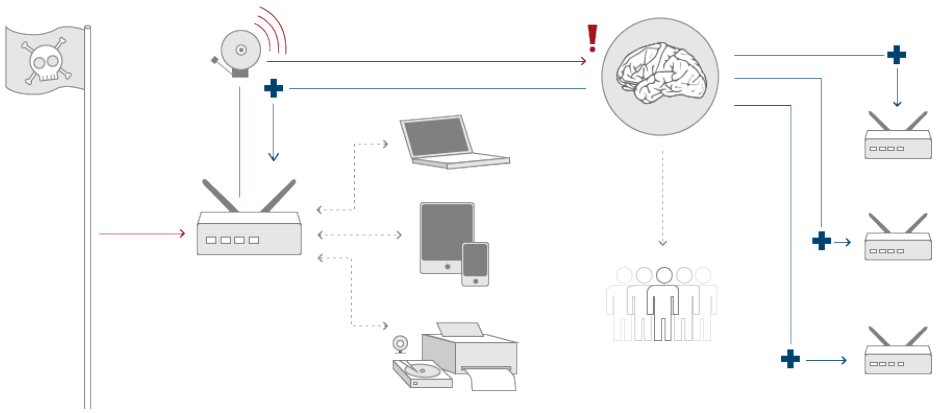
V dalších sekcích analytické části jsou rozebrány typy síťových útoků a obecně síťových anomálií, které je možné se zařízením Turris odhalit. Tématem této části je také přiblížení používaných statistických metod umožňujících detekci změn v síťovém provozu.

1.1 Projekt Turris

Turris je výzkumný projekt neziskového sdružení CZ.NIC, z.s.p.o.[4], správce české národní domény .CZ. Název projektu je odvozen z latinského slova pro věž, která často sloužila k včasnému varování před hrozícím nebezpečím.

Hlavní myšlenkou projektu je vývoj domácího WiFi routeru Turris, který neplní jen funkci routeru, ale obsahuje také programové vybavení schopné analyzovat provoz v síti a případně detekovat podezřelé datové toky. Pokud takováto situace nastane, router nahlásí možný útok centrálnímu serveru a ten pak může data s pomocí informací z ostatních připojených routerů vyhodnotit. Router zde tedy slouží jako strážní věž. Pokud centrála opravdu odhalila útok, jsou vytvořeny aktualizace firewallu, které jsou následně distribuovány do celé sítě Turris. Poté je firewall připraven bránit se proti podobným útokům a chránit tak uživatele.

Do projektu Turris se může zapojit každý, stačí se pouze zaregistrovat a počkat, zda obdržíme pozvánku. Z důvodu vysoké poptávky se však může stát, že zájemce bude muset podstoupit dlouhé čekání. Uživatel router obdrží za měsíční pronájem ve výši 1 koruny, kterou navíc z důvodu zjednodušení účtování uhradí ředitel společnosti CZ.NIC.



Obrázek 1.1: Schéma sítě Turris (zdroj obrázku [4]).

Pro lepší pochopení je možné vidět na obr. 1.1, jak taková síť Turris vypadá. Je složena z routerů, za kterými jsou připojena různá zařízení, jako například laptopy či mobilní telefony. Obrázek mozku zde značí centrální počítač, který zpracovává informace obdržené z routerů. Zpracovaná data centrálním serverem nakonec vyhodnotí skupina lidí, starající se o chod projektu Turris.

Je nutno podotknout, že projekt Turris je stále ve vývoji, a tak je možné, že informace v této práci mohou být v době čtení nekompletní či zastaralé.

1.1.1 Turris router

V této části jsou uvedeny informace o zařízení Turris, které se chová jako chytřejší směrovač. Sdružení CZ.NIC vyvíjí nejen software, ale i hardware. Veškeré výstupy však zveřejňují pod licencí open-source.[1]

1.1.1.1 Hardware

Nutnost vývoje vlastního hardware souvisí především s rychlostí připojení. Zařízení musí být dostatečně výkonné, aby zvládlo analýzu síťového provozu nejen dnes, ale i v budoucnu. Během analyzování provozu může nastat situace, kdy je router zahlcen příliš mnoha pakety a nestíhá je zpracovávat. V takovém případě je občas nějaký paket zahozen a tedy část informací je ztracena. S výkonnějším hardwarem tedy logicky klesá podíl takto zahozených paketů a analýza je přesnější.

[1] Open-source software (česky otevřený software) je počítačový software s otevřeným zdrojovým kódem. Licence umožňuje technickou dostupnost kódu, legální dostupnost, využívání či upravování software.

Vývoj první verze routeru trval od dubna roku 2013 a již v lednu 2014 byla ukončena výroba série 1000 zařízení. Od února téhož roku již začala distribuce koncovým uživatelům [4].

Parametry zařízení

- Procesor Freescale P2020 taktovaný na 1200 MHz
- 2 GB DDR3 RAM v SO-DIMM slotu
- 16 MB NOR a 256 MB NAND flash paměť
- Samostatný gigabitový WAN a 5 gigabitových LAN portů (s využitím switch chipu QCA8337N)
- Wifi 802.11a/b/g/n s 3x3 MIMO a odnímatelnými anténami
- 2x USB 2.0 port
- 1 volný miniPCIe slot
- UART, SPI a I2C připojené na pin-header pro snadnou rozšiřitelnost

1.1.1.2 Software

Router obsahuje operační systém založený na Linuxu. Vývojáři místo tvorby vlastního operačního systému zvolili existující projekt systému pro WiFi routery OpenWrt[2] [5, 4]. Díky tomu může uživatel instalovat libovolné balíčky dostupné pro tuto distribuci a obecně využít všech výhod tohoto systému.

Výsledný systém splňuje potřebné důležité vlastnosti a je postaven tak, aby byl snadno upravitelný a rozšiřitelný. Implementuje například podporu pro DNSSEC či IPv6 včetně firewallu a přechodových mechanismů z IPv4 k IPv6 [6]. Jelikož se jedná o zařízení poskytující síťovou konektivitu velkému množství uživatelů, bezpečnost zde hraje obrovskou roli.

Ke sběru dat routerem Turris slouží aplikace ucollect. Popis této aplikace a dalších funkcionalit jsou obsaženy v následujících sekcích.

1.1.2 Monitorování

Router Turris je schopný v reálném čase zpracovávat a analyzovat provoz na síťovém rozhraní [7]. Některé tyto informace pak zasílá na centrální server k analýze. Hlavní nástroje k získání dat a monitorování sítě jsou:

[2]OpenWrt je operační systém postavený na linuxovém jádře. Je primárně určen pro směrovače na vestavěných systémech

Ucollect: Asi nejdůležitější nástroj vytvořený vývojáři projektu Turris a pracující na klientských routerech i na centrálním serveru je ucollect. Tento nástroj zajišťuje sběr a odesílání dat k analýze. Možnosti, jak využít ucollect, jsou ale omezeny a to především závazky definovanými ve smlouvě.

Ucollect sbírá pouze hlavičky paketů a to v obou směrech na rozhraní do internetu (WAN). Vedou se i statistiky kolik paketů bylo zpracováno a kolik jich bylo zahozeno z důvodu nestíhání.

Nikola: Nikola analyzuje logy z firewallu (IPTables[3]). Posílá záznamy o paketech, které jsou zachyceny ve firewallu - obvykle jsou to pokusy z vnějšku připojit se na neexistující služby (např. útoky lámání hesla na SSH či skenování portů).

Logsend: Zasílá logy z automatických updatů firmwaru routeru a ze sběracího softwaru. Toto slouží k odhalování problémů s celkovým zdravím routeru.

Lcollect: Lcollect neodesílá žádná data na centrální server. Díky tomu může sbírat data i v lokální síti a může ukládat obě strany komunikace. Analýza dat probíhá pouze lokálně na zařízení uživatele a slouží pouze uživateli pro obecný přehled o dění v síti.

1.1.2.1 Firewall

Distribuovaný adaptivní firewall je hlavním prvkem aktivní ochrany. Snaží se reagovat na nové bezpečnostní hrozby. Na základě analýzy dat z jednotlivých routerů centrálním prvkem mohou vzniknout nová firewallová pravidla, která jsou následně zaslána na koncová zařízení, tedy routery Turris. Pravidla pro firewall vznikají ale i na základě informací z jiných veřejných či specializovaných zdrojů [4].

1.1.2.2 Soukromí

Jakékoliv sledování uživatelského provozu je velmi citlivá věc. Proto se sdružení okolo projektu Turris snaží o naprostou otevřenost, ať jde o zdrojové kódy aplikace, design hardwaru či smluvní podmínky užití.

Sama aplikace na analýzu síťových toků kontroluje pouze hlavičky paketů, nikoliv jejich tělo. Následná komunikace klienta s centrálním serverem je šifrovaná [4].

Všechny informace, které obdržel centrální server jsou spojeny jen s koncovými zařízeními, není zde žádná informace o identitě uživatelů. Informace

[3]IPTables je název pro nástroj Linuxu sloužící k nastavování pravidel firewallu v jádře [8].

o uživatelích jsou připojené jen pokud jsou vytvářeny přehledy pro jednotlivé uživatele a ty jsou dostupné pouze s uživatelským účtem a přístupovým heslem na stránkách projektu.

1.1.3 Ucollect

Framework ucollect je asi nejdůležitější součást programového vybavení routeru. Tento framework[4] na síťovém rozhraní klienta zpracovává pakety, které poté zpřístupní zásuvným modulům. Dále poskytuje spojení mezi jednotlivými klienty (t.j. routery Turris) a centrálním prvkem, při kterém si navzájem zasílají konfigurace nebo nasbíraná data pomocí jednotlivých zásuvných modulů [9].

Pomocí konfiguračního souboru je možné určit, na kterém síťovém rozhraní je možno pakety zachytávat, které zásuvné moduly se mají při startu aktivovat a v neposlední řadě také na jaký server se připojit a jaké přístupové jméno a heslo použít. Framework ucollect je napsán v programovacím jazyce C.

1.1.3.1 Zásuvné moduly

Ve frameworku probíhá veškerá analýza a řízení sběru požadovaných dat o provozu přes tak zvané zásuvné moduly (pluginy)[5]. Jejich výhoda tkví především v možnosti jednoduchého rozšíření a úpravy funkcionality aplikace. Jak již bylo zmíněno, ucollect se stará hlavně o sběr paketů a o komunikaci se serverem. Každá z těchto funkcí nabízí své rozhraní, které mohou jednotlivé pluginy libovolně využívat. Můžou tedy existovat pluginy, které využívají pouze komunikační rozhraní a pakety je nezajímají, či naopak využívají informace o paketech, ale svá data již neodesílají na server.

Při běhu aplikace každý plugin dostane v callbacku paket již zpracovaný, a tak může ke svým potřebám využívat strukturu, která se jmenuje packet_info. V této struktuře jsou obsaženy informace, jako například adresa, port, velikost paketu, protokol či zda jde paket směrem ven z lokální sítě nebo dovnitř [9].

Zde je popis všech pluginů v současnosti v aplikaci:

Count: Asi nejjednodušší plugin, který pouze třídí a počítá pakety podle různých informací v jejich hlavičkách. Jednou za čas server zažádá o statistiky, které plugin odešle a nastaví svá počítadla na nulu.

Zásuvný modul uchovává počítadla pro následující druhy paketů:

- všechny pakety
- pakety zaslané přes protokol IPv4

[4] Framework je softwarová struktura, sloužící jako podpora při programování. Tato struktura většinou nabízí různé knihovny, podpůrné programy či návrhové vzory.

[5] Plugin je software, který není schopen pracovat samostatně, ale slouží jako doplňkový modul jiné aplikace. Aplikaci rozšiřuje o novou funkčnost.

- pakety zaslané přes protokol IPv6
- pouze přijaté IPv4 a IPv6 pakety
- pouze odeslané IPv4 a IPv6 pakety
- TCP pakety
- UDP pakety
- ICMP pakety
- pakety se vzdáleným portem menším nebo rovným 1024
- TCP pakety s nastaveným SYN příznakem
- TCP pakety s nastaveným FIN příznakem
- TCP pakety s nastaveným SYN+ACK příznakem
- TCP pakety s nastaveným ACK příznakem
- TCP pakety s nastaveným PUSH příznakem
- pakety pro komunikaci se spojeným serverem
- IPv6 pakety zaslané skrz tunel

Bandwidth: Tento plugin pasivně měří rychlost připojení, jak rychlost stahování, tak i rychlost nahrávání. Plugin se snaží získat počet bytů za sekundu za použití okýnek, která mají různou délku v mikrosekundách. Počet okýnek i jejich délka je konfigurovatelná.

Algoritmus vkládá příchozí pakety do skupiny klouzavých okýnek v čase, která slouží jako buffer. V jednu chvíli buffer přeteče a tím je skupina okýnek podezřelá z globálního maxima. Konečná hodnota se nakonec musí vypočítat z bytů za délku okénka a výsledek je pak v bitech za sekundu.

Buckets: Pravděpodobně nejsložitější plugin v aplikaci Ucollect. Jedná se o statistické rozšíření pro detekci anomálií. Metoda je inspirována prací „Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures“ od G. Dewaele, K. Fukuda, P. Borgnat, P. Abry a K. Cho [10].

Tato metoda využívá hashovací funkci spolu se skupinou „přihrádek“ (buckets). Pokaždé když přijde paket, plugin analyzuje obdržené informace a uchová si ty, které jsou pro metodu zajímavé (například IP adresu či port). Tyto informace (sloužící jako klíče pro hashovací funkci) zahashuje a přiřadí do příslušné přihrádky. Systém se podobá hash mapě. V přihrádkách se neuchovávají celé pakety, pouze počty příslušných paketů, tudíž přiřazením do přihrádek se pouze inkrementuje příslušné počítadlo.

Po krátké době se vymění stará sada přihrádek za novou a stará sada se uchová. Jednou za čas si server vyžádá počty k analýze. Po kontrole dat

a porovnání s daty jiných klientů nebo historickými daty, může server vyžádat po klientech klíče (informace z paketů) k indexům (zahashované klíče), které nevypadají statisticky správně. Tyto klíče klient uchovává v malém množství, právě pro případ, že by si je server vyžádal.

Flow: Tento zásuvný modul slouží pro sběr síťových toků. Modul seskupí pakety, které mají stejnou adresu a port a spočítá, jakým směrem byly zaslány. Nejedná se ale o protokol NetFlow [2], protože nesbírá všechny statistiky definované v tomto protokolu. Čas od času jsou tyto informace zaslány na server. O intenzitě zasílání rozhoduje klient, a to pokud uplyne příliš dlouhá doba od minulého zaslání, nebo obdržel příliš mnoho informací o tocích v síti.

Plugin ovšem nedetekuje veškeré toky, nýbrž pouze ty, které projdou předem určeným filtrem. Ostatní toky jsou ignorovány.

Refused: Slouží ke sledování odchozích odmítnutých spojení. Jakmile uplyne nějaká doba, nebo již příliš mnoho informací je uloženo, plugin odešle informace serveru o každém nezdařeném spojení zevnitř sítě.

Účelem tohoto pluginu je především detekovat malware, který se snaží při spuštění připojit k svým serverům, které ale již nemusí být funkční, a tak spojení odmítnou.

Při každé nějakým způsobem zajímavé události plugin vytvoří záznam a jakmile je dostatek těchto záznamů a tedy je možné určit zda spojení bylo úspěšné či ne, je spojení označeno.

Za zajímavé události se považuje:

- paket s příznakem SYN, ale bez ACK - inicializace spojení
- paket s příznakem SYN a ACK - potvrzení spojení
- paket s příznakem RST - odmítnutí spojení
- ICMP paket s nedostupnou cílovou adresou
- vypršení času

Sniff: Zásuvný modul sniff je po většinu času nečinný. Nekontroluje procházející pakety ani nic jiného v síti. Jakmile ale přijde požadavek na spuštění nějakého úkolu ze serveru, plugin rozpozná, o jaký typ úkolu se jedná, získá od serveru parametry a spustí daný proces v pozadí. Modul je schopný udržovat v běhu několik předdefinovaných typů úkolů zároveň a i každý typ může běžet vícekrát například s jinými parametry. Viz příklady uvedené níže.

Plugin tedy dohlíží na procesy běžící v pozadí a ukládá jejich výstupy. Jakmile je proces ukončen, uložené výstupy jsou podle typu úkolu zpracovány a připraveny jako odpověď serveru.

Přehled možných typů úkolů:

- The NAT detector – zjišťuje zda je klient připojen pomocí veřejné IP adresy nebo přes NAT.
- The pinger – při tomto úkolu obdrží klient seznam hostitelů a zkouší provést ping na každého z nich. Serveru poté nahlásí získané IP adresy a dobu trvání požadavku.
- The certificate downloader – cílem tohoto typu úkolu je stáhnout certifikát služeb. Klient od serveru obdrží seznam služeb, pro které si stáhne certifikáty a serveru nazpět pošle informace, jež si vyžádal.
- The „NOP“ task – typ úkolu, který nic nedělá, slouží pro vnitřní užití protokolu

Spoof: Slouží k ověření, zda poskytovatelé internetového připojení správně blokují pakety s podvrženými adresami. Na základě požadavku plugin pošle serveru dva pakety. Jeden paket se správnou zdrojovou adresou, který slouží k ověření dostupnosti a druhý paket s úmyslně podvrženou IP adresou, který má být poskytovatelem připojení zablokován. Server po obdržení jednoho či dvou paketů spojí adresu s klientem a výsledek uloží do databáze.

1.1.3.2 Komunikace se serverem

Ucollect zajišťuje, aby data zaslaná lokálním pluginem byla zpracována odpovídajícím pluginem na serveru. Je možné naprogramovat serverový plugin tak, aby čekal na data od klienta, nebo také klient může čekat než server zašle požadavek, a pak teprve svá data odešle.

Server se dotazuje také proto, aby poznal, zda je zařízení připojeno. Podle smlouvy s uživatelem routeru může být zařízení Turris odpojeno od sítě pouze 720 hodin (30 dnů) za 6 měsíců.

Funguje to tedy tak, že po připojení klienta do sítě a startu aplikace ucollect se aplikace připojí k serveru. Komunikace mezi serverem a klientem je šifrovaná. Identity jsou kontrolovány pomocí SSL certifikátu. Jak klient, tak server využívá knihovnu libatsha204 [11], která umožňuje komunikaci s krypto čipem Atmel ATSHA204. Tento čip je zde použit k authentifikaci [9].

1.1.4 Lcollect

Lcollect je obdobný nástroj jako ucollect, který ovšem nezasílá žádná data serveru. Sbíraná data tedy zůstávají pouze u koncového uživatele, a proto může tento nástroj sbírat oproti nástroji ucollect i data v lokální síti. K těmto datům má přístup pouze majitel routeru Turris, a tak se nejedná o porušení ochrany soukromí.

Framework Lcollect zajišťuje přístup ke struktuře paketů. Komunikaci se serverem není možná. Lcollect obsahuje pouze jeden zásuvný modul.

Majordomo: Zásuvný modul Majordomo slouží pro analýzu toků připojených klientů uvnitř lokální sítě. Modul tedy má přístup k poměrně citlivým datům, a tak nemůže kvůli závazkům ve smlouvě běžet v aplikaci ucollect.

Tento nástroj byl vytvořen pro uživatele, aby měli přehled o dění v jejich síti. Uživatel pak může s nasbíranými daty naložit podle svého uvážení [7].

1.1.5 Centrální server

Centrální server (neboli collection master) komunikuje s klientskými routery. Collection master obsahuje serverovou část pluginů, které jsou nasazeny na klientovi. Serverové pluginy se starají především o naslouchání klientům, o zasílání požadavků na klienta a přijímání nasbíraných dat. Dále pak data určená k uchování uloží do databáze. Collection master i serverové pluginy jsou napsány v programovacím jazyce Python a jako databáze se zde používá PostgreSQL.

Pokud je třeba, server zasílá konfigurační proměnné pro jednotlivé pluginy ucollectu v dohodnutém formátu. Dále zpracovává příchozí data ze zásuvných modulů, opět v dohodnutém pořadí a formátu, a ukládá tyto hodnoty do databáze, kde každý modul má své vlastní tabulky.

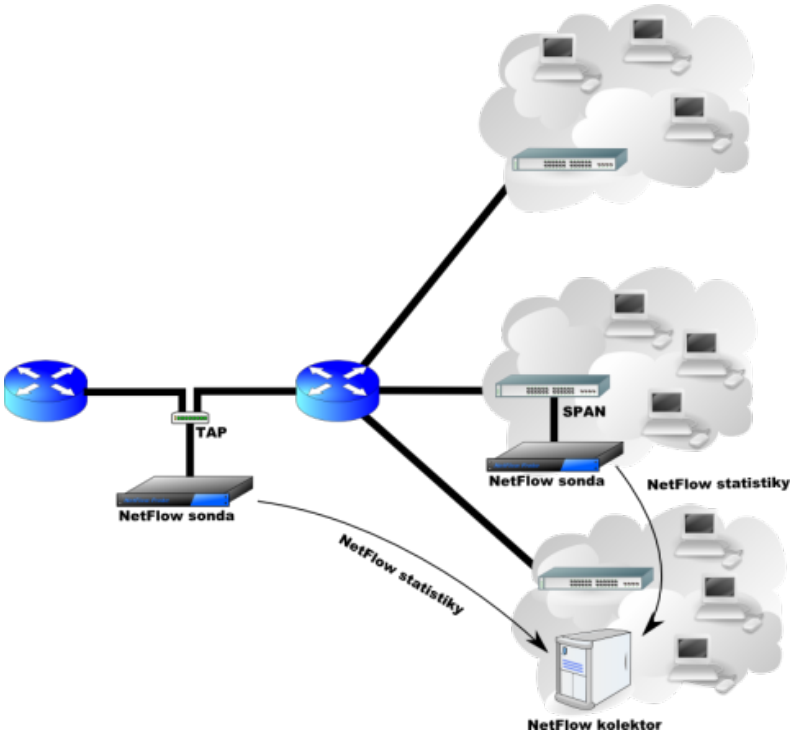
1.2 Analýza síťového provozu

Tato diplomová práce se zabývá analýzou síťového provozu a následnou detekcí anomálií v počítačových sítích. To je možné provést několika způsoby, které jsou v následujících větách popsány. Jedná se o metodu paketové analýzy a metodu síťových toků [12].

Analýza síťového provozu je důležitá pro mnoho odvětví v informačních technologiích. Využívá se například pro kontrolu správného chodu sítí nebo také pro poskytovatele internetu, kteří na základě statistik mohou svým zákazníkům účtovat ceny služeb v závislosti na množství přenesených dat.

Paketovou analýzu může v malém rozsahu provádět běžný uživatel, který s pomocí paketových analyzérů či sniffrů může shromažďovat veškerá data, která prochází sítí. Jako příklad takového softwaru lze uvést aplikaci Wireshark [13]. Při paketové analýze se v některých případech ukládají celé pakety pro následné zpracování, které je díky komplexnosti a náročnosti složité provádět v reálném čase.

S postupem času a rozvojem nových technologií se ale rychlost počítačových sítí několikanásobně zvýšila, a tak přenos dat může dosahovat až k několika desítkám či stovkám gigabitů za sekundu. Z tohoto důvodu není možné uchovávat informace o každém zachyceném paketu. Další problém paketové analýzy představuje narůstající počet šifrovaných síťových připojení, v opač-



Obrázek 1.2: Schéma moderní NetFlow architektury (zdroj obrázku [2]).

ném případě může hrát roli také případné narušení soukromí uživatelů. Bylo tedy nutné zvolit jiný přístup k analýze síťového provozu.

Za termínem síťový tok (neboli NetFlow) stojí společnost Cisco Systems. NetFlow je protokol běžící na směrovačích a jeho účelem je monitorovat provoz na základě IP toků. Znamená to tedy, že protokol neprohlíží celý paket, nýbrž pouze hlavičky paketů a z těch následně generuje a ukládá statistiky. Analýza síťových toků je výrazně rychlejší, a tak je možné data zpracovávat v reálném čase [2, 14].

Pro lepší znázornění je možné na obr. 1.2 vidět schéma moderní NetFlow architektury. Tento typ architektury využívá pasivní NetFlow sondy. Pasivní v tomto případě znamená, že sondy data v tocích pouze monitorují a nijak nemění. Statistiky poté odesílají na kolektor zvláštní linkou tak, aby se na monitorované lince chovaly zcela neviditelně a tedy schovaně proti útočníkům.

Síťový tok může být definován mnoha způsoby. Cisco standard verze 5 [2], který je nejpoužívanější, definuje tok jako jednosměrnou sekvenci paketů, které mají těchto 7 věcí společných:

- SNMP index vstupního a výstupního rozhraní
- zdrojovou IP adresu

- cílovou IP adresu
- IP protokol
- zdrojový port pro TCP nebo UDP protokol, popřípadě hodnotu 0 u ostatních protokolů
- cílový port pro TCP nebo UDP protokol, typ a kód pro protokol ICMP, popřípadě hodnotu 0 u ostatních protokolů
- ToS (Type of Service)[6]

1.3 Síťové anomálie

Za síťové anomálie lze obvykle považovat všechny události, při kterých se chování sítě odlišuje od normálu. Za anomálie mohou být zodpovědné různé faktory. Může se jednat o selhání prvku v síti, přetížení sítě nebo například úmyslný útok. Všechny tyto události ovlivní chování sítě. Jak ale vypadá normální chování se u každé sítě různí a závisí to na několika faktorech, jako jsou typy běžících aplikací v síti nebo na velikosti provozu [16].

1.3.1 Typy anomálií

V podstatě se síťové anomálie dělí na dvě kategorie. První z nich souvisí se selháním síťových prvků a obecně techniky, tedy problémy s výkonností systému. Druhou skupinu tvoří všechny problémy spojeny s bezpečností, na které se tato práce zaměří v následujících sekcích. Vybrány a popsány jsou především anomálie související s náhlou změnou poměru paketů s různými TCP příznaky od obvyklého provozu.

1.3.1.1 Nedostupnost služby

Z první kategorie síťových anomálií, tedy té nesouvisející s bezpečností, je zde uveden pouze jeden příklad a to nedostupnost služby. Nedostupnost často využívané služby se může totiž projevit jako anomálie v síti. Jestliže služba obvykle běžící na nějakém portu neběží, pak i když je tento port uzavřen, stále mohou přicházet požadavky na spojení. Místo spojení se ovšem nazpět odešle paket s příznakem RST, který značí uzavřený port. Anomálii zde tedy představuje zvýšený počet RST paketů vzhledem k paketům SYN.

[6]Typ služby je pole v IPv4 hlavičce, které sloužilo za posledních pár let k různým účelům. Podle původních představ mělo sloužit k volbě charakteru přepravní služby ideální pro dotyčný datagram. Jako například požadavek na nejmenší zpoždění či největší šířku pásma. Dnes slouží jako nosič značky pro mechanismy zajišťující služby s definovanou kvalitou [15].

1.3.1.2 DoS a DDoS útok

DoS (Denial of Service) je asi nejznámější a nejpopulárnější útok, který má za cíl donutit cílový server k pádu či restartu nebo alespoň znepřístupnit nějakou službu legitimním uživatelům. Je způsoben například zaplavením sítě falešnými pakety, a tím znesnadnit cestu sítí paketům s pravými daty. V případě DDoS (Distributed Denial of Service) je situace obdobná, rozdíl je pouze v tom, že útok probíhá z více zdrojů. Dalším způsobem může být přerušování již započatých spojení, a tak blokovat přístup ke službám. Detekce DoS útoků a jejich minimalizace je poměrně obtížná ale i tak je možné se bránit proti některým typům provedení útoku [17, 18].

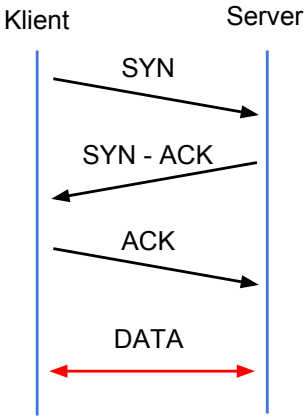
V podstatě nejčastější způsob, jak provést DoS útok, je pomocí záplavy paketů. Jsou tři základní možnosti jak záplavu provést. Jedná se o záplavu TCP pakety, ICMP pakety a UDP pakety. Tyto metody mají za cíl vyčerpat oběti šířku pásma a výpočetní čas. Dalším typem je útok využívající chyb a vyčerpání systémových prostředků. Tyto metody (kromě záplavy pomocí SYN paketů) zde popsány nejsou, ale pro příklad patří do této kategorie Ping of Death (ping smrti), Lands attacks, Nukes a další [19].

Záplava TCP pakety Zajímavý je především útok záplavou SYN paketů. Tento útok zneužívá oboustranný proces pro vytvoření TCP (Transmission Control Protocol) spojení [20]. Pro vytvoření nového spojení používá TCP protokol trojcestný handshake[7]. Klient nejprve posílá serveru inicializační paket s příznakem SYN. Server po přijetí SYN paketu zkontroluje, zda je možné spojení uskutečnit. Je-li vše v pořádku, server zarezervuje místo pro spojení a nazpět klientovi pošle paket s příznaky SYN a ACK, což značí potvrzení spojení. Klient po přijetí už serveru odešle jen paket s příznakem ACK a dále může posílat libovolná data. Pro lepší znázornění je TCP handshake zobrazen na obr. 1.3.

Útok tedy probíhá tak, že útočník zasílá oběti velké množství inicializačních paketů s příznakem SYN, tedy pakety, které slouží k navázání spojení. Tento paket má navíc často podvrženou zdrojovou adresu. Server si pro každý přijatý paket vytvoří v paměti místo pro přijímání a odesílání dat a odešle SYN-ACK paket. Na ten oběti už ale nikdo neodpovídá, a tak je dočasně vytvořeno polovičaté spojení, které zabírá oběti stejné množství paměti, jakoby bylo vytvoření spojení úplně dokončeno. Při útoku se tento proces opakuje tak dlouho, dokud se oběti nevyčerpají veškerá volná místa v paměti a služba se tak stane nedostupnou pro uživatele, kteří by ji chtěli reálně využívat [18].

Nutno ještě podotknout možnost záplavou pakety ACK, FIN apod. Zde se jedná ale pouze o zahlcení linky oběti vygenerovanými pakety.

[7] Slovo handshake zde znamená proces navázání spojení.



Obrázek 1.3: První tři šipky značí trojcestný TCP handshake

Záplava ICMP pakety ICMP (Internet Control Message Protocol) je protokol používaný v operačních systémech pro odesílání chybových zpráv, například pro oznámení zda je služba dostupná či naopak. Nástroj ping, který se používá právě pro zjištění dosažitelnosti cílového prvku používá ICMP zprávy „Echo Request“ a očekává odpověď „Echo Reply“. Tyto zprávy jsou nejčastěji používány při záplavě ICMP pakety. Odpověď „Echo Reply“ navíc zachovává velikost paketu s požadavkem a podle doporučení RFC (Request For Comments)[8] by měla být maximální velikost 548 bytů. Operační systémy Microsoft Windows a Linux ovšem umožňují pakety velikosti až 65535 bytů [18].

Záplavou ICMP pakety (může se také nazývat záplava ping) se útočník snaží posílat pakety „Echo Request“ co možná nejrychleji, tak aby oběti vyčerpal celou šířku pásma pro příjem dat. Oběť je navíc donucena odpovídat pakety „Echo Reply“ a tím si zahltí šířku pásma pro odesílání dat. Tento útok není v dnešní době příliš efektivní, protože pro úspěch je důležité, aby útočník měl větší přenosovou rychlost jako oběť. Přesto při některých malých úpravách je útok stále možný.

Záplava UDP pakety UDP (User Datagram Protocol) je bezstavový protokol a tedy se nevytváří mezi komunikujícími stranami žádné spojení. Principem je opět zahltit linku oběti obrovským množstvím paketů. Dříve se využívalo i služeb echo a chargen, které byly v operačních systémech Linux standardně spuštěny [18].

Služba echo odesílá veškerá data, která přijdou na její port zpět jako ozvěnu. Služba chargen odesílá jako odpověď náhodná data. Pokud tedy útočník zfalšuje IP adresu a port tak, aby dva napadené počítače komunikovali

[8]Request For Comments označuje skupinu dokumentů, které v informatice popisují internetové protokoly či systémy. Přestože jsou brány spíše jako doporučení než standardy, řídí se jimi většina vývojářů [21].

mezi sebou právě přes služby echo a chargen, docílíme toho, že napadené počítače budou komunikovat stále dokola.

1.3.1.3 Skenování portů

Jako další anomálii je možné označit například metodu skenování portů. Díky tomuto procesu lze zjistit, které služby na cílovém zařízení běží. Útočník následně může najít slabinu, kterou je možné zneužít. Může se jednat například o neaktualizovaný software nebo operační systém [22].

Při skenování portů mohou nastat tři situace. Při zaslání dotazu na port může cílové zařízení odpovědět a tím prozradit, že port je otevřený a daná služba běží. Tato situace je pro útočníka samozřejmě nejžádanější. Další možností je, že při testování portu se vrátí ICMP paket se zprávou, že port je nedosažitelný a tedy je zavřený. Poslední možností je, že žádná odpověď nepřijde. V takovém případě byl paket zahozený, nebo je port filtrovaný či blokovaný a tedy, že počítač je chráněn firewallem.

Skenování portů lze provést několika způsoby, zde jsou vybrané některé z nich:

Sken pomocí TCP: Nejjednodušší způsob provedení. Útočník zde využije funkci operačního systému connect(). Pokud je port otevřený tak metoda provede trojcestný TCP handshake. V opačném případě obdrží chybovou zprávu. Poté je spojení ihned ukončeno, aby se útočník vyhnul způsobení DoS útoku. Výhodou této metody je, že nejsou potřeba žádná speciální privilegia. Na druhou stranu je tato metoda poměrně snadno vypátratelná (služba mohla zalogovat aktivitu a IDS mohl vyvolat alarm), a tak je potřeba následně provést úklid.

Sken pomocí SYN paketu: Tato metoda je obdoba TCP skenování. Útočník vygeneruje přímo SYN paket a monitoruje odpovědi. Pokud je port otevřený, jako odpověď přijde paket s příznaky SYN a ACK. Útočník poté pošle RST paket, který TCP spojení ukončí předtím, než je trojcestný handshake uskutečněn. V opačném případě útočník obdrží jako odpověd na SYN paket RST paket, který značí uzavřený port. Tato metoda je obtížněji vypátratelná než skenovaní pomocí TCP.

Sken pomocí UDP paketu: Další možností je zasílání UDP paketu na zkoumaný port. Metoda má ovšem hned několik nevýhod. Pokud je port zavřený, útočník obdrží ICMP paket se zprávou o nedostupnosti. Pokud se ale žádná zpráva nevrátí útočník neví, zda je port otevřený, nebo je blokovaný firewallem.

Alternativní přístup je vytvořit UDP pakety na míru jednotlivým službám. Například vytvořit paket s DNS dotazem a poslat ho na port 53. Pokud útočník obdrží odpověď, dozví se, že DNS server je přítomen.

Sken pomocí FIN paketu: Poslední zmíněnou možností je využití paketů s příznakem FIN. Tato metoda je užitečná v případě, kdy veškeré SYN pakety jsou blokovány firewallem. FIN pakety mohou obejít firewall a port, na které jsou poslány může zareagovat. Pokud je port zavřený, na paket FIN odpoví paketem RST. Pokud je port otevřený, paket bude ignorovat a odpověď se nedostaví žádná.

1.3.1.4 TCP reset útok

Za další síťovou anomálii je možné označit TCP reset útok. Jedná se o snahu přerušit již existující TCP spojení mezi dvěma komunikujícími stranami. Útok lze provést dvěma způsoby, a to pomocí ICMP zpráv nebo pomocí TCP paketů s příznaky RST, popřípadě SYN [23, 24].

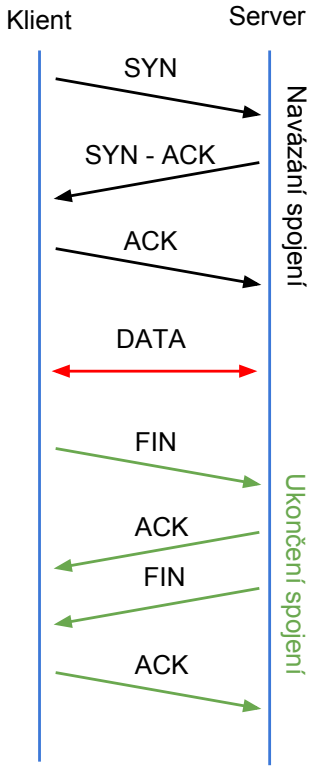
Nejprve je nutné si představit jak probíhá celé spojení pomocí TCP protokolu. Jak se provádí TCP handshake je již zobrazeno na obr. 1.3, nyní je potřeba si vysvětlit i jak se spojení ukončuje.

Na obr. 1.4 lze vidět, že po trojcestném handshaku a navázání spojení si může již klient a server mezi sebou posílat data. Poté co jsou všechna data předána, ukončení spojení může jedna strana navrhnout paketem s příznakem FIN. Jestliže druhý s ukončením spojení souhlasí, vrátí paket s příznakem ACK, a také s FIN, což znamená potvrzení návrhu na ukončení a ukončení i z druhé strany. Konec spojení nastává když i druhý FIN paket je potvrzen pomocí ACK paketu.

Reset pomocí ICMP paketu Jednou z možností jak narušit TCP spojení je využití ICMP protokolu. Proti útoku ICMP zprávami jsou dnešní operační systémy vetšinou chráněny. Nicméně to neznamená, že takový útok se již nepoužívá. Na provedení útoku stačí útočníkovi znát IP adresu a port obou komunikujících stran. Pokud jsou tyto informace známé (číslo portu je případně možné zkoušením uhodnout) stačí zaslat na IP adresu a port oběti jeden z vybraných ICMP paketů, který způsobí chybu a spojení se ukončí. Tyto ICMP pakety jsou typu 3 (cíl nedostupný) kódy 3-5 a značí „protokol nedostupný“, „port nedostupný“ a poslední „fragmentování potřebné, ale nastaveno nefragmentovat“.

Reset pomocí TCP paketu Narušit TCP spojení lze i přímo pomocí TCP paketů. Jednou možností, jak útočník může narušit spojení, je použít paket SYN, který pokud přijde v jinou dobu, než na začátku spojení, vyvolá chybu a spojení se ukončí. Druhá možnost je zaslání paketu RST, který je v protokolu TCP určen k okamžitému ukončení spojení, například pro uvolnění alokovaných zdrojů z předchozího spojení.

Pro provedení tohoto typu útoku musí útočník znát nejen IP adresu a port obou komunikujících stran, ale i jejich sekvenční čísla.



Obrázek 1.4: Znázornění celé komunikace pomocí protokolu TCP

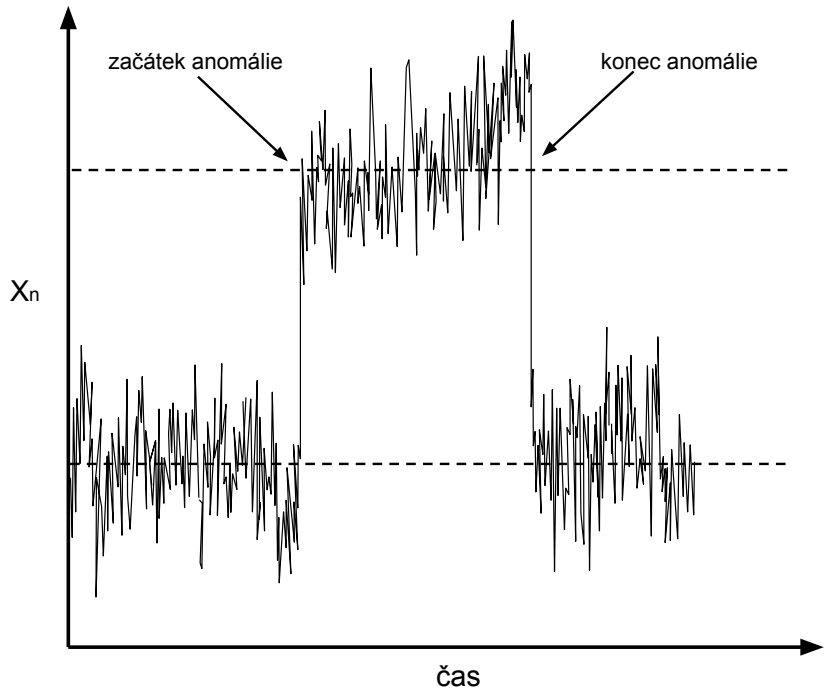
1.3.2 Detekce síťových anomálií

Jak bylo již zmíněno dříve, síťové anomálie jsou události, při kterých se chování sítě odlišuje od normálu. Aby bylo možné poskytnout obecný návod, jak detekovat síťové anomálie, je nutné vyřešit problém definice normálnosti. Myšlenka normálnosti je obvykle prezentována jako formální model, který vyjadřují vztahy mezi základními prvky zapojenými v systému. Událost nebo prvek je považován za neobvyklý, jestliže stupeň odchylky vzhledem k chování systému, specifikovaném normálním modelem je dostatečně velký [16, 25, 26].

Pro lepší znázornění je na obr. 1.5 zobrazen příklad síťové anomálie. Na následujících řádcích jsou popsány vybrané metody užívané k detekci anomálií v síti.

1.3.2.1 Statistické metody

Většina metod sloužících k detekci síťových anomálií, které jsou v následujících sekcích diskutované, potřebuje s rozvojem sítí novou kalibraci nebo trénování. Oproti tomu analýza pomocí statistiky může průběžně sledovat chování sítě. Pomocí statistické metody je možné sledovat jak anomálie způsobené pomocí



Obrázek 1.5: Ilustrace detekce anomálie v symetrickém síťovém provozu (zdroj obrázku [27]).

selhání síťových prvků, tak i síťové útoky. Obě tyto události umožňují použít sekvenční metodu detekce bodu změny.

Sekvenční detekce bodu změny využívá neparametrickou metodu cumulative sum (CUSUM) [28, 29, 30], která bude popsána v následující kapitole 1.4. Pomocí tohoto přístupu je možné detekovat určité anomálie, jako například záplavu pakety SYN, s vysokou přesností a poměrně krátkým časem detekce.

1.3.2.2 Analýza signatur

Porovnávání signatur či metody porovnání na základě pravidel byly dříve v sítích mezi nejrozšířenějšími. Obsahují databázi signatur a pravidel chování vadných či napadených systémů, které jsou použity pro rozhodování, zda nastala anomálie.

Signatury jsou popisem chování konkrétních známých útoků. V případě, že NIDS (Network Intrusion Detection System)[9] odhalí shodu chování v síti

[9]NIDS je zařízení nebo software, které monitoruje síť a aktivity systému, aby odhalil zlomyslné aktivity či porušení politiky. Následně vytvoří vyhodnocení, které předá administrátorovi sítě.

se signaturou, může upravit chod firewallu tak, aby zabránil škodám v systému. Tyto systémy porovnávání signatur úzce souvisí se schopnostmi síťového administrátora [25].

Jako příklad je možné uvést systém Snort. Tento open-source IDS porovnává každý zachycený paket se skupinou pravidel. Předchůdce pravidla Snort je boolovský vzorec složený z predikátů, které hledají speciální hodnotu například v částech hlavičky IP paketu. Snort je tedy schopen odhalit pakety útoku na základě IP adres, čísla TCP nebo UDP portů nebo ICMP kódu. Pravidla Snortu jsou uspořádána do prioritních tříd na základě závažnosti možných následků.

Systémy na porovnávání signatur jsou ale pro real-time aplikace příliš pomalé a velkou nevýhodou je neschopnost reakce na neznámé podněty či neustálá nutnost aktualizace databázových pravidel.

1.3.2.3 Behaviorální analýza

Behaviorální analýza, neboli metoda vzorů popisuje anomálie jako odchylky od normálního chování. Snaží se vypořádat s proměnlivostí v prostředí počítačových sítí. V této metodě slouží online učení k tvorbě profilů provozu dané sítě. Tyto profily jsou vytvořeny pomocí specifických znaků jako je například míra ztráty paketů či počet kolizí. Dále jsou profily rozděleny podle časových úseků, jako jsou určité části dne, dny v týdnu a zda se jedná například o víkend či den prázdnin. Anomálie je detekována pokud nově změřená data nelze napasovat do rozsahu příslušného profilu [31, 32].

Jelikož se profily tvoří pomocí učení, je zde určitá pravděpodobnost, že se metoda naučí i samotnou anomálii v domnění, že se jedná o normální provoz. Pak tato metoda může vyvolat falešný alarm.

1.3.2.4 Stavová analýza

Detekce anomálií nebo selhání prvků v síti pomocí stavových automatů modeluje sekvenci alarmů, které mají nastat před nebo během selhání systému. V podstatě se jedná o metodu, která kontroluje síťový provoz na základě přesných definic protokolů provozovaných v síti [16]. Tyto protokoly jsou definovány pomocí stavů automatu a každá činnost protokolu představuje přechod mezi stavy automatu. V případě, že chování provozu v síti je odlišné od stavu automatu, pak tohle chování může být považováno za anomálii.

Stavové automaty jsou vytvořeny, aby nejen detekovali anomálie, ale aby byly schopné i identifikovat a diagnostikovat problém. Obtížností v této metodě je především to, že nemusí zachytit všechna selhání v síti, což závisí na počtu možných stavů automatu. Vytvoření všech přechodů stavů může mít za následek obrovskou komplexitu modelu [16].

S rozvojem sítí se ovšem zvyšuje počet parametrů a tedy doba pro trénování se značně prodlužuje a to se bezpodmínečně projeví na době, než se takový stavový automat umístí do sítě.

1.4 Statistické metody

Co detekují statistické metody bylo zmíněno již dříve. Tedy, že je možné díky nim sledovat jak anomálie způsobené pomocí selhání síťových prvků, tak i síťové útoky.

Statisticky řečeno je anomálie nějaké pozorování, které je podezřelé, že ať už úplně nebo částečně není vytvořeno stochastickým modelem[10]. Statistické metody obvykle zkouší porovnat data se statistickým modelem normálního chování a rozhodnout, zda tato neznámá data do modelu pasují. Případy, kdy tato data mají malou pravděpodobnost, že byla vygenerována z naučených modelů, jsou prohlášeny za anomálie.

1.4.1 Přehled používaných statistických metod

Pro detekci síťových anomálií pomocí statistiky se využívá několik technik a modelů [33, 34]. Zde je popis vybraných metod:

Prahová metrika Metoda vytváří počty nějakých událostí, které nastávají v síťovém provozu. Jakmile tento počet překročí stanovený práh nebo se již nevleze do určeného rozmezí, metoda vyvolá alarm. Příkladem může být metoda sekvenční detekce bodu změny, která je popsána v sekci 1.4.2.

Markovský proces Tato metoda kontroluje stav systému ve stejně dlouhých intervalech a uchovává si jej. Při každé nové události se stav systému změní a jestliže pravděpodobnost výskytu nového stavu je nízká, je považován za anomálii.

Průměr a standardní odchylka V tomto případě se za anomálii považuje událost, jejíž odchylka od momentálního průměru nespadá do určeného rozmezí. Toto rozmezí se může lišit v závislosti na aktuálním měření. Výhodou této metody je, že není třeba určovat, jak vypadá „normální“ provoz.

1.4.2 Sekvenční detekce bodu změny

Detekce bodu změny se snaží ve statistické analýze určit, kdy se pravděpodobnostní rozdělení stochastického modelu změní. Problémem není jen určit, jestli změna nastala či nenastala, ale také je nutné vědět časy těchto změn.

[10] Stochastika je matematický obor, který se zabývá zkoumáním a modelováním náhodných jevů. Jedná se o souhrnný název pro teorii pravděpodobnosti a matematickou statistiku.

Problém detekce bodu změny (CPD - change point detection) lze definovat jako detekce změn ve statistickém modelu v co nejkratším čase, to znamená s minimálním průměrným zpožděním (ADD - average detection delay), při zachování míry výskytu falešných poplachů (FAR - false alarm rate) na stanovené úrovni [30, 35].

Detekce neočekávané změny byla poprvé studována Pagem v práci [36] v kontextu kontroly kvality. V této práci byla detekce bodu změny formulována jako „jeden senzor jeden kanál“, kde se rozdělení jedné sekvence pozorování změní v neznámém bodu v čase. Pod podmínkou, že pozorování jsou nezávislá a rovnoměrně rozdělená (i.i.d[11]) se známým rozdělením jak před, tak i po změně.

Proceduru analýzy bodu změny je možné aplikovat na jakoukoliv charakteristiku, pro kterou jsou data sbírána v čase. Může být aplikovaná například na průměry, odchylky, rozsah, počty a tak dále. Charakteristika se pak znázorní jako sekvence proměnných $X_1, X_2, X_3, \ldots, X_n$ naměřená v čase.

Ještě je dobré zmínit, že metody detekce bodu změny mohou být parametrické, jako například Pageova CUSUM metoda [36], nebo také další možností jsou metody neparametrické, jako například NP-CUSUM [28, 29, 30]. Obě tyto metody jsou dále popsány.

1.4.2.1 CUSUM

Jedná se o statistickou metodu představenou E. S. Pagem již v roce 1954. Název CUSUM vznikl složením slov cumulative sum (souhrnný součet). Page původně odkazoval na „číslo kvality“ θ, kterým myslel parametr pravděpodobnostního rozdělení, například průměru. Pokud je tato metoda použita na změny v průměru, pak je možné odhalit kroky v časových řadách [37].

Rovnici pro počítání souhrnných součtů pro změny v kladném směru je možné vidět v (1.1), kde x_n značí proces a ω_n značí jeho váhu [28]. Jakmile hodnota S překročí nějaký, uživatelem nastavený práh, byla nalezena změna hodnoty.

$$\begin{aligned} S_0 &= 0, \\ S_{n+1} &= max(0, S_n + x_n - \omega_n). \end{aligned} \tag{1.1}$$

Pokud je třeba detekovat změnu v záporném směru, stačí změnit operaci max na min (viz (1.2)) a jakmile se hodnota S dostane pod stanovený práh, je změna detekována.

$$\begin{aligned} S_0 &= 0, \\ S_{n+1} &= min(0, S_n + x_n - \omega_n). \end{aligned} \tag{1.2}$$

[11] i.i.d znamená independent and identically distributed, tedy nezávislá a rovnoměrně rozdělená

1.4.2.2 NP-CUSUM

Metoda NP-CUSUM neboli nonparametric cumulative sum (neparametrický souhrnný součet), je odvozena od Pageovi CUSUM metody. Oproti CUSUM metodě ovšem nemusí pracovat s nezávislými a rovnoměrně rozdělenými daty, ale naopak lze použít u dat s neznámým rozdělením a to jak před změnou, tak i po ní [28, 29, 30].

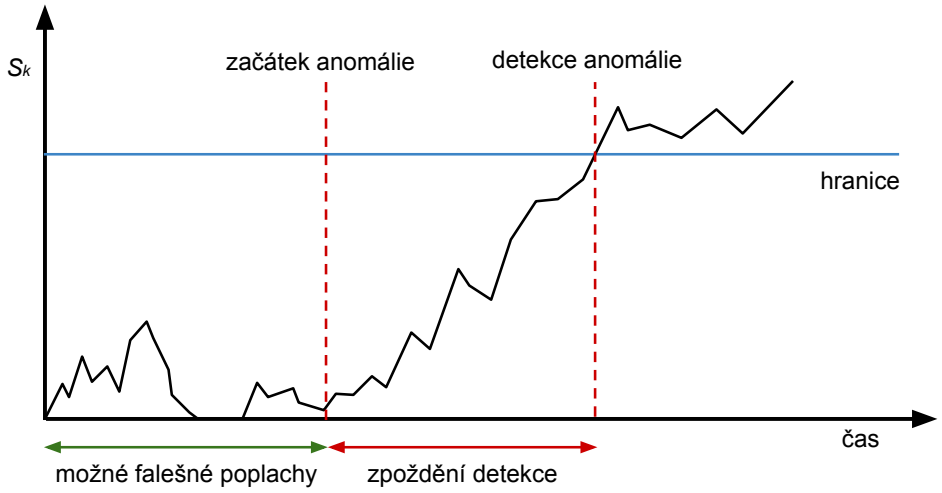
Klíčovou součástí metody je rovnice (1.3), která počítá statistiku S_n pro danou charakteristiku. V této práci je metoda NP-CUSUM implementována s charakteristikou x_n, která představuje například poměr paketů s příznakem SYN s pakety s příznaky FIN. Pokud statistika překročí předem určený práh, statistika vyhlásí poplach.

$$\begin{aligned} S_0 &= 0, \\ S_n &= max\{0, S_{n-1} + x_n - \mu - \epsilon\theta\}. \end{aligned} \tag{1.3}$$

Proměnná μ určuje střední hodnotu charakteristiky, θ_n představuje střední hodnotu charakteristiky v případě útoku a ϵ slouží jako optimalizační konstanta. Tyto tři konstanty by se dali nahradit za jedinou a její velikost by se dala vypočítat jako $\mu + \frac{\theta - \mu}{2}$. Tudíž pokud pro přehlednost ponecháme všechny konstanty jako v rovnici (1.3), pak optimalizační konstantu ϵ je možné vypočítat přibližně jako (1.4).

$$\epsilon = \frac{\mu - \theta}{2\theta}. \tag{1.4}$$

Metoda NP-CUSUM je vhodná právě pro detekci síťových anomálií [30], protože je asymptoticky optimální se zmenšující se intenzitou falešných poplachů. Jak se tato metoda chová znázorňuje obr. 1.6.



Obrázek 1.6: Ilustrace průběhu metody NP-CUSUM.

Kapitola 2

Návrh

Tato část práce popisuje detaily prostředí, ve kterém běží framework ucollect. Další sekcí bude návrh konkrétních částí pluginu, který monitoruje síťový provoz a v případě detekce síťové anomálie zašle administrátorovi systému upozornění. V poslední části této kapitoly je popsán systém nastavení pluginu pomocí konfiguračních proměnných pro detekci konkrétní síťové anomálie.

2.1 Prostředí

Jelikož vytvářím plugin do již existujícího systému, některé použité technologie není možné ovlivnit. Jako příklad je možné uvést programovací jazyk pro klientskou část pluginu je C a pro serverovou část Python. V následujícím textu jsou uvedena některá tato „omezení“.

2.1.1 PostgreSQL

Centrální server projektu Turris využívá databázi PostgreSQL (nebo jednodušeji Postgres). Jedná se o objektově-relační databázový systém založený na POSTGRES verze 4.2[12]. Na vývoji tohoto open-sourcového software se podílí globální komunita lídí a firem [38, 39].

PostgreSQL implementuje většinu SQL:2011 standardů, je transakční a má ACID[13] vlastnosti. Postgres stejně jako ostatní databázové systémy umožňuje například použít triggery či cizí klíče, podporuje funkce a procedury či zamezuje problémům s uváznutím pomocí MVCC (Multi-Version Concurrency Control)[14].

[12] POSTGRES byl databázový výzkumný projekt na University of California v Berkeley. Tento projekt byl ukončen již v roce 1994.

[13] Atomicity, Consistency, Isolation, Durability (atomicita, konzistence, izolovanost, trvalost) jsou vlastnosti databázových transakcí.

[14] MVCC zpřístupňuje uživatelům „snapshot “ databáze, díky čemuž mohou provádět změny čímž omezuje nutnost zamykání

Zajímavostí bezpochyby je, že tato PostgreSQL databáze využívá TOAST (The Oversized-Attribute Storage Technique) pro ukládání velkých atributů v tabulce, jako jsou například XML soubory. Díky této metodě pak databáze umožňuje maximální velikost tabulky 32 TB či maximální velikost řádku 400 GB.

Pro správu PostgreSQL databáze může být využito několik volně dostupných i proprietárních rozhraní.

psql Primární nástroj, který je určen pro příkazovou řádku. Pomocí tohoto nástroje může uživatel zadávat přímo SQL dotazy či je číst ze souboru.

pgAdmin Grafické rozhraní PostgreSQL databáze určené pro velké množství operačních systémů. Nejnovější verze tohoto programu je napsána v jazyce C++ a přeložena do více jak 10 jazyků. Rozhraní, jež se dříve jmenovalo pgManager, se vyvíjí od roku 1998 [40].

phpPgAdmin Administrační nástroj phpPgAdmin napsaný v jazyce PHP se prezentuje jako webová aplikace. Nástroj je založený na rozhraní phpMyAdmin, který je populární pro MySQL databáze.

2.2 Plugin

Monitorovací plugin využívá pro detekci síťových anomálií statistickou metodu NP-CUSUM zmíněnou v 1.4.2.2. Asi nejdůležitější částí rovnice pro výpočet charakteristiky je určení konstant (μ, ϵ, θ), které zásadně ovlivňují výsledky metody. Při špatném zvolení těchto konstant metoda nemusí zachytit probíhající útok, nebo naopak při normálním provozu může vyvolat falešné alarmy.

Pro určení konstant μ, ϵ, θ je nutné mít nějaká data z reálného provozu. Nejprve je nutné analýzou dat odhadnout, jak asi vypadá normální provoz v síti a jak vypadá anomálie. Poté stačí přizpůsobit konstanty tak, aby byla metoda schopná tyto anomálie detekovat. Více v části 4.2.

Rozhodl jsem se tedy vytvořit dvě verze pluginu. První verze je určena pro získání dat z reálného provozu a zároveň ověření funkčnosti metody NP-CUSUM. Druhá se již o sběr dat nestará, ale pokud metoda vyhodnotí síťový provoz jako anomální, pak zašle upozornění.

2.2.1 Plugin pro sběr dat

První verze pluginu, která je určená především pro sběr dat a otestování metody NP-CUSUM sbírá pakety s příznakem SYN, FIN, RST a pakety jak s příznaky SYN, tak zároveň s ACK. Nasbírané počty uloží do databáze a také z těchto dat spočítá charakteristiku.

2.2.1.1 Charakteristika

Charakteristika je konfigurovatelná tak, aby bylo možné počítat například poměr paketů SYN a FIN či charakteristiku modifikovat na počítání poměru SYN a SYN-ACK. Toho je docíleno pomocí jedné rovnice, kde se pomocí konfigurace váhy jednotlivých příznaků paketů může ovlivnit výsledný výpočet charakteristiky.

Rovnice pro výpočet charakteristiky (2.1) obsahuje členy w, což značí váhu konkrétního příznaku paketu. Členy c značí počet těchto paketů zachycených během předem stanoveného časového úseku.

$$x_n = \frac{w_{\text{syn}} \cdot c_{\text{syn}}}{(w_{\text{fin}} \cdot c_{\text{fin}}) + (w_{\text{rst}} \cdot c_{\text{rst}}) + (w_{\text{synAck}} \cdot c_{\text{synAck}})}. \tag{2.1}$$

Může ovšem nastat situace, kdy je spodní část rovnice rovna nule a tím by celá charakteristika nešla vypočítat. V tomto případě, ale také v případě, kdy i vrchní část rovnice je rovna nule, je vypočítání charakteristiky přeskočeno (výsledek charakteristiky je roven 0).

2.2.1.2 Návrh metody NP-CUSUM

Rovnice pro výpočet neparametrické kumulativní sumy je neupravena a použita pro přehlednost jako v analytické části - viz rovnice (1.3).

Jak jsem již zmínil v předešlé sekci, může se stát, že charakteristika je rovna nule. Pak i celý průběh metody NP-CUSUM je v daný časový úsek přeskočen a výsledek ponechán na hodnotě spočítané v předešlém kole. To znamená $S_n = S_{n-1}$. Výsledek metody tento postup poměrně ovlivní.

Uvažujme příklad. První výpočet charakteristiky byl proveden například po analýze mnoha paketů SYN a třeba jen poloviny paketů FIN. Výsledek charakteristiky je pak okolo hodnoty 2 a křivka metody NP-CUSUM poroste. Tato situace ovšem může nastat i v normálním síťovém provozu, pokud byla charakteristika spočítána v době, kdy většina TCP spojení ještě neskončila. Při druhém výpočtu by tedy mělo dorazit více paketů s příznaky FIN a tím křivku metody přiblížit opět hodnotě nula. Problém by nastal pokud by pro druhý výpočet bylo analyzováno 0 paketů s příznakem SYN a přitom mnoho paketů FIN. Pak by nový výsledek charakteristiky byl přeskočen a přiblížení křivky by nenastalo.

Pravděpodobnost nastání této situace úzce souvisí se zvolením časového intervalu pro počítání charakteristiky. Jestliže se charakteristika počítá často, počty naměřených paketů jsou nižší a pravděpodobnost výskytu nuly v rovnici je vyšší. Pokud ale počítáme charakteristiku po delších časových úsecích, prodlužuje se průměrné zpoždění detekce (ADD). Jedním z důležitých úkolů je tedy najít vhodný časový interval.

V pluginu určenému ke sběru dat je tento interval nastaven na 2 minuty, ale díky uchovávání počtů paketů s příznaky je pak možné optimalizovat tento interval podle naměřených dat.

2.2.1.3 Konfigurační proměnné

Konfigurační proměnné ovlivňují výpočet charakteristiky, interval počítání a optimalizaci metody NP-CUSUM. Proměnné se načítají z konfiguračního souboru při startu centrálního serveru. Jejich názvy a popis zobrazuje tab. 2.1.

Tabulka 2.1: Konfigurační proměnné pluginu pro sběr dat.

Název proměnné	Popis	Jednotky
version	aktuální verze pluginu	desetinné číslo
interval	časový interval počítání charakteristiky	počet sekund
aggregate_delay	doba čekání na odpovědi od klientů, před tím, než budou zpracovány	počet sekund
syn_weight	váha paketů s příznakem SYN	desetinné číslo
fin_weight	váha paketů s příznakem FIN	desetinné číslo
rst_weight	váha paketů s příznakem RST	desetinné číslo
syn_ack_weight	váha paketů s příznakem SYN-ACK	desetinné číslo
const_micro	velikost konstanty μ	desetinné číslo
const_epsilon	velikost konstanty ϵ	desetinné číslo
const_theta	velikost konstanty θ	desetinné číslo

Po startu serveru a načtení konfiguračních proměnných ze souboru server odešle proměnné týkající se metody NP-CUSUM klientovi. Klient čeká na tyto informace a jakmile je obdrží, začne analyzovat síťový provoz. Proměnné „interval“ a „aggregate_delay“ zůstávají pouze na straně serveru. Pokaždé, když uplyne stanovený interval, pošle server dotaz na klienta o zaslání informací. V tuto chvíli klient spočítá charakteristiku a hodnotu kumulativní sumy a zašle informace serveru.

2.2.1.4 Uchovávané informace

Plugin pro sběr dat uchovává více informací než je pro správnou funkcionalitu NP-CUSUM nutné. Ukládané informace slouží především k ověření funkčnosti implementace a následné ověření, jak nastavit proměnné, aby výsledky metody byli co nejpřesnější a s co nejmenším zpožděním detekce. Klientská část zašle informace serveru, který je uloží do PostgreSQL databáze. Seznam potřebných sloupců je zobrazen v tab. 2.2.

2.2.2 Plugin pro zasílání upozornění

Druhá verze pluginu, která je určená pro nasazení v projektu Turris. Plugin zasílá pouze upozornění a to jen v případě, kdy metoda NP-CUSUM vyhodnotí síťový provoz jako anomální. Protože se jedná o verzi, která půjde do reálného provozu, kladou se na ni i jiné nároky.

Tabulka 2.2: Databázové sloupce a jejich datové typy pluginu pro sběr dat.

Název	Datový typ
timestamp	timestamp without time zone
client	integer
characteristic	double
result	double
syn_count	bigint
fin_count	bigint
rst_count	bigint
syn_ack_count	bigint

2.2.2.1 Charakteristika

Výpočet charakteristiky probíhá stejně jako v rovnici (2.1). I zde může být výsledek charakteristiky nulový, a to v případě, že vrchní či spodní část zlomku je rovna nule. Tomuto se snaží zamezit určení vhodného časového intervalu pro výpočet charakteristiky.

2.2.2.2 Návrh metody NP-CUSUM

Metoda NP-CUSUM je oproti pluginu pro sběr dat mírně upravena. V původní verzi (1.3) byly konstanty μ, ϵ, θ pouze pro přehlednost a pro lepší pochopení výpočtu této metody. Pro fungování metody neparametrické kumulativní sumy jsou ale tři konstanty zbytečné a je možné je nahradit pouze jednou konstantou. V pluginu jsou tedy konstanty μ, ϵ, θ nahrazeny konstantou ω. Výsledná rovnice tedy vypadá následovně:

$$\begin{aligned} &S_0 = 0, \\ &S_n = max\{0, S_{n-1} + x_n - \omega\}. \end{aligned} \tag{2.2}$$

Konstantu ω je nutné odhadnout tak, aby odpovídala zhruba středu mezi hodnotou charakteristiky normálního provozu a hodnotou, kterou již považujeme za anomálii. Vzhledem k původnímu vzorci NP-CUSUM (1.3) je rovnice pro výpočet ω určena jako:

$$\omega = \mu + \frac{\theta - \mu}{2}. \tag{2.3}$$

Oproti návrhu pluginu pro sběr dat má zde metoda NP-CUSUM definovaný nějaký práh (threshold). Ten se při každém výpočtu porovná s hodnotou S a jakmile je práh překročen vyvolá metoda alarm. Velikost prahu má vliv na frekvenci výskytu falešných poplachů (FAR) i průměrné zpoždění detekce (ADD). Tento práh je určen empiricky během nasazení nebo pomocí trénovacích dat.

2.2.2.3 Konfigurační proměnné

V předchozí verzi pluginu jsme si mohli dovolit načítat konfigurační proměnné z konfiguračního souboru serveru, protože plugin běžel hlavně v testovacím prostředí a restart serveru bylo možné provést každou chvíli. Při reálném použití by ale bylo restartování serveru při každé změně nastavení pluginu obtížné a nepraktické. Další možností, jak tedy předat nastavení klientovi, je pomocí dat uložených v konfigurační tabulce.

Serverová část pluginu načte konfigurační proměnné z databáze a zašle je klientovi. Při změně hodnot proměnných stačí upravit hodnotu v databázi a bez nutnosti restartu serveru se informace dostane až ke klientovi. Přehled hodnot, které je nutné přidat do databáze zobrazuje tab. 2.3.

Tabulka 2.3: Databázové sloupce pro konfigurační proměnné pluginu pro zasílání upozornění

Název sloupce	Popis	Jednotky
version	aktuální verze pluginu	celé číslo
interval	časový interval počítání charakteristiky	počet sekund
syn_weight	váha paketů s příznakem SYN	desetinné číslo
fin_weight	váha paketů s příznakem FIN	desetinné číslo
rst_weight	váha paketů s příznakem RST	desetinné číslo
syn_ack_weight	váha paketů s příznakem SYN-ACK	desetinné číslo
const_omega	velikost konstanty ω	desetinné číslo
threshold	hranice, kdy už se jedná o anomálii	desetinné číslo

V tomto případě se na klientskou část odešlou všechny konfigurační proměnné včetně hodnoty „interval“. Klient dle obdržené hodnoty časového intervalu spočítá charakteristiku a sumu. Jestliže hodnota S_n překročí hodnotu zadanou v thresholdu, plugin zašle serveru upozornění na možnou anomálii. V opačném případě server nekontaktuje a pokračuje v analýze sítě a počítání statistiky.

2.2.2.4 Uchovávané informace

Plugin uchovává pouze nejnutnější data. I přenos mezi klientem a serverem je co nejmenší, aby se uživateli, který využívá router Turris, zbytečně nezaplňovala linka a nedošlo k omezení jeho internetu. Proto výsledná tabulka v databázi uchovává pouze hodnoty zobrazené v tab. 2.4.

2.3 Možnosti detekce

Rovnice pro výpočet charakteristiky (2.1) je možné přizpůsobit tak, aby detekoval anomálie. Toho je docíleno pomocí nastavení váhy jednotlivých typů

Tabulka 2.4: Databázové sloupce a jejich datové typy pluginu pro zasílání upozornění.

Název	Datový typ
timestamp	timestamp without time zone
client	integer
result	double

paketů. Jak ale tyto váhy nastavit aby detekovali konkrétní anomálie (viz sekce 1.3) je přiblíženo v následující části.

2.3.1 Detekce záplavy pakety SYN

Při záplavě pakety s příznakem SYN útočník očekává, že server odpoví na každý tento paket paketem s příznakem SYN a ACK zároveň. Útočník se již ale o dokončení trojcestného handshaku nezajímá a pouze se snaží vyplýtvat všechna volná spojení tak, aby se pro reálné uživatele stala služba nedostupná (více v sekci 1.3.1.2).

V tomto případě tedy plugin zachytí nejprve mnoho paketů SYN a podobný počet paketů SYN-ACK, ale s narůstajícím objemem paketů SYN se zmenšuje počet SYN-ACK jakmile služba přestává mít volná spojení. Poměr SYN a SYN-ACK paketů tedy nemusí vždy detekovat anomálii. Naopak tím, že útočník se nesnaží dokončit handskake a nenastane úplné TCP spojení, nemůže se také spojení přirozenou cestou ukončit a začne se projevovat značný nepoměr mezi pakety SYN a FIN.

Pomocí konfiguračních proměnných nastavíme tedy váhu paketů SYN i FIN například na 1 a ostatní váhy 0, aby charakteristika počítala pouze tyto dva typy paketů. Výsledná rovnice (po úpravě rovnice charakteristiky (2.1)) vypadá následovně:

$$x_n = \frac{1 \cdot c_{\text{syn}}}{(1 \cdot c_{\text{fin}}) + (0 \cdot c_{\text{rst}}) + (0 \cdot c_{\text{synAck}})} = \frac{c_{\text{syn}}}{c_{\text{fin}}}. \tag{2.4}$$

U této metody je možné přidat do charakteristiky RST pakety. Některé služby mohou ukončovat spojení pomocí RST paketů (místo tradičních FIN paketů). Ovšem je nutné si uvědomit, že příliš mnoho paketů RST může značit i jinou anomálii, a tedy při detekci záplavy SYN paketů nemají RST pakety takovou váhu jako pakety s příznakem FIN.

2.3.2 Detekce nedostupné služby

Nedostupná služba se projeví tak, že po zaslání paketu SYN se nazpět vrátí místo potvrzujícího paketu s příznakem SYN-ACK paket s příznakem RST. Tyto znaky jsou ale podobné jako v případě skenování portů. Více tedy v následující sekci 2.3.3.

2.3.3 Detekce skenování portů

Při skenování portů hledá útočník běžící služby na cílovém počítači. Tato metoda je detailně popsána v analytické části v sekci 1.3.1.3. Jestliže útočník zvolí pro otestování portu metodu pomocí zasílání paketů SYN, ve většině případech (tedy v těch, kdy je port zavřen) obdrží nazpět paket RST nebo vůbec žádnou odpověď, pokud firewall tyto pakety filtruje. Důležité ale je, že útočník nedostane potvrzovací paket SYN-ACK.

Možností jak detekovat tuto anomálii je poměr paketů SYN a SYN-ACK, popřípadě poměr SYN a RST. Pakety RST ale díky firewallu nemusí přicházet, a tak je lepší využít prvního poměru.

Rovnice pro nastavení charakteristiky vypadá například takto:

$$x_n = \frac{1 \cdot c_{\text{syn}}}{(0 \cdot c_{\text{fin}}) + (0 \cdot c_{\text{rst}}) + (1 \cdot c_{\text{synAck}})} = \frac{c_{\text{syn}}}{c_{\text{synAck}}}. \tag{2.5}$$

Je nutno vzít v potaz, že zásuvný modul s metodou NP-CUSUM započítá pakety s příznakem SYN-ACK i do kategorie SYN. Tudíž výsledná rovnice by v optimálním případě měla vyjít v poměru 0,5.

I v tomto případě by bylo možné do rovnice přidat počet RST paketů. Vše ale závisí na vlastnostech síťového provozu a na konkrétním nastavení vah jednotlivých typů paketů.

2.3.4 Detekce resetování TCP spojení

V analytické části 1.3.1.4 byl zmíněn i útok na běžící TCP spojení. V tomto případě útočník zasílá SYN paket, popřípadě RST paket pro přerušení spojení. V průběhu takového útoku by se tedy měl zvýšit jak počet SYN, tak RST paketů oproti normálnímu provozu.

Detekovat tuto anomálii by bylo možné pomocí stejného výpočtu charakteristiky jako v případě (2.4). Tedy po navázání spojení (2 SYN pakety) by přišel paket RST a ukončil spojení. Tímto by se nedostavily příslušné FIN pakety a metoda by vyhlásila alarm.

2.3.5 Co nelze detekovat

Vzhledem k povaze pluginu je zřetelné, že lze detekovat pouze anomálie v TCP spojeních. Veškeré útoky provedené například záplavou pakety UDP, ICMP či skenování portů použitím jiných technik, než pomocí TCP spojení, nelze tímto pluginem detekovat.

Plugin navíc sbírá statistiky pouze o paketech s příznaky SYN, FIN, RST a SYN-ACK a o ostatní pakety se nezajímá. Potom ani útoky provedené pomocí TCP paketů s ostatními typy příznaků nemohou být odhaleny. Stejně tak pokud útočník vytvoří TCP paket a v hlavičce nakombinuje příznaky, které by nikdy neměly nastat, tento zásuvný modul je nedetekuje a tedy nijak nevyhodnotí.

Z uvedených možností detekce je také zřejmé, že i když metoda vyhlásí alarm, v některých případech není možné určit přesná příčina anomálie. Například zda se jedná o nedostupnost služby nebo o skenování portů.

2.3.6 Nastavení prahu

Tato sekce se doposud zabývala nastavením vah jednotlivých paketů charakteristiky, ale přitom v konfiguračních proměnných lze nastavit i hranici (threshold), konstantu ω a časový interval pro výpočet charakteristiky. Tyto údaje spolu souvisí a určují, jak velká odchylka už je považována za anomálii (ω viz (2.3)) a po jak dlouhý čas se musí odchylka objevovat, aby metoda zahlásila anomálii.

Časový interval a hranice je nutné nějakým způsobem odhadnout a protože to není úplně jednoduché, je tomu věnována část v kapitole 4 o testování.

Kapitola 3

Realizace

V této kapitole je popsáno, jak jsem při tvorbě pluginu postupoval, co všechno je nutné naprogramovat, jaké metody využít a na co při vývoji nezapomínat. Postup je psán s předpokladem, že někdo další se bude inspirovat touto prací při tvorbě vlastního zásuvného modulu.

3.1 Vývojové prostředí

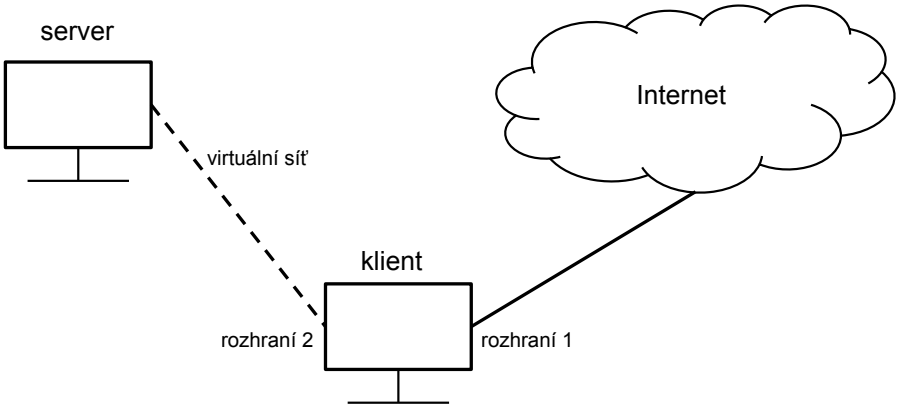
Protože aplikace ucollect, pro kterou je napsán diskutovaný zásuvný modul, běží v operačním systému routeru, bylo nutné vytvořit si vlastní testovací prostředí, sloužící k ověření naprogramované funkcionality. Rozhodl jsem se vytvořit si dva virtuální počítače. První virtuální stroj představuje klientskou část, tedy Turris router. Druhý počítač se chová jako centrální server Turris, který shromažďuje informace od svých klientů a data ukládá do databáze. Tyto dva počítače jsem propojil virtuální sítí. Pro lepší znázornění je virtuální síť vyobrazena na obr. 3.1.

Na obrázku lze vidět, že klient disponuje dvěma rozhraními. Rozhraní 1 slouží k přístupu na internet, a také na tomto rozhraní plugin analyzuje síťový provoz. Druhé rozhraní (rozhraní 2) slouží pouze pro zasílání výstupu pluginu centrálnímu serveru. I když to na obrázku není zobrazeno, server má také přístup na internet. Potřeba internetu je ale především pro instalaci programového vybavení potřebného pro běh Turris serveru.

Na obou virtuálních počítačích je nainstalován operační systém Debian 7 Wheezy s grafickým prostředím.

3.2 Instalace

Návod na instalaci aplikace ucollect na čistý linuxový operační systém je možné najít ve veřejně dostupném git repozitáři [9]. Postup instalace je zde velmi dobře popsán v dokumentu INSTALL, a tak nebude uveden v této



Obrázek 3.1: Virtuální síť pro testování pluginu

práci. Navíc v průběhu vývoje projektu Turris se může postup instalace měnit a mohlo by se stát, že zde přepsaný postup by v době čtení byl již zastaralý a mohl čtenáře akorát zmást.

Jen upozorním, že ucollect nelze spustit, pokud DNS server není zabezpečen, tedy nejedná se o DNSSEC.

3.3 Klientská část

Vyvíjený plugin má implementovanou jak klientskou část, tak i serverovou. Klientská část je psána v jazyce C a framework ucollect poskytuje rozhraní metod, které je nutné využít.

Nejdůležitější část, kterou obsahuje každý plugin je specifikace pluginu a název metod pro callback. Příklad kódu vypadá takto:

```
#ifdef STATIC
struct plugin *plugin_info_cusum(void) {
#else
struct plugin *plugin_info(void) {
#endif
    static struct plugin plugin = {
        .name = "Cusum",
        .packet_callback = packet_handle,
        .init_callback = initialize,
        .uplink_connected_callback = connected,
        .uplink_data_callback = communicate,
        .version = 1
    };
    return &plugin;
}
```

V kódu je možné vidět, že plugin pojmenovaný „Cusum“ je verze 1 a metoda, která se zavolá, když přijde na monitorované rozhraní paket se nazývá „packet_handle“. Inicializační metoda se jmenuje „initialize“, metoda pro zjištění připojení „connected“ a když server potřebuje komunikovat s modulem zavolá se metoda „communicate“.

Tyto metody může programátor libovolně pojmenovat, ale je nutné dodržet správný počet a správné typy parametrů, se kterými jsou metody v callbacku volány. Přehled hlaviček těchto metod v následujícím textu.

3.3.1 Inicializace

Metoda inicializace se volá, jakmile se spustí daný plugin. To znamená při startu serveru, popřípadě pokud je plugin ukončen například díky chybě a znovu nastartován. Hlavička této metody vypadá takto:

```
static void initialize(struct context *context)
```

Při volání má pouze jeden parametr „context“. V těle metody je obvykle inicializace proměnných důležitých pro chod modulu a ty jsou pak alokovány v proměnné context.

3.3.2 Zpracování paketů

Pokaždé, když framework ucollect zachytí paket k analýze, zavolá metodu zadanou v packet_callback, zde „packet_handle“, s parametry „context“ a „packet_info“. Viz:

```
static void packet_handle(struct context *context,
    const struct packet_info *info)
```

Struktura packet_info poskytuje programátorovi analyzované informace o paketu, které jsou přístupné přes rozhraní. Programátor tedy může zjistit tyto parametry paketu:

length: velikost dat paketu

data: data paketu v původní formě [15]

interface: název rozhraní, na kterém byl paket zachycen

hdr_length: délka hlavičky

timestamp: časové razítko z doby zachycení

addresses: zdrojová a cílová adresa paketu

[15] Data z těla paketu by se z důvodu možného porušení soukromí neměla analyzovat.

ports: zdrojový a cílový port

layer: vrstva paketu

ip_protocol: IP protokol

app_protocol: aplikační protokol (například TCP, UDP, ICMP)

direction: směr paketu

tcp_flags: příznaky TCP paketů

Některé z těchto informací poskytuje struktura i v původní nezpracované verzi například „layer_raw“ a „app_protocol_raw“.

Jako příklad použití je zde uvedena podmínka, která zjišťuje, zda se jedná o IP paket. Pokud analyzovaný paket není IP paket, metoda packet_handle se ukončí:

```
if (info->layer != 'I')
    return;
```

3.3.3 Komunikace se serverem

Další důležitou metodou pro komunikaci se serverem je uplink_data_callback, zde „communicate“.

```
static void communicate(struct context *context,
    const uint8_t *data, size_t length)
```

Tato metoda je serverem zavolána se třemi parametry. Parametr „context“, který i zde slouží k předání proměnných v rámci pluginu, dále „data“ a „length“. Proměnná „data“ uchovává pole bytů zaslaných serverem. Může se jednat například o konfigurační nastavení, časové razítko, popřípadě jiné hodnoty. Délka těchto dat je uložena v proměnné „length“.

V pluginu, vytvořeném v této práci, tato metoda očekává od serveru nejprve znak, který odliší, jaká data přišla. Pokud je na prvním místě v datech znak „C“ jedná se o konfigurační data a jestliže obdrží znak „D“ jde o zaslání časového razítka a požádání o vypočítání charakteristiky a metody NP-CUSUM. Po provedení výpočtu plugin připraví zprávu o předem domluvené velikosti a struktuře a odešle ji zpátky na server.

K odeslání zprávy na server využívají moduly metodu, jejíž hlavička vypadá následovně:

```
bool uplink_plugin_send_message(struct context *context,
    const void *data, size_t size)
```

Při zavolání této metody vložíme jako parametry použitý kontext, připravenou zprávu a nakonec i její velikost.

3.4 Serverová část

Serverová část zásuvného modulu je napsaná v jazyce Python. Tato část má za úkol především sbírání dat od klientů a jejich následné ukládání do databáze. I zde je programátorovi poskytnuto rozhraní a metody, které musí pro správnou funkcionalitu využít.

Při vývoji pluginu je nutné vytvořit třídu, která dědí třídu „Plugin“. Rodičovská třída poskytuje dvě metody:

name metoda vrací string ID pluginu, musí být stejné jako v klientské části modulu

message_from_client tato metoda je volána, když klient zašle nějaká data

V následující ukázce kódu je „init“ metoda volající se při volání instance třídy „CusumPlugin“. V této metodě se nastavují proměnné sloužící k chodu pluginu.

```
class CusumPlugin(plugin.Plugin):

    def __init__(self, plugins, config):
        plugin.Plugin.__init__(self, plugins)
        self.__interval = int(config['interval'])
        self.__aggregate_delay = int(config['aggregate_delay'])
        self.__config = config
        self.__downloader = LoopingCall(self.__init_download)
        self.__downloader.start(self.__interval, False)
        self.__data = {}
        self.__last = self.__current = int(time.time())
```

V ukázce lze vidět, že „interval“ a „aggregate_delay“ jsou nastaveny pomocí hodnot z konfiguračního souboru. Dále je do pole „downloader“ přiřazena metoda „init_download“, která žádá všechny klienty, aby zaslali své statistiky. Tato metoda se volá stále dokola po nastaveném časovém intervalu.

3.4.1 Komunikace s klientem

Jak již bylo zmíněno, příchozí data jsou zpracována pomocí metody, jejíž hlavička vypadá takto:

```
def message_from_client(self, message, client)
```

Rodič data přesměruje do správného pluginu a tato metoda, mající parametry „message“ a „client“, tedy pozná od jakého klienta zpráva přišla. V těle metody vývojář rozhoduje, co s daty udělat.

V pluginu vytvořeném v této práci rozhoduje první byte o tom, zda klient požaduje o zaslání konfiguračních dat (byte odpovídá znaku „C“), nebo

jestli klient zasílá data, o která server požádal (byte odpovídá znaku „D“). Při předávání dat mezi klientem a serverem, musí být z dat vytvořena předem definovaná struktura zprávy o předem definované velikosti, tak aby při dekompozici na jedné či druhé straně byla data správně analyzována.

Jakmile je zpráva rozebrána, metoda uloží potřebné hodnoty do databáze pomocí SQL příkazů.

3.4.2 Nastavení konfiguračního souboru

Každý plugin je nutné přidat do konfiguračního souboru, který se při startu centrálního serveru vloží jako parametr. Soubor se nazývá „collect-master.conf“ a jako příklad textu přidaného do souboru je zde uveden název vytvářeného pluginu a jedna jeho konfigurační proměnná:

```
[cusum_plugin.CusumPlugin]
interval: 120 ;Ukázka konfigurační proměnné
```

3.4.3 Databáze

Jelikož PostgreSQL databáze je na straně serveru, zmíním ji v této sekci. Protože vytvářený modul potřebuje ukládat data do databáze, je nutné pro něj vytvořit speciální tabulku. Sloupce tabulky je dobré vytvořit s optimálními datovými typy, aby zbytečně nezabíraly místo v paměti. Nastavení cizích klíčů a jiných klíčů je také samozřejmostí (v příkladu dále je unikátní klíč klient - časové razítko, cizí klíč klient).

Nakonec je ještě nutné nastavit každé tabulce všechna práva pro uživatele ucollect, práva na čtení uživateli archivist a práva na vkládání dat pro uživatele updater. Pokud by práva nebyla nastavena, do databáze by se žádná data zapsat pomocí ucollectu nedala.

Jako příklad je na následujících řádcích uveden SQL příkaz pro vytvoření tabulky pluginu cusum (pro verzi určenou ke sběru dat):

```
CREATE TABLE cusum
(
  "timestamp" timestamp without time zone NOT NULL,
  client integer NOT NULL,
  result double precision NOT NULL,
  syn_count bigint NOT NULL,
  fin_count bigint NOT NULL,
  rst_count bigint NOT NULL,
  syn_ack_count bigint NOT NULL,
  characteristic double precision NOT NULL,
  CONSTRAINT cusum_client_fkey FOREIGN KEY (client)
      REFERENCES clients (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT cusum_timestamp_client_key UNIQUE ("timestamp" , client )
)
```

```
WITH (
  OIDS=FALSE
);
ALTER TABLE cusum
  OWNER TO ucollect;
GRANT ALL ON TABLE cusum TO ucollect;
GRANT INSERT ON TABLE cusum TO updater;
GRANT SELECT ON TABLE cusum TO archivist;
```

Pokud se vývojář rozhodne umístit své konfigurační proměnné do databáze, místo konfiguračního souboru, pak v databázi existuje tabulka s názvem „config“. Stačí specifikovat jméno proměnné, její hodnotu a pro jaký plugin je určena.

Další úprava v databázi souvisí s použitím activity logu v Python skriptu. Kód vypadá například takto:

```
activity.log_activity(client, 'cusum')
```

Aby se do databáze uložila zpráva o aktivitě, je nutné přidat do tabulky s názvem „activity_types“ řádek s id a názvem pluginu.

3.5 Přehled

S realizací pluginu jsme u konce, a tak je dobré zopakovat, co vše je nutné zařídit, aby plugin fungoval. Tyto věci je potřeba udělat:

1. klientská část v jazyce C
2. serverová část v jazyce Python
3. tabulka v PostgreSQL databázi
4. přidat plugin do konfiguračního souboru na serveru
5. nastavit na klientovi, které pluginy použít - viz instalace 3.2, konfigurace klienta
6. upravit soubory Makefile pro správnou kompilaci modulu

Kapitola 4

Testování

Poslední kapitola této práce patří testování vytvořeného pluginu a především analýze nasbíraných dat. Tato část je obzvlášť důležitá, neboť spolehlivost statistické metody NP-CUSUM, je závislá na správném určení parametrů funkce. Proto zkusíme odhadnout, jaké konstanty zvolit tak, abychom docílili nízkého počtu falešných poplachů a zároveň anomálii detekovali s co nejmenším zpožděním.

4.1 Data

Testování zásuvného modulu bylo provedeno ve vývojovém prostředí popsaném v sekci 3.1. Jako součást testování byla modulem sbírána i data z provozu v těchto virtuálních strojích. Protože síťový provoz ovšem nebyl dostatečně velký a zajímavý, bylo možné tímto způsobem pouze ověřit správnou funkcionalitu modulu či zachytávání a analýzy paketů.

Z tohoto důvodu jsem požádal společnost CZ.NIC, která stojí za projektem Turris, o poskytnutí většího počtu dat nasbíraných jejich routery. Společnost mi nakonec poskytla anonymizovaná vybraná data z 978 Turris routerů sbíraná po dobu 9 dní.

4.2 Analýza dat

Poskytnutá data obsahují záznamy o počtu zachycených TCP paketů. Data byla sbírána centrálním serverem každých 5 minut a každý řádek dat uchovává informace o číslu klienta, počtu a typu příznaků. Jedná se o zhruba 13 milionů záznamů v databázi.

Pro zpracování takhle velkého množství dat jsem využil program R Studio, který k matematickým operacím využívá jazyk R a který je vhodný i pro vykreslení přehledných grafů.

4.2.1 Obecné statistiky

Pro hledání anomálií v datech je nutné nejdříve určit, jak vypadá „normální“ provoz. Roztřídil jsem data podle klientů i dle pětiminutových časových úseků a vypočítal jsem průměrné hodnoty v poměru paketů $\frac{\text{SYN}}{\text{FIN}}$ a $\frac{\text{SYN}}{\text{SYN-ACK}}$.

Tabulka 4.1: Statistika poměru paketů SYN a FIN a poměru SYN a SYN+ACK

Třídění	Poměr	Průměr	Medián
klienti	SYN/FIN	1,70	1,15
klienti	SYN/SYN+ACK	1,86	1,12
časový interval	SYN/FIN	1,99	1,58
časový interval	SYN/SYN+ACK	2,66	1,89

Z tabulky 4.1 lze vidět, že pokud data vytřídíme podle klientů, dosahujeme lepších výsledků (myšleno blížíme se specifikaci protokolu TCP), než v případě rozdělení dle časových úseků. Například medián hodnoty poměru paketů SYN/FIN u poloviny klientů menší než 1,15 a u 70% klientů je poměr do hodnoty 1,3 (což je stále menší než průměr).

Tyto lepší výsledky u klientů předznamenávají, že anomálie bude možné detekovat, protože se týkají pouze menší části klientů. Cílem tedy je, nalézt nějaké anomálie u některého klienta.

4.2.2 Poměr paketů SYN a FIN

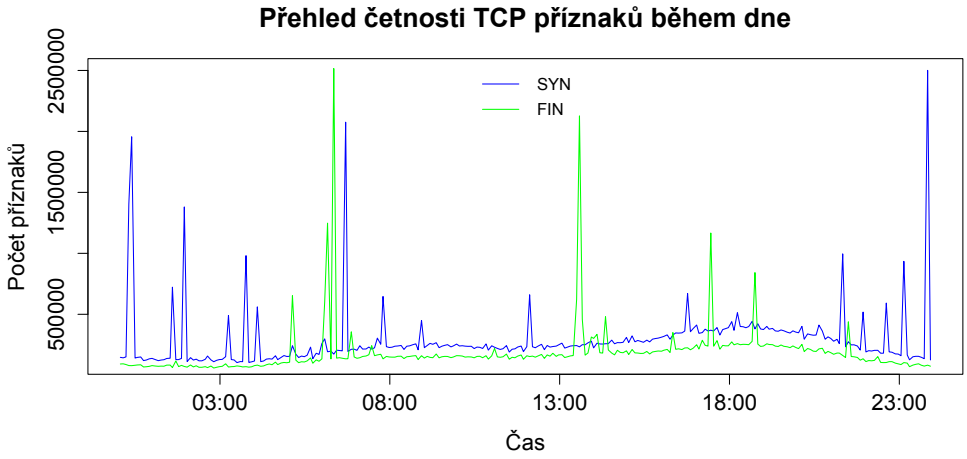
Při hledání anomálií nás zajímá především poměr zvolených paketů. Pro ukázku by bylo vhodné zobrazit i počet paketů, ze kterého se poměr počítá a jak se tento počet mění během dne. Na grafu 4.1 jsou znázorněny počty paketů SYN a FIN během dne na všech klientech dohromady.

Z grafu 4.1 lze vyčíst, že největší provoz je zaznamenán okolo 18. hodiny a nejvíce SYN paketů bylo zaznamenáno v nočních hodinách. Během dne byl zaznamenán i nárůst paketů FIN. Na dalším grafu 4.2 je zobrazen již poměr $\frac{\text{SYN}}{\text{FIN}}$ z téhož dne. Výsledek tohoto poměru není překvapivý a opět znázorňuje, že anomálie nastaly převážně v nočních hodinách.

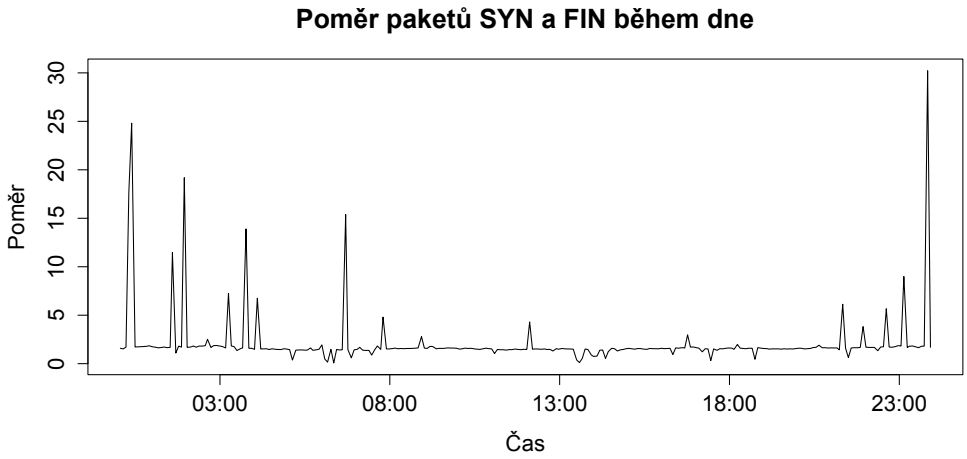
Zobrazená data jsou agregovaná ze všech klientů, proto v následující části se zaměříme na konkrétní klienty, u kterého byla či nebyla zachycena nějaká anomálie a zkusíme aplikovat metodu NP-CUSUM pro její detekci.

4.2.2.1 Klient bez anomálie

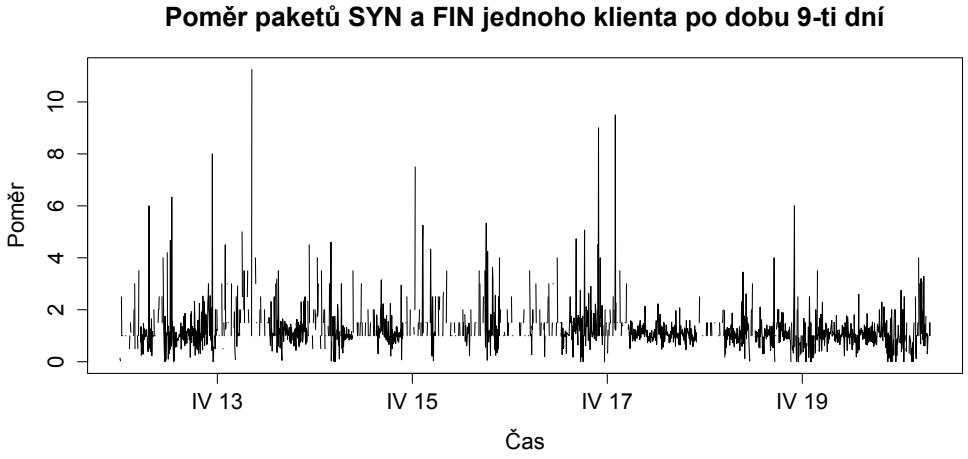
Nejprve jsem vybral klienta, který díky poměru počtu paketů $\frac{\text{SYN}}{\text{FIN}}$ i $\frac{\text{SYN}}{\text{SYN-ACK}}$ zhruba odpovídal celkovému naměřenému mediánu. Graf 4.3 zobrazuje poměr paketů SYN a FIN vybraného klienta.



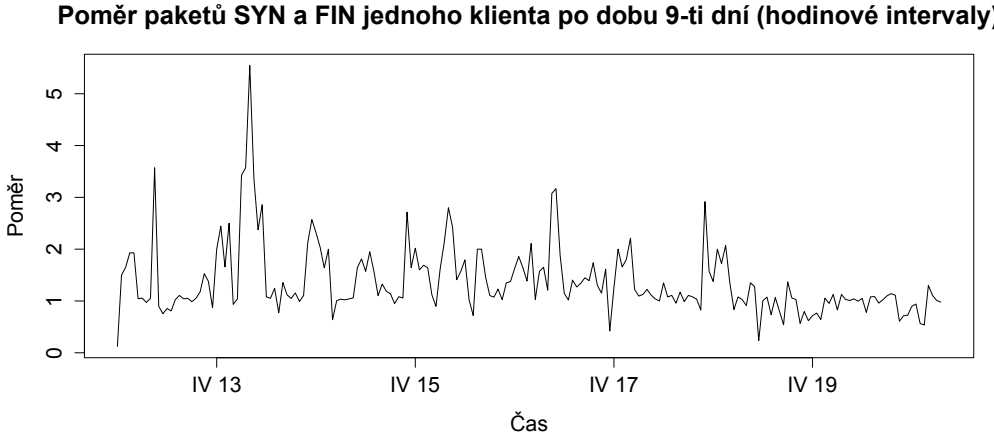
Obrázek 4.1: Počet paketů SYN a FIN zachycených během jednoho dne.



Obrázek 4.2: Poměr paketů SYN a FIN zachycených během jednoho dne.



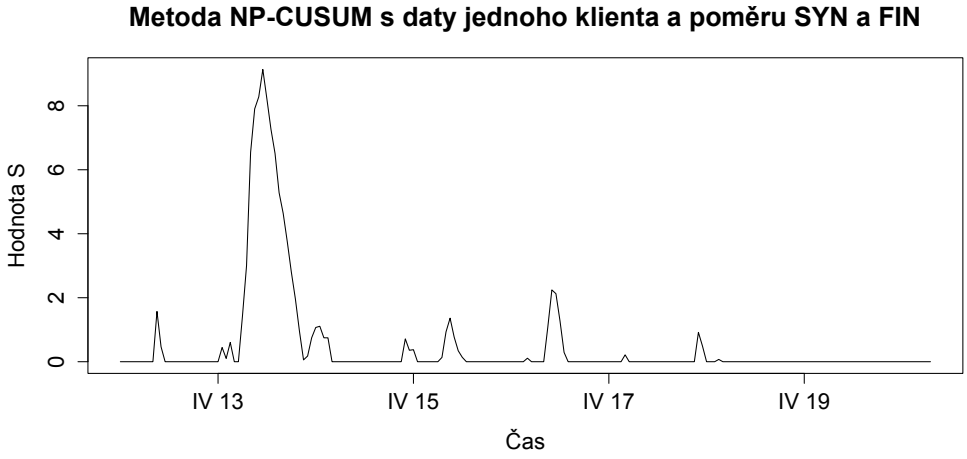
Obrázek 4.3: Poměr paketů SYN a FIN klienta bez anomálie po dobu 9-ti dní.



Obrázek 4.4: Poměr paketů SYN a FIN klienta bez anomálie po dobu 9-ti dní, ale počítáno v hodinových intervalech.

Přestože průměrný poměr odpovídal hodnotě přibližně 1,15, dle grafu 4.3 se zdá, že poměr se blíží k hodnotám až kolem 10. Nemusí se ovšem jednat o anomálii. Pouze ale o špatné nastavení časového intervalu. Pokud přijde za časový interval malý počet paketů, poměr je snadno ovlivněn. Proto můžeme namísto zobrazovaných 5 minutových intervalů, počítat charakteristiku po časovém intervalu jedné hodiny, výsledný graf 4.4 již neobsahuje tolik odchylek a maximální hodnota poměru paketů klesla na polovinu.

Jak vypadá křivka vykreslená pomocí dat z metody NP-CUSUM při nastavení hodnoty $\omega = 2$ (v rovnici (2.2)) zobrazuje graf 4.5. V jednu chvíli to



Obrázek 4.5: Průběh metody NP-CUSUM s daty klienta bez anomálie při využití poměru paketů SYN a FIN.

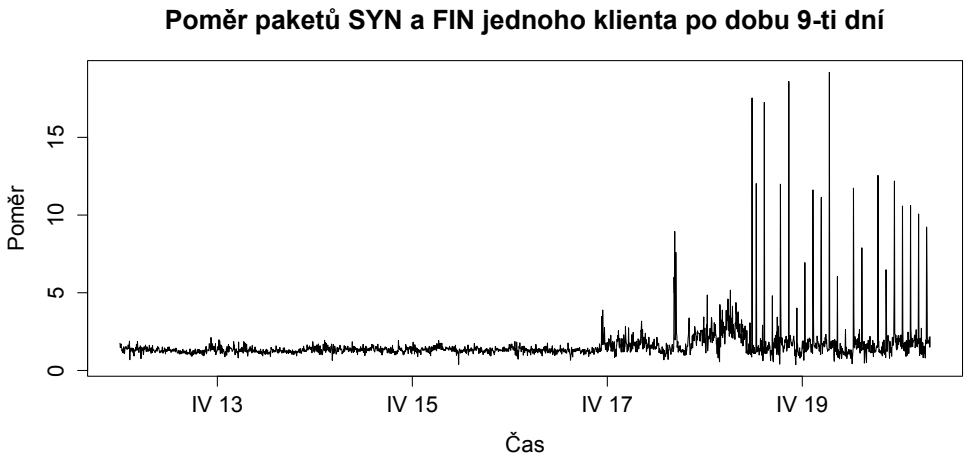
vypadá, že metoda objevila síťovou anomálii, ale při detailní kontrole dat se v tomto případě jedná spíše o počítání charakteristiky z malého počtu dat. Pokud by byl práh (threshold) nastaven v takto malých hodnotách, pak by se jednalo o falešný alarm. Data by bylo možné sloučit do ještě větších časových intervalů, ale tím by se již podstatně zvětšila průměrná doba detekce.

4.2.2.2 Klient s anomálií

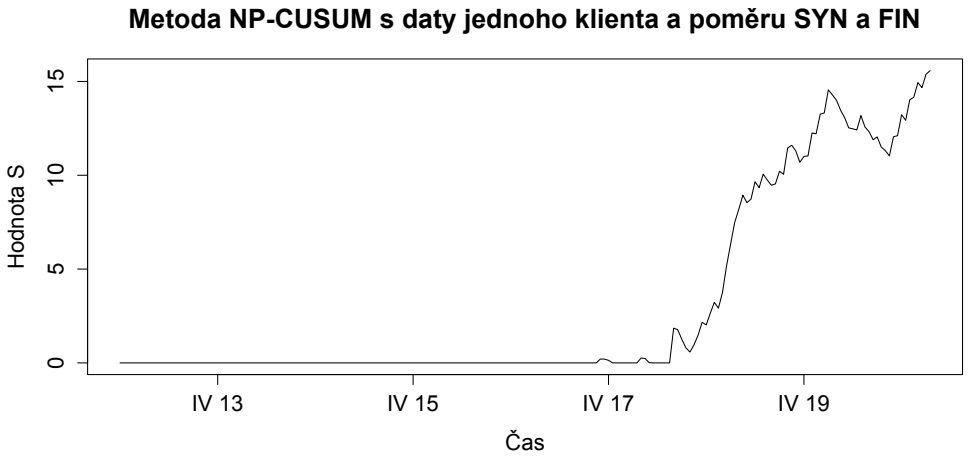
Mezi obdrženými daty jsem nalezl klienta, jehož provoz byl dostatečně velký, tak aby poměr paketů nebyl ovlivněn a zároveň byla v průběhu sledovaných devíti dní detekována anomálie. Na grafu 4.6 je možné vidět, jak se v posledních dnech detekce zvýšil poměr paketů SYN a FIN.

Po převedení na hodinové časové intervaly a použití poměru jako charakteristiky pro metodu NP-CUSUM, dosahuje statistika v nejvyšším bodě hodnoty 15. Celý průběh metody NP-CUSUM pak zobrazuje graf 4.7. Nastavení konstant pro metodu NP-CUSUM v tomto případě vypadá takto: váha paketů SYN a FIN = 1, váha RST a SYN+ACK = 0, $\omega = 1{,}8$.

Dle nasbíraných dat byl v době anomálie zvýšený počet paketů SYN oproti paketům s příznakem FIN, ale zároveň byl zvýšený počet paketů SYN i proti paketům se SYN+ACK. To znamená, že při anomálii nebyl proveden trojcestný handshake a protože počet paketů nebyl nijak závratný oproti jiným dnům, zřejmě se nejedná ani o záplavu pakety SYN a následné vyčerpání možných TCP spojení. Anomálie tedy mohla vzniknout nedostupností nějaké služby či při skenování portů.



Obrázek 4.6: Poměr paketů SYN a FIN klienta s anomálií po dobu 9-ti dní.

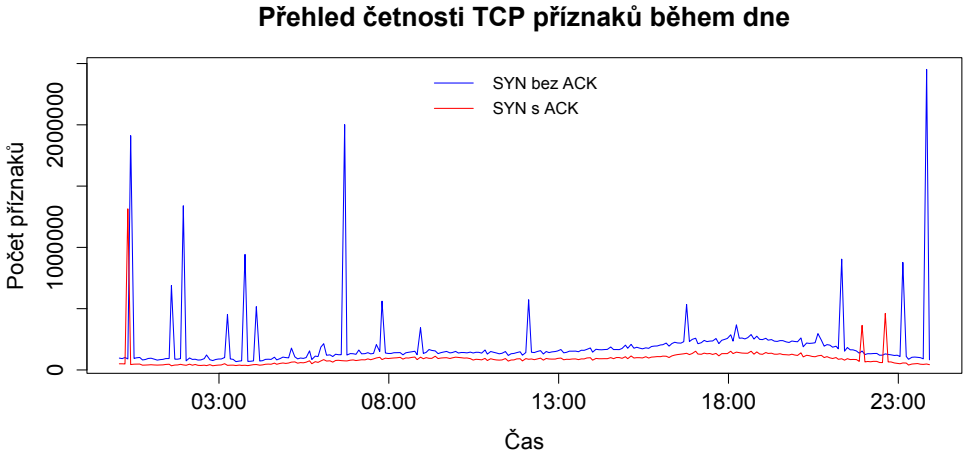


Obrázek 4.7: Průběh metody NP-CUSUM s daty klienta s anomálií při využití poměru paketů SYN a FIN.

4.2.3 Poměr paketů SYN a SYN+ACK

Pro správné počítání charakteristiky s využitím poměru paketů SYN a SYN - ACK je nutné nastavit v rovnici (2.2) váhu SYN paketů na 0,5 a váhu SYN - ACK na 1, protože počet SYN paketů zahrnuje i pakety SYN - ACK. Jelikož princip je podobný jako při počítání $\frac{\text{SYN}}{\text{FIN}}$, nebude zde tento poměr rozepsán příliš detailně.

Pro ukázku je opět zobrazen graf 4.8 , který představuje počty paketů SYN bez ACK a počet paketů SYN s ACK zachycených během jednoho dne.



Obrázek 4.8: Počet paketů SYN a SYN-ACK zachycených během jednoho dne.

4.2.4 Určení parametrů

Jak již bylo v práci několikrát zmíněno, na zvolených parametrech závisí typ detekované anomálie, velikost detekovatelné odchylky od normálního provozu či míra falešných poplachů a průměrné zpoždění detekce.

Při analýze dat bylo nalezeno mnoho různých anomálií, které byly nalezeny za pomoci užití charakteristiky poměru paketů SYN a FIN či SYN a SYN - ACK. Z pozorování jsem odvodil, že parametr ω přímo souvisí s délkou časového intervalu. Čím je časový interval kratší, tím vyšší musíme parametr ω nastavit, abychom předešli falešným poplachům. Je to z důvodu malého počtu zachycených paketů, kde při výsledku poměru hraje každý jeden paket zásadní roli.

Dalším kritériem pro určení délky časového intervalu je počet případů, kdy při počítání charakteristiky vyjde dělenec či dělitel 0. Celá charakteristika je pak dle implementace pluginu nulová a do metody NP-CUSUM se nepromítne. Při pětiminutových intervalech se objevuje 0 zhruba v každém 10. případu, kdežto při hodinových intervalech nuly téměř úplně vymizí.

Pokud chceme nalézt optimální nastavení parametrů, záleží na tom, jak velkou odchylku od normálu již chceme detekovat. Dle analýzy dat můžeme považovat jako vhodné nastavení volbu parametru ω mezi hodnotami 1.5 - 2. A časový interval mezi půl hodinou až hodinou.

Metoda NP-CUSUM je vhodná především tam, kde je větší množství zachycených paketů a kde anomálie měla delší pozvolné trvání. Nastavení prahu, kdy metoda vyvolá alarm, je už libovolné. Čím menší práh, tím dříve bude odhalena anomálie, ale zvýší se i míra falešných poplachů. Tento parametr tedy také souvisí s parametrem ω a časovým intervalem.

Závěr

Cílem práce bylo seznámit se s projekem Turris, se síťovými útoky a statistickými metodami zaměřenými na detekci bodu změny. Cílem bylo také vytvořit a otestovat zásuvný modul pro výzkumný projekt Turris, který by na základě analýzy síťových toků pomocí statistických metod byl schopen odhalit síťové anomálie.

Výsledkem této práce jsou dva podobné funkční zásuvné moduly, které se mírně liší konfiguračními proměnnými a zapisovanými daty do databáze. Oba tyto moduly implementují statistickou metodu NP-CUSUM, která je schopná detekovat při různém nastavení různé síťové anomálie. V kapitole realizace je popis implementace těchto pluginů popsán tak, aby bylo možné proces vývoje jednoduše zopakovat a čtenář byl schopen na základě textu vytvořit další zásuvný modul pro projekt Turris.

Testování funkcionality pluginu proběhlo ve virtuálním prostředí. Vhodnost metody NP-CUSUM bylo možné otestovat pomocí analýzy dat z reálného prostředí, které jsem na základě dohody o jejich následném nezveřejnění, obdržel od sdružení CZ.NIC, které stojí právě za projektem Turris. Díky velkému rozsahu obdržených dat bylo možné při analýze detekovat několik anomálií. Původ anomálií bylo možné pouze odhadnout, jelikož data byla z důvodu ochrany soukromých údajů anonymní a neměl jsem více informací o prostředí, ve kterém byla sbírána.

Zadání diplomové práce jsem tedy splnil a během vypracovávání této práce jsem se seznámil s mnoha novými metodami a technologiemi. Především projekt Turris mě překvapil svými možnostmi a potenciálem. Mnou implementovaný plugin byl v tomto okamžiku předán sdružení CZ.NIC a očekávám, že data budou k dispozici v době obhajoby této práce. Také doufám, že modul napomůže detekci anomálií v sítích uživatelů routerů Turris.

Jako oblast dalšího rozvoje této práce bych viděl spíše implementaci nějakého nového modulu pro projekt Turris, než úpravu stávajícího. Díky této práci by to nyní měl mít vývojář značně ulehčené, také zde implementované moduly mu mohou sloužit jako inspirace. I tak je ale možné mnou vytvořený

modul vylepšit například přidáním experimentální optimalizace parametrů implementovaného algoritmu, nebo dalšího způsobu detekce anomálií s pomocí jiné statistické metody či implementování kontroly již nalezené anomálie, která by snížila počet falešných poplachů.

Literatura

[1] Špringl, P.: Monitorování provozu sítě. [cit. 3.5.2015]. Dostupné z: http://www.systemonline.cz/it-security/monitorovani-provozu-site.htm

[2] NetFlow. Duben 2015, page Version ID: 656748318. Dostupné z: http://en.wikipedia.org/w/index.php?title=NetFlow&oldid=656748318

[3] The Top 20 Free Network Monitoring and Analysis Tools for Sys Admins. [cit. 3.5.2015]. Dostupné z: http://www.gfi.com/blog/the-top-20-free-network-monitoring-and-analysis-tools-for-sys-admins/

[4] Turris. [cit. 20.3.2015]. Dostupné z: https://www.turris.cz/cs/

[5] OpenWrt. [cit. 2.4.2015]. Dostupné z: https://openwrt.org/

[6] Rohleder, D.: Přechodové mechanismy k IPv6. [cit. 3.5.2015]. Dostupné z: http://ics.muni.cz/bulletin/articles/667.html

[7] Turris - uživatelská dokumentace [Project: Turris]. [cit. 22.3.2015]. Dostupné z: https://www.turris.cz/doc/

[8] iptables. Duben 2015, page Version ID: 655946091. Dostupné z: http://en.wikipedia.org/w/index.php?title=Iptables&oldid=655946091

[9] Turris - GitLab. [cit. 10.4.2015]. Dostupné z: https://gitlab.labs.nic.cz/turris/ucollect/

[10] Dewaele, G.; Fukuda, K.; Borgnat, P.; aj.: Extracting hidden anomalies using sketch and non Gaussian multiresolution statistical detection procedures. In *Proceedings of the 2007 workshop on large scale attack defense*, LSAD '07, ACM, Srpen 2007, ISBN 9781595937858, s. 145–152, doi:10.1145/1352664.1352675.

[11] Turris - libatsha204. [cit. 3.5.2015]. Dostupné z: https://gitlab.labs.nic.cz/turris/libatsha204

[12] Minařík, P.: Pokročilá analýza provozu datových sítí. [cit. 3.5.2015]. Dostupné z: http://www.systemonline.cz/clanky/pokrocila-analyza-provozu-datovych-siti-2-dil.htm

[13] Wireshark About. [cit. 12.4.2015]. Dostupné z: https://www.wireshark.org/about.html

[14] NetFlow/IPFIX - INVEA-TECH. [cit. 10.4.2015]. Dostupné z: https://www.invea.com/cs/sitova-reseni/netflow/ipfix

[15] Almquist, P.: Type of Service in the Internet Protocol Suite. [cit. 8.4.2015]. Dostupné z: https://tools.ietf.org/html/rfc1349

[16] Thottan, M.; Ji, C.: Anomaly detection in IP networks. *IEEE Transactions on Signal Processing*, ročník 51, č. 8, 2003: s. 2191–2204, ISSN 1053-587X, doi:10.1109/TSP.2003.814797.

[17] Tixteco, L. P.; A, E. A.; Hdez, A. F. M.; aj.: DoS Attacks Flood Techniques. *International Journal of Combinatorial Optimization Problems and Informatics*, ročník 3, č. 2, Květen 2012: s. 3–13, ISSN 2007-1558.

[18] Denial of Service (DoS) útoky: typy využívající chyb a vyčerpání systémových prostředků (1.). [cit. 8.4.2015]. Dostupné z: http://www.lupa.cz/clanky/typy-vyuzivajici-chyb-a-vycerpani-systemovych-prostredku-1/

[19] Rajapandian, P.; Alagarsamy, K.: Intrusion Detection in Dos Attacks. *International Journal of Computer Applications*, ročník 15, č. 8, Leden 2011, ISSN 0975-8887, doi:10.5120/1967-2634.

[20] Transmission Control Protocol. Duben 2015, page Version ID: 657316471. Dostupné z: http://en.wikipedia.org/w/index.php?title=Transmission_Control_Protocol&oldid=657316471

[21] RFC-Editor Webpage. [cit. 5.4.2015]. Dostupné z: http://www.rfc-editor.org/index.html

[22] Skenování portů: teorie - Lupa.cz. [cit. 8.4.2015]. Dostupné z: http://www.lupa.cz/clanky/skenovani-portu-teorie/

[23] TCP reset attack. Leden 2015, page Version ID: 644858705. Dostupné z: http://en.wikipedia.org/w/index.php?title=TCP_reset_attack&oldid=644858705

[24] Resetovací útoky na TCP spojení - Lupa.cz. [cit. 8.4.2015]. Dostupné z: http://www.lupa.cz/clanky/resetovaci-utoky-na-tcp-spojeni/

[25] Bhuyan, M. H.; Bhattacharyya, D. K.; Kalita, J. K.: Network Anomaly Detection: Methods, Systems and Tools. *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS*, ročník 16, č. 1, 2014: s. 303–336, ISSN 1553-877X, doi:10.1109/SURV.2013.052213.00046.

[26] Dutta, D.; Choudhury, K.: Network Anomaly Detection using PSO-ANN. *International Journal of Computer Applications*, ročník 77, č. 2, Leden 2013, ISSN 0975-8887, doi:10.5120/13368-0968.

[27] Yu, M.: A Nonparametric Adaptive Cusum Method And Its Application In Network Anomaly Detection. *International Journal of Advancements in Computing Technology*, ročník 4, č. 1, 2012.

[28] Blažek, R.; Kim, H.; Rozovskii, B.; aj.: A novel approach to detection of 'Denial-of-Service' attacks via adaptive sequential and batch-sequential change-point detection methods. In *Proc. 2nd IEEE Systems, Man, and Cybernetics Information Assurance Workshop*, West Point, NY, June 2001.

[29] Tartakovsky, A.; Rozovskii, B.; Blazek, R. B.; aj.: A Novel Approach to Detection of Intrusions in Computer Networks via Adaptive Sequential and Batch-sequential Change-point Detection Methods. *IEEE Transactions on Signal Processing*, ročník 54, č. 9, September 2006: s. 3372–3382.

[30] Tartakovsky, A. G.; Rozovskii, B. L.; Blažek, R. B.; aj.: Detection of intrusions in information systems by sequential change-point methods. *Statistical Methodology*, ročník 3, č. 3, 2006: s. 252–293, ISSN 1572-3127, doi:10.1016/j.stamet.2005.05.003.

[31] Invea-Tech: Behaviorální analýza datového provozu v praxi. [cit. 12.4.2015]. Dostupné z: `https://www.invea.com/data/articles/2014-06-DSM.pdf`

[32] Ertoz, L.; Eilertson, E.; Lazarevic, A.; aj.: Minds-minnesota intrusion detection system. *Next Generation Data Mining*, 2004: s. 199–218.

[33] Jyothsna, V.; Prasad, V. R.; Prasad, K. M.: A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, ročník 28, č. 7, 2011: s. 26–35. Dostupné z: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.1390&rep=rep1&type=pdf`

[34] Butun, I.; Morgera, S. D.; Sankar, R.: A survey of intrusion detection systems in wireless sensor networks. *Communications Surveys & Tutorials, IEEE*, ročník 16, č. 1, 2014: s. 266–282. Dostupné z: `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6517052`

[35] Taylor, D. W. A.: Change-Point Analysis: A Powerful New Tool For Detecting Changes. [cit. 12.4.2015]. Dostupné z: http://www.variation.com/cpa/tech/changepoint.html

[36] Page, E. S.: Continuous inspection schemes. *Biometrika*, 1954: s. 100–115. Dostupné z: http://www.jstor.org/stable/2333009

[37] CUSUM. Listopad 2014, page Version ID: 633625161. Dostupné z: http://en.wikipedia.org/w/index.php?title=CUSUM&oldid=633625161

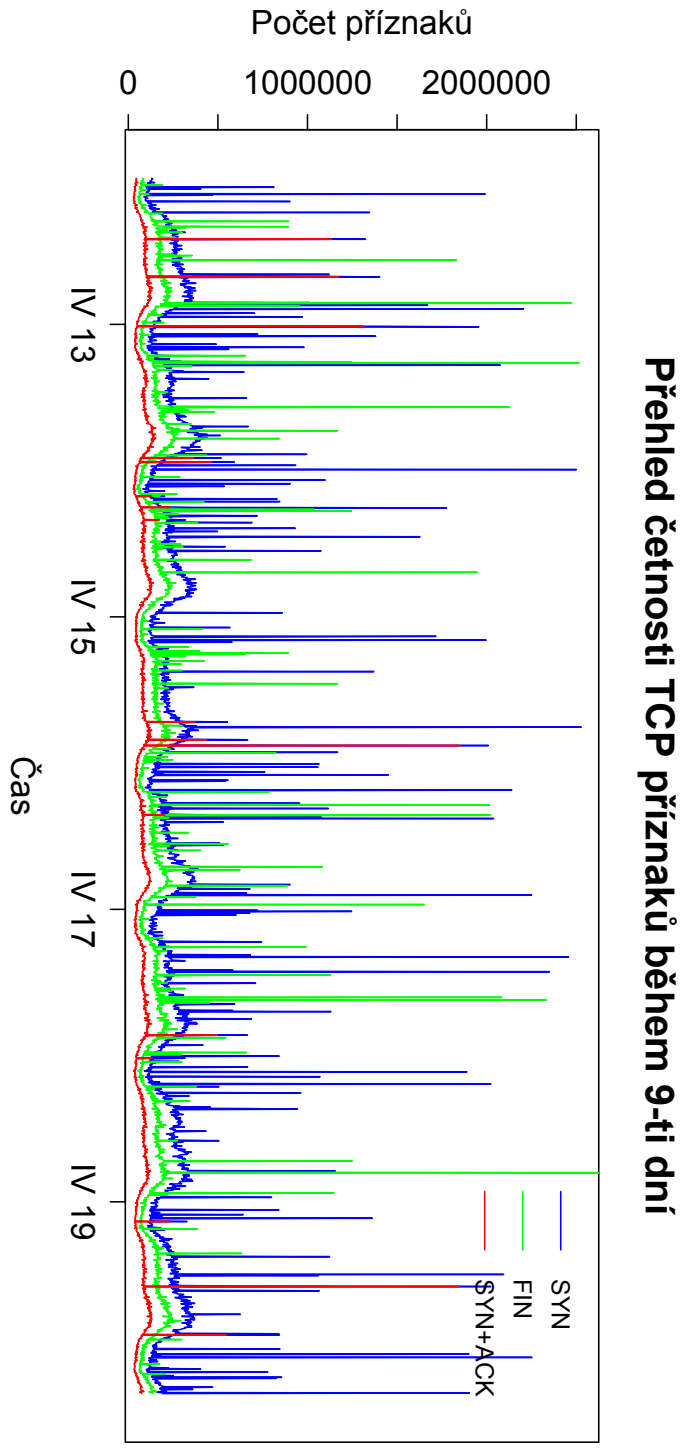
[38] PostgreSQL. Duben 2015, page Version ID: 659440432. Dostupné z: http://en.wikipedia.org/w/index.php?title=PostgreSQL&oldid=659440432

[39] PostgreSQL. [cit. 15.4.2015]. Dostupné z: http://www.postgresql.org/

[40] pgAdmin: Advocacy - Introduction. [cit. 18.4.2015]. Dostupné z: http://www.pgadmin.org/advocacy/

Příloha **A**

Přehled počtů TCP paketů



Obrázek A.1: Počet paketů SYN, FIN a SYN-ACK zachycených na všech klientech během 9-ti dnů. V počtu paketů SYN jsou zahrnuty i pakety SYN-ACK.

Příloha B

Seznam použitých zkratek

LAN Local Area Network

WAN Wide Area Network

DNS Domain Name System

DNSSEC Domain Name System Security Extensions

IP Internet Protocol

SSH Secure Shell

NAT Network Address Translation

TCP Transmission Control Protocol

UDP User Datagram Protocol

ICMP Internet Control Message Protocol

DoS Denial of Service

DDoS Distributed Denial of Service

RFC Request For Comments

IDS Intrusion Detection System

SQL Structured Query Language

SNMP Simple Network Management Protocol

ToS Type of Service

NIDS Network Intrusion Detection System

CPD Change Point Detection

ADD Average Detection Delay

FAR False Alarm Rate

CUSUM Cumulative Sum

NP-CUSUM Non-Parametric Cumulative Sum

ACID Atomicity, Consistency, Isolation, Durability

MVCC Multi-Version Concurrency Control

TOAST The Oversized-Attribute Storage Technique

XML Extensible Markup Language

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `ucollect` ..... zdrojové kódy frameworku
    - `pluginy` ..... zdrojové kódy modulů
      - `plugin_hotovy` ..... plugin pro zasílání upozornění
      - `plugin_testovaci` ..... plugin pro sběr dat
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS