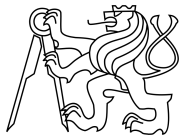
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Automatická analýza nahlášených bezpečnostních incidentů
Student:	Bc. Adam Plánský
Vedoucí:	Ing. Tomáš Čejka
Studijní program:	Informatika
Studijní obor:	Počítačová bezpečnost
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Nastudujte formát IDEA [1] pro reprezentaci detekovaných bezpečnostních incidentů hlášených detekčními systémy (IDS, honeypot apod.).
Proveďte rešerši aktuálních hrozeb a útoků v počítačových sítích s cílem zmapovat typy, závažnosti a četnosti bezpečnostních incidentů.
Seznamte se s typy IDEA zpráv, které jsou v současnosti odesílány do systému Warden [2] pro sbírání a sdílení informací o bezpečnostních incidentech.
Na základě analýzy dat dodaných vedoucím navrhněte algoritmus pro prioritizaci a výběr nahlášených incidentů za účelem včasného spuštění záchytu důkazního materiálu probíhajících incidentů a jejich forenzní analýzy pomocí existujícího systému popsaného v [3].
Navržený algoritmus implementujte jako prototyp v jazyce Python.
Výsledné řešení otestujte a vyhodnoťte pomocí dat dodaných vedoucím.

Seznam odborné literatury

[1] https://idea.cesnet.cz/en/index
[2] https://warden.cesnet.cz/cs/index
[3] Z. Rosa, T. Cejka, M. Zadnik, and V. Puš, "Building a Feedback Loop to Capture Evidence of Network Incidents," in *12th International Conference on Network and Service Management (CNSM 2016)*, Montreal, Canada, 2016.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 19. prosince 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Diplomová práce

Automatická analýza nahlášených bezpečnostních incidentů

Bc. Adam Plánský

Vedoucí práce: Ing. Tomáš Čejka

9. května 2017

Poděkování

Za odborné vedení, cenné rady a věcné připomínky při vytváření diplomové práce děkuji panu Ing. Tomáši Čejkovi. Dále děkuji své rodině za celoživotní podporu.

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 9. května 2017 .....

Odkaz na tuto práci

Plánský, Adam. *Automatická analýza nahlášených bezpečnostních incidentů.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Detekční systémy na počítačových sítích detekují více bezpečnostních událostí, než je možné zpracovat manuálně. Tato práce se zabývá snížením počtu těchto bezpečnostních událostí, což pomůže bezpečnostním týmům se zpracováním detekovaných událostí. Algoritmus je přizpůsobitelný cílovému nasazení pomocí konfiguračních parametrů. Navržený systém redukuje počet bezpečnostních událostí na jednotky procent.

Klíčová slova bezpečnostní incident, selekce bezpečnostních incidentů, automatizované zpracování dat, páteřní sítě, záchyt paketů

Abstract

An intrusion detection systems in computer networks detect more security events than can be manually handled. This thesis is concerned with reduction of these security events, which should help security teams process the detected events. The algorithm is customizable for production deployment with configuration parameters. The proposed system reduces the number of security events to single-digit percentages.

Keywords security incident, selection of security incidents, automatic data processing, backhaul, packet capture

Obsah

Seznam obrázků

Úvod

Žijeme v době masového rozšíření elektronických zařízení, která jsou nedílnou součástí našeho života. Většina z nás bere počítače, mobilní telefony a další zařízení, které spolu komunikují skrze internet, jako samozřejmost. Ovšem za všemi výhodami moderních technologií se skrývá mnoho nástrah, které nesmíme opomíjet. Základní potřebou každého operátora síťové infrastruktury je zajistit bezproblémový chod sítě a bránit ji proti kybernetickým hrozbám.

V dnešní době existuje nespočet útoků na počítačové sítě a zařízení uvnitř sítě. Motivace útočníků je různá, ať už jde o finanční zisk, odstavení určité služby, či získání důvěryhodných dat, je proto nutné se proti těmto útokům umět bránit. Zejména díky relativní anonymitě na internetu a snadné vidiny zisku jsou kybernetické útoky [1] velice časté.

Monitorování síťového provozu a následné zpracování, oznámení, či zamezení síťového útoku, je nezbytnou součástí moderních počítačových sítich. Sledování síťového provozu pomocí plně automatizovaných nástrojů předcházíme bezpečnostním incidentům, nebo se o bezpečnostním incidentu alespoň dozvíme, a dokážeme tak minimalizovat vzniklé škody. Existuje mnoho nástrojů, které se nasazují na různá místa počítačové sítě. Ať už to jsou lokální sítě, či páteřní sítě, bez těchto plně automatizovaných mechanismů bychom odhalili jen hrstku útoků, a proto je nutné se touto problematikou zabývat.

Analýza provozu páteřní sítě v reálném čase, kde skrze jednotlivé sondy proudí síťový provoz v řádech desítek Gb/s detekuje stovky bezpečnostních incidentů za minutu.

Bezpečnostní incidenty, které jsou na síti detekovány, jsou předány k vyhodnocení příslušné členům bezpečnostních týmu nebo systému. Existují plně automatizované nástroje, které dokáží selektivně zachytit vzorek škodlivého provozu. Pomocí těchto systémů je získáván vzorek dat, který je dále analyzován.

Počet bezpečnostních incidentů na páteřní síti je ovšem o mnoho větší, než jsme schopni manuálně zpracovávat.

Cílem práce je navrhnout a implementovat systém pro automatickou se-

lekci a prioritizaci detekovaných bezpečnostních incidentů na síti CESNET2. Navržený algoritmus rozhodne, které bezpečnostní události má smysl analyzovat ve vyšší míře detailu.

Bezpečnostních incidentů se detekuje neúnosné množství, a proto musíme snížit jejich počet a vybrat pouze důležité nahlášené bezpečnostní incidenty pro následné zpracování. Vybrané incidenty poté budou zachyceny a následně podrobeny automatizované paketové analýze. Tatko uložená data slouží jako důkazní materiál při síťových útocích. Dále tyto záchyty mohou vylepšovat současné detekční algoritmy. Výsledek této práce pomůže bezpečnostním týmům v síti CESNET2 při řešení bezpečnostních incidentů.

Práce je rozdělena do pěti částí:

Kapitola 1 popisuje existující formáty bezpečnostních události a jejich životní cyklus. Kapitola 2 popisuje analýzu infrastruktury sítě CESNET2, na které bude nasazen výsledný systém. Kapitola 3 popisuje návrh systému pro selekci bezpečnostních událostí. Kapitola 4 popisuje implementaci systému pro selekci bezpečnostních událostí. Kapitola 5 popisuje výsledky testování implementovaného systému z reálného provozu na páteřní síti CESNET2.

Kapitola 1

Úvod do problematiky

Bezpečnostní incident [2] je kybernetická bezpečnostní událost, která může způsobit narušení bezpečnostní informací v informačních systémech nebo narušení bezpečnosti služeb a sítí elektronických komunikací. Bezpečnostní událostí se myslí jev, který můžeme pozorovat uvnitř systému, či sítě. Pod pojmem narušení bezpečnosti se nemyslí přírodní pohroma, výpadek proudu, či jiné události, které nemůžeme přímo ovlivnit.

Existují různé datové reprezentace (formáty) bezpečnostních událostí. Jednotlivé datové reprezentace se od sebe liší zejména tím, jakým způsobem popisují bezpečnostní incident. Některé z nich popisují celý útok, kde může být několik bezpečnostních incidentů, jako jednu bezpečnostní událost. Jiné reprezentace popisují každý bezpečnostní incident jednotlivě. Dále se liší modifikovatelností své struktury a rozšiřitelností.

1.1 Životní cyklus bezpečnostní události

Na určitých bodech sítě jsou monitorovací sondy, které exportují síťové toky do detekčních systémů. Tyto detekční systémy provádějí automatické detekce škodlivého provozu, které jsou založeny na vzorkování, anomáliích, či jiném detekčním algoritmu.

Při detekci bezpečnostní události se vytvoří zpráva, která je většinou čitelná jak pro člověka, tak pro počítačové zpracování. V těle této zprávy jsou zapsány veškeré důležité informace, které detekční algoritmus zjistil v průběhu útoku. Typickými položkami ve zprávě jsou čas detekce, typ útoku, zdrojová adresa, cílová adresa, jméno detektoru. Dále se vyskytují v těle zprávy další položky, které jsou závislé na formátu bezpečnostní události, typu útoku nebo specifikaci struktury incidentu.

Jeden z hlavních problému detekčních systému je chybná detekce bezpečnostních incidentů, kdy je normální provoz nesprávně označen za škodlivý.

Dalším problémem je velké množství identických zpráv od jednoho zdroje. Tento problém je řešen v Kapitole 3.

Po přijmutí bezpečnostní události existuje několik možností, které mohou nastat:

- Zaslání zprávy o bezpečnostní události odpovídající osobě. Zpráva může být odeslána například emailem nebo do příslušného systému, který eviduje bezpečnostní incidenty. Bezpečnostní expert poté vyhodnotí závažnost situace a provede odpovídající opatření.
- Plně automatizované zakročení systému. Systém nastaví ostatní zařízení do stavu, které zabraní bezpečnostní události napáchat škody v počítačové infrastruktuře. Dále systém informuje odpovědnou osobu.
- Odchycení škodlivého provozu a zahájení plně automatizované analýzy odchyceného vzorku dat.

Plně automatizovaná analýza vzorku dat ze škodlivého provozu se zabývají systémy, které jsou tématem této práce. Po zachycení škodlivého vzorku dat je tento vzorek podroben automatické analýze. Toto automatické vyhodnocení dat není běžné a značně tím ulehčuje práci bezpečnostním specialistům.

1.2 Reprezentace bezpečnostní události

Existuje mnoho různých formátů pro reprezentaci bezpečnostních incidentů. Jednotlivé reprezentace se od sebe liší zejména popisem bezpečnostního incidentu. Některé popisují celý průběh útoku, kde v jedné bezpečnostní události se vyskytuje několik bezpečnostních incidentů, tak jak probíhaly v čase. Jiné popisují pouze jeden bezpečnostní incident.

Dále se liší datovým formátem, kdy nejčastěji se používá formát XML nebo formát JSON.

Některé reprezentace mají pevně danou strukturu, která určuje např. maximální hloubku zanoření položek. Jiné toto omezení nemají a různé implementace tak mohou mít velice odlišnou strukturu reprezentace bezpečnostní události.

1.2.1 IDEA

IDEA [3] neboli Intrusion Detection Extensible Alert je komunikační formát využívaný sdružením CESNET. IDEA bere v úvahu výhody a nevýhody odstatních formátů a pokouší se o maximální zjednodušení při práci s nimi – snaží se najít rovnováhu mezi komplexitou a použitelností. Tento formát byl primárně navržen pro hlášení bezpečnostních událostí do velkých bezpečnostních systémů pro sdílenou informaci. V síti CESNET2 je pro tyté účely používán systém WARDEN [4].

Ukázka formátu IDEA zprávy ve formátu JSON:

```
{
   "Type" : [ "Flow",  "Statistical"   ],
   "PacketCount" : 799,
   "Category" : [ "Availability.DDoS"  ],
   "Format" : "IDEA0",
   "ByteCount" : 926353,
   "ID" : "9df9bb33-c2c1-4312-95f3-e5827c944584",
   "Target" : [
      {
         "IP4" : [    "195.113.241.34"  ],
         "InByteCount" : 926353,
         "Proto" : [  "udp", "dns"],
         "InPacketCount" : 799,
         "InFlowCount" : 416
      }
   ],
   "DetectTime" : "2016-11-10 19:29:28Z",
   "Description" : "DNS amplification",
   "Source" : [
      {
         "IP4" : ["63.243.194.2"],
         "InByteCount" : 28872,
         "Proto" : [ "udp", "dns"],
         "Type" : [ "Backscatter"],
      }
   ],
   "Node" : [
      {
         "Name" : "cz.cesnet.nemea.
            amplificationdetector",
         "SW" : [    "Nemea",      "
            amplification_detection"    ]
      }
   ],
   "FlowCount" : 416
}
```

IDEA používá datový formát JSON. Ostatní formáty (popsané dále v této kapitole) používají datový formát XML a rozdíl mezi těmito formáty je znatelný. Rychlost při zpracování souboru, velikost přenášeného souboru i čitelnost pro člověka hraje ve prospěch JSON formátu.

IDEA formát je navržen primárně pro sdílení bezpečnostních událostí do velkých center pro sdílenou informaci. IDEA bere to nejlepší ze stávajících

formátů a přizpůsobuje to svým potřebám.

IDEA formát je vhodný pro vytváření nových programů, které s ním pracují. Přesně definovaná struktura IDEA dokumentu zajišťuje bezproblémové zpracování ve všech systémech. Dokument je vždy tvořen klíčem a hodnotou, kde hodnota může mít jen datové typy, které jsou definované v JSON a to jsou datové typy: řetězec, číslo, slovník, pole, boolean a NULL. Existují knihovny pro práci s JSON formátem pro mnoho různých programovacích jazyků. To jsou hlavní důvody, proč je práce s IDEA formátem snadná.

IDEA definuje přesně dané hodnoty klíčů ve specifikaci. Tyto klíče lze rozšířit. Pokud se strana, která odesílá IDEA zprávy, dohodne se stranou, která přijímá tyto zprávy, na jasně dané struktuře dokumentu, může být IDEA formát bez problémů rozšířen o nově definované položky. Jen je nutné použít unikátní jméno pro rozšiřující klíč nově zvolené položky ve struktuře dokumentu.

IDEA formát má několik povinných položek. Tyto položky je nutné v každé zprávě uvést, a jsou to: Format, ID, DetectTime, Category. Format určuje identifikátor verze IDEA. ID je globální identifikátor zprávy, doporučuje se používat UUID verze 5. DetectTime je čas detekce události. Category je druh bezpečnostní události, která nastala. Položka Category je klíčová pro navrhovaný algoritmus v Kapitole 3, proto tu jsou rozebrány nejčastěji se vyskytující kategorie a jejich subkategorie, celý seznam je dostupný v dokumentaci [5] k IDEA formátu. Každá kategorie obsahuje několik subkategorií, které specifikují bezpečnostní událost podrobněji.

- Recon.Scanning – Útoky, které mají za úkol objevit slabé místo v cílovém systému. Nejčastěji se jedná o proces shromažďování dat o počítači - např. Port scanning.
- Attempt.Exploit – Pokus o kompromitování systém, či narušení jakékoli běžící služby uvnitř systému. Jedná se o využití zranitelnosti se standardizovaným identifikátorem CVE - např. buffer overflow, či cross site scripting.
- Attempt.Login – Slovníkové útoky na službu, kde se útočník snaží uhádnout heslo pomocí předpřipravených slovníků.
- Anomaly.Traffic – Anomální chování sledovaného provozu. Nemusí se jednat o útok, ale je vhodné prošetřit nejasné chování subjektu uvnitř sítě.
- Abusive.Spam – Nevyžádaná pošta. Příjemce dostal elektronickou poštu o kterou nežádal.
- Intrusion.Botnet – Počítač je součástí botnetu. Botnet je uskupení počítačů, které byly napadeny a ovládnuty. Útočník má kontrolu nad napadeným počítačem a může ho používat například pro DDoS útok.

- Availability.DoS – Útok, který se snaží o vyřazení počítačové služby. Nejznámější DoS útoky jsou SYN Flood, ICMP flood, či DNS amplifikační útok.

- Availability.DDoS – DDoS funguje na stejném principu jako DoS útok. Jediným rozdílem je použití velkého množství počítačů pro DOS útok.

1.2.2 IDMEF

IDMEF [6] je zkratka pro The Intrusion Detection Message Exchange Format a jedná se velice univerzální komunikační formát používaný např. v nástroji SNORT. Datová reprezentace je XML dokument.

Formát není příliš vhodný pro vývoj nových aplikací, protože jeho struktura je příliš složitá. Datová struktura může být neomezeně hluboká a pro popis jednotlivých atributů používá složité identifikátory, např. pro popis IP adresy je to „Alert.Source.Node.Address.address“, dále mezi jeho hlavní nevýhody patří globální svázání dvojce klíč/hodnota ke zprávě, nikoli ke specifické zdrojové či cílové adrese.

IDMEF je silný formát, který dokáže popsat nejširší spektrum všech bezpečnostních incidentů do detailu. Bohužel tato komplexnost datového formátu má za následek nepřiměřenou složitost. IDMEF dokument často obsahuje několika úrovňovou rekurzi XML atributů. Formát se snaží popsat vše, a proto se musejí psát komplexní knihovny pro vytváření a následný načtení těchto zpráv. Proto tento formát není příliš vhodný pro nasazení na páteřní sondy, kde se snažíme o co největší efektivitu.

1.2.3 IODEF

IODEF [7] je zkratka pro The Incident Object Description Exchange Format a rozšiřuje IDMEF formát. IODEF vzniknul pro potřeby CSIRT týmů, který tento formát používají pro přeposílání bezpečnostních incidentu do svých systému. IODEF slučuje několik bezpečnostních událostí na síti do jedné zprávy. Díky této funkcionalitě popisuje IODEF celý průběh bezpečnostního incidentu v čase.

Při vytváření incidentu je velká volnost při sestavování dokumentu a to s sebou nese riziko u přijímače, který nemusí sestavený dokument správně analyzovat.

1.2.4 STIX

STIX [8] je zkratka pro The Structured Threat Information eXpression jazyk a je poměrně nový, vznikl v roce 2012 a pro výměnu dat používá XML dokumenty. Motivací pro vznik STIX formátu je neustále se zvětšující počet kybernetických útoku a jejich nutná evidence. STIX byl vytvořen ve spolupráci s experty z různých odvětví jako jsou vlády, univerzity, či bezpečnostní

experti a snaží se definovat nový standard pro kybernetické hrozby. Jazyk se snaží být plně expresivní, flexibilní, rozšiřitelný s možností automatizace, a zároveň čitelností pro člověka.

Hlavní výhodou je, že téměř vše je volitelné a má pevně danou strukturu. Uživatelské skupiny (komunity) se předem mohou dohodnout na pevně daných strukturách STIXu a mají garantovanou bezproblémovou výměnu bezpečnostních incidentů mezi sebou.

Kapitola 2

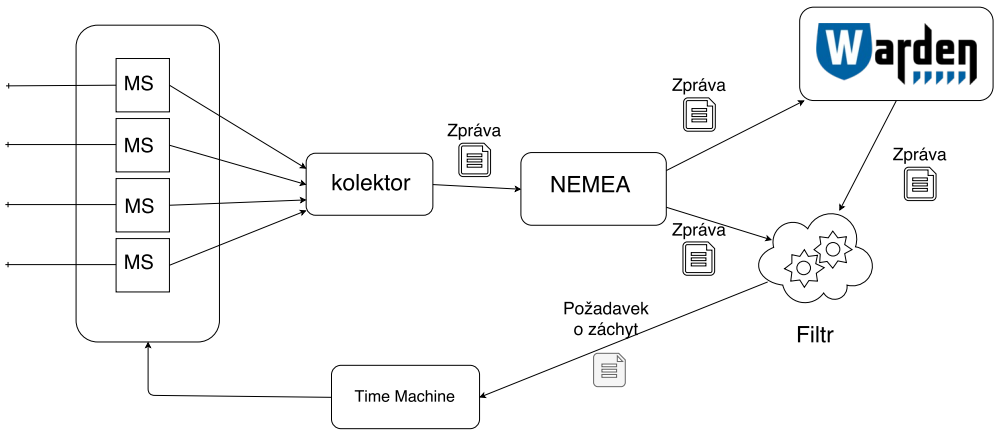
Analýza problematiky

Práce bude nasazena v národní akademické síti CESNET2 [9]. CESNET2 je vysokorychlostní počítačová síť, propojující zejména univerzitní města v České republice. Tato síť má spojení se zahraničím, kde se přenosové kapacity sítě pohybují v řádek 100 Gb/s. Monitorovací sondy se nacházejí na hraničních bodech této sítě. V této kapitole jsou popsány všechny prvky infrastruktury sítě CESNET2. Dále je provedena analýza bezpečnostních událostí, které slouží pro vstup navrhovaného systému.

2.1 Aktuální řešení

2.1.1 Monitorovací sondy

Monitorovací sonda (MS) je zařízení umístěné uvnitř sítě. Menší monitorovací sondy se nacházejí na vnitřním okruhu sítě. Na hraničních linkách se poté nacházejí monitorovací sondy s propustností 100 Gb/s.



Obrázek 2.1: Infrastruktura sítě CESNET2

Monitorovací sonda přijímá data ze sítě, rozděluje je na pakety a následně je agreguje do síťových toků [10]. Výstupem z monitorovacích sond jsou síťové toky ve formátu IPFIX [11] nebo NetFlow [12].

2.1.2 IPFIXcol

IPFIXcol [13] je zkratka pro IPFIX collector. Jedná se o systém pro práci se síťovými toky. Aplikace je rozšířitelná a podporuje mnoho zásuvných modulů, které se starají o rozšíření IPFIX formátu nebo převod do jiného formátu.

V současné infrastruktuře se IPFIXcol používá pro ukládání síťových toků a jejich následné převedení do UniRec formátu. Tento formát se používá v systému NEMEA.

2.1.3 Systém NEMEA

Systém NEMEA je systém, který bude posílat bezpečnostní události do systému pro automatický výběr detekovaných událostí popsaný v Kapitole 3.

NEMEA [14] [15] je zkratka pro Network Measurement Analysis. Tento systém je určen pro proudové zpracování síťových toků a detekci anomálií v reálném čase. Vstupem mohou být soubory (ve formátech CSV [16] nebo nfdump [17]), síťové rozhraní, nebo síťové toky (ve formátu UniRec [18])

UniRec je datový formát pro ukládání a přenášení malých dat. Hlavní výhodami tohoto formátu jsou: rychlost, efektivita a dostatečná flexibilita.

Moduly v systému se spouštějí jako nezávislé procesy. Každý modul může mít více rozhraní, vstupní/výstupní, a každé rozhraní je pouze jednosměrné. Jednotlivé moduly komunikují mezi sebou pomocí UniRec zpráv. Systém NEMEA je plně modulární a každý uživatel tohoto systému si může vytvořit vlastní modul nebo použít předpřipravené moduly systému.

NEMEA obsahuje řadu předpřipravených modulů (popsány níže). Detekční moduly se starají o detekci škodlivého provozu ze síťových toků. Reportovací moduly navazují na detekční moduly a převádějí detekované události do pevně daných struktur, např. do IDEA formátu.

Níže jsou popsány některé typické moduly, které na NEMEA systému běží a jsou volně přístupné veřejnosti:

DNS Tunnel Detektor

Tento detektor je založen na anomálním chování DNS protokolu. Pomocí DNS protokolu lze obejít bezpečnostní politiky. Za normálních okolností by veškerý síťový provoz od neověřeného zdroje v síti byl zahazován, toto chování je běžné u veřejných hotspotů nebo u firemních firewallů. Toho lze obejít skrze DNS protokol. Veškerá data od neověřeného zdroje jsou zabalena do DNS protokolu.

Modul detekce je založen na prahových hodnotách pro DNS dotazy, které řádově překračují hodnoty běžného DNS provozu.

Black List Filter

Detektor používá veřejně dostupné seznamy podezřelých IP adres. Tyto seznamy si pravidelně aktualizuje a pokud se v síťovém toku objeví nějaká IP adresa z těchto seznamů, tak detektor vygeneruje hlášení.

HostStats

Detektor HostStats je modul, který sleduje statistické vlastnosti jednotlivých IP adres a pokouší se odhalit jejich podezřelé chování.

Statistiku pro určitou adresu periodicky porovnává se sadou předem připravených pravidel a v případě porušení je tato událost nahlášena. V současné době HostStats detektor obsahuje pravidla pro následující útoky: skenování sítě, DoS útoky, prolamování SSH hrubou silou a DNS amplifikační útoky.

Booter Detektor

Booter [19] je webová služba, která se většinou prezentuje jako stress test. Ve skutečnosti se jedná spíše o placený DDoS útok.

Booter server není zapojen do DDoS útoku, proto tyto služby není lehké odhalit. Na webovém serveru pouze potencionální útočník zadá na koho chce provést DDOS útok a následný DDoS již proběhne nezávisle na Booter serveru.

Booter detektor funguje na podobném principu jako Black List Filter detektor. Detektor si stahuje aktuální seznam booterů z veřejně dostupných seznamů na internetu a následně je porovnává s IP adresami ze síťových toků. Pokud je nalezena shoda je vytvořeno hlášení o útoku.

Horizontal Address Scan Detektor

Horizontální skenování sítě je založeno na procházení velkého rozsahu IP adres pro jeden port. Může se například jednat o procházení celého subnetu IP adres, kde se útočník snaží najít otevřený port 23 pro službu telnet.

Vertical Port Sken Detektor

Vertikální skenování sítě je založené na podobném principu jako je horizontální skenování sítě. Vertikální skenování sítě se ovšem zaměřuje na jednu IP adresu a skenuje celý rozsah jeho portů. Algoritmus je založen na podobném principu jako algoritmus u horizontálního skenování.

VoIP fraud Detektor

VoIP [20] je zkratka pro Voice over IP. Tato technologie umožňuje přenos digitálního hlasu, či videohovoru skrze IP protokol.

VoIP fraud útok hádá prefix PSTN [21] brány. Při uhádnutí tohoto prefixu je hovor spojen v opačném případě je hovor odmítnut. Hlavní motivací útočníka může být využití telefoní služby zadarmo, například hovory do zahraničí nebo způsobení finanční škody majiteli PSTN brány. Útok je podobný skenování sítě. Pokud se útok provádí pomalu, je velice těžké tento útok odhalit.

Detektor na VoIP fraud používá rozšířený síťový tok o položku SIP protokolu a samotný algoritmus detekce je popsaný v článku [22].

Bruteforce Detector

Bruteforce detector je zaměřený na slovníkové útoky proti službám jako jsou SSH [23] nebo TELNET [24]. Útočník se snaží prolomit internetové protokoly pomocí předdefinovaných slovníků. Spoléhá na slabá hesla uživatelů. Při použití slovníků se šance na uhádnutí hesel zvyšuje.

2.1.4 Původní verze filtrování bezpečnostních událostí

Všechny detekované události z jednotlivých detektorů jsou přeposlané do filtru bezpečnostních událostí.

Filtr událostí je program, který redukuje počet bezpečnostních událostí na vstupu pomocí vnitřního algoritmu na omezený počet událostí na výstupu.

Původní řešení používá nedokonalý algoritmus. Každých deset minut se vybere nejčastěji nahlášená událost. V případě více událostí se stejným počtem nahlášení je zvolena novější událost. Tato událost je poté odeslána pro podrobenější anlýzu příslušné osobě, či do systému tomu určenému. Tato událost nemůže být algoritmem znova vybrána následující hodinu.

2.1.5 Automatický záchyt důkazního materiálu

Pro vyfiltrované události je možné získat podrobnější data pomocí systému Time Machine.

Systém je určen pro zachytávání dat. Při požadavku o záchyt dat se tato data dělí na historická data a živá data. Historická data jsou data, která byla zachycena před požadavkem o záchyt. Živá data jsou data, která byla zachycena po požadavku o záchyt.

Z důvodů nasazení na vysokorychlostní sítě po kterých proudí desítky, či stovky Gb/s, se historická data ukládají pouze krátkodobě, a to jen začátky toků, konkrétně n paketů toku do kruhového bufferu. Po naplnění kruhového bufferu se vždy přepisují nejstarší data.

Výstupem stroje času jsou datové soubory. Každý soubor je pro jednu IP adresu a obsahuje data pro jednu bezpečnostní událost.

Požadavek o záchyt

Pokud chceme vytvořit požadavek o záchyt dat pro systém Time Machine je nutné použít jeho komunikační protokol. Pomocí tohoto protokolu se nadefinují vstupní parametry pro požadavek o záchyt a vytvoří pravidlo při filtrování provozu. Vstupní parametry vypadají následovně:

- IP adresa – hlavní klíč při filtrování
- Směr – směr zachycované IP adresy. Zdrojová, cílová nebo zachycení obousměrně
- Počet paketů – Počet zachycených paketů
- Čas záchytu – Čas, po který se mají data zachytávat

Pro účel filtrování dat si Time Machine vytváří tabulku pravidel, která umožňuje záchyt dat k několika událostem současně. Každý řádek (každé pravidlo) je indexován pomocí směru provozu a IP adresy. Jednotlivá pravidla si udržují:

- Počet zachycených paketů
- Limit počtu paketů
- Čas začátku toku
- Časový limit
- Název souboru

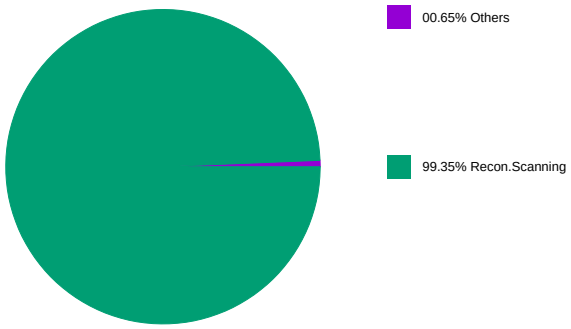
Pokud pravidlo překročí časový limit nebo paketový limit, je záchyt ukončen a zachycený datový provoz je uložen do souboru. Stroj času ukládá datový provoz v paketovém formátu pcap používaný knihovnou libpcap [25].

2.2 Analýza detekovaných bezpečnostních událostí

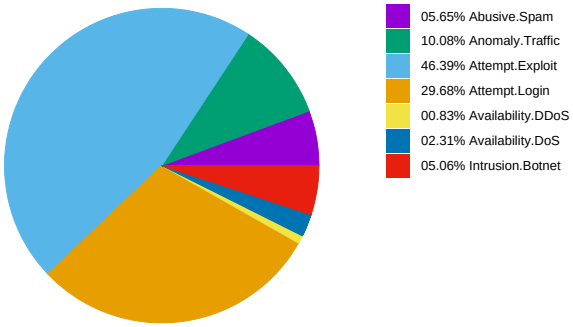
Pro analýzu bezpečnostních událostí ze sítě CESNET2 byly použity bezpečnostní události ze systému WARDEN. Pro tuto analýzu byla použita data z měsíců září a říjen roku 2016 a ledna roku 2017. Každý měsíc je zasláno do systému WARDEN okolo 70 000 000 bezpečnostních událostí.

Pro srovnání systém NEMEA vygeneruje za měsíc zhruba 4 000 000 událostí. Veškeré události ze systému NEMEA jsou přeposílána do systému WARDEN.

Pro následující grafy byla použita data ze všech tří měsíců a tato data byla sloučena do jednoho statistické celku.



Obrázek 2.2: Rozdělení bezpečnostních událostí v síti CESNET2



Obrázek 2.3: Rozdělení bezpečnostních událostí v síti CESNET2 bez kategorie “Recon.Scanning”

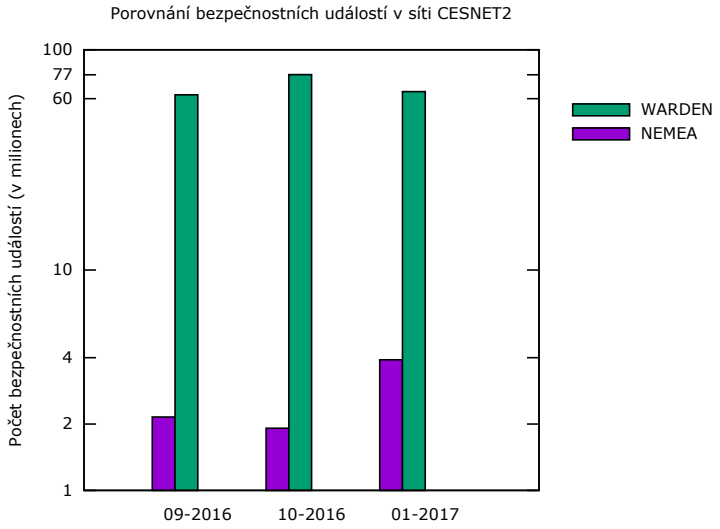
Na obrázku 2.2 je zastoupení jednotlivých kategorií. Kategorie „Recon.Scanning“ má dominantní zastoupení a pro přehlednější graf byly všechny ostatní kategorie sloučeny do kategorie „Others“.

Na obrázku 2.3 je zobrazené procentuální zastoupení všech kategorií z kategorie „Others“ z obrázku 2.2.

Na obrázku 2.4 jsou porovnány počty bezpečnostních události v systému WARDEN a v systému NEMEA. Narůst počtu u systému WARDEN je zapříčiněný přidáním nového detektoru. Tento narůstající trend počtu bezpečnostních událostí bude v budoucnosti pokračovat, protože budou přibývat nové detektory a také bezpečnostní incidenty uvnitř sítě.

2.2.1 WARDEN

WARDEN [4], je systém pro sdílení informací o detekovaných bezpečnostních událostech. Systém vznikl primárně pro potřeby dvou akreditovaných CSIRT týmů, a to CESNET-CERTS [26] a CSIRT-MU [27].



Obrázek 2.4: Porovnání počtu bezpečnostních událostí v systémeu WARDEN a NEMEA'

V současné době je projekt WARDEN zaměřen převážně pro potřeby národního výzkumu a vzdělávání pro jeho členy a další zapojené instituce.

Bezpečnostní týmy mohou s incidenty na svojí síti naložit několika způsoby. Buď je mohou zaslat správcům, které spravují příslušnou síť, pod kterou bezpečnostní incident patří. Další možností je úplné ignorování bezpečnostního incidentu nebo zaslání zprávy o incidentu na sběrné místo, kde se budou tyto incidenty uchovávat. Jelikož bezpečnostní incidenty jsou velice citlivá data, je nutné posílat data pouze do systému, který je dostatečně zabezpečený. Systém WARDEN splňuje tyto požadavky.

Účastníci zasílají bezpečnostní události, detekované v jejich sítí, do systému WARDEN a mohou si stahovat veškeré informace o jejich síti z toho systému.

Kapitola 3

Návrh systému pro filtrování bezpečnostních událostí

Tato kapitola popisuje návrh systému pro filtrování bezpečnostních událostí. Systém bude pojmenován „Filtr bezpečnostních událostí“.

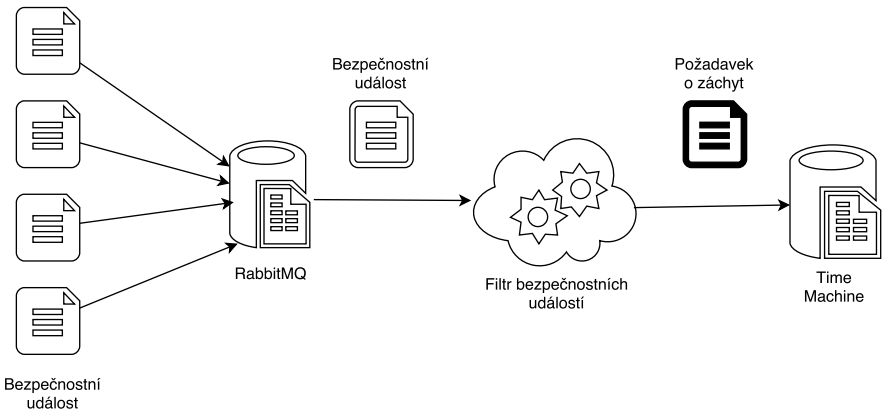
Cílem navrhovaného systému je redukovat množství bezpečnostních událostí na vstupu na přijatelné množství událostí na výstupu. Přijatelné množství událostí se liší od velikosti sítě až po prostředky vyhrazené pro tyto účely. Každá síť je unikátní, a proto je nutné, aby tento systém byl konfigurovatelný pro konkrétní potřeby dané sítě.

Analýzou bezpečnostních událostí ze systémů WARDEN a NEMEA a po konzultaci s bezpečnostními odborníky ze sítě CESNET2 byly stanoveny následující požadavky pro Filtr bezpečnostních událostí:

- Redukce počtu bezpečnostních incidentů na únosné množství.
- Upřednostňování méně častých bezpečnostních událostí.
- Různé formáty bezpečnostních událostí na vstupu.
- Snadná rozšiřitelnost na různé reprezentace bezpečnostních událostí.
- Ke každé kategorii bude přiřazena cena, která bude mít vliv na výsledné skóre.
- Plná kontrola nad parametry záchytu pomocí konfiguračních souborů.

3.1 Popis systému pro filtrování bezpečnostních událostí

Systém pro filtrování bezpečnostních událostí, znázorněn na obrázku 3.1, spolupracuje s dalšími dvěma aplikacemi: RabbitMQ a Time Machine, které jsou popsané dále.



Obrázek 3.1: Architektura systému pro filtrování bezpečnostních událostí

Na vstupu jsou bezpečnostní události, které jsou posílány do systému RabbitMQ (Podsekce 3.1.1). RabbitMQ se stará o přeposílání bezpečnostních událostí do filtru. Filtr událostí příjme zprávu, uloží si ji do vnitřního uložiště, vyhodnotí její důležitost, a pokud je skóre události vysoké, pošle požadavek o záchyt do systému Time Machine.

3.1.1 RabbitMQ

RabbitMQ [28] je message broker (program, který zprostředkovává přeposílání zpráv mezi aplikacemi). Tato aplikace funguje jako klient-server. Všechny požadavky obstarává server.

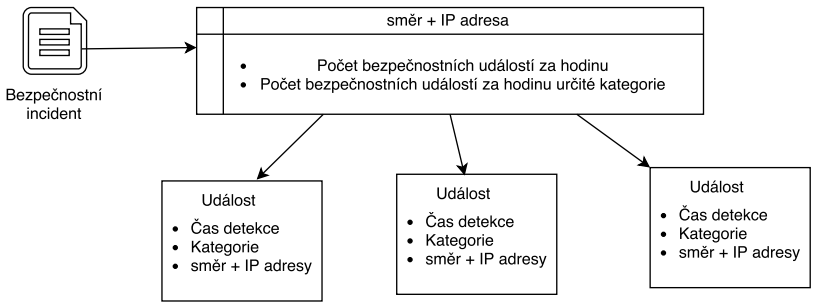
Hlavní funkcí RabbitMQ je předávání bezpečnostních událostí do filtru bezpečnostních událostí. RabbitMQ byl zvolen pro svojí snadnou obsluhu a bezpečné přeposílání zpráv skrze systémy.

Systém nabízí knihovny pro různé programovací jazyky, různé druhy výměny zpráv a je dobře zdokumentovaný. Proto je velice snadné rozšířit stávající sestavu o další detekční systémy, nebo napojit systém pro sdílené informace o detekovaných událostech. Jelikož je nutné dbát na bezpečnost při předávání bezpečnostních událostí mezi jednotlivými systémy, je veškerý provoz šifrován protokolem TLS, který je součástí systému RabbitMQ.

3.1.2 Bezpečnostní událost

Systém je primárně navržen pro bezpečnostní události ve formátu IDEA, který je používaný uvnitř sítě CESNET2. Celý systém pro filtrování bezpečnostních události je navržen primárně pro práci s datovými soubory ve formátu JSON.

Přestože je primárně určen pro formát IDEA, tak lze do systému posílat i jiné formáty. Pro systém filtrování bezpečnostních událostí je navržen speciální překladač, který transformuje jednotlivé formáty bezpečnostních incidentů na vstupu do vnitřní struktury pro filtrování událostí. Nevýhoda této



Obrázek 3.2: Struktura hašovací tabulky pro práci s bezpečnostními událostmi

transformace je, že je vyžadován JSON formát na vstupu, protože pro tento účel je použit nástroj jq [29], který umí pracovat pouze s JSON formátem. To ovšem neznamená, že systém nelze rozšířit na jiný formát dokumentu, např. XML.

Pro podporu ostatních formátu dokumentu je nejjednodušší převést dokument do JSON formátu a poté již je zaručená bezproblémová funkčnost.

Jq je nástroj, pomocí kterého lze snadno filtrovat, či upravovat datový soubor.

3.2 Filtr bezpečnostních událostí

Systém pro filtrování bezpečnostních událostí se stará o redukci bezpečnostních události na únosné množství podle potřeb bezpečnostních týmů.

Na vstupu tento systém má bezpečnostní událost. Bezpečnostní událost je nejdřív upravena do příslušné datové struktury, se kterou celý systém následně pracuje.

Následně proběhne uložení do vnitřního úložiště. Poté jsou načteny všechny potřebné informace k bezpečnostní události z konfiguračních souborů a četnosti jednotlivých kategorií z vnitřních struktur, potřebných pro vyhodnocovací algoritmus aplikace. Následně vypočte algoritmus výsledné skóre události.

V závislosti na nastavení navrhovaného systému se skóre vyhodnotí jako dostatečné nebo nedostatečné. Nedostatečné skóre znamená, že se bezpečnostní událost nebude nadále zpracovávat. Dostatečné skóre znamená, že událostí bude přeposlána na podrobnější analýzu odpovídající osobě, či systému.

3.2.1 Datová struktura pro uložení bezpečnostní události

Datová struktura je zobrazena na obrázku 3.2. Jedná se o hašovací tabulku, která používá jako klíč dvojici „směr + IP adresa“. Směr může nabývat dvou hodnot: „S“ nebo „T“. „S“ znamená, že se jedná o zdrojovou IP adresu (Source), a „T“ o cílovou IP adresu (Target). Hašovací tabulka uchovává udá-

losti pouze za poslední hodinu. Pokud je bezpečnostní událost starší více než jednu hodinu, tak je z tabulky odstraněna.

Každá bezpečnostní událost použije všechny svoje zdrojové a cílové adresy, které použije jako klíč do hašovací tabulky. Pokud již hašovací tabulka obsahuje tuto dvojici „směr + IP adresa“, tak se zvýší čítač počtu bezpečnostních událostí za hodinu a čítač příslušné kategorie. Dále se přidá upravená bezpečnostní událost na konec pole všech bezpečnostních událostí příslušného klíče v hašovací tabulce.

Upravená bezpečnostní událost obsahuje ze všech políček, které obsahuje, pouze čas detekce, kategorie a „směr + IP adresa“, kde „směr + IP adresa“ má opačný směr oproti hašovacímu klíči. Pokud je hašovací klíč zdrojová IP adresa, tak v poli bezpečnostních událostí příslušného klíče bude vždy cílová IP adresa.

Ukládání všech IP adres a všech směru je nutné, protože chceme mít možnost zachytávat veškerý provoz jak od zdroje útoku, tak od cíle útoku.

3.2.2 Relativní četnost výskytu kategorie

Relativní četnost kategorie bezpečnostní události je další důležitá část pro algoritmus ohodnocování bezpečnostní události.

Každá bezpečnostní událost obsahuje alespoň jednu kategorii. Tato kategorie je uložena do hašovací tabulky četnosti. Údaje jsou uloženy jako absolutní četnost a až při dotázaní, jaké je zastoupení jednotlivé kategorie, je vypočítána relativní četnost kategorie.

Hašovací tabulka si uchovává údaje za celou dobu běhu a je perzistentní.

3.2.3 Algoritmus

V následující sekci bude popsán algoritmus pro ohodnocování bezpečnostních událostí.

Cílem navrhovaného algoritmu je přiřadit každé události skóre. Podle skóre lze redukovat bezpečnostní události na únosné množství. Únosné množství zachytávaných bezpečnostních událostí se liší od velikosti bezpečnostního týmu. Algoritmus je navržen tak, aby množství bezpečnostních událostí bylo závislé na konfiguraci navrhovaného systému.

V konfiguraci tohoto algoritmu lze ovlivnit cenu pro jednotlivé kategorie a parametry pro navrhovaný algoritmus popsaný níže.

Algoritmus je rozdělen na dvě části. První část přiřazuje skóre skenům. Jak plyne z obrázku 2.2, kategorie „Recon.Scanning“ je dominantní kategorií na síti CESNET2, a proto je řešena samostatně (první část algoritmu (3.2)). Všechny ostatní kategorie jsou ohodnocovány podle druhé části algoritmu (3.3).

Při vstupu bezpečnostní události do systému jsou ke každé položce přiřazeny tyto údaje:

- Category – Název kategorie příslušné bezpečnostní události. Např. „Availability.DDoS“

- Price – Cena kategorie pro příslušný bezpečnostní incident. Doporučuje se použít rozsah $\langle 0, 10\rangle$.
- CntHour – Počet výskytů dvojice kategorie a IP adresa za poslední hodinu
- Probability – Relativní četnost výskytu kategorie bezpečnostní události. Rozsah $(0, 1\rangle$.

Celý ohodnocující algoritmus je znázorněn ve výpisu Algoritmus 1. Výsledné skóre může nabývat všech reálných čísel. Větší skóre znamená větší závažnost bezpečnostní události. Události ohodnoceny skóre < 1 jsou považovány za nedůležité.

if *Category == "Scan" and*
((CntHour + p) mod modul == 0 or RandomPick(n)) **then**
  score = CatScore ;
else if *Category != "Recon.Scanning"* **then**
  $score = \text{sgn}(-CntHour + 4) * (2^{Price-1}(1 - Probability) + 2^{Price-1})$
else
  score = 0 ;
end

Algoritmus 1: Ohodnocující algoritmus bezpečnostní události

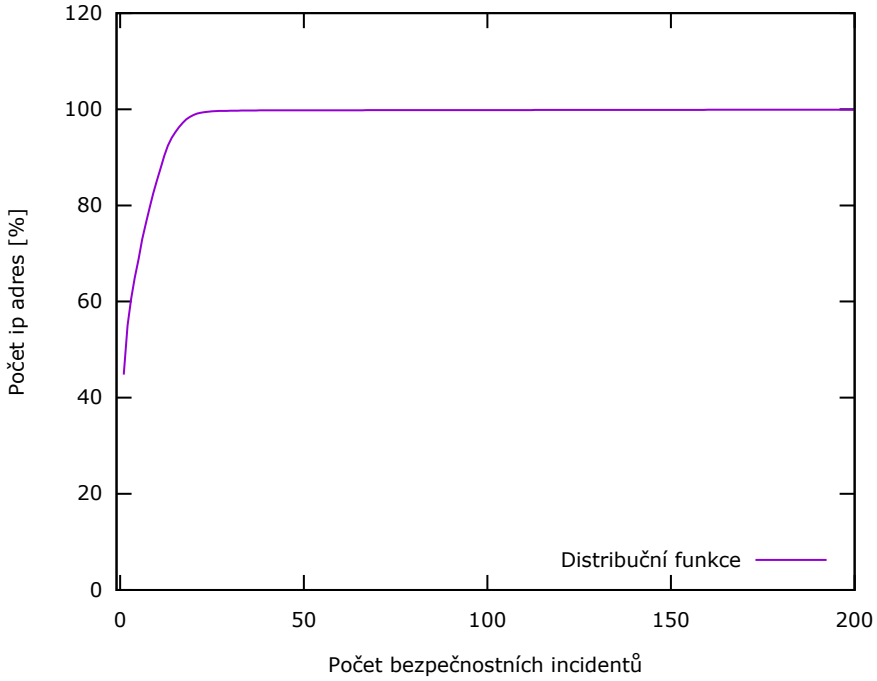
3.2.4 Ohodnocovací algoritmus skenů

První část algoritmu (3.1) se stará pouze o skeny a je rozdělena na dvě části. První část je pro opakující se skeny (3.2) a druhá část je pro skeny, které se vyskytly v systému právě jednou.

$$(CntHour + p)\ mod\ modul\ ==\ 0\ or\ RandomPick(n) \tag{3.1}$$

Na obrázku 3.3 je distribuční funkce, která říká, kolik IP adres průměrně obsahuje jednotlivý počet bezpečnostních incidentů za hodinu. V tomto případě se jedná pouze o skeny. Okolo 50 % IP adres detektory nahlásí bezpečnostní událost pouze jednou. Toto jsou skeny, které se neopakují, a jejich výběr je prováděn náhodně – Algoritmus 2, kde parametr n udává, jak často se vybere bezpečnostní událost náhodně. Pro $n == 250$ se událost vybere náhodně v průměru jednou za 250 skenů.

Z grafů na obrázcích 3.4 a 3.5 plyne algoritmus pro opakující se skenování sítě, který má tvar (3.2). Tato rovnice má dva parametry: p a *modul*, které ovlivňují počet zachycených událostí.



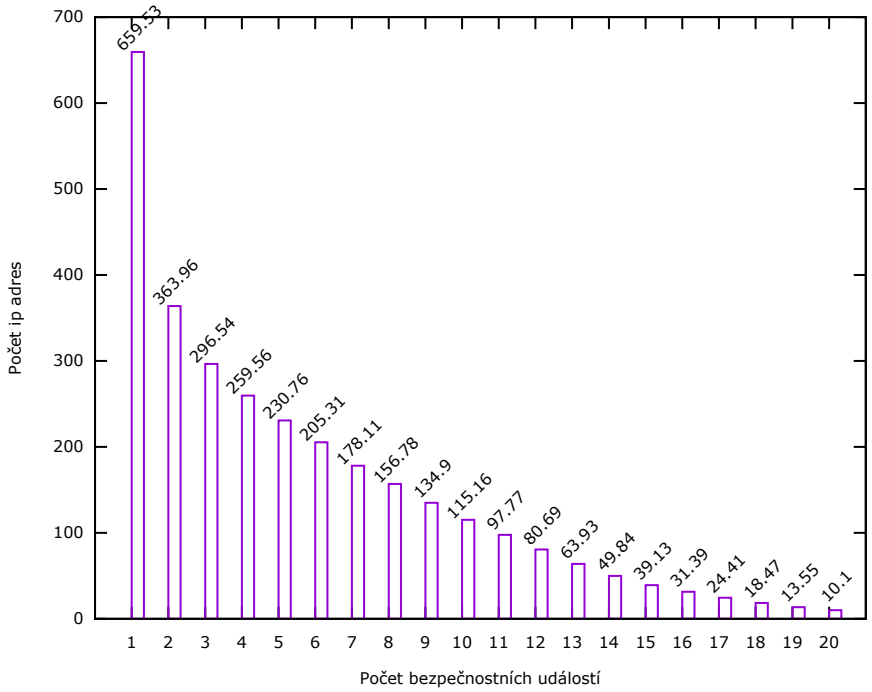
Obrázek 3.3: Distribuční funkce pro kategorii sken

function RandomPick(n)
  return randomInt(n) == 0
Algoritmus 2: Randomizovaný výběr bezpečnostní události

Rovnice (3.2) vyjadřuje, jaké IP adresy se mají zachytávat v závislosti na jejich výskytu v hodinové statistice, pokud zvolíme parametry $p = 95$ a $modul = 100$. Poté pokud se v hodinové statistice pro jednotlivou IP adresu s kategorií skan vyskytne stejný útok popáte, stopáté, dvěstěpáte za hodinu, tak dostane kladné ohodnocení. Tyto parametry vyhovují síti NEMEA, ve které bude nasazená. V průběhu času se může počet bezpečnostních incidentů měnit, a proto jsou tyto parametry konfigurovatelné.

$$CntHour + p \ (mod \ modul) \tag{3.2}$$

První část algoritmu (3.2) pro výběr skenů se zaměruje pouze na opakující se skeny. Opakující se skeny budou s větší pravděpodobností opravdu skeny. Pokud detektor nahlásí sken pouze jednou pro konkrétní IP adresu, je menší pravděpodobnost, že se nejedná o sken, než když detekční algoritmus označí desetkrát tu samou IP adresu za hodinu. Skenování sítě většinou trvá déle než několik vteřin, a z tohoto předpokladu detektor zahlásí více než jednu



Obrázek 3.4: Průmerný počet kategorie sken za hodinu (NEMEA)

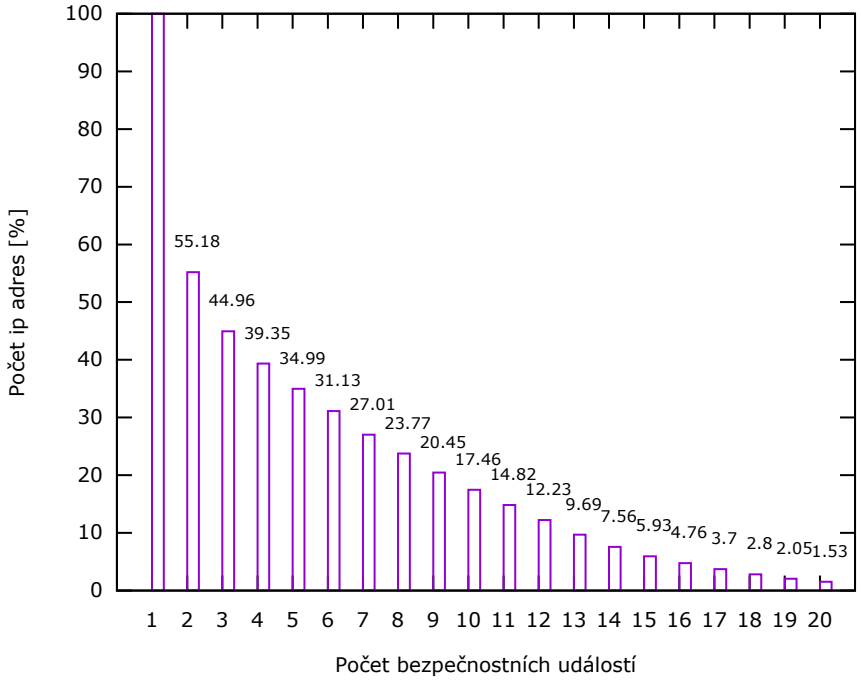
bezpečnostní událost tohoto druhu, kterou ohodnotíme. Na druhou stranu ale chceme zachytávat i skeny, které nemají dlouhého trvání, a ty vybíráme randomizovaně. To jsou hlavní důvody, proč je algoritmus pro skeny rozdělen na dvě části.

3.2.5 Ohodnocovací algoritmus ostatních kategorií

Druhá část algoritmu (3.3) a její značná část vychází z grafu 3.6. Graf je distribuční funkce a znázorňuje průměrný počet bezpečnostních událostí (bez skenů) a jejich závislost na počtu IP adres. Detektory detekují 98% všech bezpečnostních incidentů pro konkrétní IP adresu v hodinové statistice méně než třikrát.

$$\text{sgn}(-CntHour + 4) * (2^{Price-1}(1 - Probability) + 2^{Price-1}) \tag{3.3}$$

Průměrné statisitky ze systému NEMEA jsou znázorněny na obrázcích 3.8 a 3.7. Průměrně systém NEMEA detekuje okolo 40 unikátních událostí za hodinu. Graf na obrázku 3.8 znázorňuje opakující se bezpečnostní události. Pouze 13 % těchto událostí se opakuje. 87 % událostí se neopakuje. Obrázek 3.7 znázorňuje konkrétní počty za hodinu.



Obrázek 3.5: Průměrný počet kategorie sken v procentech za hodinu (NEMEA)

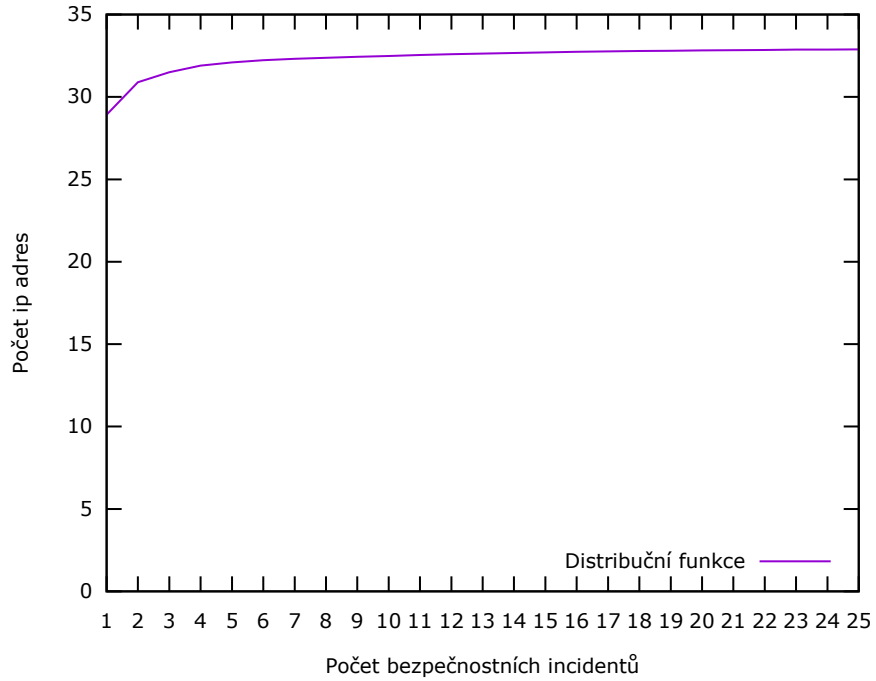
Po detailnějším prozkoumání zbývajících událostí (detekce více než třikrát za hodinu) se vytvořilo toto omezení. Detektor ve většině těchto případů poslal několik zpráv v jedné vteřině, a proto jsou události, které nastaly více než třikrát za hodinu, zanedbány.

$$\text{sgn}(-CntHour + 4) \tag{3.4}$$

Toto chování zajišťuje první část algoritmu (3.4) a to funkce signum. Funkce signum nabývá diskrétních hodnot {-1,0,1}. Pokud detektor nahlásí pro dvojici IP adresa a kategorie více než tři bezpečnostní události za hodinu, tak rovnice (3.4) zajistí nulové, či záporné celkové skóre.

Druhá část ohodnocovacího algoritmu (3.5) přiřazuje konkrétní skóre. Vzorec přiřazuje ohodnocení bezpečnostní události v závislosti na ceně kategorie a relativní četnosti výskytu kategorie. Cena se volí v konfiguračním souboru. Čím větší cena, tím větší skóre. Pro dvě kategorie se stejnou cenou bude mít větší skóre ta, která má menší relativní četnost (Probability) – větší prioritu mají méně časté události.

$$(2^{Price-1}(1 - Probability) + 2^{Price-1}) \tag{3.5}$$



Obrázek 3.6: Průměrná hodinová četnost dvojice kateogrie a IP adresa bez skenů (NEMEA)

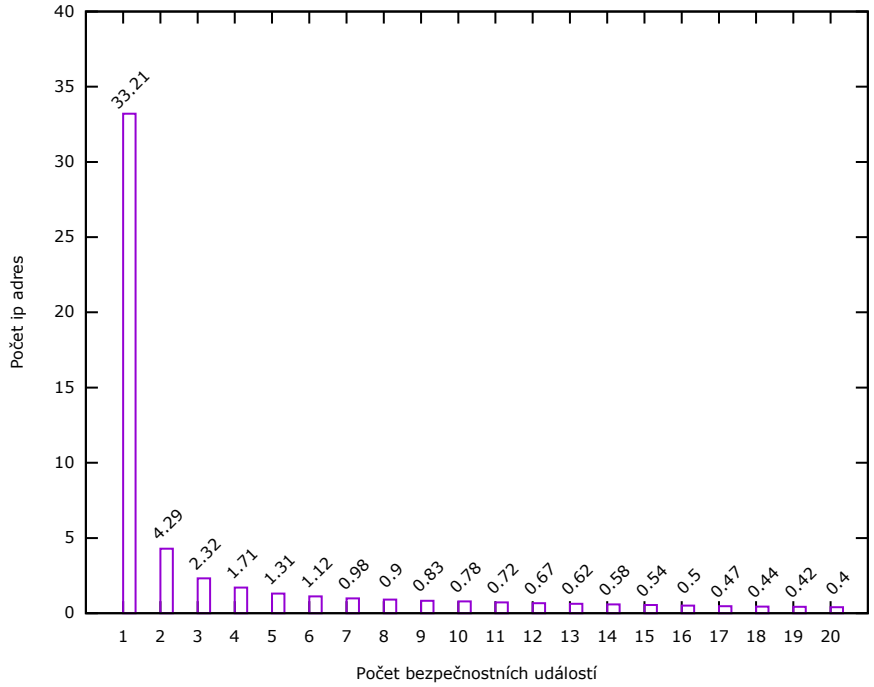
Pro výpočet skóre používá algoritmus exponenciální funkci o základu dva s *Price* – cena kategorie v exponentu. Tato volba proběhla s ohledem na druhý parametr tohoto algoritmu a to je *Probability* – relativní četnost výskytu dané kategorie (3.6). Protože menší relativní četnosti mají mít větší skóre, je tu použit doplněk k jevu *Probability*:

$$(1 - probability) \tag{3.6}$$

Dále je nutné přepočítat relativní četnost výskytu kategorie z intervalu $\langle 0, 1\rangle$ na interval ceny. Každá četnost bude ležet v intervalu (3.7).

$$\langle 2^{Price-1}, 2^{Price}\rangle \tag{3.7}$$

Exponenciální funkce o základu dva byla zvolena pro její rychlý růst. Díky této funkci dokážeme lépe porovnávat jednotlivé skóre mezi sebou. Dále dokážeme snadno porovnávat bezpečnostní události se stejnou cenou, ale rozdílnou pravděpodobností. To je vhodné například v situacích, kdy máme omezené možnosti pro záchyt. V situaci, kdy systém pro záchyt bezpečnostní události dokáže obsloužit pouze jeden záchyt, tak vždy vyhraje bezpečnostní událost s větší cenou.



Obrázek 3.7: Průměrný počet bezpečnostních událostí za hodinu (NEMEA bez skenů)

Pro převod relativní četnosti na interval $\langle a, b\rangle$ je použit obecný vzorec (viz. (3.8)). :

$$(b - a) * (probability) + a \tag{3.8}$$

Rovnice má po konečné úpravě tvar (3.3).

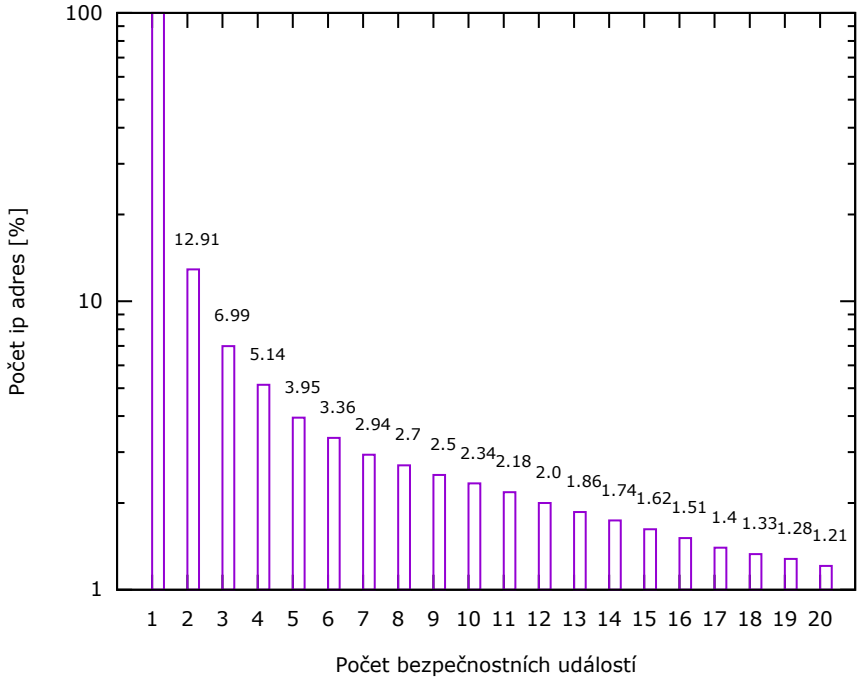
3.2.6 Limit počtu bezpečnostních událostí za hodinu

Limit počtu bezpečnostních událostí za hodinu je ochrana proti opakujícímu se odesílání stejné dvojice „kategorie + IP adresa“. Tato ochrana se vztahuje pouze na část algoritmu pro skeny v algoritmu 1.

Parametr *limit* slouží pro kontrolu bezpečnostních událostí za hodinu. Pokud počet dvojice „kategorie + IP“ adresa překročí hodnotu *limit*, tak veškeré události obdrží nulové skóre.

Algoritmus pro ostatní kategorie má tuto ochranu již v návrhu, kde se přiřazuje záporné skóre všem událostem, které se objeví v systému více než třikrát za hodinu.

Limit je nastaven na číslo 500, pro základní nastavení algoritmu. Tedy čtyřikrát za hodinu se maximálně může zachytit jeden sken pro stejnou IP



Obrázek 3.8: Průmerný počet bezpečnostních událostí za hodinu v procentech (NEMEA bez skenů).

adresu. Tento výpočet plyne z rovnice (3.2), pro nastavení parametrů $p = 95$ a $modul = 100$.

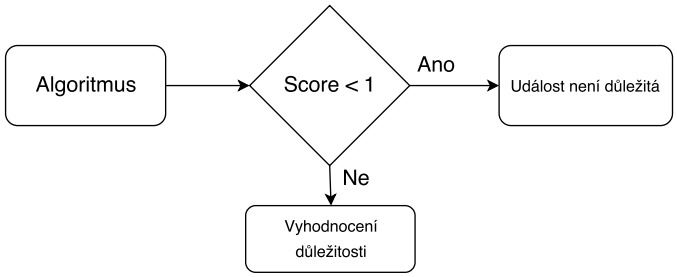
3.3 Vyhodnocení skóre

Poté, co algoritmus 1 přiřadí bezpečnostní události skóre, tak se toto skóre musí vyhodnotit. Toto vyhodnocení nabývá dvou stavů: událost je důležitá, nebo událost není důležitá. Pokud je událost důležitá, potom je poslána příslušné osobě, či do příslušného systému pro podrobnější analýzu. Pokud událost není důležitá, tak je událost zahozena.

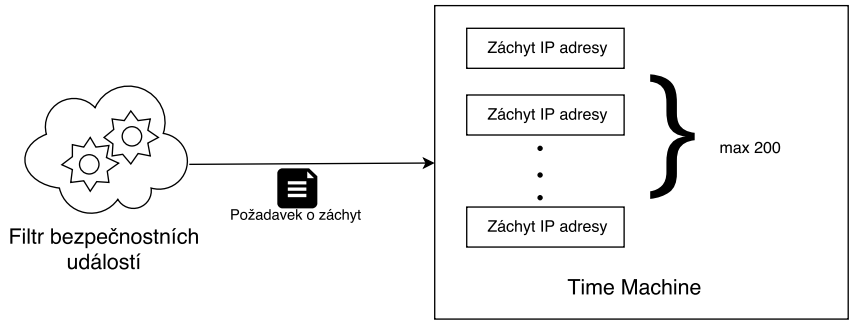
Vyhodnocení důležitosti se liší podle systému za navrhovaným Filtrem. Pokud máme neomezené prostředky, můžeme si dovolit označit za důležité všechny zprávy, které mají skóre ≥ 1 (viz. Obrázek 3.9).

Pokud nemáme neomezené prostředky, tak je několik možností, jak lze danou situaci řešit:

- Nastavení ceny pro skeny (a ostatní nedůležité kategorie) na 0. Veškeré ostatní bezpečnostní události budou důležité.



Obrázek 3.9: Vyhodnocení důležitosti.



Obrázek 3.10: Omezení počtu IP adres v systému Time Machine.

- Vybírat bezpečnostní událost s nejvyšším skóre za určitě časové období na které jsou prostředky. Např. každou hodinu je vybrána událost s nejvyšším skóre.

- Pokud máme systém, který zachytává bezpečnostní událost, tak důležitost nově příchozí události je závislá na události, která se právě zachytává. Tento záchyt trvá určitou dobu. Pokud má nově příchozí událost větší skóre, než zachytávaná událost, tak je zachytávaná událost označena za nedůležitou a nově příchozí je označena za důležitou. Takto je řešen systém Time Machine.

3.3.1 Time Machine

Time Machine, který je použit jako systém navazující na Filtr bezpečnostních událost, navrhovaný v této sekci, má omezené prostředky. Spojení těchto dvou systémů znázorňuje obrázek 3.10.

Systém dokáže teoreticky zachytávat až 1024 IP adres současně. Toto číslo je pouze teoretické a závisí na velikosti provozu pro jednotlivé IP adresy. Při velkém datové provozu by mohli vznikat ztráty dat. Proto bylo vytvořeno omezení na maximální počet 200 IP adres.

Pokud se zachytává méně než 200 IP adres, tak jsou veškeré bezpečnostní události se skóre ≥ 1 zachytávány. To znamená, že se vytvoří požadavek o záchyt podle konfiguračního souboru, který určuje, jak dlouho se má daná IP adresa zachytávat/maximální počet zachycených paketů.

Pokud se zachytává více než 200 IP adres a skóre nově příchozí bezpečnostní události je větší, než některá ze zachytávaných IP adres, tak je záchyt IP adresy s nejmenším skóre zrušen a je vytvořen požadavek o záchyt. Pokud není skóre větší než alespoň jedna zachytávaná IP adresa, tak je událost označena za nedůležitou.

Kapitola 4

Implementace systému automatického filtrování bezpečnostních událostí

Filtr bezpečnostních událostí byl navržen jako nový izolovaný systém, nezávislý na formátu bezpečnostních událostí na vstupu. Výstupem Filtru bezpečnostních událostí jsou vyfiltrované bezpečnostní události. Tyto vybrané události se poté transformují na požadavky o záchyt, které jsou určené pro Time Machine.

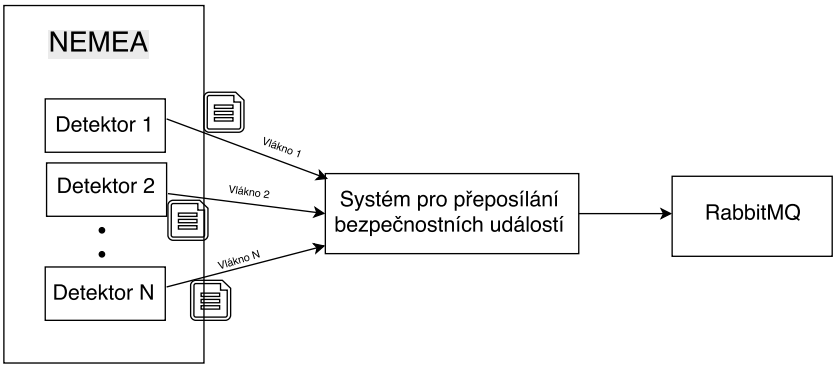
V této kapitole budou popsány všechny důležité části celého systému. Jedná se o systém pro přeposílání bezpečnostních událostí do systému RabbitMQ. Dále RabbitMQ, který je používán pro asynchronní výměnu bezpečnostních událostí mezi systémy. A nakonec samotný Filtr bezpečnostních událostí, který zpracovává veškeré bezpečnostní událost a rozhoduje, zda-li jsou tyto události důležité. Pokud je událost vyhodnocena za důležitou, vytvoří se požadavek o záchyt do systému Time Machine. Všechny aplikace jsou implementovány v programovacím jazyku Python 2.

4.1 Systém pro přeposílání bezpečnostních událostí

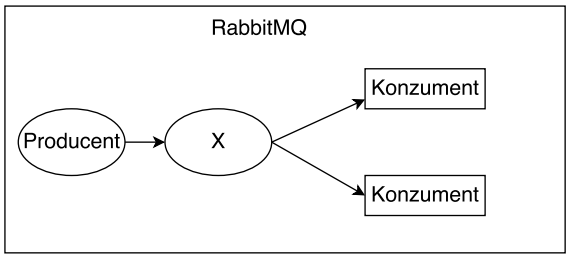
Obrázek 4.1 znázorňuje, jak je řešeno přeposílání nahlášených událostí ze systému NEMEA. V rámci této práce vznikl modul, který sbírá výsledky detekce, a přeposílá je do RabbitMQ fronty.

Pro požadavky Filtru bezpečnostních události byly detektory v systému NEMEA nakonfigurovány tak, aby bylo možné pomocí jejich výstupních rozhraní, které používají *Unix sockets*, přijímat IDEA zprávy.

Systém pro přeposílání bezpečnostních událostí se připojí na výstupní rozhraní detektorů v systému NEMEA. Pro každý detektor si vytvoří samostatné



Obrázek 4.1: Systém pro přeposílání bezpečnostních událostí.



Obrázek 4.2: Systém pro přeposílání bezpečnostních událostí.

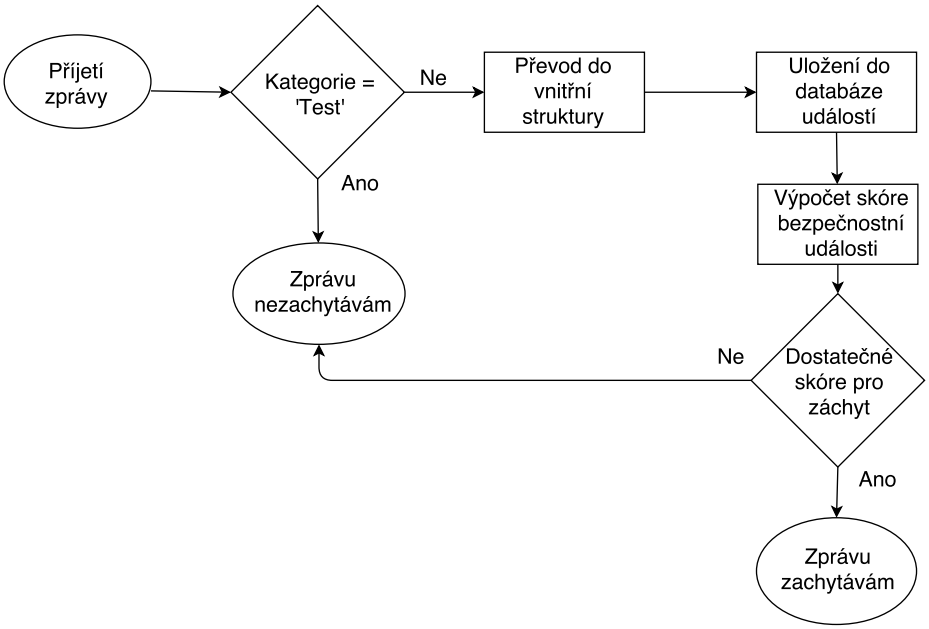
vlákno. Pokud detektor detekuje bezpečnostní událost, systém pro přeposílání bezpečnostních událostí se postará o přeposlání IDEA zprávy do služby RabbitMQ. Tato služba se postará o rozeslání této události na všechny připojené konzumenty zpráv, na Filtr bezpečnostních událostí.

RabbitMQ je v celém systému používán pro asynchronní výměnu IDEA zpráv mezi aplikacemi, tato služba používá AMQP [30] protokol pro výměnu zpráv. Aplikace funguje jako klient-server aplikace. Kde RabbitMQ na všech diagramech je chápán jako server a všechny aplikace, které jsou spojené s RabbitMQ, jsou klienti.

Veškerá funkčnost je obstarávána serverem, ale klienti si nastaví vlastnosti, s kterými chtějí pracovat. Tyto vlastnosti jsou:

- Producent/konzument – zda-li je klient odesílatel, nebo příjemce
- Fronta – typ fronty pro výměnu zpráv
- Exchange point – místo na RabbitMQ, kam se ukládají zprávy

Obrázek 4.2 zobrazuje, v jakém režimu RabbitMQ je naimplementován. Jedná se o typ fronty rozvětvení (fanout). Pokud se přihlásí více konzumentů pro odběr, tak všichni konzumenti dostanou vždy všechny zprávy od producentů.



Obrázek 4.3: Diagram aktivit.

V systému je prozatím pouze jeden producent (NEMEA) a jeden konzument (Filtr bezpečnostních událostí), ale pomocí RabbitMQ lze toto schéma snadno rozšířit na vícero producentů a vícero konzumentů, kde producenti mohou být dalšími detekční systémy. Producenti mohou být např. nové testovací verze Filtru detekčních událostí nebo filtr, který bude zpracovávat pouze detektory, nacházející se v testovacím režimu.

4.2 Filtr bezpečnostních událostí

Implementace Filtru bezpečnostních událostí má na vstupu několik konfiguračních souborů, které definují chod aplikace. Tyto konfigurační soubory je nutné přizpůsobit konkrétní počítačové síti.

Na obrázku 4.3 je popsán diagram aktivit pro příjem bezpečnostní události. Nejdříve je zpráva přijata a poté jsou odděleny všechny testovací zprávy. Testovací zprávy mají v položce kategorie hodnotu „Test“. Testovací zprávy nejsou nadále zpracovávány. Pokud se nejedná o testovací zprávy, je následně zpráva převedena do datové struktury „Dictionary“ a použity pouze položky, které jsou předdefinované v konfiguračním souboru „mapping“ (Podsekce 4.2.2). S takto upravenou zprávou následně celý program pracuje.

Každá zpráva je následně uložena do hašovací tabulky. Poté je vypočítáno skóre bezpečnostní události podle algoritmu popsaného v podsekci 3.2.3, jednotlivé ceny kategorií se berou ze zdrojového souboru „static_prices.json“

(Výpis 4.1). Pokud je skóre dostatečně velké, je vytvořen požadavek o záchyt v systému Time Machine.

```
[
 {
   "Default": {
     "Score": 5,
     "Direction": "S",
     "Timeout": 300,
     "Packets": 10000
   }
 },
 {
   "Abusive": {
     "Harassment": {
       "Score": 4,
       "Direction": "BS",
       "Timeout": 200,
       "Packets": 2000
     },
]
```

Výpis 4.1: Zkrácený výpis konfiguračního souboru „static_prices.json“

4.2.1 Přijímání zpráv

Přijímání zpráv je implementováno pomocí pythonové knihovny „pika“ [31], která implementuje AMQP [30] protokol. Pomocí této knihovny se nadefinují vlastnosti pro RabbitMQ server. Tyto vlastnosti pro implementaci „Filtru bezpečnostních událostí“ jsou:

- Konzument
- Fronta typu „fanout“ – rozvětvení
- Exchange point „broadcast_idea“

Toto přijímání zpráv je v blokujícím režimu. Pokud přijde na RabbitMQ bezpečnostní událost, zpráva je ihned vyzvednuta z fronty na RabbitMQ serveru. Pokud se nejedná o validní JSON soubor, tak tato bezpečnostní událost není nadále zpracovávána. Pokud se jedná o validní JSON soubor, tak je tato zpráva dále převedena do vnitřní struktury aplikace.

4.2.2 Převod bezpečnostní události do vnitřní struktury aplikace

Převod bezpečnostní události do vnitřní struktury aplikace je univerzální převod bezpečnostní události do hašovací tabulky „Filtru bezpečnostních událostí“.

Současná implementace umí pracovat pouze s bezpečnostními událostmi na vstupu ve formátu JSON. Tento požadavek plyne z použití Pythonové knihovny jq [32], která se stará o manipulaci s těmito zprávami. Tato knihovna jq volá pouze aplikaci jq [29]. Jq obsahuje vlastní jazyk pro filtrování a různé transformace struktury JSON. Tento jazyk je použit při výběru položek z bezpečnostních událostí v konfiguračním souboru „Mapping“.

```
'.Format' == 'IDEA0'
  '.DetectTime' -> 'DetectTime'
  '.Category' -> 'Category'
  'if has("Target") then .Target[].IP4 else null end' -> 'TargetIP4'
  'if has("Target") then .Target[].IP6 else null end' -> 'TargetIP6'
  'if has("Source") then .Source[].IP4 else null end' -> 'SourceIP4'
  'if has("Source") then .Source[].IP6 else null end' -> 'SourceIP6'

'. | keys[]' == 'IDMEFMessage'
  '.IDMEFMessage.Alert.Source.Node.Address.address.__text' -> 'Source'
  '.IDMEFMessage.Alert.Target.Node.Address.address.__text' -> 'Target'
  '.IDMEFMessage.Alert.CreateTime.__text' -> 'DetectTime'
  '.IDMEFMessage.Alert.Classification.Reference[].name.__text' -> 'Category'
```

Výpis 4.2: Konfiguračního soubor „Mapping“

Konfigurační soubor „Mapping“ obsahuje definici převodu bezpečnostní události do datové struktury v jazyku Python „Dictionary“. Tento konfigurační soubor popisuje, které formáty bezpečnostních událostí lze zpracovávat. V tomto příkladu 4.2 lze zpracovat pouze dva formáty: IDEA a IDMEF. A dále je zobrazeno mapování mezi hašovací tabulkou a bezpečnostní událostí. Levá strana pravidel je jq syntaxe pro výběr položek z příslušné zprávy. Levá strana pravidla je oddělena od pravé pomocí symbolu: →. Práva strana pravidla je klíč v hašovací tabulce, který se má použít pro uložení výstupu z levé strany pravidla.

Pro pochopení je zde uveden příklad. Tento příklad má na vstupu bezpečnostní událost v IDEA formátu ve zkrácená podobě (výpis 4.3) a používá se mapovacího konfiguračního soubor z výpisu 4.2. Převod této bezpečnostní události pomocí tohoto konfiguračního souboru má za následek vznik nové hašovací tabulky z výpisu 4.4

```
{
   "Source" : [
      {
         "Port" : [
            64236
         ],
         "Type" : [
            "Malware"
         ],
         "IP4" : [
            "147.230.154.176"
         ],
         "Note" : "Malware: B106"
```

```
        }
    ],
    "ID" : "1-1478769718.326326-oPmD3haYNiZQ",
    "Target" : [
        {
            "IP4" : [
                "204.95.99.142"
            ],
            "Port" : [
                1991
            ]
        }
    ],
    "DetectTime" : "2016-11-09T22:58:49",

    "Category" : [
        "Intrusion.Botnet",
        "Malware"
    ],
    "Format" : "IDEA0",

}
```

Výpis 4.3: Zkrácená verze bezpečnostní události v IDEA formátu

```
{
    "Category" : ["Intrusion.Botnet","Malware"],
    "SourceIP4" : ["147.230.154.176"],
    "TargetIP4" : ["204.95.99.142"],
    "DetectTime": "2016-11-09T22:58:49"
}
```

Výpis 4.4: Bezpečnostní událost převedena do vnitřní strukturz

4.2.3 Uložení bezpečnostní události do uložiště

Filtr bezpečnostních událostí obsahuje dvě uložiště, které jsou reprezentovány jako dvě na sobě nezávislé hašovací tabulky. Obě tyto uložiště jsou vyžadovány algoritmem, který přiřazuje skóre jednotlivým událostem.

První hašovací tabulka slouží pro ukládání bezpečnostních událostí, a pro následující popis bude označována jako uložiště bezpečnostních událostí. Toto uložiště si uchovává veškeré zprávy, které nejsou starší než jedna hodina. Čas, který se používá, je čas detekce. Pokud je čas detekce starší než jedna hodina, tak je tento záznam z uložiště odebrán. Toto uložiště se neukládá na pevný disk a je pouze v paměti RAM, proto při přerušení programu jsou informace obsažené v této u ztraceny.

Jako klíč hašovací tabulky se používá dvojice „směr + IP adresa“, kde směr je buď zdroj označen písmenem „S“ jako Source nebo cíl označen hodnotou „T“ jako Target. Klíč potom může vypadat například takto: „S1.2.3.4“. Hodnota každého klíče je další hašovací tabulka, která obsahuje: pole bezpečnostních událostí pro konkrétní IP adresu, počet položek v poli a počet jednotlivých kategorií bezpečnostních událostí v poli. Každá bezpečnostní událost je do uložiště uložena tolikrát, kolik obsahuje zdrojových a cílových IP adres. Pokud obsahuje jednu zdrojovou IP adresu a jednu cílovou IP adresu, je uložena do uložiště dvakrát - viz ukázka 4.5, kdy pole bezpečnostní události

obsahuje tyto položky: čas detekce, kategorie, detektor, „směr + IP adresa“. Nová bezpečnostní událost je vždy přidána na konec tohoto pole.

Příklad uložiště bezpečnostních událostí je znázorněn ve výpisu 4.5.

```
{
    {"T195.113.253.123":
    {'cnt': 1, 'Recon.Scanning': 1, 'alerts': [
      ['2017-04-25 16:13:42', ['Recon.Scanning'],
      ['cz.cesnet.nemea.vportscan'], ['S201.214.56.9']]
    ]}},

    {"S201.214.56.9":
    {'cnt': 1, 'Recon.Scanning': 1, 'alerts': [
      [ '2017-04-25 16:13:42', ['Recon.Scanning'],
      ['cz.cesnet.nemea.vportscan'], ['T195.113.253.123'] ]
    ]}
}
```

Výpis 4.5: uložiště bezpečnostních událostí

Druhá hašovací tabulka je pro ukládání četností jednotlivých kategorií bezpečnostních událostí, pro popis je označena jako uložiště absolutních četností kategorií. Každá zpráva obsahuje minimálně jednu kategorii, která popisuje o jaký bezpečnostní incident se jedná. Každý název kategorie slouží jako klíč pro uložiště četností a hodnota se vždy zvýší o jedna. I když v uložišti bezpečnostních události máme většinu zpráv duplicitně, tak tento čítač není duplicitní a popisuje reálnou absolutní četnost bezpečnostních událostí. Pro příklad: z uložiště bezpečnostních událost popsané výše 4.5, bude vypadat uložiště relativní četnosti následovně 4.6.

Ve vyhodnocovacím algoritmu se tato absolutní četnost přepočítá na relativní četnost.

Příklad uložiště absolutních četností kategorií je znázorněn ve výpisu 4.6.

```
{'Recon.Scanning': 1.0, 'cnt': 1.0}
```

Výpis 4.6: uložiště absolutních četností kategorií

4.2.4 Výpočet skóre bezpečnostní události

Výpočet skóre bezpečnostní události vychází přímo z Algoritmu 1, který je popsán v Kapitole 3 – Návrh systému pro filtrování bezpečnostních událostí. Tak jak je teoreticky popsán v této kapitole, je i implementován s drobnými úpravami.

Konfigurační soubor „algorithm_parameters.json“ slouží pro úpravu parametrů algoritmu pro vyhodnocení důležitosti bezpečnostní události. Dále bude popsán význam všech parametrů souboru.

```
{
    "first" : 5,
    "every" : 100,
    "limit" : 500,
    "max_parallel_capture_cnt" : 200,
    "random_scan" : 250
}
```

Výpis 4.7: konfigurační soubor „algorithm_parameters.json“

Parametr „random_scan“ určuje, jak často se vybere náhodná bezpečnostní událost (pouze skeny).

Pro výpočet skóre algoritmu skenu (Kapitola 3 – rovnice (3.2)) uživatel nezadává přímo parametry p a *modul*, které jsou popsané v Algoritmu 1. Namísto toho uživatel zadává parametry: *first* a *every*, z kterých se vypočtou původní parametry, popsané v návrhu. Výpočet parametrů p a *modul* probíhá následovně:

```
p = every - first
modul = every
```

Parametr limit udává hodinový limit počtu dvojice „kategorie + IP adresa“, při jehož překročení se budou veškeré bezpečnostní události této dvojice ignorovat.

Parametr „max_parallel_capture_cnt“ udává maximální počet paralelních požadavků o záchyt současně.

Každá bezpečnostní událost obsahuje nejméně jednu kategorii. Do algoritmu vstupuje pouze jedna kategorie. Pokud událost obsahuje více kategorií, je vybraná ta s nejvyšší cenou. Pokud zpráva obsahuje více stejných kategorií se stejnou cenou, potom je vybraná první v pořadí z konfiguračního souboru “static_prices.json”. Uložiště absolutních četností kategorií v tomto ohledu nehraje roli.

Konfigurační soubor „static_prices.json“ (Výpis 4.1) určuje cenu jednotlivých kategorií a parametry záchytu. Cena kategorie je v konfiguračním souboru uvedena jako „Score“. Ceny jednotlivých kategorií se při spuštění programu načtou do hašovací tabulky. Kategorie v IDEA formátu se skládá z kategorie a podkategorie oddělený tečkou. Např. „Abusive.Harassment“, kde „Abusive“ je hlavní kategorie a „Harassment“ je podkategorie. Ve zdrojovém souboru lze této funkcionality dosáhnout pomocí vnořených objektů viz Výpis 4.8.

Klíče jednotlivých objektů se spojují v rekurzi pomocí tečky až do té doby dokud nenarazíme na nejhlubší bod rekurze. V tomto bodě je vytvořen název kategorie. Takto vytvořený název je použit v hašovací tabulce jako klíč a hodnota tohoto klíče je nová hašovací tabulka se všemi údaji, které jsou uvedeny v nejhlubším bodě této rekurze. Konfigurační soubor z výpisu 4.8 je potom převeden na hašovací tabulku ve výpisu 4.9. Kategorie „Default“ je výchozí

kategorie pro všechny kategorie bezpečnostní události. To znamená, že pokud není uvedena kategorie v konfiguračním souboru, nebo pro uvedenou kategorii chybí hodnoty z nejhlubšího bodu JSON objektu, tak se použije hodnoty z kategorie „Default“

```
[
  {
    "Default": {
      "Packets": 10000,
      "Direction": "S",
      "Score": 10,
      "Timeout": 300
    }
  },
  {
    "Abusive": {
      "Harassment": {
        "Score": 10
        "Packets": 5000,
      },
      "Violence": {
        "Score": 10
        "Timeout": 100,
      },
    }
  }
 ]
```

Výpis 4.8: Konfigurační soubor “static_prices.json”

```
{
  {  "Default": {"Packets":5000,  "Direction":"S",
     "Score":10,"Timeout":300  } },
{"Abusive.Harassment": {"Score": 10, "Packets": 5000 } },
{"Abusive.Violence": {"Score": 10, "Timeout": 100 } }
}
```

Výpis 4.9: Hašovací tabulka kategorií

4.2.5 Skóre pro záchyt

Po vyhodnocení skóre jednotlivým bezpečnostním událostem je nutné rozhodnout, zda je tato událost důležitá, či není.

Toto rozhodnutí závisí na tom, kolik současných požadavků o záchyt může probíhat. Pro zjednodušení tohoto popisu je jeden požadavek o záchyt chápán

jako jeden časový interval. Těchto akcí je možno mít maximálně 200 současně. V každém okýnku je jedna IP adresa.

Pro vysvětlení, jaké požadavky jsou zachytávány, bude dále uvažováno pouze jedno okénko pro záchyt. Při ohodnocení bezpečnostní události může nastat několik situací:

- Skóre < 1 – IP adresa se nezachytává
- Okno záchytu je prázdné a skóre $>= 1$ – IP adresa se zachytává.
- Okno záchytu není prázdné a skóre aktuální události je menší, než zachytávaná událost – IP adresa se nezachytává
- Okno záchytu není prázdné a skóre aktuální události má o 20% větší skóre než událost, která se zachytává – IP adresa se nezachytává
- V ostatních případech se IP adresa nezachytává

Pokud jsou splněny požadavky aby se IP adresa zachytávala je vytvořena požadavek o záchyt.

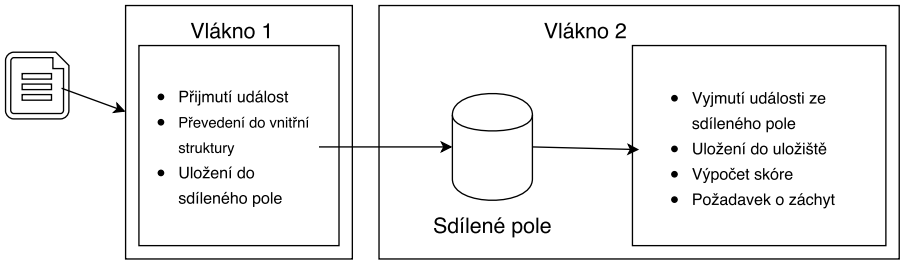
4.2.6 Parametry záchytu

Konfigurační soubor „static_prices.json“ (Výpis 4.1) umožňuje konfigurovat parametry záchytu. Pomocí těchto parametrů dokážeme pro jednotlivé kategorie vytvářet rozdílné požadavky o záchyt. Pokud daná kategorie neobsahuje v konfiguračním souboru parametry záchytu, tak jsou použity parametry záchytu z kategorie „Default“.

Konfigurační soubor obsahuje tři parametry pro záchyt: „Direction“, „Timeout“ a „Packets“. Parametr „Timeout“ určuje časový limit záchytu. Parametr „Packets“ určuje paketový limit záchytu. Parametr „Direction“ určuje směr záchytu.

Směr záchytu může nabývat pěti hodnot: „S“, „T“, „BS“, „BT“, „BB“. Vysvětlení těchto parametrů je následovné:

- „S“ – zachytávám zdrojovou IP adresu, směr zdroj
- „T“ – zachytávám cílovou IP adresu, směr cíl
- „BS“ – zachytávám zdrojovou IP adresu, v obou směrech
- „BT“ – zachytávám cílovou IP adresu, v obou směrech
- „BB“ – zachytávám cílovou i zdrojovou IP adresu, v obou směrech



Obrázek 4.4: Diagram zobrazující vlákna aplikace.

4.2.7 Popis více vláken

Aplikace je rozdělena na dvě vlákna, která běží paralelně. První vlákno se stará o příjem bezpečnostní události a převod do vnitřní struktury aplikace. Druhé vlákno se stará o uložení bezpečnostní události do uložiště, výpočet skóre a při splnění kritérii pro požadavek o záchyt – vytvoří požadavek o záchyt.

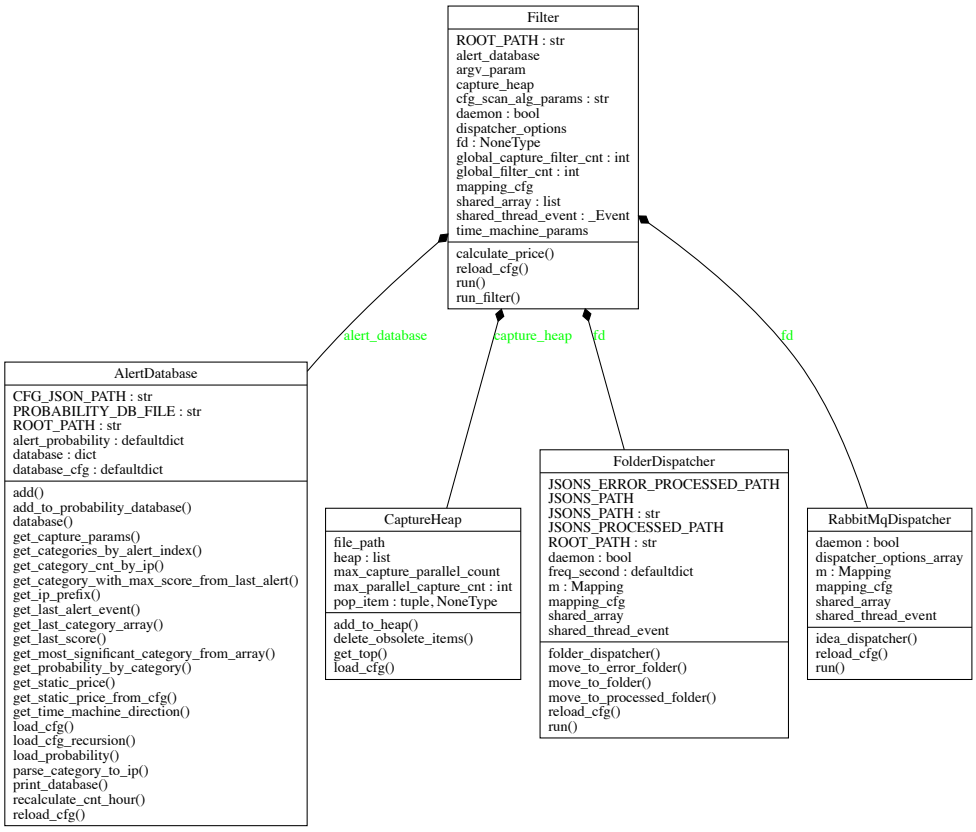
Implementace používá třídu „threading.Thread“, ze které jsou dvě hlavní třídy zděděné. Pro výměnu je používáno sdílené pole, kde první vlákno přidává na konec tohoto pole bezpečnostní události a druhé vlákno tyto bezpečnostní události vybírá. Obě dvě vlákna běží v nekonečném cyklu. První vlákno čeká v blokujícím čekání na přijmutí zprávy. Po přijmutí zprávy předá tuto zprávu do sdíleného pole a nastaví příznak pro výběr z tohoto pole pro druhé vlákno na „True“. Tím se druhé vlákno aktivuje a začne vybírat bezpečnostní události ze sdíleného pole. Pokud je sdílené pole prázdné, tak nastaví svůj vnitřní příznak na „False“ a čeká tak v blokujícím volání, dokud se tento příznak znova nenastaví na „True“.

4.2.8 Diagram tříd

Jednotlivé části programu jsou rozdělené na samostatné třídy. Tyto třídy jsou znázorněny v diagramu tříd na obrázcích 4.5 a 4.6.

Obrázek 4.5 znázorňuje třídy, které jsou na sobě závislé. Hlavní třídou celé aplikace je třída „Filter“, která řídí celý běh aplikace. Tato třída se spouští v novém vlákně. V této třídě je prováděno vyhodnocování skóre a následné vyhodnocování důležitosti bezpečnostní události. Dále zajišťuje komunikaci se systémem Time Machine a vytváření požadavků o záchyt.

Třída „RabbitMqDispatcher“ zajišťuje příjímání zpráv z RabbitMQ serveru. Tato třída se spouští v novém vlákně. Při přijetí nové zprávy tato třída zajistí převedení do interní datové struktury bezpečnostních událostí. Dále je tato událost uložena do sdíleného pole, kde si jí třída „Filter“ dále zpracuje. Pro testovací účely existuje v aplikaci ještě třída „FolderDispatcher“, která pracuje podobně jako „RabbitMqDispatcher“, ale přijímá bezpečnostní události ze složky, nikoli z RabbitMQ.



Obrázek 4.5: Třídy se závislostí.

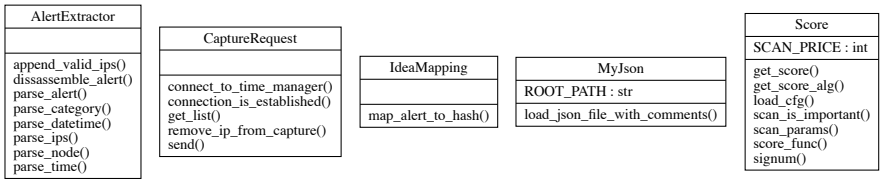
Obrázek 4.6 znázorňuje všechny třídy, které nemají závislosti. Tyto třídy mají všechny metody statické.

4.2.9 Textový interpretr

Napověda k programu je dostupná v Příloze B.1. Příkazy jsou interpretované ve třídě *Shell*.

Textový interpretr slouží pro manipulaci s programem. Momentálně obsahuje program pouze čtyři příkazy:

- help – zobrazí nápovědu
- start – spustí program



Obrázek 4.6: Třídy bez závislosti.

- reload_config – načte všechny změny v konfiguračních souborech do aktuální konfigurace běžícího souboru
- exit – ukončí program

Kapitola 5

Testování systému automatického filtrování bezpečnostních událostí

5.1 Testovací prostředí

Implementované systémy byly testovány na dvou různých strojích. Většina testů byla provedena na produkčním stroji, který je nasazen na síti CESNET2. Tento počítač má k dispozici 4GB RAM a dvojice procesorů Intel Xeon E5-2670.

Pro test maximálního počtu bezpečnostních událostí, které systém dokáže obsloužit, byl použit počítač s procesorem Intel Xeon E3-1230V2 s operační pamětí 16GB RAM.

5.2 Testovací konfigurace

Pokud není explicitně uveden konfigurační soubor u určitého testu, pak konfigurace algoritmu je následující:

```
{
  "first": 5,
  "every": 100,
  "limit": 500,
  "max_parallel_capture_cnt": 200,
  "random_scan": 250
}
```

Výpis 5.1: Konfigurační sestava pro testování

5.3 Generátor bezpečnostních událostí

Pro testování funkčnosti Filtru bezpečnostních událostí byl vytvořen generátor bezpečnostních událostí.

Vstupem generátoru bezpečnostních událostí je soubor s bezpečnostními událostmi. Tento soubor obsahuje na každém řádku jednu bezpečnostní událost. Pro testování byly použity data ze systému NEMEA (leden 2017).

Tento generátor dat převádí historická data na současná. Tento převod převede položky („DetectTime“ a „CreateTime“) na aktuální datum a čas. Následně každou bezpečnostní událost uloží do souboru nebo pošle do RabbitMQ. Pro události uložené do souboru, slouží třída „FolderDispatcher“, která zpracovává tyto soubory s událostmi.

5.4 Dynamická analýza kódu

Pro dynamickou analýzu kódu byl použit nástroj „pprofile“ [33]. Pomocí tohoto nástroje byly odhaleny hlavní výkonnostní nedostatky programu, které byly opraveny.

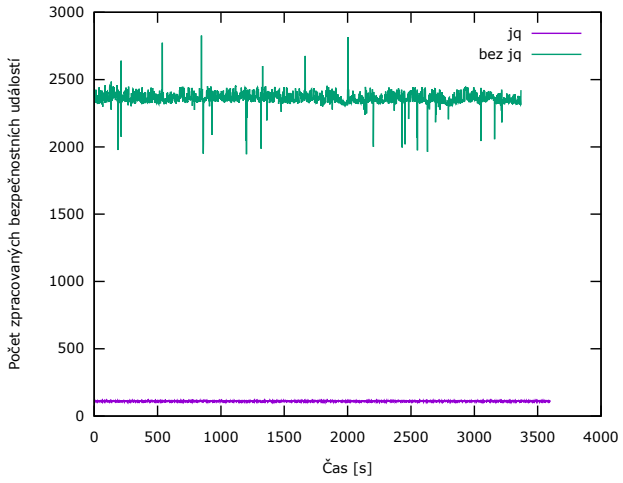
5.5 Maximální počet bezpečnostních událostí zpracovaných za jednu vteřinu

Pro měření maximálního počtu zpracovaných bezpečnostních událostí, za jednu vteřinu bylo vytvořeno testovací prostředí mimo produkční server. K testování byl použit generátor bezpečnostních událostí přeposílající veškeré události na vstupu na RabbitMQ server, umístěný na stejném stroji.

V základní konfiguraci Filtru bylo dosaženo zpracování okolo 110 bezpečnostních událostí za jednu vteřinu (Obrázek 5.1). Pro systém NEMEA i pro systém WARDEN je toto zpracování událostí dostatečné. Filtr zpracuje bez problému všechny bezpečnostní události z těchto systémů.

Dynamická analýza kódu objevila hlavní výkonnostní slabinu, ta se nachází v knihovně „jq“. Při každém vstupu se musí vyhodnotit transformace, která je popsaná v řetězci (konfigurační soubor 4.2). Tento řetězec obsahuje složitější výrazy, jako jsou podmínky. Tyto výrazy se musí převést do vnitřní interpretace, a to se provádí pro každý vstup zvlášť. Tento převod je velice náročný a zpomaluje chod celé aplikace.

Odstraněním knihovny „jq“ a provedením transformace uvnitř kódu, se výsledky zlepšily. Bylo dosaženo zpracování okolo 2300 bezpečnostních událostí za jednu vteřinu.



Obrázek 5.1: Maximální počet bezpečnostních událostí, které dokáže Filtr bezpečnostních událostí obsloužit za jednu vteřinu.

5.6 Vytížení procesoru a paměti

Obrázek 5.2 znázorňuje vytížení procesoru a využití paměti RAM. Vytížení procesoru na produkčním serveru se dlouhodobě pohybuje na průměru okolo 0.37% a RAM paměť okolo 0.6%.

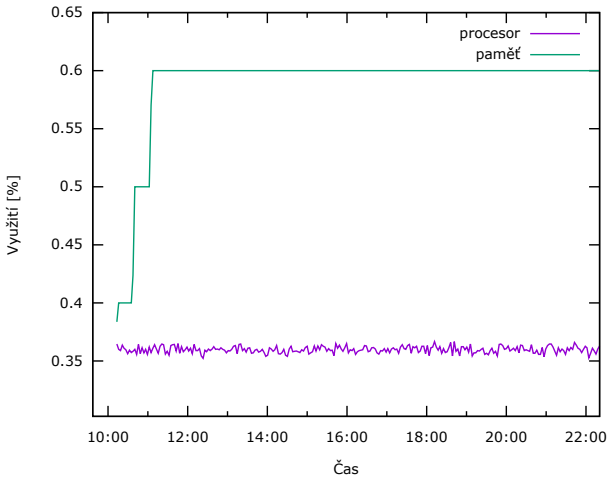
Zelená čára znázorňuje vytížení paměti RAM. Skoky v grafu jsou zapříčiněny plněním úložiště bezpečnostními událostmi. Tyto události si Filtr udržuje v úložišti pouze poslední hodinu, proto přibližně po hodině od spuštění programu se využití paměti RAM zastaví na konstantní hodnotě.

5.7 Testování konfigurací algoritmu

Toto testování měří celkovou redukci bezpečnostních událostí v časovém intervalu 24 hodin. Pro testování byly zvoleny čtyři různé konfigurace Filtru bezpečnostních událostí. Testování probíhalo na produkčním serveru. Aplikace byly spuštěny současně, na vstupu byly identické bezpečnostní události.

- Konfigurace 1 je výchozím nastavením aplikace.

```
{
  "first": 5,
  "every": 100,
  "limit": 500,
  "max_parallel_capture_cnt": 200,
  "random_scan": 250
```



Obrázek 5.2: Vytížení procesoru a paměti

```
}
```

Výpis 5.2: Konfigurace 1

- Konfigurace 2 obsahuje stejné nastavení algoritmu jako Konfigurace 1. Ale cena kategorie sken je nastavena na 0. Tato konfigurace vždy označí kategorii sken za nedůležitou.
- Konfigurace 3 zanedbává randomizaci v algoritmu. Dále je upraven parametr „first“ na hodnotu 2.

```
{
  "first": 2,
  "every": 100,
  "limit": 500,
  "max_parallel_capture_cnt": 200,
  "random_scan": -1
}
```

Výpis 5.3: Konfigurace 3

- Konfigurace 4 zanedbává opakující se skeny a vybírá mezi skeny náhodně. V průměru každým 100 skenům je přiřazeno kladné skóre bezpečnostní události.

```
{
  "first": -1,
  "every": -1,
```

```
    "limit": 500,
    "max_parallel_capture_cnt": 200,
    "random_scan": 100
}
```

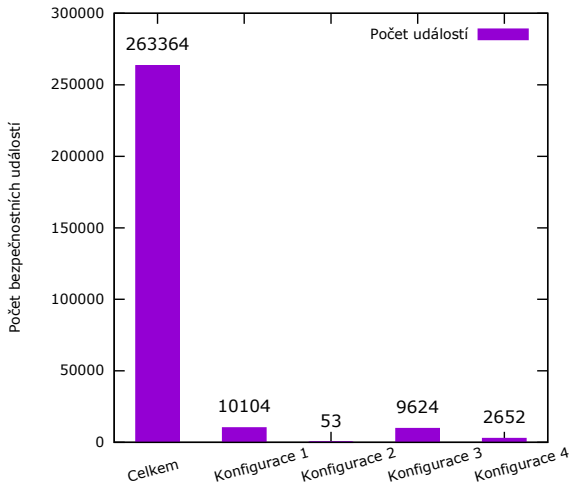
Výpis 5.4: Konfigurace 4

Výsledek tohoto testu je na obrázku 5.3. Za 24 hodin bylo celkem detekováno 263 364 bezpečnostních událostí. Filtr bezpečnostních událostí dokázal pro každou konfiguraci redukovat množství těchto událostí na jednotky procent. Nejhůře dopadla Konfigurace 1, která zaznamenala 10 104 nahlášených událostí za toto časové období, ale i přesto je tato konfigurace dostatečná, protože celkově se snížilo počet událostí na 3,836%.

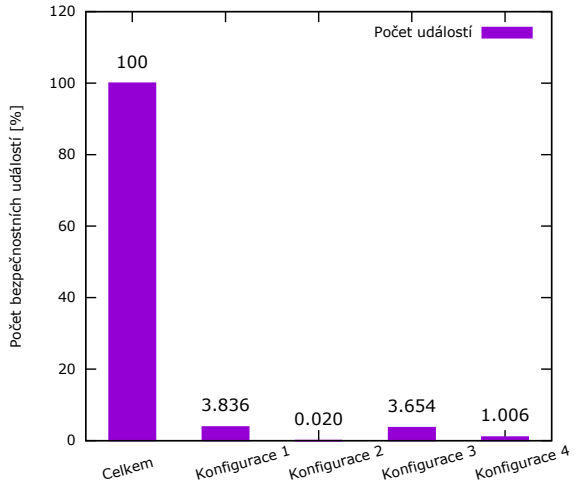
Nejlépe dopadla Konfigurace 2. Tato konfigurace zanedbává veškeré skeny a proto detekovala pouhých 53 bezpečnostních událostí.

Pro porovnání zde budou rozepsány rozdíly v konfiguracích: Konfigurace 1 a Konfigurace 3. Konfigurace 1 a Konfigurace 3 mají velice podobné výsledky celkové redukce. Obě tyto konfigurace dosahují redukce okolo 3,7%. Konfigurace 1 ovšem bere v úvahu i skeny, které se vyskytnou pouze jednou za hodinu, Konfigurace 3 tyto skeny nikdy neoznačí za důležité. Oproti tomu Konfigurace 3 dle obrázku 3.5 zachytí až okolo 55 % IP adres a Konfigurace 1 pouze 35 % IP adres (v tomto čísle nejsou zahrnuty randomizované výběry). Každá konfigurace upřednostňuje jiný typ skenů, které jsou označeny za důležité.

Všechna konfigurační nastavení dosáhla uspokojivých výsledků, kde se počet detekované bezpečnostní událostí redukoval na jednotky procent (Obrázek 5.4). Výchozí nastavení aplikace (Konfigurace 1) označilo v tomto testu za důležité v průměru 7 bezpečnostních události za minutu. Bezpečnostní týmy mají možnost volby při volení konfiguračních parametrů. Tyto parametry ovlivňují výběr bezpečnostních událostí a proto je nutné je přizpůsobit konkrétním potřebám.



Obrázek 5.3: Redukce bezpečnostních událostí



Obrázek 5.4: Redukce bezpečnostních událostí v procentech

Závěr

Na počítačových sítí probíhá denně velké množství bezpečnostních incidentů. Všechny tyto incidenty jsou automaticky detekovány pomocí detekčních systémů a je o nich vytvořena zpráva s podrobnými informacemi. Hlavním problémem je počet těchto detekovaných bezpečnostních událostí. Tento počet je ovšem o mnoho větší, než jsme schopni manuálně zpracovávat.

Cílem práce bylo vytvořit systém pro selekci bezpečnostních událostí. Systém byl pojmenován Filtr bezpečnostních událostí. Přínosy této práce je možno rozdělit do několika bodů.

Prvním krokem byla analýza současného stavu. Byl popsán životní cyklus bezpečnostní události a jednotlivé formáty, které se používají pro reprezentaci. Dále byly popsány jednotlivé výhody a nevýhody jednotlivých reprezentací. Podrobněji byla popsána reprezentace IDEA formátu, tento formát je používán jako hlavní na síti CESNET2.

Druhým krokem bylo provedení analýzy sítě CESNET2. Systém cílí na použití v této síti, proto bylo důležité popsat jednotlivé prvky celé síťové infrastruktury. Jednalo se především o systém NEMEA, který detekuje bezpečnostní události a posílá je do Filtru bezpečnostních událostí, a dále systém Time Machine, sloužící pro zachytávání dat ze síťového provozu. Dále byly analyzovány bezpečnostní události ze sítě.

Třetím krokem byl návrh systému pro redukování bezpečnostních událostí. Navrhovaný algoritmus je určen pro malé sítě, ale i pro vysokorychlostní sítě, které detekují stovky tisíc bezpečnostních událostí každý den. Algoritmus ohodnotí každou bezpečnostní událost a obsahuje několik parametrů, definujících chování algoritmu. Tyto parametry je nutné přizpůsobit konkrétním potřebám. Podle skóre je následně událost vyhodnocena, zda-li je určena k dalšímu zpracování. Při vhodném nastavení parametrů lze dosáhnout redukce bezpečnostních událostí na výsledný počet okolo 3% oproti počtu na vstupu.

Čtvrtým krokem byla implementace systému. Systém byl implementován podle algoritmu popsaného v kapitole Návrh. Je implementován univerzální mapující mechanismus na převod libovolné reprezentace bezpečnostní události

do vnitřní struktury aplikace pomocí konfiguračního souboru. Dále jsou implementovány parametry záchytu závislé na kategorii bezpečnostní události, podle kterých lze každou kategorii zachytávat odlišně.

Pátým krokem bylo testování systému. Zde byly prezentovány výsledky navrhovaného systému. Systém redukuje události na vstupu na jednotky procent a úspěšně tím řeší problém velikého množství bezpečnostních událostí na sítích. Díky tomuto systému se zachytávají daleko přínosnější výsledky, než tomu bylo v předchozí verzi. Tyto výsledky usnadňují práci bezpečnostním týmům spravující tuto síť. Dále jsou zde prezentovány výkonnostní testy, které ukazují dostatečnou výkonnost tohoto systému v reálném nasazení.

Práce je nasazena na páteřní síti CESNET2, kde aktivně slouží pro selekci bezpečnostních událostí.

Literatura

[1] McAfee Associates, I.: Net Losses: Estimating the Global Cost of Cybercrime. 2014. Dostupné z: https://www.mcafee.com/us/resources/reports/rp-economic-impact-cybercrime2-summary.pdf

[2] Národní bezpečnostní úřad: Zákon č. 181/2014 Sb. o kybernetické bezpečnosti a o změně souvisejících zákonů (zákon o kybernetické bezpečnosti). 2014. Dostupné z: https://www.nbu.cz/download/pravni-predpisy/container-nodeid-1347/zkb-181-2014-sb.pdf

[3] CESNET, z.s.p.o.: IDEA: Classifications and enumerations. 2016. Dostupné z: https://idea.cesnet.cz/en/

[4] CESNET, z.s.p.o.: WARDEN. 2017. Dostupné z: https://warden.cesnet.cz/cs/index

[5] CESNET, z.s.p.o.: Classifications and enumerations. 2016. Dostupné z: https://idea.cesnet.cz/en/classifications

[6] Debar, H.; Curry, D.; Feinstein, B.: The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765, RFC Editor, March 2007, http://www.rfc-editor.org/rfc/rfc4765.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc4765.txt

[7] Danyliw, R.; Meijer, J.; Demchenko, Y.: The Incident Object Description Exchange Format. RFC 5070, RFC Editor, December 2007, http://www.rfc-editor.org/rfc/rfc5070.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc5070.txt

[8] Stix. Dostupné z: https://stixproject.github.io/

[9] CESNET, z.s.p.o.: Síť CESNET2. 2017. Dostupné z: https://www.cesnet.cz/sluzby/pripojeni/sit-cesnet2/

[10] Brownlee, N.; Mills, C.; Ruth, G.: Traffic Flow Measurement: Architecture. RFC 2722, RFC Editor, October 1999.

[11] Claise, B.; Trammell, B.; Aitken, P.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. STD 77, RFC Editor, September 2013, http://www.rfc-editor.org/rfc/rfc7011.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc7011.txt

[12] Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954, RFC Editor, October 2004, http://www.rfc-editor.org/rfc/rfc3954.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc3954.txt

[13] CESNET: ipfixcol. https://github.com/CESNET/ipfixcol, 2017.

[14] Tomas Cejka, Vaclav Bartos, Marek Svepes, Zdenek Rosa, Hana Kubatova: NEMEA: A Framework for Network Traffic Analysis. In *12th International Conference on Network and Service Management (CNSM 2016)*, 2016.

[15] CESNET: Nemea. https://github.com/CESNET/Nemea, 2017.

[16] Shafranovich, Y.: Common Format and MIME Type for Comma-Separated Values (CSV) Files. RFC 4180, RFC Editor, October 2005, http://www.rfc-editor.org/rfc/rfc4180.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc4180.txt

[17] Haag, P.: Nfdump. 2010. Dostupné z: http://nfdump.sourceforge.net

[18] CESNET: Nemea. https://github.com/CESNET/Nemea-Framework/tree/master/unirec, 2017.

[19] J. J. Santanna and R. van Rijswijk-Deij and R. Hofstede and A. Sperotto and M. Wierbosch and L. Z. Granville and A. Pras: Booters - An analysis of DDoS-as-a-service attacks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, ISSN 1573-0077, s. 243–251, doi:10.1109/INM.2015.7140298.

[20] Uzelac, A.; Lee, Y.: Voice over IP (VoIP) SIP Peering Use Cases. RFC 6405, RFC Editor, November 2011.

[21] Livingood, J.; Shockey, R.: IANA Registration for an Enumservice Containing Public Switched Telephone Network (PSTN) Signaling Information. RFC 4769, RFC Editor, November 2006.

[22] Cejka, T.; Bartos, V.; Truxa, L.; aj.: *Using Application-Aware Flow Monitoring for SIP Fraud Detection*. Cham: Springer International Publishing, 2015, ISBN 978-3-319-20034-7, s. 87–99, doi:10.1007/978-3-319-20034-7_10. Dostupné z: http://dx.doi.org/10.1007/978-3-319-20034-7_10

[23] Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251, RFC Editor, January 2006, http://www.rfc-editor.org/rfc/rfc4251.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc4251.txt

[24] Postel, J.; Reynolds, J.: Telnet Protocol Specification. STD 8, RFC Editor, May 1983, http://www.rfc-editor.org/rfc/rfc854.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc854.txt

[25] Libpcap. 2017. Dostupné z: http://www.tcpdump.org/

[26] CESNET, z.s.p.o.: CESNET-CERTS. 2017. Dostupné z: https://csirt.cesnet.cz/cs/index

[27] CSIRT-MU: CSIRT-MU. 2017. Dostupné z: https://csirt.muni.cz/

[28] RabbitMQ: RabbitMQ. 2017. Dostupné z: https://www.rabbitmq.com/

[29] stedolan: jq. https://github.com/stedolan/jq, 2017.

[30] AMQP: Advanced Message Queuing Protocol. 2017. Dostupné z: http://www.amqp.org/

[31] pika: pika. https://github.com/pika/pika, 2017.

[32] mwilliamson: jq.py. https://github.com/mwilliamson/jq.py, 2017.

[33] vpelletier: pprofile. https://github.com/vpelletier/pprofile, 2017.

Příloha A

Seznam použitých zkratek

JSON JavaScript Object Notation

XML Extensible markup language

DDoS Distributed Denial of Service

DoS Denial of Service

IDEA Intrusion Detection Extensible Alert

IDMEF Intrusion Detection Message Exchange Format

IODEF Incident Object Description Exchange Format

STIX Structured Threat Information eXpression

IPFIX Internet Protocol Flow Information eXport

IP Internet Protocol

CSV Comma-separated values

UniRec Unified Record

NEMEA Network Measurements Analysis

PSTN Public Switched Telephone Network

SIP Session Initiation Protocol

SSH Secure Shell

TELNET Teletype Network

DNS Domain Name System

Příloha B

Nápovědy programů

B.1 filter

```
-h, --help     show this help message and exit
-f, --folder   All input json files are taken from ../jsons/
               folder if is not specified with -fp parameter
               (default: False)
-fp FOLDER_PATH, --folder_path FOLDER_PATH
               Folder path from where are json files
               taken (default: ../jsons/)
-RMQ, --RabbitMQ       All input json files are
                       taken from rabbitmq server
                       (default: False)
-RMQhostname RABBITMQ_HOSTNAME,
      --RabbitMQ_hostname RABBITMQ_HOSTNAME
      RabbitMQ hostname (default: localhost)
-RMQport RABBITMQ_PORT, --RabbitMQ_port RABBITMQ_PORT
      RabbitMQ port (default: 5672)
-RMQusername RABBITMQ_USERNAME,
      --RabbitMQ_username RABBITMQ_USERNAME
      RabbitMQ username (default: guest)
-RMQpassword RABBITMQ_PASSWORD,
       --RabbitMQ_password RABBITMQ_PASSWORD
      RabbitMQ password (default: guest)
-tmm, --time_machine_manager
      All output is sent to time machine manager.
      (default: False)
-tmm_hostname TIME_MACHINE_MANAGER_HOSTNAME,
```

```
--time_machine_manager_hostname TIME_MACHINE_MANAGER_HOSTNAME
        Time machine manager hostname. (default: localhost)
-tmm_port TIME_MACHINE_MANAGER_PORT,
--time_machine_manager_port TIME_MACHINE_MANAGER_PORT
        Time machine manager port. (default: 37564)
-no_tmm, --no_time_machine_manager
        Time machine manager is disable, all filter
        will be printed only to STDOUT (default: False)
  -cfg_mapping CFG_MAPPING
        Path to mapping config (default: ../config/mapping)
  -cfg CFG
        Path to config (default: ../config/static_prices.json)
  -cfg_algorithm_parameters CFG_ALGORITHM_PARAMETERS
        Path to scan algorithm parameters
        (default: ../config/algorithm_parameters.json)
  -probability_db_file PROBABILITY_DB_FILE
        Path to database file for probability
        (default: ../config/probability_db)
```

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `filter.tar.gz` ..... balík obsahující zdrojové kódy
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis` ..... složka s LaTeX soubory