VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MODUL NA ZPRACOVÁNÍ ZPRÁV ZE SYSTÉMU MIKROTIK ROUTEROS PRO IBM QRADAR

MIKROTIK ROUTEROS MODULE FOR IBM QRADAR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **VÁCLAV SYSEL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK HRANICKÝ, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

147209

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Sysel Václav**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Modul na zpracování zpráv ze systému Mikrotik RouterOS pro IBM QRadar**
Kategorie:	Počítačové sítě
Akademický rok:	2022/23

Zadání:

1. Seznamte se se systémem IBM QRadar SIEM.
2. Seznamte se s funkcemi operačního systému MikroTik RouterOS z pohledu síťového administrátora. Zaměřte se na formát zpráv typu Syslog, které systém generuje.
3. Po domluvě s vedoucím navrhněte modul typu DSM pro zpracování zpráv ze směrovačů MikroTik. Zaměřte se na identifikaci jednotlivých typů zpráv a informace, které jsou zajímavé z hlediska detekce bezpečnostních incidentů a monitoringu provozu na síti.
4. Navrhněte také jednoduché rozšíření typu Add-on pro vizualizaci informací získaných z vašeho DSM.
5. Navržené subsystémy implementujte.
6. Demonstrujte použitelnost vašeho řešení při zpracování vybraných typů síťového provozu.
7. Zhodnoťte dosažené výsledky.

Literatura:

- Chakrabarty, B., Patil, S. R., Shingornikar, S., Kothekar, A., Mujumdar, P., Raut, S., & Ukirde, D. (2021). *Securing Data on Threat Detection by Using IBM Spectrum Scale and IBM QRadar: An Enhanced Cyber Resiliency Solution*. IBM Redbooks.
- S. Gupta, B. S. Chaudhari and B. Chakrabarty, "Vulnerable network analysis using war driving and security intelligence," *2016 International Conference on Inventive Computation Technologies (ICICT)*, 2016, pp. 1-5, doi: 10.1109/INVENTIVE.2016.7830165.
- SIA Mikrotīkls. MikroTik router OS documentation, v. 28. 2022 [online] URL https://help.mikrotik.com/docs/display/ROS/RouterOS

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 4

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Hranický Radek, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Cílem této práce je návrh a implementace rozšíření pro systém IBM QRadar. Jedná se o bezpečnostní systém, který za pomoci sběru informací z počítačové sítě detekuje kybernetické útoky. Smyslem rozšíření je umožnit sběr, zpracování a vizualizaci informací z operačního systému MikroTik RouterOS. Navržený software tak poskytuje administrátorům sítí s prvky MikroTik lepší povědomí o dění v chráněné síti a ucelený přehled o bezpečnostní situaci.

Abstract

The goal of this work is to design and implement an extension for the IBM QRadar system. It is a security system that detects cyber attacks by collecting information from the computer network. The purpose of the extension is to enable the collection, processing and visualization of information from MikroTik RouterOS operating system. The designed software provides administrators of networks with MikroTik elements with a better awareness of what is happening in the protected network and a comprehensive overview of the security situation.

Klíčová slova

QRadar, IBM, RouterOS, MikroTik, modul DSM, SIEM, bezpečnost

Keywords

QRadar, IBM, RouterOS, MikroTik, DSM, SIEM, security

Citace

SYSEL, Václav. *Modul na zpracování zpráv ze systému Mikrotik RouterOS pro IBM QRadar*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Hranický, Ph.D.

Modul na zpracování zpráv ze systému Mikrotik RouterOS pro IBM QRadar

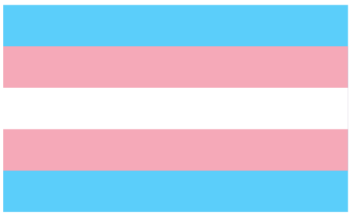
Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením Ing. Radka Hranického, Ph.D. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

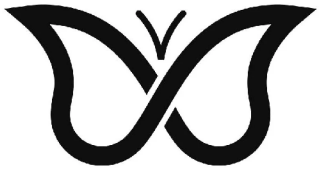
.....
Václav Sysel
8. května 2023

Poděkování

Ráda bych poděkovala Ing. Radkovi Hranickému, Ph.D., za vedení mé práce, za poskytnutí materiálů, pomoc s organizací a podporu. Dále bych ráda poděkovala své drahé polovičce Marcy nejen za gramatickou korekturu. Mým kamarádům, jmenovitě Kyo, Meyo, Emye, Vezzovi a dalším, za pomoc s udržováním příčetnosti. Bez jejich pomoci bych se dávno vydala ve spáry smrti a tato práce by nedošla zdárného konce. Navíc bych ráda poděkovala své psycholožce a psychologovi za psychickou pomoc a Poradenskému centru Alfons za psychickou a organizační podporu.



Obrázek 1: Transgender Pride vlajka



Obrázek 2: ADHD motýlek

Obsah

Kapitola 1

Úvod

Zabezpečení musí být tím lepší, čím více lidem musí odolat. Například dveře od domu musí zastavit pouze sousedy, případně vzácné návštěvníky. Samo umístění je v podstatě zabezpečením. Na internetu je však třeba se ochránit před značně větším počtem lidí. Toto je jeden z důvodů, proč je digitální zabezpečení tak obrovským tématem a proč mu musí být věnováno mnohem více pozornosti. Dostupnost, kterou internet poskytuje, je výhodou i nebezpečím. Vede se neustálý závod ve zbrojení, způsobů útoků je bezpočet a ochránit se před nimi nesnadným úkolem. Jednou z možností, jak se bránit, je sledovat podezřelé chování na síti. Proto vznikly systémy SIEM. Ty sbírají informace o síti z více zdrojů, uskladňují je na jednom místě a díky velkému množství poznatků mohou provést dedukce, které by jinak byly nemožné. QRadar od společnosti IBM je právě takovým systémem.

Z podstaty věci však může být QRadar pouze tak účinný, z kolika zdrojů může informace sbírat, čím více, tím lépe. Bohužel mnoho zařízení stále není podporováno. Smyslem této práce je tedy přispět troškou do mlýna a rozšířit sadu podporovaných zařízení o další, konkrétně od společností MikroTik. Zařízení MikroTik umožňují velikou škálu nastavení a jsou zároveň relativně dostupná, což z nich dělá ideálního kandidáta. Zařízení MikroTik v době psaní práce nebyla podporována ani ze strany QRadar, ani MikroTik, ani z třetí strany.

Shrnuto, cíl této práce je dvojí. Vytvořit modul, který umožní podporu těchto zařízení, a vytvořit doplněk, který bude moci zobrazit informace získané tímto způsobem. Tímto by práce měla přispět k většímu bezpečí na internetové síti.

1.1 Přínos práce

V práci se mi podařilo prozkoumat strukturu zpráv posílaných ze zařízení MikroTik RouterBOARD 750. Dle této struktury jsem navrhla regulární výrazy pro zachytávání informací z těchto zpráv a dále jsem implementovala modul DSM, který umí zpracovávat tyto zprávy pro použití v systému QRadar. Modul DSM je ve stavu, kdy je možné ho nasadit v reálném prostředí. V práci jsem taktéž navrhla rozšíření typu Add-on, které vizualizuje vytvořené události. Toto rozšíření jsem dále implementovala. V současném stavu je funkční a je vhodné k použití jako základ pro další vývoj.

1.2 Související řešení

Nejdříve bych ráda zmínila pár řešení, která se snaží zpracovávat zprávy syslog samy o sobě. Na poli zpracování zpráv je zajímavým řešením tzv. matematická analýza provedená na půdě Masarykovy Univerzity. Dříve analyzovali zprávy sémanticky, tato metoda však nemohla postihnout možnosti, kdy například dojde k změně protokolu (z IPv4 na IPv6), či změně struktury zpráv. Ač se nepodařilo vytvořit matematický model, který by uspokojivě určil stabilitu systému, udělali mnoho pokroků na tomto poli [14].

V jiném souvisejícím řešení implementovali dynamickou analýzu generických zpráv, za účelem zjištění podezřelého chování. Toto řešení je založené na skrytých Markovových modelech [15].

Následují související řešení, která implementují moduly DSM, či souvisí s implementací. Například v tomto souvisejícím řešení byl využit QRadar k detekci zranitelných bezdrátových sítí. V práci „Vulnerable network analysis using war driving and security intelligence“ za pomoci programu Vistumbler monitorovali bezdrátové sítě během jízdy v autě. Tato data v reálném čase zpracovávali pomocí systému QRadar a byli schopni dosáhnout analýzy sítí během jízdy v reálném čase [4]. V práci „Next-Generation Firewall, Deep Learning Endpoint Protection and Intelligent SIEM Integration“ integrovali pokročilý firewall „Sophos Firewall“. Záznamy z tohoto firewall a síťových prvků poslali do systému QRadar. Výsledkem bylo snížení trvání detekce incidentu a snížení trvání odezvy na incident [1].

Obecně při zpracování záznamů syslog systém QRadar je problém velké množství zpráv, kde každá sama o sobě má malý význam. Mnoho routerů posílý zprávy syslog, které jsou vhodné spíše k ladění než k diagnostice sítě. V jednom souvisejícím řešení se na toto dilema soustředili a vyvinuly systém „SyslogDigest“, který zpracovává mnoho drobných zpráv s neúplnými informacemi do malého množství „síťových“ zpráv s kompletními informacemi [13].

V době psaní práce existují řešení pro zařízení od jiných firem. Například routery a switche firmy Cisco jsou podporovány modulem DSM „Cisco IOS“, který je oficiálně podporován i ze strany firmy IBM. Implementovány jsou všechny zprávy.[1] Switche firmy Juniper Networks jsou podporovány modulem „Juniper Networks EX Series Ethernet Switch“. Implementovány jsou taktéž všechny zprávy.[2] Od firmy Huawei jsou podporovány některé routery série AR a S. V tomto případě jsou však implementovány pouze události protokolu IPv4.[3] Ač existuje podpora pro alternativní zařízení, pro zařízení od firmy MikroTik v době psaní práce neexistovala žádná podpora ani ze třetích stran.

1.3 Struktura práce

Práce se skládá z 8 kapitol. Úvod a Závěr jsou obsaženy v kapitolách 1 a 8. Tyto dvě kapitoly shrnují podstatu práce. V kapitole 2 je představen systém IBM QRadar. V kapitole 3 je představena firma MikroTik a systém RouterOS, podrobněji jsou v sekci 3.3 popsány zprávy, které tento systém produkuje. V kapitole 4 je popsán návrh integrace těchto zpráv do systému QRadar pomocí modulu DSM a kapitola 5 je věnována návrhu aplikace pro vizualizaci dat získaných touto integrací. Samotná implementace je pak popsána v kapitole 6. V kapitole 7 je pak popsáno, jak bylo výsledné řešení ověřeno.

[1] https://www.ibm.com/docs/en/dsm?topic=cisco-ios [cit. 2023-05-02]

[2] https://www.ibm.com/docs/en/dsm?topic=networks-juniper-ex-series-ethernet-switch [cit. 2023-05-02]

[3] https://www.ibm.com/docs/en/dsm?topic=configuration-huawei[cit. 2023-05-02]

Kapitola 2

IBM QRadar

Tato kapitola se věnuje systému IBM Security QRadar SIEM (dále IBM QRadar či QRadar). Nejdříve je zběžně představena firma IBM, co je to systém SIEM, základní struktura QRadar a popis zpracování záznamů o událostech.

2.1 IBM

Firma IBM je dominantním hráčem na poli informačních technologií. Tato americká mezinárodní firma založená roku 1911 za své existence drasticky přispěla k vývoji informačních technologií. K jejich úspěchům patří mimo jiné návrh programovacího jazyka Fortran, vynález karty s magnetickým proužkem, vynález diskety, vynález čárového kódu UPC, poskytnutí počítačů a softwaru pro misi Apollo. V neposlední řadě je potřeba zmínit IBM PC, s jehož nástupem došlo k masivnímu rozšíření osobních počítačů a jehož design se stal prakticky standardem.

V současné době firma poskytuje infrastrukturu, software a poradenské služby v oblasti IT. Jednou ze současných oblastí zájmu je automatizace repetitivních všedních úkolů pomocí umělé inteligence [5].

2.2 SIEM

Pro vysvětlení termínu SIEM je nejdříve vhodné uvést termíny IDS a IPS. IDS (Intrusion Detection System) je software či hardware, který monitoruje síťový provoz za cílem detekce nežádoucí aktivity, jako je například ilegální nebo škodlivá činnost, či porušení zásad zabezpečení. IPS (Intrusion Prevention System) je software či hardware, který taktéž detekuje tuto nežádoucí aktivitu a zároveň provádí akce za účelem jejího zastavení, například zahozením packetů [3]. SIEM (Security Information and Event Management) sám o sobě žádný sběr neprovádí, informace získává od existujících IDS, IPS a jiných zdrojů.

V kyberbezpečnosti se za SIEM (Security Information and Event Management) považuje série technologií, která je zodpovědná za provádění analýzy, zmírnění hrozeb a zaznamenávání bezpečnostních událostí v dané síti. SIEM poskytuje obecný pohled na veškerou technickou infrastrukturu, včetně specifických dat ohledně bezpečnostních událostí a zmírnění jakéhokoliv vektoru bezpečnostní hrozby, které jsou v prostředí nalezeny.

SIEM obsahují několik funkcí, jakými jsou například spravování bezpečnostních informací (Security Information Management) a spravování bezpečnostních událostí (Security Event Management), které jsou zkombinovány do jednoho řešení. Pro lepší pochopení SIEM

je vhodné si představit řešení, které sbírá data z bezpečnostních zdrojů za účelem analýzy a korelace. Následně provádí akce na základě možných hrozeb. Správa SIEM poskytuje různé funkce v následujících odvětvích:

1. sběr událostí a záznamů,
2. korelace pravidel,
3. správa zdrojů záznamů,
4. přizpůsobivost,
5. normalizace dat a jejich uložení,
6. reporty a dodržení předpisů.

Toto řešení se snaží řešit scénáře, ve kterých lidé nemohou analyzovat pokročilé hrozby použitím běžných nástrojů pro pozorování. Děje se tak za pomoci podnikové technické infrastruktury a sjednocením všech prvků. Těmito prvky jsou většinou agenti v hierarchickém modelu, kteří sbírají události z koncových zařízení, serverů a síťového vybavení [2].

2.3 Obecné informace

Systém QRadar byl původně vyvinut firmou Q1 Labs, údajně na začátku roku 2000. Původně se jednalo o systém odhalení průniku, který se vyvinul do systému prevence průniku, následně se vyvinul do systému SIEM. V roce 2011 firma IBM odkoupila Q1 Labs. QRadar se přejmenoval na IBM Security QRadar SIEM a firma Q1 Labs se stala součástí IBM.[1]

QRadar je poskytován jako služba, virtuální stroj a fyzické zařízení.[2] QRadar poskytuje interoperabilitu s třetími stranami, takže mnoho řešení může být integrováno, což dělá tento produkt škálovatelným a více robustním. IBM QRadar je v současnosti jedním z nejvíce populárních SIEM řešení na trhu. QRadar pomáhá rychle odhalit existující a potenciální rizika skrz své schopnosti pokročilé analýzy. Poskytuje mnoho funkcí jako centralizovaný přehled, flexibilní nasazení, automatickou inteligenci, počítačové učení, proaktivní lovení hrozeb a další.

QRadar sbírá události z různých prostředků, které jsou přítomny, a sbírá surové packety dat ze sítě ke korelaci. Navíc umožňuje obnovení relace pro forenzní analýzu [2].

2.4 Základní architektura

Jádro architektury systému QRadar se dá reprezentovat třemi vrstvami. V první vrstvě dochází ke sběru informací, v druhé k jejich zpracování a testování, ve třetí vrstvě jsou informace zpřístupněny uživateli.

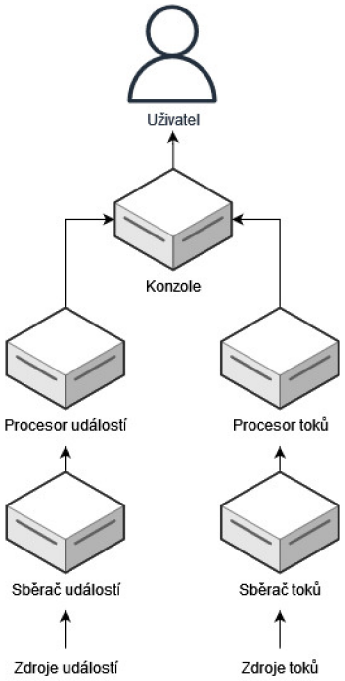
QRadar je založen na sběru tzv. událostí a toků. V tomto případě je událostí myšlena jakákoliv informace, která reprezentuje událost, která nastala v pozorovaném prostředí. Může se jednat o přihlášení, e-mail, připojení VPN, odmítnutí bránou firewall, připojení proxy a jakoukoliv jinou událost hodnou zaznamenání. Tokem je myšlena informace o síťovém provozu, nebo informace o komunikaci mezi dvěma hosty na síti. QRadar normalizuje surová data toku na IP adresy, porty, počty bytů, počty packetů a ostatní informace. Tyto

[1] https://www.comparitech.com/net-admin/qradar-siem-review-alternatives/ [cit. 2022-12-17]
[2] https://www.ibm.com/products/qradar-siem/pricing [cit. 2022-12-17]

informace se nazývají záznamy flow a reprezentují síťovou komunikaci. V první vrstvě jsou tato data sbírána. Tento sběr probíhá pomocí tzv. sběrače událostí a sběrače toků. Může se jednat o samostatné zařízení, či součást zařízení „vše v jednom“.

V druhé vrstvě jsou data událostí a data toků zpracována subsystémem Custom Rule Engine, který generuje incidenty a výstrahy. Incidentem je myšlena nežádoucí situace v síti, například krádež dat, porušení GDPR a jiné. V této vrstvě jsou navíc data událostí a data toků uložena. Vrstva může být modifikována a rozšířena o další funkce, pro podstatu této práce však není nutné tyto informace znát. Zpracování probíhá v tzv. procesoru událostí a procesoru toků. Může se jednat o samostatné zařízení, či součást zařízení „vše v jednom“.

Ve třetí vrchní vrstvě jsou data zpřístupněna uživatelům k vyhledávání, analýze, reportování a zkoumání incidentů a výstrah. Taktéž je možné spravovat celý QRadar. Tyto akce jsou proveditelné z tzv. konzole, která může být samostatným zařízením, či součástí zařízení „vše v jednom“ [8]. Struktura je připodobněna na obrázku 2.1.



Obrázek 2.1: Vyobrazení hlavních částí systému QRadar

Tato práce se zabývá zpracováním zpráv typu Syslog, což spadá do událostí. Následovat tedy bude podrobnější pohled na zpracování událostí.

2.5 Zpracování událostí

QRadar přijímá záznamy událostí z různých zdrojů událostí, které jsou na síti. Za zdroj událostí se považuje zdroj dat, jako například firewall nebo systém prevence průniku (anglicky IPS), které vytváří záznamy o událostech.

Záznamy jsou přijímány použitím protokolů, jako jsou například Syslog, syslog-tcp a SNMP. QRadar taktéž může navázat odchozí připojení, aby vyzvednul záznamy, a to pomocí protokolů jako SCP, SFTP, FTP, JDBC, Check Point OPSEC a SMB/CIFS.

Když jsou záznamy posbírány, či získány ze zdrojů, jsou nejdříve umístěny do vstupních front. Velikosti front se liší podle protokolu a metody, které jsou použity, a z těchto front jsou záznamy zpracovány a normalizovány. V tuto chvíli se vynucuje licence, a to v podobě událostí za sekundu (EPS). Pokud je počet událostí za sekundu překročen, události zůstávají ve frontách. Pokud jsou fronty zaplněny, události jsou zahozeny. Pokud je limit splněn, zprávy jsou dále normalizovány. Proces normalizace spočívá v převodu surových dat do formátu, který má pole, se kterým může QRadar pracovat. QRadar rozeznává známé zdroje záznamů podle zdrojové IP adresy nebo podle jména zařízení, které je uvedeno v hlavičce.

Zpracování probíhá za pomoci kódového modulu DSM (device support module). Každý typ zdroje má svůj vlastní modul [7].

QRadar zpracovává a spojuje události ze známých zdrojů událostí. Události z nových nebo neznámých zdrojů, které nebyly detekovány v minulosti, jsou přeposlány subsystému analýzy provozu (provádí automatickou detekci).

Když je nový zdroj událostí objeven, do konzole je zaslána konfigurační žádost o přidání nového zdroje. Pokud je automatická detekce vypnuta, nebo je překročen licenční limit, nové zdroje nejsou přidány.

Normalizované události jsou zpracovány v procesoru událostí. Události jsou zpracovány za pomoci subsystému Custom Rule Engine (dále CRE). CRE porovnává data za pomoci definovaných pravidel, udržuje informace o systémech, které se účastnily v minulosti incidentů. Když události splní pravidlo, vytvoří se oznámení o incidentu a je zasláno magistrátu na konzoli.

Procesor taktéž přeposílá data událostí konzoli, kde je uživatel může ve skutečném čase prohlížet. Události jsou uloženy do databáze Ariel, která je založena na časové řadě. Data jsou uložena na procesoru, který danou událost zpracoval.

Magistrát je komponent na konzoli, který vytváří a spravuje incidenty. Když obdrží upozornění od procesoru, odpoví různými akcemi, například upozorněním, zasláním e-mailu, zprávou Syslog, SNMP, vytvořením nové události nebo vytvořením incidentu. Incidenty jsou uloženy do databáze PostgreSQL [8].

Modelová situace postupu zpracování událostí je ukázána na diagramu 2.2. Na diagramu jsou tři zdroje událostí, v této situaci Firewall, VPN a Linux. Ty zasílají záznamy o událostech sběrači událostí. Záznamy jsou umístěny do vstupních front. VPN a Linux používají stejný protokol a jejich záznamy jsou tedy přiřazeny do stejné fronty. Firewall používá rozdílný protokol, záznam je tedy přiřazen do jiné fronty. Po ověření licence jsou záznamy zpracovány pomocí DSM modulů. Firewall, VPN a Linux jsou rozdílné typy zdrojů událostí, tedy každý záznam je zpracován jiným modulem DSM. Výstupem zpracování jsou normalizované události, které jsou poslány procesoru událostí.

S událostmi jsou v procesoru udělány tři akce. První je uložení do databáze Ariel. Z této databáze pak mohou být události vyhledávány uživatelem. Další akcí je přeposlání do konzole, aby si uživatel mohl zobrazit události v reálném čase. Poslední akcí je zpracování pomocí CRE. Události jsou porovnány s pravidly. V této situaci bylo jedno z pravidel splněno a došlo k vytvoření incidentu, který je přeposlán konzoli, konkrétně komponentě magistrát.

V konzoli si uživatel může zobrazit události v reálném čase, které jsou přeposílány z procesoru událostí. Může i vyhledávat události, což zasílá žádost do databáze Ariel, která je na procesoru. Komponentou konzole magistrát je zpracován příchozí incident, na který je reagováno podle pravidel. Incident je taktéž uložen do databáze PostgreSQL. Tím jsou události plně zpracovány a prošly celým systémem. Jak je vidno, část zpracování, která je závislá

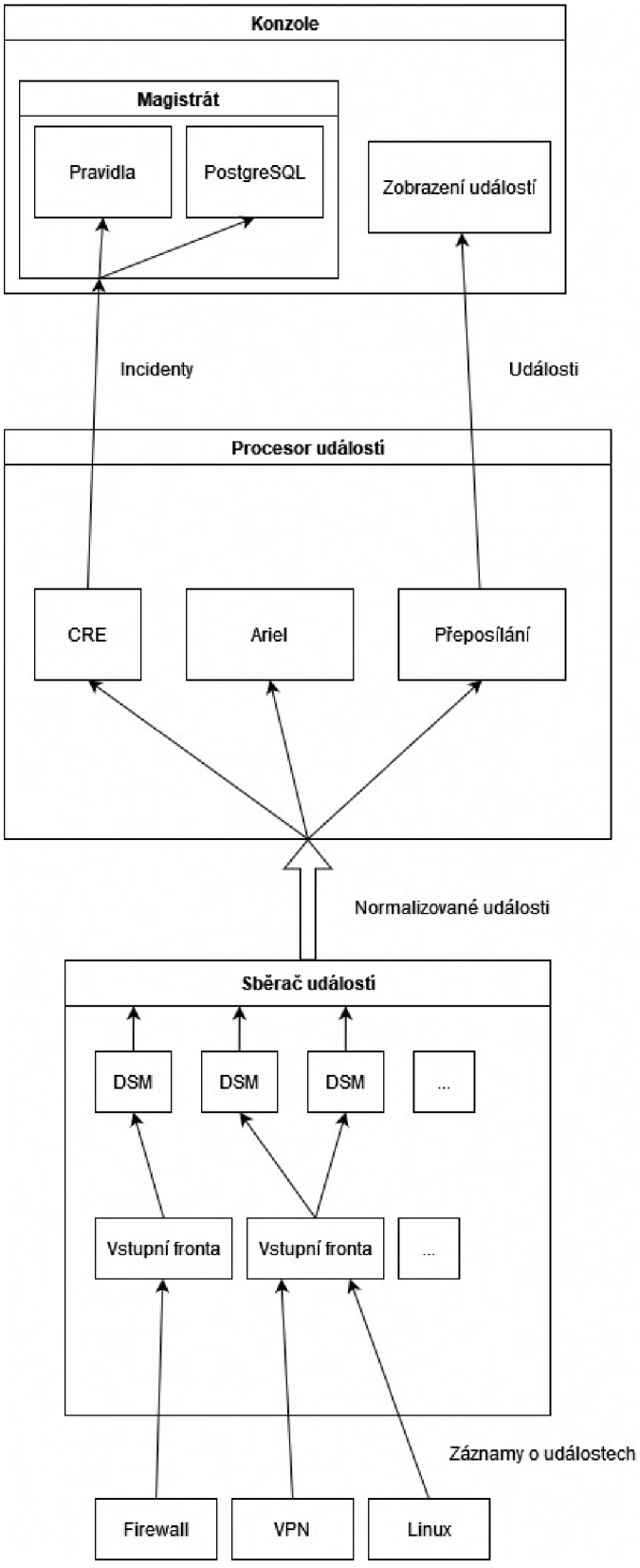
na typu zdroje událostí a která systému QRadar chybí k úspěšnému zpracování záznamů ze systému RouterOS, je modul DSM. V sekci 2.6 bude tento modul blíže vysvětlen.

2.6 Modul DSM

DSM (Device Support Module) je kódový modul, který zpracovává přijaté události z více zdrojů událostí a převádí je do standardního formátu, který může být zobrazen jako výstup. Každý druh zdroje událostí má svůj odpovídající DSM. Například IBM Fiberlink MaaS360 DSM zpracovává a normalizuje události ze zdrojů událostí IBM Fiberlink MaaS360 [7]. Vstupem modulu je záznam o události a výstupem je normalizovaná událost, která má určité vlastnosti obsahující informace ze záznamu. Například událost o připojení nového zařízení bude mít vlastnost „MAC adresa“, která bude obsahovat adresu připojeného zařízení.

DSM může zpracovávat záznamy několika způsoby. Umí číst některé formáty používané pro výměnu dat, konkrétně JSON, LEEF, CEF a XML. Na data lze nahlížet jako na seznam párů „jméno hodnota“. Na záznam můžeme nahlížet jako na seznam hodnot rozdělených oddělovačem. V poslední řadě lze využít regulárních výrazů a zachytit skupinu v rámci výrazu. V jednom modulu DSM lze využít několik těchto nástrojů pro zpracování záznamů [7].

Zpracované informace se ukládají do vlastností událostí. Sytém QRadar poskytuje možné vlastnosti událostí, lze však definovat i nové vlastnosti. Předdefinované vlastnosti v sobě mají kontrolu typu, tzv. před uložením vlastnosti se ověří, že je v očekávaném formátu (například při pokusu uložit „jksflkaj“ do IP adresy se informace neuloží, při pokusu uložit „192.168.1.53“ se adresa uloží). Pro každou vlastnost lze nastavit více způsobů, jak se vlastnost má zachytit, tyto varianty jsou seřazené a bude použita první, která přinese výsledek.



Obrázek 2.2: Diagram zpracování událostí

Kapitola 3

MikroTik RouterOS

V této kapitole je nejdříve krátce představena firma MikroTik, co je operační systém RouterOS a dále je popsán formát zpráv Syslog, který tento systém produkuje.

3.1 MikroTik

MikroTik je litevská firma, která byla založena roku 1996. Momentálně má své sídlo v Rize. Již od svého založení se zabývala vývojem routerů a bezdrátových systémů pro poskytovatele internetových služeb. V roce 1997 vytvořila systém RouterOS a od roku 2002 vytváří vlastní hardware pod jménem RouterBOARD. V současné době firma MikroTik poskytuje hardware a software pro připojení k internetu ve většině zemí světa [10]. Mezi zákazníky firmy MikroTik patří významné společnosti jako například CERN, NASA, HP, Motorola, Mitsubishi či Vodafone.[1]

3.2 Operační systém RouterOS

RouterOS je samostatný routerový operační systém založený na linuxovém jádru. Je předinstalovaný na zařízeních značky MikroTik, ale je možné ho nainstalovat i na vlastní zařízení či na virtuální stroj.

Systém poskytuje mimo jiné směrování, například pomocí protokolů RIP v1, RIP v2, RIPng, OSPFv2, OSPFv3, BGP v4, BGP a BFD, dále MPLS, Firewall, VPN, bezdrátovou konektivitu, DHCP, Hotspot, QoS, proxy a jiné. Systém lze konfigurovat několika způsoby, a to pomocí programu WinBox pro operační systém Windows, přes webový prohlížeč, přes aplikaci na chytrém mobilu, přístup pomocí MAC, API a pomocí příkazové řádky, která je přístupná přes lokální terminál, sériovou konzoli, telnet a ssh [11].

3.3 Analýzy formátu zpráv

RouterOS je schopen zaznamenávat různé systémové události a informace o stavu. Záznamy mohou být uloženy do paměti, na disk, zaslány e-mailem nebo na Syslog server.

Během psaní práce byly informace o formátu těchto zpráv nedostačující. Kontaktovala jsem zastoupení společnosti MikroTik a požádala jsem o podrobnější dokumentaci či příklady zasílaných zpráv. Dle jejich slov momentálně dokumentaci či více příkladů nemají a

[1] https://mikrotik.com/customers [cit. 2022-12-17]

v současné chvíli je neplánují přidat. Musela jsem tedy udělat vlastní analýzu. V odpovědi bylo poukázáno na virtualizovanou verzi jejich systému, kterou je možné stáhnout zdarma. Toto naštěstí nebylo potřeba a vedoucí práce mi poskytnul zařízení MikroTik RB750. Zařízení jsem aktualizovala na verzi 6.48.6 a zprávy odchytávala pomocí softwarových nástrojů Rsyslog a Wireshark. Zachycené zprávy si bude možné prohlédnout v přiloženém médiu po dokončení práce.

Zprávy využívají protokolu Syslog [12], v základním nastavení však nedodržují formát těchto zpráv. Zprávy typu Syslog dle standardu RFC3164 [9] využívají tento formát:

```
příklad:  <23> Oct 10 14:22:34 router info: device connected
          on interface eth1
formát:   <priorita> <časová značka> <hostname> <tag>: <zpráva>
```

Zachycené zprávy se skládají pouze ze dvou částí. Druhou částí je samotná zpráva, tato část si nevyžaduje žádného další vysvětlení, a proto je zmíněna nejdříve. První částí jsou tzv. témata, která mohou být úrovně priorit zpráv nebo části systémů, kterých se zpráva týká, například `dns`, `ssh`, `system` a jiné. Pokud se zpráva týká více témat, jsou zobrazena všechna, i když nebyla zvolena v nastavení. (Následující příklad by se zobrazil, i kdyby bylo zvoleno pouze téma info k záznamu.)

```
příklad:  system,info log action changed by admin
formát:   <témata> <zpráva>
```

V nastavení lze nastavit možnost `bsd-syslog`, která dle manuálu dodržuje standard RFC 3164. Zprávy pak vypadají takto:

```
příklad:  DAEMON.INFO: Dec 10 13:45:04 MikroTik log action changed by admin
formát:   <zařízení>.<priorita>: <časová značka> <hostname> <zpráva>
```

Tento formát je již blíže standardu RFC 3164 a například Rsyslog, stejně jako jiný software, ho již rozeznává. Bohužel zařízení a priorita je nastavitelná uživatelem, na informace v těchto částech se proto není možno spoléhat, protože mohou být jakékoliv.

Z tohoto důvodu je v práci dále pracováno se základním formátem zpráv. Ač není standardní, zprávy vytvořené tímto nastavením jsou vždy stejné a vyžadují méně úprav v nastavení. V případě využití `bsd-syslog` nastavení by bylo nutné zveřejnit pravidla, jakými byly tyto zprávy zachytávány, u základního formátu stačí pouze uvést, jaké téma tuto zprávu vytvoří. U všech zpráv se navíc předpokládá, že není nastaven žádný prefix. Zprávy jsou tedy zachytávány v základním formátu bez prefixu.

Kvůli nestandardnosti zpráv má každý systém tendenci dát před samotnou zachycenou zprávou doplňující informace, zde je příklad, jak je zpráva zachycena pomocí Wireshark (jak byla odeslána), následně pomocí Rsyslog a IBM QRadar.

```
Wireshark:  system,info log action changed by admin
   Rsyslog: Dec 1 16:19:15 router.lan system,info log rule changed by admin
    Qradar: <182>Dec 08 12:48:54 192.168.88.1 system,info user user1
            removed by admin
```

Rsyslog předepíná samotné zachycené zprávě čas zachycení a jméno zařízení, Qradar předepíná prioritu (která se zdá být neměnná), čas a IP adresu. Během implementace však QRadar zachytával zprávy ve stejném formátu jako Wireshark, návrh modulu DSM byl

proto upraven, aby vyhovoval tomuto formátu. Pro stručnost bude dále v práci zobrazována pouze zkrácená verze, neboli jak zprávu zachytává Wireshark. Je však nutné mít toto chování na paměti.

3.4 Metodika tvorby a záchytu zpráv

Pro generaci dat jsem využila MikroTik RB750 s verzí RouterOS 6.48.6. K portu WAN jsem připojila Raspberry Pi 3, které zprostředkovávalo připojení k internetu. K routeru samotnému byl připojen stolní počítač, na kterém běžely dva virtuální stroje. Na prvním virtuálním stroji běžela linuxová distribuce Debian a server Rsyslog, tento stroj sloužil k ukládání záznamů. Na druhém virtuálním stroji běžela linuxová distribuce Kali Linux, která je navržena pro penetrační testování a forenzní analýzu. K routeru byl navíc připojen laptop.

Záznam probíhal 1., 6. a 7. prosince, největší část však proběhla 6. a 7. prosince. Během záznamu jsem na routeru zapnula zaznamenávání všech témat s výjimkou témat `raw` a `packet`. Tato dvě témata přeposílají síťové toky procházející systémem rozdělené do několika záznamů, což vytváří neúměrně velký počet zpráv. Zpracování tohoto druhu výstupu je možné, nicméně toto není běžný způsob, jakým QRadar zpracovává události. QRadar očekává jednu zprávu, jednu událost. Tento způsob by byl pro QRadar náročný, zejména na licenci, kterou je omezen. Z tohoto důvodu jsem tyto zprávy vynechala. Ve snaze zachytit pokud možno co nejvíce událostí, jsem každým zařízením vytvořila nějaký provoz, reprezentující možné použití internetu. Laptop simuloval běžnou internetovou činnost jako například brouzdání webu, sledování videí, psaní zpráv atd. Stolní počítač simuloval administrátorskou činnost, například administrátorský přístup k routeru, změnu pravidel firewall, změnu portů, změnu uživatelů, úpravu serveru dhcp atd. A Kali Linux simuloval povrchní útok na systém, jmenovitě sken portů, útok ping flood a útok hrubou silou. K těmto útokům byly použity nástroje nmap (sken portů), hping3 (ping flood), crunch (generování hesel), ncrack (útok hrubou silou na ssh) a mkbrutus (útok hrubou silou na api).

Zaznamenala jsem alespoň 300 tisíc záznamů. Některé záznamy budou odstraněny z osobních důvodů, ostatní jsou uloženy na přiloženém médiu. Z důvodu takovéto velikosti jsem záznamy třídila pomocí skriptů napsaných v programu Python, které jsou taktéž přiloženy na médiu. Zpracování záznamů probíhá ve dvou fázích. V první jsou záznamy nejdříve ořízнuty o data, která jim přidal Rsyslog. Data jsou abecedně seřazena a jsou vyřazeny duplicitní záznamy. Toto zpracování nestačí k odhalení jednotlivých typů, protože některé záznamy obsahují identifikátory, které si router uchovává pro zpracování (například pořadové číslo, které si router uchovává pro identifikaci volání DNS). V dalším kroku tedy dochází k agresivnějšímu zpracování. Některé informace v záznamech jsou anonymizovány (například nalezené IP adresy, MAC adresy, pořadová čísla), každý originální a anonymizovaný záznam je uložen do páru a data jsou opět abecedně seřazena, tentokrát dle anonymizovaných verzí. Skript prochází anonymizovanými verzemi a z každého druhu záznamu vybere 5 prvků. Skript pro anonymizaci není dostatečně dobrý a výsledkem je tedy 60 tisíc prvků, které stále nejsou dostatečně unikátní. Jedná se však už o malý počet, který je dobře seřazený, zbylé prvky jsem tedy ručně přebrala a rozdělila. Za použití výše uvedené metodiky jsem identifikovala 12 kategorií zpráv. Tyto budou dále rozepsány podrobněji.

3.5 Kategorie zpráv

Každou kategorii krátce představím a zmíním, proč tuto kategorii považuji či nepovažuji za relevantní. Následovat bude pár příkladů z každé kategorie, případně generalizace u širších kategorií.

3.5.1 Přihlášení, odhlášení

Tento druh záznamů oznamuje úspěšné přihlášení či odhlášení uživatele ze systému. Záznam poskytuje informace o konkretním uživateli, IP adrese a způsobu přístupu. Tato kategorie je důležitá, protože informace kdo přistoupil, odkud a jak jsou velmi důležité. Zde je několik příkladů z této kategorie:

```
system,info,account user admin logged in from 192.168.88.251 via winbox
system,info,account user nikole logged in from 192.168.88.247 via ssh
system,info,account user admin logged out from 192.168.88.251 via winbox
system,info,account user nikole logged out from 192.168.88.247 via ssh
```

3.5.2 Selhání přihlášení

Zde je ohlášen neúspěšný pokus o přístup k systému. Zpráva má podobnou skladbu jako úspěšné přihlášení. Poskytnuto je jméno uživatele, IP adresa a způsob přístupu. Tato kategorie je důležitá i více než přihlášení a odhlášení, protože velký počet selhání přístupu značí na možný útok. Zde je několik příkladů z této kategorie:

```
system,error,critical login failure for user jsdkf from 192.168.88.251
via telnet
system,error,critical login failure for user admin from 192.168.88.247
via ssh
system,error,critical login failure for user admin from 192.168.88.248
via api
```

3.5.3 DHCP přiřazení

Tyto zprávy upozorňují na přiřazení IP adresy k MAC adrese. Obsažené informace jsou IP adresa, MAC adresa a jméno serveru DHCP. Tato kategorie je důležitá, protože díky ní lze pozorovat počet zařízení na síti, dle MAC adresy i odhadnout bližší informace. Zde je několik příkladů z této kategorie:

```
dhcp,info defconf assigned 192.168.88.250 to 00:00:00:00:04:20
dhcp,info defconf assigned 192.168.88.247 to 08:00:27:22:46:4F
dhcp,info defconf deassigned 192.168.88.250 from 00:00:00:00:04:20
dhcp,info defconf deassigned 192.168.88.247 from 08:00:27:22:46:4F
```

3.5.4 DHCP klient

Zde je oznámeno, jakou IP adresu a na jakém portu zařízení obdrželo či ztratilo. Tato kategorie je důležitá, protože adresa ukazuje na zkoumanou síť zvenčí a dle zpráv lze odhadnout, jestli síť má připojení k internetu. Zařízení MikroTik mohou být nastavena, aby měla více WAN portů. Názvy portů mohou být navíc konfigurovatelné. Tyto dva faktory znesnadňují odhad, jak vypadá síť, nicméně se stále jedná o důležité informace. Zde je několik příkladů z této kategorie:

```
dhcp,info dhcp-client on ether1 got IP address 10.42.0.12
dhcp,info dhcp-client on ether1 lost IP address 10.42.0.12 -
lease stopped locally
```

3.5.5 Připojení portů

Tyto zprávy oznamují připojení a odpojení portů k routeru. Ve zprávě je obsaženo jméno portu a informace o navázaném připojení. Tato kategorie je důležitá, protože značí případnou fyzickou manipulaci se zařízením či může upozornit na náhodné poškození kabeláže. Jméno portu je méně užitečné, protože je nastavitelné uživatelem. Automatická identifikace portu tedy nemusí být triviální. Zde je několik příkladů z této kategorie:

```
interface,info ether1 link up (speed 100M, full duplex)
interface,info ether1 link down
interface,info ether5 link up (speed 100M, full duplex)
interface,info ether5 link down
interface,info Group2-eth5 link up (speed 100M, full duplex)
interface,info Group2-eth5 link down
```

3.5.6 Chybové zprávy

Během průzkumu byly zachyceny tyto chybové zprávy. Jsou od sebe navzájem odlišné a poskytují v porovnání s ostatními kategoriemi různorodé informace. Tato kategorie je odlišná od ostatních a z mého pohledu je těžké nějak shrnout, jaké informace by se z této kategorie daly získávat. Počet těchto zpráv je v jednotkách, kdežto ostatní kategorie měly desítky či stovky kusů. Nízký výskyt a vysoká různorodost dělá tuto kategorii problematickou. Vysoká priorita těchto zpráv je však dělá důležitými. Zde je několik příkladů z této kategorie:

```
dhcp,critical,error dhcp-client on ether1 lost IP address 10.42.0.12 - lease
expired
ssh,error can't agree on KEX algorithms
dns,error DoH server connection error: resolving error
```

3.5.7 DNS

Další kategorií jsou zprávy ohledně požadavků DNS. Poskytnutými daty jsou IP adresa zařízení, ze kterého požadavek přišel, identifikátor, který router používá pro uchovávání informací o požadavku, a požadavek samotný. V případě odpovědi se jedná opět o identifikátor požadavku, dotazovanou adresu a odpověď. Tato data jsou důležitá, protože poskytují informace, na jaké stránky se zařízení pokusila přistoupit. Důležité je poznamenat, že do tématu `dns` spadají taktéž zprávy s tématem `packet`, při nastavování záznamu tohoto druhu je nutné nastavit, aby zprávy s tématem `packet` byly vynechány. Zde je několik příkladů z této kategorie:

```
dns query from 192.168.88.247: #1012 google.com. A
dns done query: #1012 google.com 142.250.186.46
dns query from 192.168.88.247: #1013 google.com. AAAA
dns done query: #1013 google.com 2a00:1450:4001:827::200e
dns query from 192.168.88.247: #1113 222.77.75.77.in-addr.arpa. PTR
dns done query: #1113 www.seznam.cz
```

```
dns done query: #13793 dns server failure
dns done query: #13803 dns name exists, but no appropriate record
```

Jistou podkategorií jsou zprávy označené „local query“. Domnívám se, že se jedná o DoS (DNS over HTTPS), protože tento druh zpráv se podle mého vědomí začal objevovat až po zapnutí DoS, stále se však objevovaly i normální záznamy. Zde je několik příkladů z této kategorie:

```
dns local query: #412 dns.quad9.net. A
dns done query: #412 dns.quad9.net 9.9.9.9
```

3.5.8 Systémové změny

Tyto zprávy se odesílají, když na routeru dojde k nějaké změně v nastavení. Zaznamenán je uživatel, který změnu provedl, o jakou kategorii změny se jedná a jaká změna byla konkrétně provedena. Změny v systému jsou významným zásahem, a proto je tato kategorie důležitá. Zde je několik příkladů z této kategorie:

```
system,info address added by admin
system,info bridge port changed by admin
system,info dhcp lease removed by admin
system,info dhcp server lonely_port added by admin
system,info dns changed by admin
system,info filter rule moved by admin
system,info user vaclav removed by nikole
system,info <změna> by <uživatel>
```

3.5.9 Firewall

Tato kategorie reaguje na pravidla nastavená ve firewall routeru. V základním nastavení žádná pravidla tyto zprávy nevytváří a je nutno pravidla upravit, než začnou podávat nějaký výstup v podobě záznamů. Z tohoto důvodu je nesnadné odhadnout, jak tyto zprávy budou vypadat, co vlastně budou znamenat a z jakých pravidel byly vytvořeny. Z tohoto důvodu tato kategorie nebude dále v práci zpracována, zde je uvedena pouze pro úplnost. Zde je několik příkladů z této kategorie:

```
firewall,info dstnat: in:ether1 out:(unknown 0), src-mac 00:e0:4c:3f:c5:b1,
proto TCP (SYN), 10.42.0.1:1694->10.42.0.12:80, len 40
firewall,info dstnat: in:ether1 out:(unknown 0), src-mac 00:e0:4c:3f:c5:b1,
proto TCP (SYN), 10.42.0.1:1930->10.42.0.12:80, len 40
firewall,info input: in:ether1 out:(unknown 0), src-mac 00:e0:4c:3f:c5:b1,
proto TCP (SYN), 10.42.0.1:1540->10.42.0.12:84, len 40
firewall,info input: in:ether1 out:(unknown 0), src-mac 00:e0:4c:3f:c5:b1,
proto TCP (SYN), 10.42.0.1:1540->10.42.0.12:840, len 40
```

3.5.10 DHCP podrobně

Tato část podrobněji popisuje konání dhcp serverů a klientů. Ač by tyto informace mohly být užitečné, méně podrobné informace uvedené v kategorii 3.5.3 jsou dle mého mínění dostačující. Z tohoto pohledu je tato kategorie duplicitní a nebude s ní v práci dále počítáno. Zde je několik příkladů z této kategorie:

```
dhcp,debug , authorative or unicast, new lease, acquire
dhcp,debug 192.168.88.247 entering bound
```

```
dhcp,debug defconf received discover id 1695089620
from 0.0.0.0 '1:0:0:0:0:4:20'
dhcp,debug defconf received request id 100214494
from 192.168.88.247 '1:8:0:27:22:46:4f'
dhcp,debug defconf sending ack with id 1004980415 to 192.168.88.247
dhcp,debug defconf sending offer with id 2575625519 to 192.168.88.250
dhcp,debug lease 192.168.88.249 expired
dhcp,debug,state dhcp-client on ether1 entering <renewing...> state
```

3.5.11 SSH

Tyto záznamy jsou výhradně z témat `ssh` a `debug`. Jakožto takové obsahují mnoho informací, ale jsou příliš podrobné a mohly by zahltit QRadar. Dostatečný pohled o přístupu k SSH poskytují kategorie 3.5.1 a 3.5.2. Z těchto důvodů nebudou záznamy z této kategorie zpracovány. Zde je několik příkladů z této kategorie:

```
ssh,debug auth req: admin ssh-connection publickey
ssh,debug auth retry: 4
ssh,debug authorization service requested
ssh,debug channel #0 local window consumed: 8 left: 27ffde
ssh,debug client version: SSH-2.0-OpenSSH_7.1
ssh,debug getPrime bits: 7680[2048-8192] returned: 2048
ssh,debug enc algo CS: aes128-cbc,3des-cbc,blowfish-cbc,aes192-cbc,
aes256-cbc,aes128-ctr,aes192-ctr,aes256-ctr
ssh,debug host key algo: ecdsa-sha2-nistp384
ssh,debug transport state: 1 -> 2
```

3.5.12 Směrování

Tyto zprávy mají stejný problém jako témata `packet` a `raw`. Informace jsou rozprostřeny do více zpráv. Ze stejných důvodů, jako byla vyřazena témata packet a raw, si dovoluji vyřadit i tuto kategorii. Zde je několik příkladů z této kategorie:

```
route,debug Accept withdraw 10.42.0.0/24
route,debug,event Address added or changed
route,debug,event     local=10.42.0.12
route,debug,event     network=10.42.0.0/24
route,debug,event     interface=ether1
route,debug Commit prefix 10.42.0.0/24
route,debug,calc Tag next hop for recalculation
route,debug,calc     address=10.42.1.1
```

3.5.13 Ostatní

Poslední kategorií jsou zbylé záznamy z různých témat. Vzhledem k jejich různorodosti a nízké prioritě si je dovoluji zanedbat a vyřadit ze zpracování. Zde je několik příkladů z této kategorie:

```
bridge,info "lonely_bridge" mac address changed to D4:CA:6D:24:41:65
bridge,stp ether5 forwarding
bridge,stp ether5 learning
certificate,debug start CRL update
```

```
certificate,debug trust store updated
info fetch: file "digicert-root-ca.pem" downloaded
```

3.6 Nastavení pravidel pro reprodukci zpráv

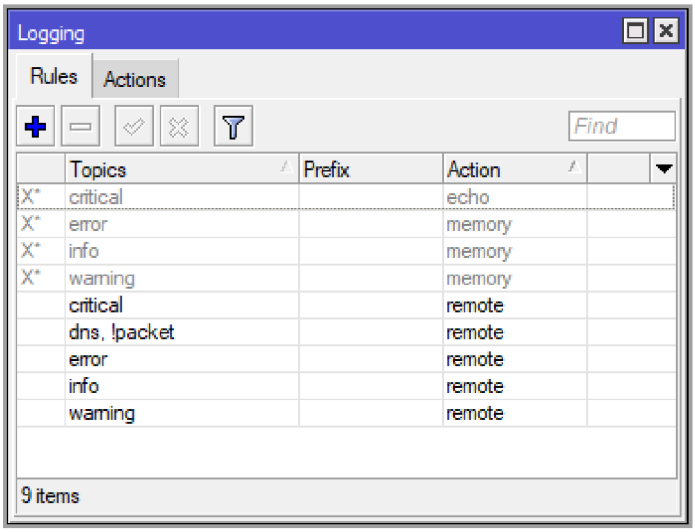
Po zjištění druhů zpráv, které vychází ze systému a zhodnocení, které by bylo vhodné ke zpracování, je nutno zjistit, jak těchto zpráv dosáhnout. Jak bylo zmíněno dříve, zprávy jsou generovány podle témat, pokud alespoň jedno z témat, ke kterým zpráva patří, je uvedeno v pravidlech. Je tedy nutno zjistit, jaká témata jsou obsažena v jednotlivých kategoriích.

Kategorie	Témata
Přihlášení, odhlášení	system,info,account
Selhání přihlášení	system,error,critical
DHCP přiřazení	dhcp,info
DHCP klient	dhcp,info
Připojení portů	interface,info
Chybové zprávy	dhcp,critical,error ssh,error dns,error
DNS	dns
Systémové změny	system,info

Tabulka 3.1: Kategorie záznamů a témata, která obsahují

Jak je vidět v tabulce 3.1, nejběžnějším tématem je `info`, které pokrývá všechny kategorie kromě Selhání přihlášení, Chybové zprávy a DNS. Tématem `error` jsou pokryta témata Selhání přihlášení a Chybové zprávy. Zbývá pouze DNS, které je možno pokrýt tématem `dns`. Jak bylo řečeno v subsekci 3.5.7 o záznamech DNS, je nutné nastavit vynechání témat `packet` pro téma dns. V manuálu od MikroTiku jsou jako základní témata uvedena `critical`, `debug`, `error`, `info`, `packet`, `raw` a `warning`. Jak již bylo dříve řečeno, `raw` a `packet` by příliš zatěžovaly systém a u `debug` můžeme předpokládat totéž. Když vybereme témata pro pokrytí kategorií, zbývají `critical` a `warning`. Ač tato témata nejsou teoreticky nutná, v reálném nasazení doporučuji vytvořit pravidla i pro ně. Ač návrh DSM modulu s potenciálními zprávami z těchto témat nebude počítat, jejich priorita je tak vysoká, že by bylo vhodné tyto zprávy zachytit, pokud nějaké budou zaslány.

Pro shrnutí: pro reprodukci zachytávaných zpráv je nutné nastavit minimálně 3 pravidla, a to pravidlo pro téma `info`, pravidlo pro téma `error` a pravidlo pro téma `dns` s vynecháním tématu `packet`. Pro bezpečnost doporučuji přidat navíc dvě pravidla pro zachytávání témat `critical` a `warning` respektive, tedy pět pravidel celkem. Příklad nastavení těchto pravidel pomocí aplikace Winbox je znázorněn na obrázku 3.1.



Obrázek 3.1: Nastavení zpráv v aplikaci Winbox

Kapitola 4

Návrh modulu DSM

V této kapitole bude představen návrh modulu DSM. Modul DSM je částí zpracování událostí systému QRadar, která za pomoci různých metod převádí zprávy na události s vlastnostmi. Podrobnější popis je v sekci 2.6. Konkrétně zde bude v sekcích popsáno zpracování jednotlivých vlastností události.

Vzhledem k tomu, že zprávy jsou určeny ke čtení lidmi, nejvhodnější způsob jejich zpracování pomocí počítače je využitím regulárních výrazů. Regulární výrazy v systému QRadar umožňují definovat formátovací řetězec, který umožňuje vybrat skupiny v rámci výrazu a připojit k nim text. Například „`$1`“ způsobí, že výsledkem bude první skupina, „`$1 $2 assigned`“ bude výsledkem první skupina, druhá skupina a slovo `assigned` oddělené mezerami atd. Část řetězce obsahující „`$0`“ referuje na celý regulární výraz.

V každé sekci bude představena jedna z extrahovaných vlastností, její význam a datový typ. Následovat bude regulární výraz, který ji zachytává, a formátovací řetězec. Pro lepší pochopení regulárního výrazu bude přidán příklad, který bude obarven. Žlutá barva označuje část zprávy, která byla zachycena regulárním výrazem. Zelená barva označuje část výrazu, která bude použita pro vlastnost. Červená barva označuje část, která nesmí následovat, tzv. záporné tvrzení o následujícím.

4.1 Event category

Toto je jedna ze dvou vlastností, které QRadar používá pro kategorizaci událostí. Dle konfiguračního manuálu pro moduly DSM se nezdá, že tyto kategorie mají nějaký konkrétní význam jako například IP adresa či jméno uživatele [7]. Naopak se zdá, že se jedná o arbitrární kategorii, která slouží k jakési generalizaci událostí v rámci jednoho modulu DSM. U některých DSM se může jednat o číslo, název, či dokonce je toto pole neměnné a všechny události mají stejnou kategorii. Z tohoto důvodu pro tuto kategorii byla vybrána témata, která patří k danému záznamu.

Jedná se o textový formát, název témat tedy vyhovuje tomuto formátu. Bylo by vhodné, kdyby témata byla rozdělena a zaznamenaná událost tedy měla více kategorií. To však dle mých znalostí není možné.

Tato vlastnost je zachycena metodou `Generic List`, která se na záznam dívá jako na list prvků oddělených znakem. Za oddělovací znak v tomto případě byla vybrána mezera a je zachycen první prvek. V původním návrhu se zachytával pátý prvek, protože QRadar předepínal informace ke zprávě. Tento přístup jsem původně zvolila, protože prvky předcházející tématům měly vždy konstantní počet mezer. Regulární výraz v tomto případě

by byl neúměrně složitý oproti tomuto jednoduchému přístupu. Po zjištění nesrovnalostí během implementace jsem se rozhodla tento postup zachovat, změněna byla pouze pozice. Řešení se tedy vymyká implementaci ostatních, je však jednoduché a umožňuje snadnou úpravu, kdyby v nějakém prostředí QRadar začal opět předepínat informace před zprávu. Následující příklad nedodržuje barevné schéma ostatních příkladů. V tomto příkladu zelená stále značí získanou kategorii, žlutá však slouží ke zvýraznění předcházejících mezer:

```
system,info,account user admin logged in from 192.168.88.247 via ssh
dns query from 192.168.88.247: #1013 google.com. AAAA
system,info device changed by admin
```

Zde je ukázka chování původního návrhu:

```
<182>Dec 08 12:48:54 192.168.88.1 system,info device changed by admin
```

4.2 Event ID

Toto je druhá vlastnost, kterou QRadar používá pro kategorizaci událostí. Stejně jako u Event category, výběr se zdá být arbitrální. U některých DSM se jedná o čísla či celá slova, například u Exabeam „105“, u FireEye „IOC Hit Found“ atd. Pro MikroTik touto vlastností budou generalizované popisky ze samotných záznamů. Jedná se o textový formát, je tedy možné použít takřka cokoliv. Kvůli různorodosti možných zpráv bylo vytvořeno celkem 9 regulárních výrazů, který každý zachytává jinou kategorii. Výrazy je možné si prohlédnout v tabulce 4.1.

Regulární výraz	Formátovací řetězec
„(user) (.+?) (logged) (in|out)“	„$1 $3 $4“
„login failure“	„$0“
„dhcp.+?(assigned|deassigned)“	„$1 ip address“
„(dhcp-client).+?(got|lost) IP“	„$1 $2 ip address“
„link (up|down)“	„$0“
„dns (query|done|local)“	„$0“
„(dhcp server|user|pool) (.+?) (added|changed|moved|removed)“	„$1 $3“
„ (port) (added|changed|removed) by “	„$1 $2“
„info (.+) by“	„$1“

Tabulka 4.1: Regulární výrazy používané pro Event ID

Každý výraz patří k nějaké kategorii z Kategorií zpráv, ze které se snaží získat akci. Kategorii Přihlášení a odhlášení 3.5.1 zpracovává výraz „`(user) (.+?) (logged) (in|out)`“, s jehož pomocí se zachytávají skupiny 1, 3 a 4. Jedná se o přímočaré řešení, které hledá přímo slova ve zprávě:

```
system,info,account user admin logged in from 192.168.88.251 via winbox
system,info,account user admin logged out from 192.168.88.251 via ftp
```

Pro kategorii Selhání přihlášení 3.5.2 je tento výraz ještě prostší a jednoduše hledá slova „`login failure`“. Zachytáván je celý výraz:

```
system,error,critical login failure for user admin from 192.168.88.251
via api
```

Kategorii DHCP přiřazení 3.5.3 zpracovává výraz „`dhcp.+?(assigned|deassigned)`“. Je zachycena skupina 1 a je za ni připnuto „`ip address`“. Teoreticky by bylo možné použít zachycené „IP“ psané hůlkovým písmem jako ID, v tomto případě je však nahrazeno tým-

též, psanými malými písmeny, protože většina ID bude v malých písmenech. Tento výraz by mohl být i prostší, je zde však takto dlouhý, aby nedošlo k náhodnému zachycení případného záznamu, který se během práce nepodařilo objevit. V těchto příkladech výstupem regulárního výrazu bude „assigned ip address“ a „deassigned ip address“:

```
dhcp,info defconf assigned 192.168.88.251 to E2:01:50:72:B0:EA
dhcp,info defconf deassigned 192.168.88.250 from 00:00:00:00:04:20
```

Pro kategorii DHCP klient 3.5.4 je výraz „`(dhcp-client).+?(got|lost) IP`“, za ID budou použity zachycené skupiny 1 a 2, za které bude připnuto „`ip address`“. Výraz sleduje podobnou filosofii jako předchozí, tedy kontroluje více částí zprávy, aby nedošlo ke kolizi. V následujících příkladech by výstupem bylo „dhcp-client got ip address“ a „dhcp-client lost ip address“:

```
dhcp,info dhcp-client on ether1 got IP address 10.42.0.12
dhcp,info dhcp-client on ether1 lost IP address 10.42.0.12 - lease
stopped locally
```

Pro kategorii Připojení portů 3.5.5 byl napsán výraz „`link (up|down)`“, za ID se použije celý zachycený text, jedná se opět o přímočaré řešení problému:

```
interface,info ether1 link up (speed 100M, full duplex)
interface,info ether1 link down
```

Pro kategorii DNS 3.5.7 byl vymyšlen výraz „`dns (query|done|local)`“, jako ID bude použit celý zachycený text. Jako v předchozích případech není to nejefektivnější řešení, ale je jednoduché, což poskytuje málo prostoru případným chybám.

```
dns query from 192.168.88.247: #1013 google.com. AAAA
dns done query: #10 www.facebook.com 157.240.30.35
dns local query: #1002 dns.quad9.net. A
```

Pro kategorii Systémové změny 3.5.8 bylo nejtěžší vymyslet nějaký smysluplný výraz. Nabízelo se zachytit popis akce, který se vyskytuje mezi tématy a „by <user>“, některé zprávy zde však ukazují jméno upravované části systému, tedy by takto zachycený úsek obsahoval jméno a byl tedy nevhodný pro použití jako identifikátor. Bylo tedy nutné vymyslet dva výrazy. Jeden pro záznamy, které jméno upravované části systému neobsahují, a druhý pro záznamy, které jméno obsahují. Výraz pro změny se jménem bude předcházet obecnějšímu výrazu, tedy pokud zpráva bude mít v sobě jméno, použije se jako první a k vyhodnocení obecnějšího nedojde. Je nutné podotknout, že stále může nastat, že se zachytí nevhodná část, kdyby se objevila zpráva, která má pojmenovanou změnu, která nebyla nalezena během zkoumání systému. Výraz pro pojmenované změny je „`(dhcp server|user|pool) (.+?) (added|changed|moved|removed)`“, první skupina zachytává části systému, prostřední pojmenování a poslední typ změny, pro identifikátor bude použita první a poslední skupina. V těchto případech bude jako identifikátor použito „dhcp server changed“ a „user added“:

```
system,info dhcp server lonely_port changed by admin
system,info user nikole added by admin
```

Během sbírání záznamů nebyl zachycen záznam, který by obsahoval „port“. Během implementace však některé záznamy nebyly korektně zpracovány, ukázalo se, že mají špatný identifikátor události, protože nereagují na ostatní regulární výrazy. Než pátrat nad chybou návrhu ostatních výrazů bylo jednodušší přidat další, tedy výraz „ `(port) (added|changed|removed) by `“. Pro ostatní záznamy bude použit výraz založený na úvodní domněnce, tedy „`info (.+) by`“, a bude použita právě tato prostřední část neboli první skupina. Zde jsou příklady:

```
system,info address changed by admin
system,info filter rule moved by admin
```

Kategorie Chybových zpráv 3.5.6 nemá regulární výraz pro Event ID. Vzhledem k nahodilosti této skupiny by bylo těžké vymyslet nějaký ucelený výraz. Zbytek systému QRadar bude v tomto případě reagovat na vlastnost Event category a Event ID bude zanedbáno.

4.3 Source IP address

Zdrojová IP adresa je vlastnost, která se dále zpracovává v subsystému Custom Rule Engine. Tato vlastnost je zvláště zajímavá, protože může být využita k identifikaci osoby na síti. Společně s MAC adresou pak lze identifikovat konkrétní zařízení. Ve studovaných záznamech se vyskytuje u tří kategorií: u přístupu k RouterOS, u dns záznamů a u přiřazení IP adres k MAC adresám. Všechny tyto situace jsou užitečné při sledování aktivity na síti, a tedy je sledování zdrojové IP těchto aktivit důležité. Pro kategorie dns a přístupů můžeme použít regulární výraz, který reaguje na slovo „from“, které u těchto záznamů předchází IP adresu:

Regulární výraz:	„`from (.+?)(:? )`“
Formátovací řetězec:	„`$1`“

Zde jsou příklady použití výrazu:
`system,info,account user admin logged in from 192.168.88.247 via ssh`
`dns query from 192.168.88.247: #1013 google.com. AAAA`
U zpráv popisujících přiřazení dhcp serverem využijeme podobnou filosofii, tentokrát však reagujeme na předcházející slovo „assigned“:

Regulární výraz:	„`assigned (\S+?) `“
Formátovací řetězec:	„`$1`“

Zde je příklad použití výrazu:
`dhcp,info defconf assigned 192.168.88.247 to 08:00:27:22:46:4F`

Výrazy jsou navrženy tak, že jsou schopny případně zachytit i IPv6 adresu, ač se žádná taková při testování nevyskytla, výrazy by měly být teoreticky schopné ji zachytit, stejné výrazy tedy budou použity i u vlastnosti Source IPv6 address.

4.4 Source MAC address

Zdrojová MAC adresa je obdobně jako IP adresa velice důležitá vlastnost, protože pomocí záznamů z DHCP můžeme k IP adrese přiřadit MAC adresu neboli můžeme k uživateli přiřadit konkrétní zařízení, které bylo použito na síti. MAC adresa taktéž poskytuje důležité informace o zařízení samotném. QRadar například může rozpoznat, jestli byl do sítě zapojen router, což je podezřelá aktivita sama o sobě. Ze zkoumaných zpráv jediné místo výskytu jsou záznamy z DHCP serveru. Adresám předchází slovo „to“ či „from“, na tato slova bohužel nemůžeme reagovat, protože dochází k duplicitě s IP adresou. Navržený výraz tedy reaguje na delší úsek, konkrétně na slovo „assigned“, pro kontrolu ověří výskyt IP adresy a pak zabere MAC adresu. Výraz je takto složitý, aby nedošlo k náhodné kolizi s nepředpokládanými záznamy:

Regulární výraz:	„`assigned (\S+?) (to|from) (.+?)$`“
Formátovací řetězec:	„`$3`“

Příklady chování výrazu na zprávách:
`dhcp,info defconf assigned 192.168.88.250 to 00:00:00:00:04:20`
`dhcp,info defconf deassigned 192.168.88.250 from 00:00:00:00:04:20`

4.5 Username

Uživatelské jméno je další z vlastností, které mohou být použity k identifikaci osob na síti. Vyskytuje se ve třech kategoriích: přihlášení/odhlášení, selhání přihlášení a systémové změny. V tomto případě bude uživatelské jméno pouze tak užitečné, jak granulární budou přihlašovací údaje. Pokud budou pouze jedny, informace o uživateli nebude mít takovou hodnotu, jako když každý administrátor bude mít své vlastní. Stále se však jedná o hodnotnou informaci, kterou je důležité zachytit. Ač to není v dokumentaci řečeno, uživatel nemůže mít mezeru ve jméně, tedy je možno použít metaznak „\w“. První výskyt je při úspěšném přihlášení či odhlášení, zde je vhodné reagovat na okolní slova, tedy „user“ a „logged“. Je potřeba reagovat na obě slova, protože uživatel se může jmenovat například „user1“, což by mohlo způsobit nechtěné problémy:

Regulární výraz:	„`user (\w+) logged`“
Formátovací řetězec:	„`$1`“

Příklady chování výrazu:
`system,info,account user admin logged in from 192.168.88.251 via ssh`
`system,info,account user admin logged out from 192.168.88.251 via ssh`
Na záznamy úprav v systému je použit stejný princip, je však potřeba reagovat širším způsobem. Ve stejné filosofii jako ostatní výrazy je výraz komplikovanější, aby nedošlo k nechtěným přesahům:

Regulární výraz:	„`((changed)|(added)|(removed)|(moved)) by (\w+)`“
Formátovací řetězec:	„`$6`“

Zde jsou příklady použití výrazu:
`system,info dhcp network added by admin`
`system,info address removed by admin`

Selhání přístupu je trochu složitější situace sama o sobě, uživatelská jména v tomto případě nemusí být skuteční uživatelé, ale pouze překlepy. Naprosto odlišná neznámá jména však mohou poukazovat na možný útok, tedy se stále jedná o hodnotnou informaci. Podobně jako v případu běžných přihlášení se není možné spolehnout pouze na slovo „user“, protože uživatel se takto může jmenovat. Výraz tedy reaguje na předcházející slovosled:

Regulární výraz:	„`failure for user (\w+)`“
Formátovací řetězec:	„`$1`“

Zde je znázorněno chování výrazu:
`system,error,critical login failure for user adl from 192.168.88.247 via telnet`

4.6 URL

URL, která se vyskytuje v záznamech dns, je velice důležitá informace, protože poukazuje, na jaké části sítě se dané zařízení pokusilo přistoupit. V kombinaci s IP adresou je možné dohledat MAC adresu, a tedy určit, které zařízení konkrétně navštívilo jakou stránku. Společně s informacemi o tom, jaké stránky jsou rizikové, je možné vyvodit riziko napadení, například jaké zařízení se pokusilo připojit na botnet. Jelikož URL může vypadat mnoha

způsoby a ve své nejjednodušší formě se může jednat o prosté slovo. Psaní výrazu, který by zachytával samotnou URL, by bylo netriviální. Filosofie tohoto regulárního výrazu je, že místo hledání URL se hledá pozice, kde se nachází. Výraz tedy identifikuje, že se jedná o zprávu ohledně dns, a pak se zastaví o identifikátor dotazu a zachytí URL:

Regulární výraz:	„`dns .+?(#\d+)(?! dns server failure) (.+?)(\s|$)`“
Formátovací řetězec:	„$2“

Zde jsou příklady použití výrazu:
`dns query from 192.168.88.247: #1013 google.com. AAAA`
`dns done query: #10 www.facebook.com 157.240.30.35`

Ve výrazu musí být tzv. záporný lookahead, který dovolí pokračovat ve výrazu, pokud nenásleduje „dns server failure“. V opačném případě by totiž výraz jako URL zachytil „dns“, jak je znázorněno v následujícím příkladu. Jedná se o pokročilé pravidlo, které však bylo nutné použít v této situaci:
Bez „`(?! dns server failure)`“:
`dns done query: #401 dns server failure`
S „`(?! dns server failure)`“:
`dns done query: #401 dns server failure`

4.7 DNS Record Type

Touto částí se dostáváme k vlastnostem, které nejsou v základní verzi systému QRadar a je potřeba je vytvořit. To znamená, že předinstalovaná pravidla na ně nebudou reagovat. Nicméně tyto informace mohou být natolik užitečné, že si zasluhují být taktéž zaznamenány.

Tato vlastnost uchovává informace o tom, jaký druh záznamu daný DNS dotaz požadoval. Tato vlastnost je důležitá, protože poskytuje podrobnější náhled na to, jak bylo k dané stránce přistupováno. Regulární výraz je založen na podobné myšlence jako vlastnost URL, tedy se zjistí kontext a pak podle pozice se zachytí druh záznamu:

Regulární výraz:	„`dns ((local )?query)(.+?) (\w+?)$`“
Formátovací řetězec:	„$4“

Příklady chování výrazu:
`dns query from 192.168.88.247: #1013 google.com. AAAA`
`dns local query: #24437 cloud2.mikrotik.com. A`

4.8 DNS Query Result

Výsledek DNS dotazu je důležitou informací, protože změna výsledku v dotazech na stejnou adresu může poukazovat na podvrh v DNS záznamu. Výraz na zachycení této informace je postaven na stejném principu jako v předchozích dvou případech. Ověření, že se jedná o výsledek dotazu pomocí „`dns done query`“, navazující ověření, že nedošlo k selhání pomocí „`(?!dns server failure)`“ a následuje zachycení výsledku. Kompletní výraz vypadá takto:

Regulární výraz:	„`dns done query: #\d+ (?!dns server failure)(\S+ )?(.+)$`“
Formátovací řetězec:	„$2“

Zde jsou příklady chování výrazu:
`dns done query: #10 www.facebook.com 157.240.30.35`
`dns done query: #10000 prg03s10-in-f14.1e100.n`
`dns done query: #401 dns server failure`

4.9 Used application

Tato vlastnost uchovává informace o tom, jaká metoda byla použita k přístupu k RouterOS. Zachytává přihlášení, odhlášení a selhání přihlášení. Tato informace může být užitečná, protože některé možnosti nejsou šifrované, takže přístup s jejich použitím může způsobit únik hesla. QRadar může být nastaven, aby upozornil, když bude právě takový přístup zaznamenán. Výraz pro zachycení této informace reaguje na slovo „via“, které se nachází na konci záznamu. Jedná se o jednodušší výraz než v ostatních případech, ale jedná se o tolik specifickou situaci, že je dostačující:

Regulární výraz:	„`via (.+?)$`“
Formátovací řetězec:	„`$1`“

Zde jsou příklady chování výrazu:
`system,info,account user admin logged in from 192.168.88.247 via ssh`
`system,error,critical login failure for user aba from 192.168.88.247 via ssh`

4.10 Interface name

Touto vlastností se dostáváme do vlastností, které jsou méně užitečné, ale stále jsou zachytávány pro úplnost. Jméno připojeného portu je méně užitečné, protože může být změněno uživatelem, a tedy předpokládání nějakých akcí může být netriviální, ale stále může být užitečné při vytváření vlastních pravidel. Taktéž tuto vlastnost bude využívat rozšíření typu add-on popsané v kapitole 5. Využívá se zde stejného principu jako u jména uživatele, ověření okolních slov a zachycení jména uprostřed:

Regulární výraz:	„`info (.+?) link`“
Formátovací řetězec:	„`$1`“

Zde jsou příklady použití výrazu:
`interface,info ether1 link up (speed 100M, full duplex)`
`interface,info ether1 link down`

4.11 Interface properties

Tato vlastnost uchovává informaci ohledně vlastností připojeného portu, konkrétně rychlost a obousměrnost komunikace. Jedná se o jednu z informací, která je vhodná spíše pro testovací účely, je však jako předchozí zachytávána pro úplnost. Regulární výraz této vlastnosti vyhledává sousloví „link up“ a zachytí vnitřek následujících závorek. Jedná se o jeden z jednodušších výrazů, situace je však natolik specifická, že je dostačující:

Regulární výraz:	„`link up \(.+?\)`“
Formátovací řetězec:	„`$1`“

Příklad použití výrazu:
`interface,info ether1 link up (speed 100M, full duplex)`

4.12 WAN IP

Tato vlastnost reprezentuje IP adresu, jakou má RouterOS. Tato informace je stále částečně využitelná. Šlo by například nastavit pravidlo, které při změně WAN IP spustí skript, který aktualizuje záznam DNS. Existují jednodušší způsoby, jak tohoto dosáhnout, například nastavit podobnou akci na samotném routeru, stále se však jedná o validní možnost. QRadar může takto aktualizovat záznamy z více routerů a není potřeba nastavovat každý router samostatně. Výraz zachytávající tuto informaci patří k jedněm z komplikovanějších. I zde se uplatňuje myšlenka zachytávání místa, kde se informace vyskytuje, místo samotného formátu IP adresy. Výraz nejdříve ověří, že se jedná o dhcp klienta, pak se výraz zastaví o slova „IP address“ a zachytí informaci za tímto slovem:

Regulární výraz:	„`dhcp-client.+?IP address (.+?)(\s|$)`“
Formátovací řetězec:	„`$1`“

Následují příklady chování výrazu:
`dhcp,info dhcp-client on ether1 got IP address 10.42.0.12`
`dhcp,info dhcp-client on ether1 lost IP address 10.42.0.12 - lease stopped locally`

4.13 DHCP server name

Poslední zachytávaná vlastnost je zde čistě jenom pro úplnost. Zachytává jméno DHCP serveru, který přiřadil, nebo zrušil přiřazení IP adresy. Regulární výraz je opět založen na myšlence ověření, zda se jedná o záznam, který tuto informaci obsahuje a odchycení místa, kde informace je. V porovnání s ostatními výrazy je však tento naopak. Většina výrazů v této práci zachytává informaci za nějakou známou částí, zde je zachytávána informace před známou částí. Navíc jsou zde přidány variace, které se nevyskytují v popsaných záznamech, ale vyskytují se v záznamech tématu debug. Jelikož tato vlastnost je spíše experimentální, výraz je tedy trošku širší než u ostatních druhů:

Regulární výraz:	„`dhcp(\S+)? (.+?) ((de)?assigned|received|sending)`“
Formátovací řetězec:	„`$2`“

Zde jsou příklady chování výrazu:
`dhcp,info defconf assigned 192.168.88.251 to E2:01:50:72:B0:EA`
`dhcp,info defconf deassigned 192.168.88.250 from 00:00:00:00:04:20`
`dhcp,debug defconf sending ack with id 100214494 to 192.168.88.247`

Kapitola 5

Návrh rozšíření Add-on

Tato kapitola je věnována návrhu rozšíření typu Add-on, které bude zobrazovat vizualizované informace, které byly zachycené pomocí modulu DSM. Nejdříve bude zběžně vysvětleno, jak fungují aplikace v prostředí QRadar, dále bude popsána činnost rozšíření a pak zobrazen i vizuální návrh.

5.1 Aplikace pro QRadar

Za aplikaci v rámci QRadar se považuje rozšíření, které bylo vyvinuto nějakým vývojářem a upravuje uživatelské rozhraní systému QRadar. Aplikace může přidávat nové záložky, nové metody API, nové přehledy, nová menu, nová tlačítka, stránky pro konfiguraci a jiné úpravy v rámci uživatelského rozhraní.

V této práci, když se zmiňuje rozšíření typu add-on, je tím myšleno přidání nové záložky do uživatelského rozhraní. Pro vývoj tohoto druhu aplikace lze využít QRadar App Framework SDK. Jedná se o webovou aplikaci, napsanou v jazyce Python, běžící s využitím aplikačního rámce Flask, komunikující se systémem QRadar pomocí volání API. Při nasazení v systému QRadar aplikace běží v izolovaném prostředí s využitím software Docker. Webové stránky takto běžící aplikace jsou pak přesměrovány do záložky v uživatelském prostředí [6].

5.2 Princip činnosti modulu Add-on

S vedoucím bylo diskutováno, jestli by toto rozšíření mohlo být použito pro správu nasazených zařízení se systémem RouterOS. Výhodou by bylo mít bezpečnost a administraci na jednom místě. Ač je to teoreticky možné, tak tuto funkci by nebylo možné zaručit. Aby bylo možné provést správu systému RouterOS ze systému QRadar, QRadar potřebuje mít přímou viditelnost na RouterOS neboli musí mít přístup k IP adrese, na které RouterOS je. Toto není možné zaručit, protože vztah RouterOS ke QRadar je klient, server. RouterOS je svým vztahem klient, který chce uložit své záznamy na QRadar, při nasazení je tedy QRadar dostupný pro RouterOS, ale RouterOS nemusí být dostupný pro QRadar. Z tohoto důvodu, po konzultaci s vedoucím, bylo domluveno, že rozšíření bude sloužit pouze pro vizualizaci dat již obsažených v systému QRadar.

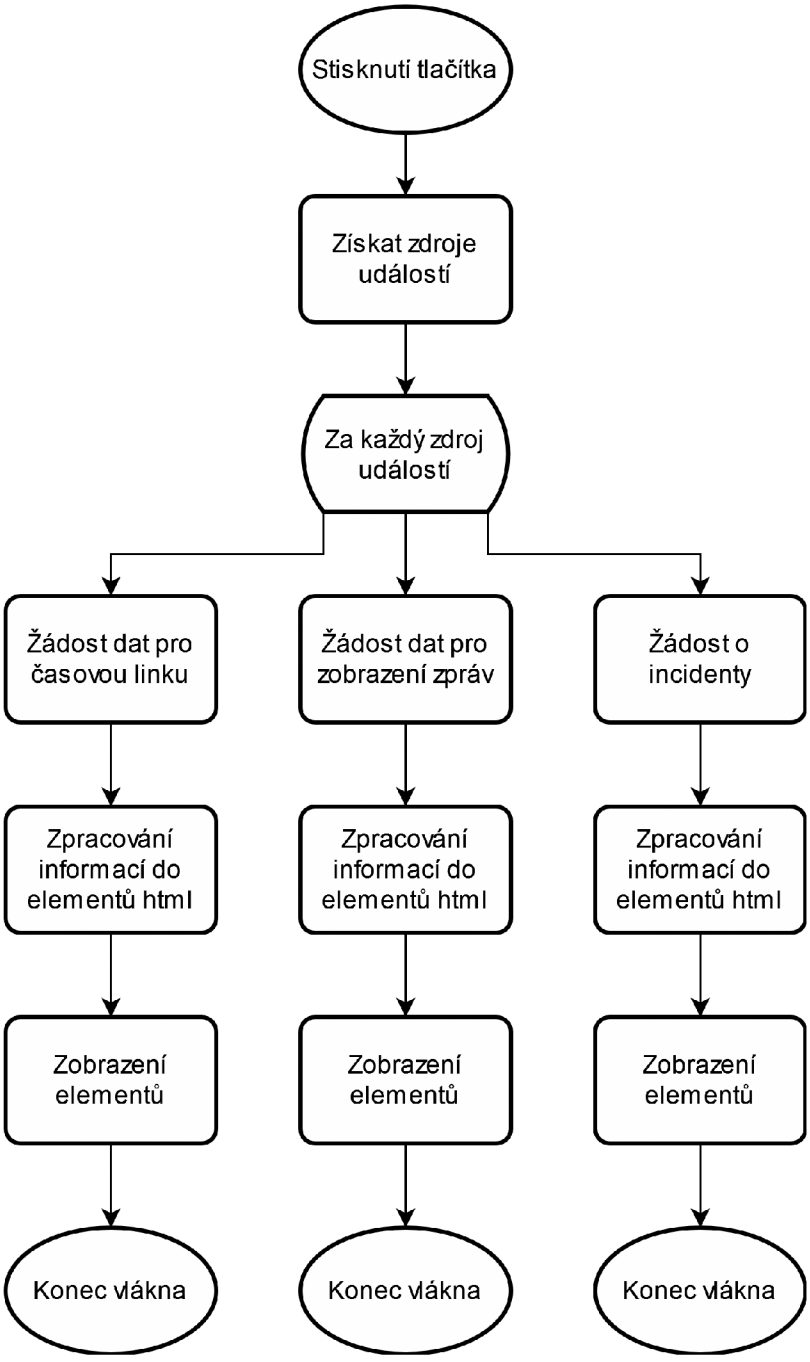
Toto rozšíření je určeno čistě k vizualizaci informací již obsažených v systému QRadar v přehlednějším formátu. Rozšíření lze připodobnit k tenkému klientu a QRadar k serveru. Rozšíření nebude žádná data ukládat, pouze je obdrží od systému QRadar, setřídí a zobrazí.

Protože je systém RouterOS určen pro použití jako síťový prvek, bylo by vhodné, kdyby rozšíření ukazovalo informace s ohledem na síťový provoz, například připojená zařízení, aktivní adresy sítí atd. Při implementaci se však ukázalo, že toto je nesnadné zajistit. Informace, které by byly vhodné ke kvalitnímu zobrazení síťového provozu, jsou obsaženy v tématu `debug`. Toto téma však generuje velké množství zpráv, které by potenciálně mohly zahltit systém QRadar, či by zpracování těchto zpráv trvalo dlouho. Bylo by vhodné toto téma prozkoumat v navazující práci, v této práci jsem se vzhledem k okolnostem rozhodla vizualizovat události, které systém QRadar zpracoval. Konkrétně je vizualizovat na časové ose, dále zobrazit seznam zobrazených zpráv a v třetí řadě seznam všech incidentů, které souvisí s daným zařízením.

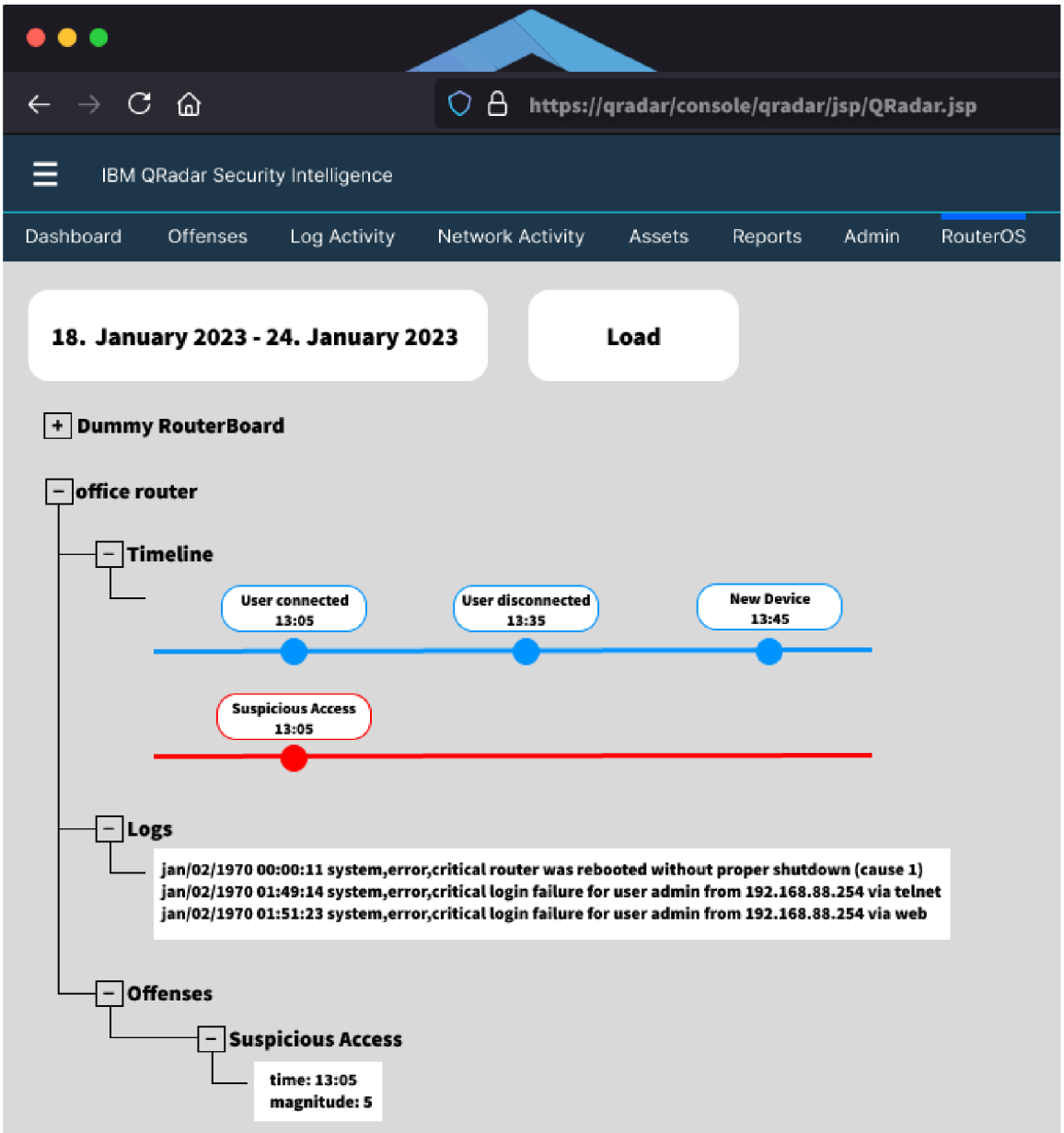
Program je webová aplikace frameworku Flask. Návrhem chování programu je myšleno jedno načtení dat po vyžádání uživatelem při stisknutí tlačítka. Na obrázku 5.1 je vyobrazen návrh chování programu. Jedná se o jednoduchý přímočarý program, který vytváří podprocesy. Uživatel spouští program klepnutím tlačítka. V první řadě se program dotáže systému QRadar na dostupné zdroje událostí, které používají modul DSM. Pro každý zdroj událostí se vytváří tři nová volání do systému QRadar, která běží nezávisle na sobě. QRadar ukládá události v databázi Ariel, která funguje na podobném principu jako SQL. Dvě ze tří volání žádají o data právě z této databáze. První volání volá databázi Ariel za účelem získání událostí pro zobrazení časové linky, druhé volání žádá data, která se použijí pro zobrazení příchozích zpráv. Třetí volání kontaktuje systém QRadar k získání incidentů daného zdroje událostí. Tato data jsou nezávisle na sobě zpracována a zobrazena.

5.3 Vizuální návrh

Na obrázku 5.2 je vyobrazen vizuální návrh. Návrh je založen na rozvržení informací, které používá například Správce zařízení v operačním systému Windows. Tedy prvky stejné informační hodnoty jsou seřazené pod sebou. Podřazené prvky, které pak obsahují bližší informace o nadřazeném prvku prvku, jsou pak posunuty trochu na pravou stranu, což symbolizuje jejich podřazenost. Tento návrh obsahuje tři druhy prvků, které byly zmíněny v sekci 5.2. Všechny informace jsou reflektovány vzhledem k časové lince vybrané uživatelem. Prvním prvkem je časová linka, která přehledně vizualizuje události a incidenty, které se staly. Události jsou zobrazeny modrou barvou, protože se jedná o relativně běžnou neutrální barvu. Incidenty jsou zobrazeny na červené ose, protože symbolizují nějaký problém, což je reflektováno červenou. Druhým prvkem jsou všechny zprávy přijaté ze zařízení MikroTik. Tyto zprávy jsou jednoduše seřazeny pod sebou a před ně je připnuto datum, kdy byly zachyceny. Pouze pár prvků je zobrazeno a je možno zobrazit ostatní posuvníkem. Na obrázku je vyobrazeno pouze pár prvků jako příklad, samotná implementace pak bude ukazovat více zpráv. Třetím jsou incidenty, které byly vyvolány na základě zpráv z daného routeru. Každý incident bude mít svůj vlastní podprvek, který bude obsahovat podrobnější informace. V obrázku návrhu je zobrazen pouze čas a závažnost, v implementaci pak bude více informací, které jsou relevantní k danému incidentu. Všechny tyto prvky má každé zobrazené zařízení a je možné je skrýt či zobrazit. Tento návrh je velice jednoduchý a v budoucí práci bych ho ráda rozšířila. Tento návrh vnímám jako položení základů, namísto hotového kompletního řešení.



Obrázek 5.1: Diagram chování programu



Obrázek 5.2: Návrh rozšíření pro QRadar

Kapitola 6

Implementace

V této kapitole je nejdříve krátce popsána implementace modulu DSM a následně rozšíření typu Add-on, společně s rozhodnutími a relevantními detaily.

6.1 Modul DSM

Implementace modulu DSM je velice prostá. Použila jsem editor DSM[1], který je vestavěný v systému QRadar. Tento editor poskytuje grafické rozhraní pro tvorbu a editaci modulů DSM. Dle návrhu uvedeného v kapitole 4 jsem vytvořila modul a jednotlivým zpracovaným vlastnostem jsem přidala odpovídající regulární výrazy. Během implementace jsem nalezla několik překlepů a chybných napsání regulárních výrazů, které jsem zpětně reflektovala v návrhu. Současný návrh obsahuje již upravené verze. Nad rámec plánované práce jsem vytvořila qid mappings a qid records. Qid records se používají pro klasifikování událostí. Qid mappings za pomoci dvojice vlastností EventID a Event category přiřazují dané události qid record. Pro každou nalezenou a předpokládanou událost jsem vytvořila qid record, kterému jsem přiřadila kategorie událostí dle dostupné dokumentace[2] a vytvořila jsem patřičný qid mapping. Bohužel neexistují příklady nebo postupy, jak tato mapování udělat. Oporou mi byla dokumentace kategorií, která je však dlouhá a nepřehledná pro adekvátní nastudování v omezeném časovém rámci této bakalářské práce. Výsledná mapování jsou kombinací kategorií, které se mi povedly najít, a mého vlastního úsudku. Výsledné qid records a qid mappings jsou uložené v souborech `qid_record.csv` a `qid_mappings.csv` na přiloženém médiu. Výsledný modul DSM s mapováními jsem pak exportovala a v době vydání práce je dostupný na přiloženém paměťovém médiu a mém osobním respositáři GitHub[3].

Dovolím si osobní poznatek, který není relevantní k výsledku, ale je užitečný, kdyby někdo následoval mé kroky. Při editaci u vlastností, které jsou vestavěné v systému QRadar, je nutné přepsat jejich chování zaškrtnutím políčka Override. Při exportu se nově vytvořené vlastnosti exportují jako nové vlastnosti a lze je najít v exportovaném souboru xml. Přepsané vestavěné vlastnosti se však exportují jako rozšíření zařízení. V exportovaném xml se dá toto rozšíření najít v elementu `device_ext`, je však kódované ve formátu BASE64[4]. Z tohoto důvodu nejsou všechny implementované regulární výrazy na první pohled viditelné.

[1] https://www.ibm.com/docs/en/qsip/7.4?topic=qradar-dsm-editor-overview [cit. 2023-4-18]
[2] https://www.ibm.com/docs/en/qsip/7.4?topic=administration-event-categories [cit. 2023-4-18]
[3] https://github.com/yellowfox-star-is/qradar-mikrotik-dsm [cit. 2023-04-24]
[4] https://www.ietf.org/rfc/rfc4648.txt [cit. 2023-4-19]

6.2 Rozšíření typu Add-on

Toto rozšíření je webová aplikace vestavitelná do systému QRadar, která je založená na frameworku Flask. Je naprogramovaná za pomocí jazyků Python a Javascript, se systémem QRadar komunikuje za pomoci volání REST API. Jedná se o jednoduchý program, zapouzdřený pomocí systému Docker. Firma IBM poskytuje sadu pro vývoj softwaru[5], která usnadňuje vytváření kontejnerů systému Docker a volání API. Důležitou částí SDK je knihovna `qpylib`[6], která poskytuje jednoduché rozhraní pro komunikaci s frameworkem QRadar App a volání REST API. Všechna volání na systém QRadar jsou zprostředkována pomocí této knihovny, což usnadňuje autentizaci, zpracování výsledků volání, automaticky doplňuje adresu systému QRadar, či zabaluje jiné opakující se prvky ve voláních.

Vzhledem k vlastní preferenci jazyka Python oproti jazyku Javascript jsem implementovala poněkud unikátní strukturu programu. Framework Flask obsluhuje několik adres URL, nejdůležitější je kořenová URL `/`, pod kterou je schována šablona `monitor.html` obsahující část programu napsanou v javascriptu. Tato část se stará o zobrazování dat za pomoci vytváření elementů HTML. Na tuto část bude dále referováno jako na zobrazovací vrstvu. Pak jsou zde jiné adresy URL, při jejich navštívení jsou zavolány funkce z Python modulu `get_data.py`, které vyzvednou relevantní data ze systému QRadar pomocí volání API a zobrazí je serializovaná ve formátu JSON. Na tuto část bude dále referováno jako na dotazovací vrstvu. Kořenová adresa tedy volá na jiné adresy, ze kterých získává data ve formátu JSON, která následně zobrazí. Dotazovací vrstva tedy obaluje API pro pohodlnější volání.

Adresářový strom si můžete prohlédnout na obrázku 6.1. Složka `app` obsahuje samotnou aplikaci Flask. Modul `__init__.py` slouží pro vytvoření a načtení Flask aplikace. Tento modul načítá základní konfiguraci aplikace (například cestu k ostatním modulům). Následně načte moduly `view.py` a `dev.py`.

Modul `dev.py` byl předvytvořen za pomoc SDK, slouží k základnímu ladění za pomoci knihovny `qpylib`. Tento modul nebyl nijak upraven a byl ponechán kvůli kompatibilitě. Modul `view.py` slouží pro definování jednotlivých cest URL a volá primárně modul `get_data.py`. Navíc provádí lehkou serializaci a deserializaci dat mezi prostředími. Konkrétně za pomoci funkce `pass_data` převádí získané objekty z modulu `get_data.py` do formátu JSON. Za pomoci funkce `url_to_stamp` převádí data ve formátu ISO, získaného ze zobrazovací části, do časového razítka Epoch Unix. Důležité je zmínit, že tato konverze využívá funkce `fromisoformat` z modulu `datetime`.

Ve verzi jazyka Python, který využívá systém QRadar, tato funkce není implementovaná. Tato funkce je do současné verze instalována pomocí balíčku `backports_datetime_fromisoformat`. Tento balíček musel být kompilován pro konkrétní verzi jazyka Python, aby mohl být použit v souladu s pravidly frameworku QRadar App (Framework neumožňuje instalaci ze zdrojů dostupných z internetu, pouze z lokálních zdrojů. Framework však neobsahuje kompilátory nutné pro kompilaci tohoto balíčku, proto musel být předkompilován). Zvýšení verze jazyka Python v budoucích verzích může způsobit nekompatibilitu. Při aktualizaci na novější verzi je tedy nutné si na toto dávat pozor. Pokud však uživatel nebude měnit verzi obrazu prostředí, nemělo by k nekompatibilitě dojít.

Složky a soubory `container`, `store`, `manifest.json` a `qenv.ini` obsahují informace pro Docker a QRadar SDK pro úspěšné vytvoření a nasazení kontejneru. Nyní se podíváme na některé důležitější části podrobněji. Složka `container` obsahuje složku `pip` obsahující sou-

[5] https://exchange.xforce.ibmcloud.com/hub/extension/517ff786d70b6dfa39dde485af6cbc8b [cit. 2023-04-19]

[6] https://github.com/IBM/qpylib [cit. 2023-04-19]

bor `backports_datetime_fromisoformat-2.0.0-cp36-cp36m-linux_x86_64.whl`. Jedná se o kompilovaný balíček `backports_datetime_fromisoformat` zmíněný dříve. Když je balíček v této cestě, SDK provede automatickou instalaci při tvorbě kontejneru Docker. Soubor `manifest.json` obsahuje, jaký obraz kontejneru se má použít. Soubor `qenv.ini` obsahuje informaci o lokaci serveru QRadar na síti a klíč pro administraci. Tento soubor se nepoužívá při vestavění, ale je důležitý, pokud aplikace běží mimo systém QRadar. Tímto jsem zmínila nejdůležitější kousky nastavení prostředí.

Soubor `README.md` obsahuje obecné informace ohledně projektu, tento soubor je zde hlavně kvůli umístění na repositáři GitHub. Soubor `LICENSE.md` obsahuje licenci pro využití tohoto projektu. Nyní bych se podrobněji věnovala souborům `monitor.html`, `get_data.py` a jejich navazujícím souborům.

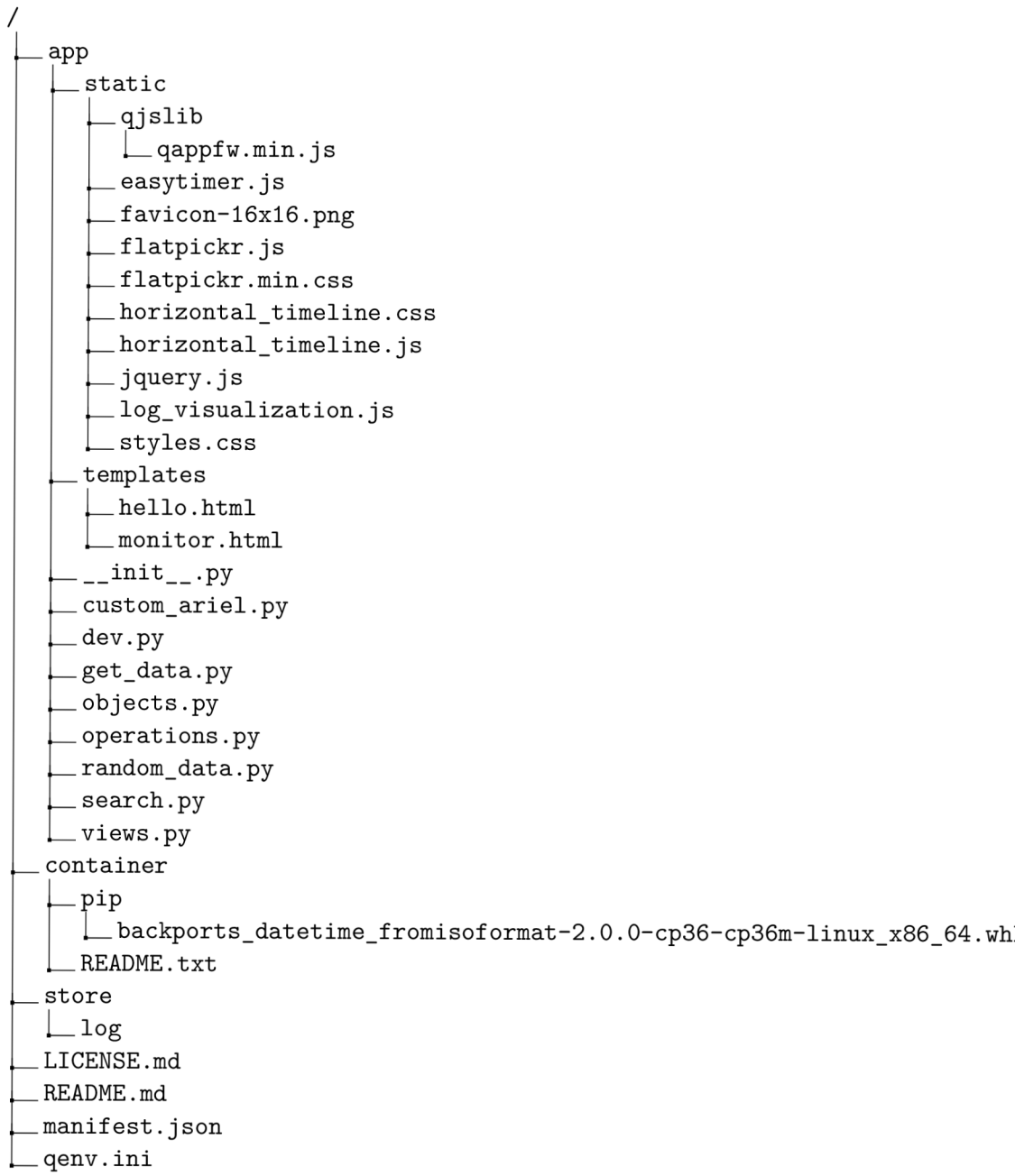
6.2.1 Zobrazovací vrstva

Zobrazovací vrstva je primárně tvořena souborem `monitor.html` a CSS/JavaScriptovými soubory. Samotný html soubor je velice prostý. Na začátku importuje všechny relevantní soubory. Následuje element `settings-bar`, který obsahuje elementy pro načítání dat. Konkrétně textový vstup `load-date`, po jeho klepnutí se objeví kalendář pro výběr data a času. Tento kalendář pochází z knihovny flatpickr[7], která je uložená v souborech `flatpickr.js` a `flatpickr.min.css`. Dále tento element obsahuje tlačítko `load-button` a elementy `wait_timer` a `fetch_counter`. Mimo element `settings-bar` následuje v souboru prázdný element `__root`.

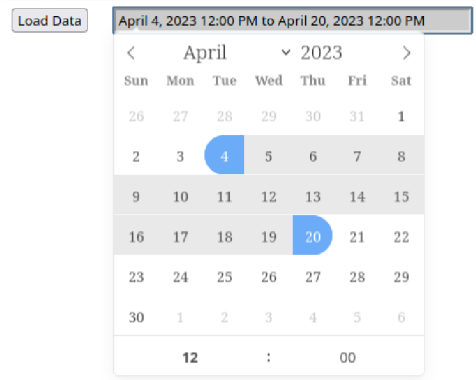
Po klepnutí tlačítka `load_button` se vezme rozmezí dat a časů z kalendáře a začnou se volat data pomocí funkce `update_router_elements`, která bude vysvětlena později. V elementu `wait_timer` se začne odpočítávat trvání zpracování dat. Tento element je zde pro vědomí uživatele, aby věděl, že aplikace stále pracuje. V elementu `fetch_counter` se zobrazí stávající počet požadavků. Když jsou všechny požadavky splněny a data načtena, časovač se zastaví. Postup si můžete prohlédnout na obrázcích 6.2, 6.3 a 6.4. Časovač stojí na objektu `wait_timer`, který je vytvořený za pomoci knihovny easytimer[8], která je uložená v souboru `easytimer.js`. Jedná se o jednoduchý časovač, který při každé změně sekundy zapíše aktuální čas do elementu `wait_timer`. Počet požadavků a kontrola časovače je závislá na třídě `SpecialCounter` a elementu `fetch_counter`. Jedná se o jednoduchý objekt s vnitřním počtem a metodami `write_to_element`, `raise` a `lower`. Metoda `write_to_element` zapisuje aktuální počet požadavků do elementu `fetch_counter`. Metody `raise` a `lower` pak zvyšují a snižují vnitřní počet o jedna. Při zvýšení či snížení dojde k zavolání metody `write_to_element` a kontrole časovače. Pokud je počet větší jak nula a časovač je zastavený, dojde k jeho spuštění. Pokud se počet stane nulovým a časovač běží, dojde k jeho zastavení. Tímto způsobem je vizualizována práce na pozadí. Objekt `fetch_counter` je předán jako reference funkci `update_router_elements`, která ho předává dále. Metoda `raise` je volána před započetím každého dotazu a `lower` je volána po jeho skončení. Všechny získané informace jsou uloženy do elementu `__root`.

Data jsou zobrazena v elementech třídy accordion a panel, kde accordion je tlačítko, které při klepnutí mění styl následujícího elementu, v tomto případě se mění styl elementu `div` mezi styly `none` a `block`. Kód pro tuto funkčnost a vizuální styl byl převzat ze stránky W3Schools[9].

[7] https://github.com/flatpickr/flatpickr [cit. 2023-04-20]
[8] https://github.com/albert-gonzalez/easytimer.js [cit. 2023-04-20]
[9] https://www.w3schools.com/howto/howto_js_accordion.asp [cit. 2023-04-20]



Obrázek 6.1: Adresářový strom rozšíření



Obrázek 6.2: Výběr rozmezí pomocí kalendáře

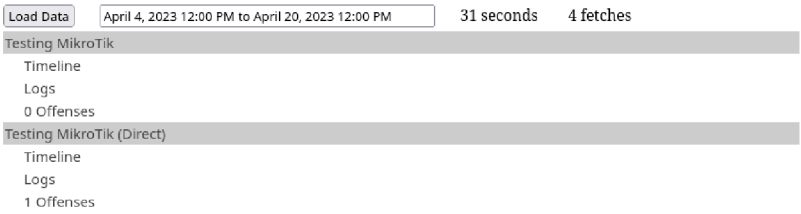
Data jsou vizualizována třemi způsoby. První je časová linka, která obsahuje body všech událostí a aktivních incidentů na dvou linkách (Aktivním incidentem je myšlen takový incident, který nebyl označen uživatelem v systému QRadar jako uzavřený.). Modrá linka obsahuje události a červená linka obsahuje incidenty. Při najetí myší na daný bod se zobrazí zpráva, která danou událost způsobila, nebo název incidentu. Navíc se zobrazí datum a čas vzniknutí události nebo incidentu. Linka byla implementována za pomoc kódu „Simple Responsive Horizontal Timeline“[10]. Tento kód je původně implementován za pomoci knihovny jQuery, se kterou však neumím pracovat. Z tohoto důvodu byl kód přeložen do „čistého“ jazyka JavaScript bez použití této knihovny. Tento překlad byl vytvořen pomocí ChatGPT[11]. Výsledný překlad jsem pak opravila, ověřila jeho funkčnost a upravila pro své potřeby. Tento výsledný kód je uložen ve funkci `timeline`.

Druhou vizualizací je seznam všech zpráv, které byly obdrženy v daném rozmezí. Tato vizualizace je velice jednoduchá, na řádcích je zobrazen čas, kdy byla zpráva obdržena, a zpráva samotná. Poslední vizualizací jsou otevřené incidenty. Tyto incidenty jsou nezávislé na vybraném časovém rozmezí. Toto chování je z důvodu, že incidenty považuji za natolik důležité, že jsou zobrazovány všechny. Tato data a části jsou shromážděny a vytvořeny pro každý zdroj událostí typu Mikrotik RouterOS (systém implementovaný v této práci). V příkladech je možné vidět zdroj událostí „Testing MikroTik“ a „Testing MikroTik (Direct)“.

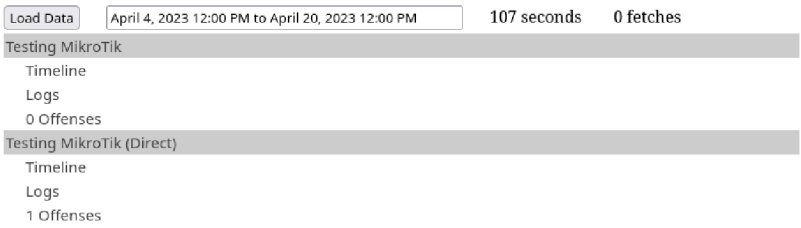
Nejdůležitější částí zobrazovací vrstvy je soubor `log_visualization.js`. Tento soubor se skládá téměř výhradně z funkcí dvou typů. První typ vytváří elementy HTML, či vkládá hotové elementy do stránky HTML. Druhý typ se dotazuje na adresy URL, ze kterých získává informace ze systému QRadar a naplňuje je do elementů. Během vývoje přibyly funkce, které momentálně nejsou používány, ale byly ponechány, kdyby se jejich funkcionalita vracela do programu. Jako příklad popíši funkci `update_router_elements` a návazné funkce pro bližší vysvětlení funkcionality.

Funkce `update_router_elements` přijímá ID „kořenového elementu“, do kterého mají být vytvořené nové elementy vloženy, rozsah časů a `fetch_counter` pro vizualizaci postupu práce. Před zavoláním `fetch` je zvýšen `fetch_counter`, čímž je okamžitě spuštěna vizualizace trvání a počet dotazů. Následně je dotazována adresa `/get/routers`, ze které je vyzvednut seznam všech „routerů“ (zdrojů událostí typu Mikrotik RouterOS). Bližší popis

[10] https://codepen.io/Seigiard/pen/MWwoqQ [cit. 2023-04-20]
[11] https://openai.com/blog/chatgpt [cit. 2023-04-20]



Obrázek 6.3: Načítání dat, čekání na dokončení volání



Obrázek 6.4: Všechna data načtena

o tom, jak fungují jednotlivé adresy, je v sekci 6.2.2. Funkcí `get_router_element_top` je ověřeno, zda byl daný router předtím zpracován. Pokud již má element vytvořen, je zavolána funkce `update_router` k aktualizaci tohoto elementu. Pokud element neexistuje, je navíc zavolána funkce `create_router`, která ony elementy vytvoří. Po zavolání funkcí nad všemi elementy je `fetch_counter` snížen. Funkce `create_router` je prostá funkce, která podle jména vytvoří páry elementů accordion a panel pro daný router a tři druhy vizualizace, časovou linku, zprávy a incidenty. Všechny tyto elementy jsou pak vloženy do „kořenového elementu“. Každý element má taktéž identifikátor, podle kterého je nalezen, a následně naplněn daty z ostatních funkcí.

Naplnění dat dochází pomocí dříve zmíněné funkce `update_router`. Tato funkce však žádná volání nedělá, místo toho volá funkce `update_offenses_element` pro naplnění elementu s incidenty, `update_raw_container` pro naplnění elementu se zprávami a `make_timeline` pro naplnění elementu s časovou linkou. Všechny tři funkce fungují koncepčně stejně. Začínají zvýšením `fetch_counter` a voláním adres, jen v tomto případě jsou součástí adresy proměnlivé argumenty. U všech adres se jedná o ID routeru, u časové linky a zpráv je navíc časové rozmezí. ID routeru systém zná z úvodních informacích o routerech. Zde jsou ony adresy:
`/get/offenses/{router_id}`
`/get/raw/{router_id}/{time_start}/{time_end}`
`/get/timeline/{router_id}/{time_start}/{time_end}`
Time start a time end jsou rozmezí, mezi kterým se má hledat.
Adresy `/get/raw` a `/get/timeline` lze zavolat i bez času. V tomto případě `get_data.py` pak zobrazí informace z posledního měsíce.

Zde je příklad, jak může vypadat zavolaná adresa:
`/get/raw/162/2023-04-04%2012:00/2023-04-20%2012:00`
Po obdržení dat funkce převedou JSON do objektů, na jejichž základě začnou vytvářet elementy HTML. U incidentů a zpráv se jedná pouze o vytvoření elementů div, které reprezentují tato data. U časové linky zpracování dochází za pomoci funkce `timeline`.

Je důležité poznamenat, že současná implementace nepočítá s tím, že by mohlo dojít k neobdržení dat. Pokud dojde k takové situaci, časovač bude stále počítat čas, počet volání zůstane neměnný a uživatel se nijak nedozví, že došlo k chybě.

6.2.2 Dotazovací vrstva

Dotazovací vrstva je primárně složena ze souborů `get_data.py` a `search.py`, v druhé řadě pak ze souborů `objects.py` a `custom_ariel.py`. Smyslem této vrstvy je dělat volání na systém QRadar a poskytovat výsledky. Konkrétně v souboru `views.py` jsou definovány cesty, při jejichž zavolání jsou zavolány patřičné funkce z modulu `get_data.py`. Výstupem z těchto funkcí jsou objekty, které jsou pak ve `view.py` serializovány do formátu JSON a v zobrazovací vrstvě v soboru `log_visualization.js` následně deserializovány. Volání na systém QRadar probíhají pomocí knihovny QPyLib, jedná se však o jednoduchá volání REST API. QPyLib však všechna volání vhodně zabaluje a poskytuje kontrolu adresy a autentikace, proto byl použit.

Všechny dotazy probíhají pomocí protokolu HTTPS, během kterého dochází k ověřování certifikátu SSL. Při instalaci systému QRadar, předinstalovaný certifikát je neověřitelný. Z tohoto důvodu by volání selhala, pokud by nedošlo k nahrazení certifikátu podepsaného známou autoritou. Snažila jsem se najít nějakou okliku, pomocí které by šlo stáhnout certifikát lokální certifikační autority, kterou si systém QRadar vytvoří, přidat ho mezi ostatní certifikáty, a tímto způsobem ověřit. Po mnoha testech a různých způsobech se mi však nepovedlo najít takový způsob, který by šlo reprodukovat v prostředí, které systém QRadar potřebuje. Z tohoto důvodu jsem nakonec implementovala drastičtější řešení ve formě vypnutí ověření SSL certifikátu. Toto řešení sice umožňuje, aby byl systém napaden, ale vzhledem k tomu, že při nasazení a vestavení rozšíření do systému QRadar je výsledné řešení uzavřené před vnější sítí, považuji toto riziko za přijatelné. Z tohoto důvodu bylo však nutné udělat ještě jednu úpravu. Část knihovny QPyLib, konkrétně část, která se stará o komunikaci s databází Ariel, neumožňuje vypnutí verifikace SSL. Tato část však volá jiné části knihovny QPyLib, které toto vypnutí umožňují. Rozhodla jsem se proto vytvořit kopii, kterou jsem uložila do souboru `custom_ariel.py`. Tato upravená verze má jediný rozdíl, a to sice má boolovskou proměnnou `VERIFY`, jejíž obsah předává při volání zbytku knihovny QPyLib. Tímto způsobem dojde k vypnutí verifikace a pokud se zbytek knihovny změní, nemělo by dojít k nekompatibilitě.

Tato vrstva získává informace dvěma způsoby, voláními REST API a hledáním v databázi Ariel. Jelikož hledání v databázi je rozděleno na několik částí, byl pro zabalení těchto částí a snadnější manipulaci vytvořen moddul `search.py`. Modul `search.py` se primárně skládá ze čtyř funkcí. První tři funkce jsou velice prosté a prakticky zabalují funkce z modulu `custom_ariel.py`. Funkce `search_start` na vstupu požaduje validní dotaz jazyka databáze Ariel. Vrací identifikátor hledání. Funkce `search_status` na vstupu očekává identifikátor hledání a vrací stav. Funkce `search_result` na vstupu očekává identifikátor hledání, které již bylo dokončeno, a vrací výsledky hledání. Funkce `search` pak používá těchto tří funkcí pro kompletní hledání. Začne hledání, čeká, dokud hledání není dokončeno, a následně vy-

zvedne výsledky a vrátí je. Je nutné podotknout, že žádná část neočekává, že by mohlo dojít k chybě. Při chybě během hledání dojde k pádu programu.

Během implementace došlo ke změně plánu, která data bude potřeba získávat. Z tohoto důvodu jsou v modulu `get_data.py` i funkce, které momentálně nejsou používány. Zmíním se tedy pouze o funkcích, které jsou v současné implementaci používány. Důležitou proměnou je proměnná `VERIFY`, která je předávána ve všech voláních API k vypnutí verifikace SSL.

Funkce `get_routers` začíná tím, že zavolá funkci `get_event_source_type_id`. Tato funkce nedělá nic jiného než to, že posílá požadavek o vyzvednutí informace o všech typech zdrojích událostí, jejichž typ má jméno Mikrotik RouterOS (jméno modulu DSM). Tento název je uložen v globální proměnné modulu. Očekává se, že bude obdržen pouze jeden typ a jeho ID se vrátí. V případě obdržení více typů funkce vyvolá selhání programu. S tímto identifikátorem funkce `get_routers` volá opět na systém QRadar o obdržení všech zdrojů událostí získaného typu předtím. Tímto způsobem je kód přenositelný, i když bude instalován na jiném systému QRadar, a typ zdroje událostí Mikrotik RouterOS bude mít jiný identifikátor. Po obdržení seznamu je překopírován název a identifikátor daného zdroje událostí (routeru) a výsledný list objektů je předán. Touto funkcí začíná aktualizace informací ve zobrazovací vrstvě. Tato funkce je volána z adresy `/get/routers/`. Ještě jednodušší funkcí je funkce `get_offenses`, která je na adrese `/get/offenses/`. Tato funkce provádí pouze jedno volání k získání všech incidentů, daného identifikátoru routeru, které jsou ověřené. Všechna příchozí data jsou vrácena bez úpravy.

Trochu složitějšími funkcemi jsou `get_raw` a `get_timeline`, které jsou na adresách `/get/raw/` a `/get/timeline/`. Obě funkce začínají relativně podobně, skládáním dotazu pro databázi Ariel. Dotazy jsou podobné dotazům z databáze SQL. Oba dotazy z obou funkcí jsou si podobné, ptající se na starttime (čas, kdy přišla zpráva), endtime (čas, kdy skončilo zpracování zprávy) a payload (zprávu samotnou). Funkce časové linky se navíc ptá o qid, které se používá pro identifikaci typu události. Jelikož obě funkce mohou být volány jak s časovým rozmezím, tak bez něj, je dotaz tomu uzpůsoben a je přidána či odebrána podmínka, která omezuje požadované události. Po vytvoření dotazu je volán modul `search.py`. Výsledek hledání je pak ověřen a v úvodu je zpracován funkcí `process_payloads`.

Uložení událostí v databázi Ariel má dvě neblahé vlastnosti. První je, že události nemají žádný unikátní identifikátor, podle kterého by šly identifikovat. Druhou je, že zpráva uložená ve sloupci payload je enkódovaná pomocí BASE64, tedy nečitelná pro lidi. Funkce `process_payload` se snaží vyřešit oba tyto problémy nejdříve dekódování zprávy. V druhé řadě použitím zprávy, starttime a endtime se pokusí vytvořit identifikátor. K jeho použití je využit hashovací algoritmus FNV-1a[12], 128 bitů. Hex tohoto bitového hashe je pak použit jako identifikátor. Vycházím z předpokladu, že mezi každou událostí, zprávou, starttime a endtime, alespoň jeden prvek bude unikátní a algoritmus vytvoří dostatečně odlišně vypadající identifikátor.

Zpracované události jsou pak vráceny. Funkce `get_raw` pouze pomocí funkce `copy_from_dict_to_dict` překopíruje starttime jako timestamp do výsledného listu slovníků a vrátí výsledek. Funkce `get_timeline` dělá zpracování navíc. Pomocí stejné funkce kopíruje starttime jako timestamp, ale navíc kopíruje i qid, které následně rozšiřuje pomocí funkce `populate_qid`. Funkce `populate_qid` projede všechny události. Když najde qid, které nezná, zavolá systém QRadar o informaci o daném qid. Výsledek si pak uloží. Ze známých qid pak k události překopíruje název a popis. Tyto informace nejsou momentálně

[12] http://www.isthe.com/chongo/tech/comp/fnv/index.html [cit. 2020-04-21]

v časové lince využívány, ale mohly by být užitečné, kdyby události byly rozřazeny do více linek. Funkce `get_timeline` pak volá funkci `get_offenses` k získání incidentů. Všechny tyto získané informace jsou pak zpracovány pomocí funkce `format_to_timeline` do struktury objektů, kterému rozumí funkce `timeline` ze zobrazovací vrstvy. Považuji za důležité podotknout, že jsou do této struktury nakopírovány i incidenty, které nejsou z daného časového rozmezí. Během testování však tyto incidenty nebyly vidět ve vizualizaci, může se to však později projevit.

Tímto jsou popsány všechny relevantní části rozšíření.

Kapitola 7

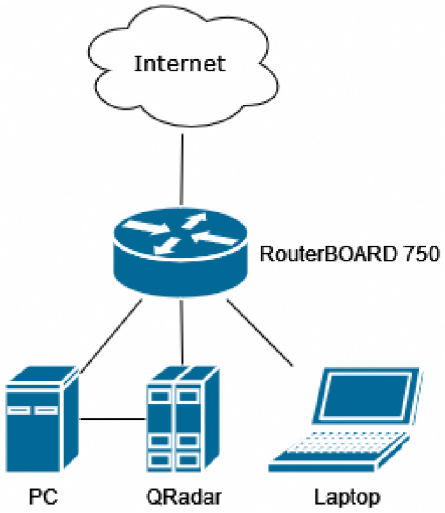
Testování řešení

Tato kapitola pojednává o testování navrženého řešení. Testování navržení řešení probíhalo experimentem. V experimentech jsem vytvořila modelové prostředí za využití obrazů systémů veřejně dostupných z internetu. Tato modelová prostředí jsem následně pozorovala a kontrolovala jejich chování. Konkrétně se prostředí skládá ze zařízení MikroTik RouterBOARD 750, systému QRaradar Community Edition, bežícího ve virtualizačním nástroji VirtualBox, a mého osobního laptopu, či jiných virtualizovaných zařízení. Do systému QRadar jsem se pokusila nainstalovat rozšíření pomocí exportovaných balíčků. Systém QRadar měl k dispozici 6 vláken z procesoru AMD Ryzen 5 2600 a 10 GiB DDR4 RAM paměti. Po vytvoření datového toku jsem pozorovala, zda-li se objevují zprávy v systému QRadar, zda-li jsou úspěšně identifikovány, zda-li mají zpracované vlastnosti a zda-li je rozšíření schopné tato data úspěšně reprezentovat.

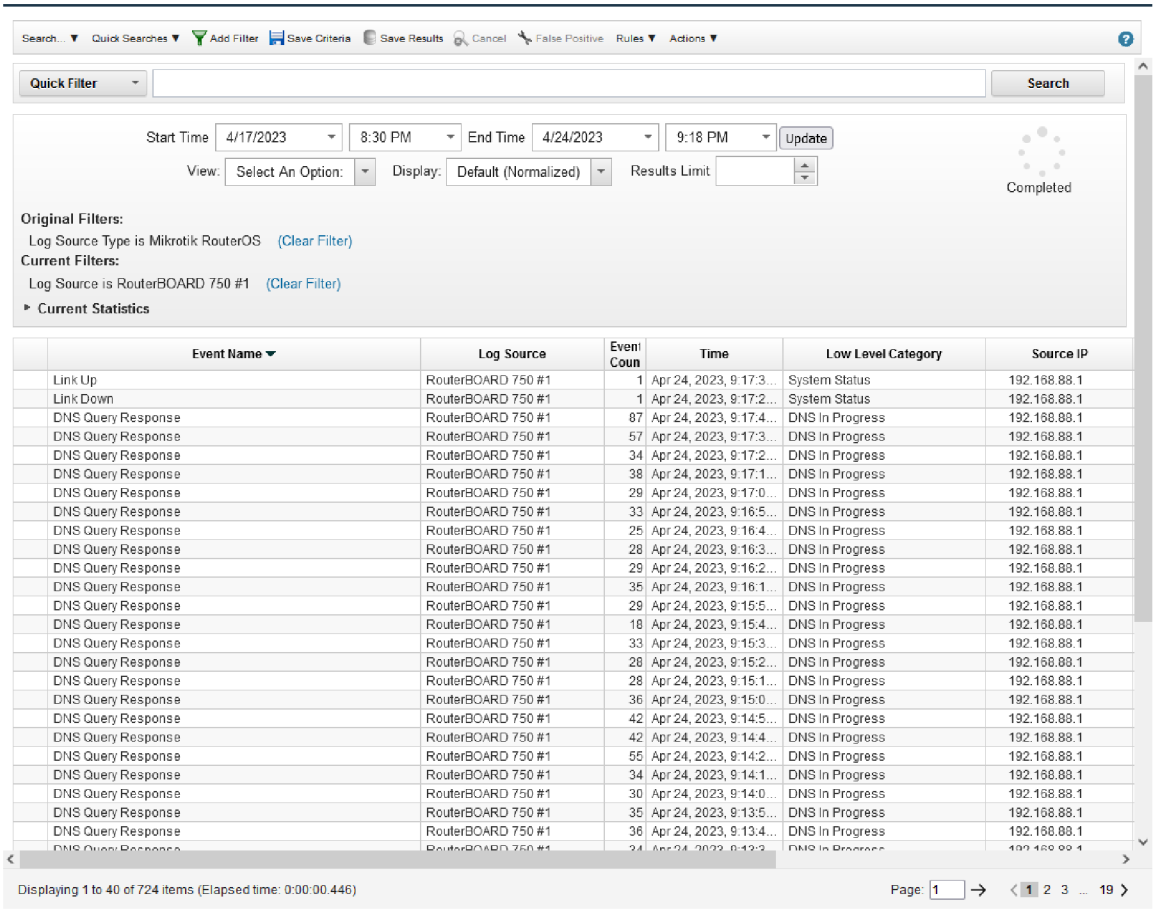
7.1 Normální provoz

V prvním testu chceme ověřit, zda-li všechny systémy byly úspěšně implementovány a zda-li jsou přenositelné. Konkrétně se v tomto testu soustředíme na zpracování událostí. V prvním testu byly součástí prostředí můj osobní počítač, systém QRadar, zařízení MikroTik RouterBOARD a můj osobní laptop. Systém QRadar běžel virtualizován na mém počítači. Systém QRadar a laptop byly připojeny k zařízení MikroTik, které se chovalo jako router a bylo připojeno k internetu. Diagram sítě je možné vidět na obrázku 7.1. Laptopem byl generován datový tok formou pozorování videí a hledání otázek pomocí internetového vyhledávače. Očekávaným výsledkem je zobrazení událostí v systému QRadar. Tyto události by taky měly mít název popisný jejich stavu, což reprezentuje správné mapování QID. Zprávy by měly obsahovat vlastnosti, vzhledem k jejich typu. Vzhledem k typu akce se očekává, že zprávy budou převážně ohledně rezoluce DNS adres a minimum zpráv ohledně akcí serveru DHCP. V rozšíření Add-on bylo pak vybráno rozpětí datumů, které odpovídají trvání tohoto testu.

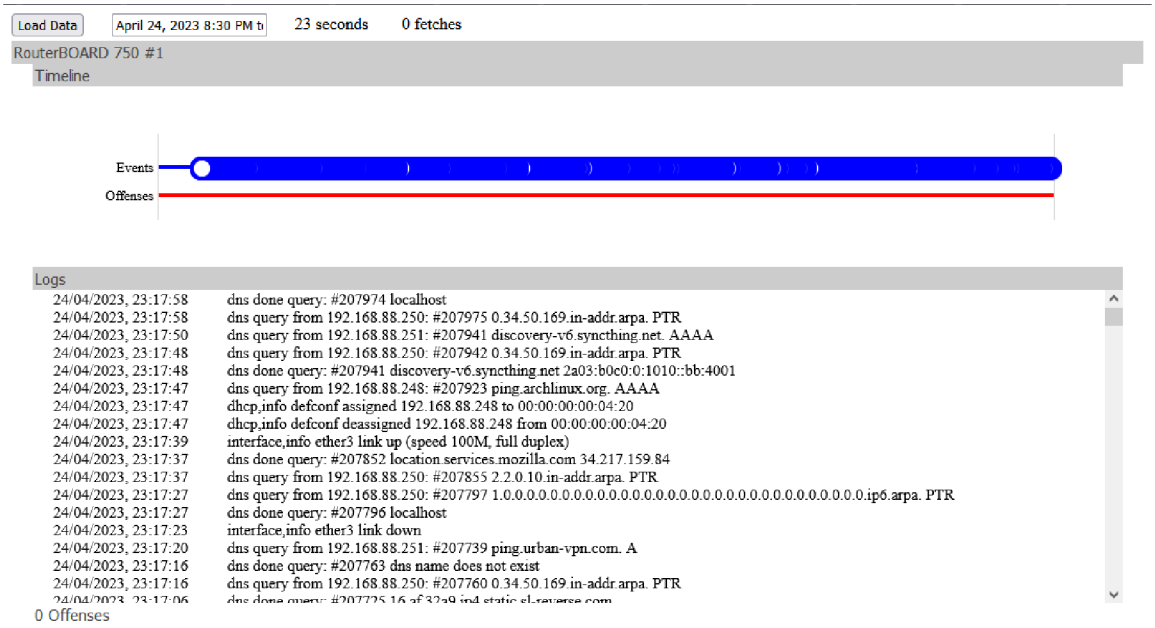
Po 48 minutách běhu jsem laptop odpojila a prohledala záložku „Log Activity“. V ní jsem dala vyhledat všechny zprávy z daného zdroje událostí. Celkem bylo nalezeno 724 událostí, z toho 721 patřilo rezoluci DNS, 4 serveru DHCP. Nebyly nalezeny žádné nezpracované zprávy. Na obrázku 7.2 si můžete prohlédnout zpracované zprávy v systému QRadar. Rozšíření Add-on po 23 sekundách úspěšně zobrazilo všechny zprávy a časovou linku. Rozšíření si můžete prohlédnout na obrázku 7.3.



Obrázek 7.1: Diagram sítě prvního experimentu



Obrázek 7.2: Záložka „Log Activity“ ukazující události úspěšně zpracované modulem DSM



Obrázek 7.3: Záložka rozšíření Add-on ukazující vizualizované informace

7.2 Útok Brute-Force

Druhý test má za úkol otestovat úspěšnou reakci systému QRadar na incident, který je vytvořen pouze za nově vytvořených událostí, a tímto zjistit, jestli mají události správné mapování a jestli je systém schopný reagovat na tyto události bez úpravy pravidel.

V druhém testu bylo využito prostředí z prvního testu s drobnou úpravou. Na osobní laptop jsem nainstalovala nástroj ncrack. Ze seznamu běžných hesel jsem stáhla seznam 100 nejpoužívanějších hesel.[1] Na konec tohoto seznamu jsem přidala skutečné heslo, které RouterBOARD používá. Při volání nástroje jsem vybrala útok na ssh připojení. Tímto způsobem je emulován úspěšný útok na RouterBOARD. Toto chování by mělo vytvořit minimálně jeden incident, který by měl být zobrazitelný v záložce „Offenses“, a vizualizace tohoto incidentu by se měla zobrazit na rozšíření Add-on.

Během vývoje se mi povedlo snadno vyvolat toto chování, během testování se mi však nedařilo toto chování reprodukovat. Nejdříve jsem se pokusila útočit ze svého laptopu. Zařízení RouterBOARD se bránilo a dočasně zastavilo možnost dalších pokusů. Po chvíli jsem mohla útok opakovat a úspěšně dokončit, na straně systému QRadar to však nevyvolalo žádnou odezvu. Pokusila jsem se reprodukovat chování, které původně vedlo k vyvolání incidentu, avšak bez úspěchu. V dalším kroku jsem se pokusila dát systému QRadar chvíli na vzpamatování, zkusila jsem vypnout zařízení MikroTik a vypla jsem záznamy DNS, aby systém QRadar mohl věnovat více výpočetní síly k zjištění tohoto incidentu. To však bylo bez úspěchu. Nakonec jsem zkusila vypnout shromažďování událostí, opět bez úspěchu.

Navzdory mým pokusům se mi nepodařilo vyvolat očekávané chování a reprodukovat chování z implementace. Z tohoto důvodu jsem se rozhodla exportovat události, které vedly k vytvoření incidentu během implementace ve formátu XML, a přiložit je na paměťové

[1] https://github.com/danielmiessler/SecLists [cit. 2023-04-24]

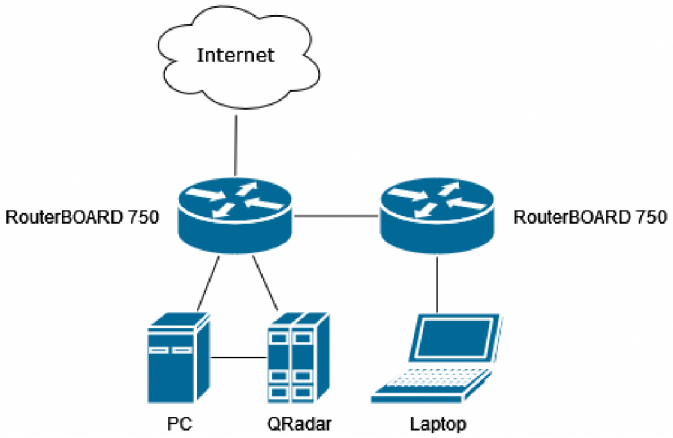
médium. Tento export může být nalezen v souboru `2023-04-24-data_export.xml.zip`. Tento export je taktéž dostupný na mém účtu GitHub.[2]

7.3 Více routerů

Třetí test má za úkol vyzkoušet model, který je více reprezentativní skutečného nasazení. Taktéž, jestli je systém schopen úsečně rozpoznat od sebe několik zdrojů událostí.

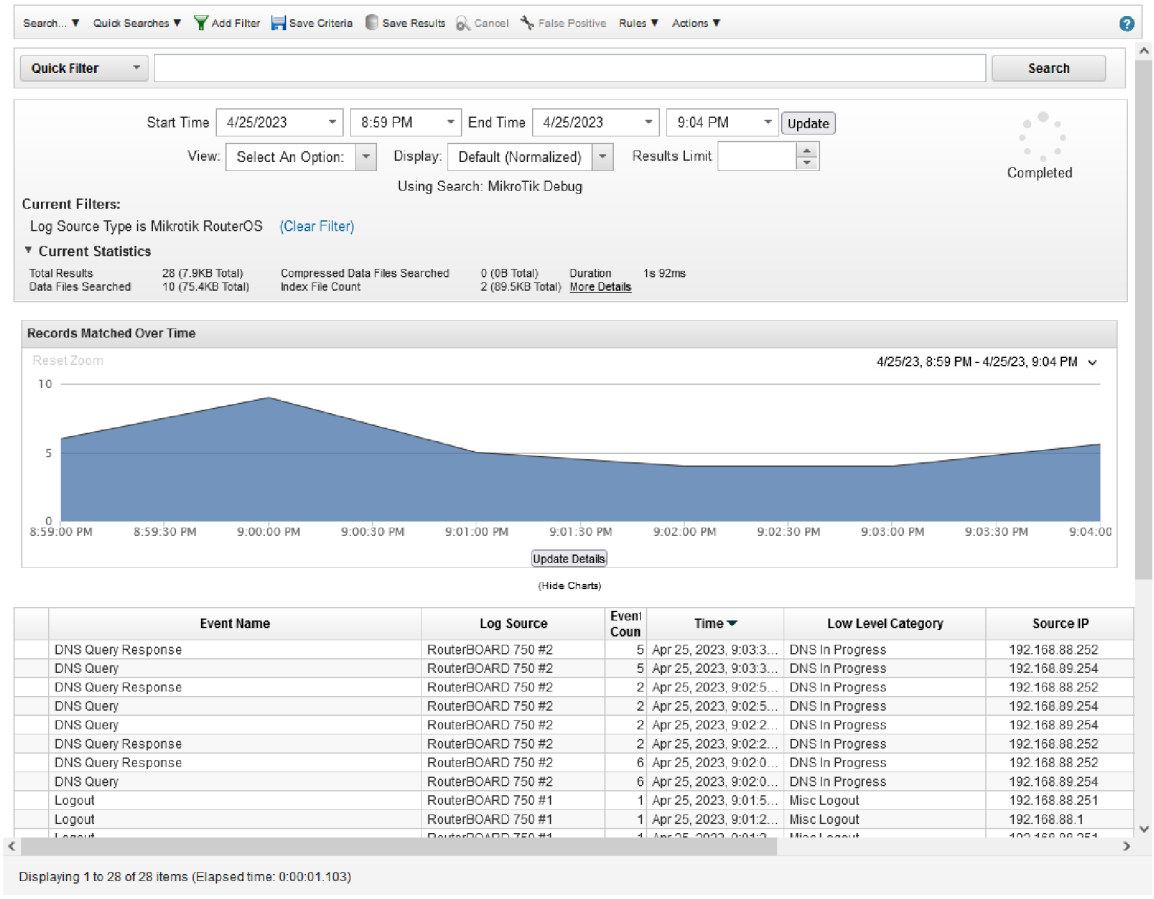
Toto prostředí je podobné prvnímu prostředí. Do kořenového routeru, který má starší verzi RouterOS, konkrétně verzi 5.9., byl zapojen původní router, model RouterBOARD 750, verze RouterOS 6.48.6. Systém QRadar byl připojen ke kořenovému routeru. Ke druhému routeru byl připojen osobní laptop. Kořenový router spravoval síť 192.168.88.1. Druhý router spravoval síť 192.168.89.1. Diagram sítě je možné vidět na obrázku 7.4.

Po 5 minutách byl experiment zastaven. Celkově bylo nashromážděno 28 událostí, z toho 22 patřilo kořenovému routeru a 6 druhému routeru. Tento nepoměr byl způsoben tím, že kořenový router má starší verzi operačního systému, která nepodporuje zasílání zpráv o rezoluci DNS, které tvoří většinou zpráv. Ze všech událostí 22 tvořily události DNS a 6 událostí patřilo přihlášení k routeru. Tato přihlášení byla vytvořena mnou, aby byly vůbec nějaké záznamy vidět, bez záznamů DNS přichází do systému QRadar malé množství zpráv. Na obrázku 7.5 jde vidět události v systému QRadar. Rozšíření trvalo 10 sekund data načíst a úspěšně zobrazilo všechny zdroje událostí a všechny události. Rozšíření si můžete prohlédnout na obrázku 7.6.

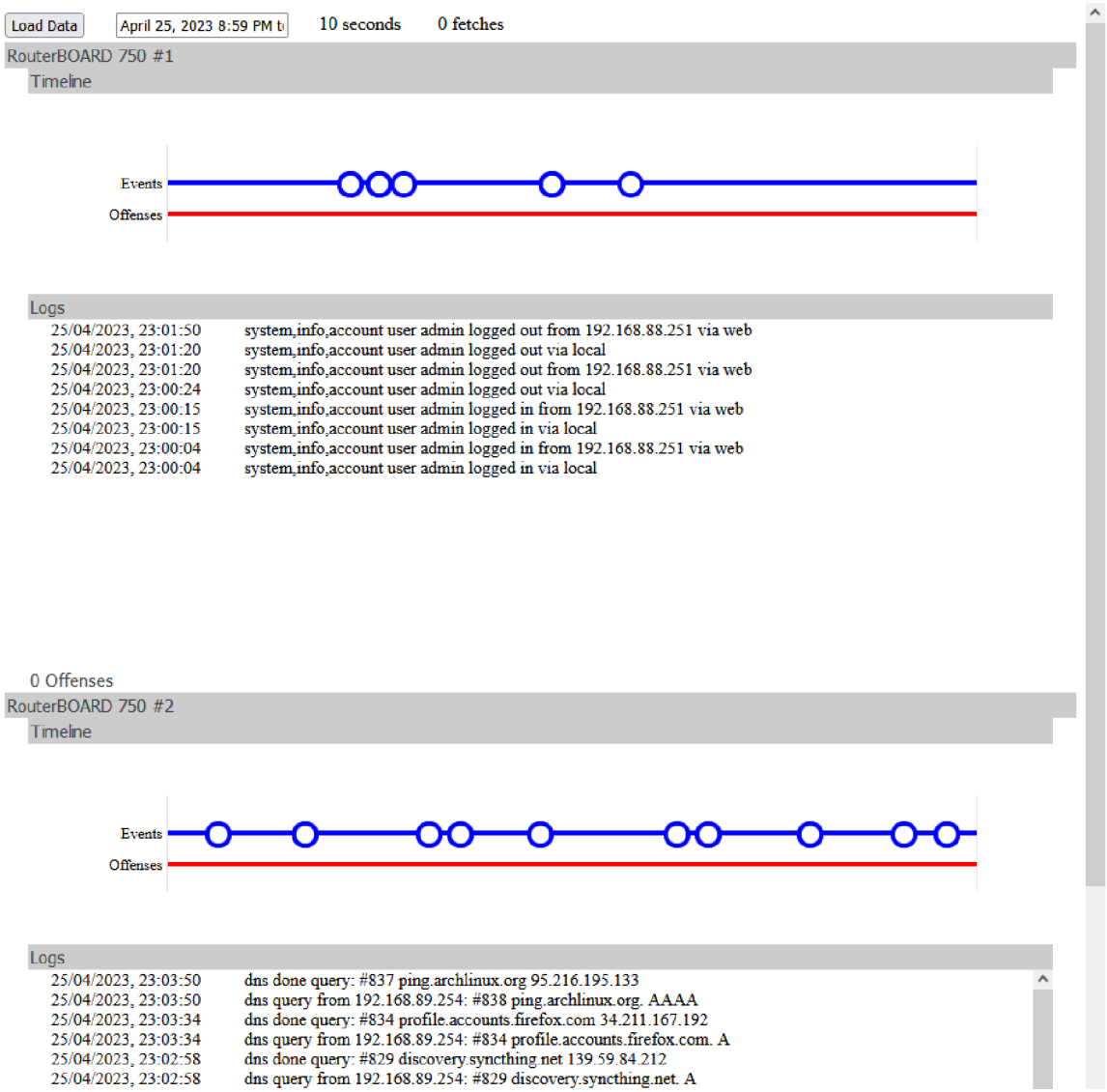


Obrázek 7.4: Diagram sítě třetího experimentu

[2] https://gist.github.com/yellowfox-star-is/da85256f1d28ee10cc2323d2fc2bfec9 [cit. 2023-04-24]



Obrázek 7.5: Záložka „Log Activity“ ukazující rozdílné zdroje událostí



Obrázek 7.6: Záložka rozšíření Add-on ukazující vizualizované informace více routerů

7.4 Zhodnocení experimentů

Rozšíření Add-on je přenositelné a chovalo se očekávaným způsobem, načítání dat bylo rychlé. Během implementace načítání dat trvalo velmi dlouho, toto se během experimentů však neprojevilo. Modul DSM je taktéž přenositelný a úspěšně zpracoval všechny příchozí události bez jakýchkoliv nalezených problémů. Je však nutné podotknout, že nebyly vůbec testovány zprávy IPv6, či zprávy Wi-Fi.

Kapitola 8

Závěr

V této práci jsme si představili problematiku IBM QRadar, zpracování událostí, operační systém RouterOS a formát zasílaných zpráv. Během analýzy vznikl vzorek zasílaných zpráv, což je samo o sobě prospěšné, protože takovýto vzorek nebyl dostupný. Na základě provedené analýzy byl představen návrh modulu DSM pro zachytávání těchto informací, včetně rozšíření typu Add-on pro zobrazení těchto informací. Následně jsem navržený modul DSM a rozšíření Add-on implementovala, testovala a opravila nedostatky. Výstupem této práce je samostatný modul DSM, který umí zpracovávat běžné zprávy ze systému RouterOS a rozšíření Add-on, které umí takto zpracované informace vizualizovat.

Na poli jak modulu DSM, tak rozšíření Add-on je mnoho způsobů jak navazovat. Rozšíření je například možné dát hezčí vzhled, rozšířit vizualizované vlastnosti, optimalizovat rychlost zpracování dat či umožnit výběr zdrojů událostí, které se vizualizují. Modul DSM bych rozšířila o zpracování zpráv Wi-Fi, protokol IPv6 a pak zkoumat nějaké zprávy méně využívaných protokolů, které jsem neměla čas prozkoumat. Při implementaci jsem si například všimla, že nová verze RouterOS podporuje protokol WireGuard, který nebyl v této práci zpracován. Dále by bylo užitečné testovat rozdílná zařízení firmy MikroTik a ověřit, zda-li jsou všechny jejich zprávy úspěšně zpracovány, případně dle nových druhů zpracování upravit.

Dovolím si pár poznámek pro možné pokračovatele mé práce, ohledně systému QRadar. Implementace pro systém QRadar z pohledu studenta je velice náročná a pomalá. Dokumentace k určitým částem systému QRadar chybí, je skrytá, či je naopak tak rozsáhlá, že je nečitelná v poskytnutém časovém rámci. Volná licence pro akademické účely tohoto systému je zastaralá. V době psaní práce byla Komunitní Edice systému QRadar na verzi 7.3.3, přičemž komerční verze 7.4.3 ztrácí podporu. Tedy licence je pro zastaralou verzi. Navíc systém QRadar je velice náročný na výpočetní sílu a jeho provoz na uživatelském hardware je pomalý a náchylný k pádům.

Literatura

[1] Abusamrah, I., Madhoun, A. a Iseed, S. *Next-Generation Firewall, Deep Learning Endpoint Protection and Intelligent SIEM Integration* [online]. Palestine Polytechnic Universities - Computer Engineering, 2021 [cit. 2023-05-26]. Dostupné z: https://scholar.ppu.edu/bitstream/handle/123456789/7533/%d9%85%d8%b4%d8%b1%d9%88%d8%b9%20%d8%a7%d9%84%d8%aa%d8%ae%d8%b1%d8%ac%20%28%d8%a7%d8%a8%d8%b1%d8%a7%d9%87%d9%8a%d9%85%20%d9%88%20%d8%b3%d8%a7%d8%b1%d8%a9%20%d9%88%d8%a7%d9%81%d9%86%d8%a7%d9%86%29.pdf.

[2] Chakrabarty, B., Patil, S. R., Shingornikar, S., Kothekar, A., Mujumdar, P. et al. *Securing Data on Threat Detection by Using IBM Spectrum Scale and IBM QRadar: An Enhanced Cyber Resiliency Solution* [online]. 3. vyd. IBM Redbooks, září 2021 [cit. 2022-11-19]. 64 s. ISBN 9780738460017. Dostupné z: https://www.redbooks.ibm.com/abstracts/redp5560.html.

[3] Chakraborty, N. Intrusion detection system and intrusion prevention system: A comparative study. *International Journal of Computing and Business Research (IJCBR)*. 2013, sv. 4, č. 2, s. 1–8.

[4] Gupta, S., Chaudhari, B. S. a Chakrabarty, B. Vulnerable network analysis using war driving and security intelligence. In: *2016 International Conference on Inventive Computation Technologies (ICICT)*. 2016, sv. 3, s. 1–5. DOI: 10.1109/INVENTIVE.2016.7830165.

[5] IBM. *About IBM* [online]. [cit. 2022-12-17]. Dostupné z: https://www.ibm.com/about.

[6] IBM. *IBM QRadar Application Framework Guide* [online]. 2020 [cit. 2022-12-19]. Dostupné z: https://www.ibm.com/docs/en/SS42VS_SHR/pdf/b_qradar_appframework_devguide.pdf.

[7] IBM. *IBM QRadar DSM Configuration Guide April 2021* [online]. 2021 [cit. 2022-10-06]. Dostupné z: https://www.ibm.com/docs/en/SS42VS_7.5/pdf/b_siem_deployment.pdf.

[8] IBM. *IBM QRadar 7.5 Architecture and Deployment Guide* [online]. 2022 [cit. 2022-11-05]. Dostupné z: https://www.ibm.com/docs/en/SS42VS_7.5/pdf/b_siem_deployment.pdf.

[9] Lonvick, C. *The BSD Syslog Protocol* [RFC 3164 (Informational)]. Request for Comments (RFC) 3164. Internet Engineering Task Force (IETF), srpen 2001. Obsoleted by RFC 5424. Dostupné z: http://www.ietf.org/rfc/rfc3164.txt.

[10] MIKROTĪKLS. *About us* [online]. [cit. 2022-11-25]. Dostupné z: https://mikrotik.com/aboutus.

[11] MIKROTĪKLS. *RouterOS Documentation* [online]. [cit. 2022-11-24]. Dostupné z: https://help.mikrotik.com/docs/display/ROS/RouterOS. Path: Getting started; Software Specifications;.

[12] OKMIANSKI, A. *Transmission of Syslog Messages over UDP* [RFC 5426 (Proposed Standard)]. Request for Comments (RFC) 5426. Internet Engineering Task Force (IETF), březen 2009. Dostupné z: http://www.ietf.org/rfc/rfc5426.txt.

[13] QIU, T., GE, Z., PEI, D., WANG, J. a XU, J. What Happened in My Network: Mining Network Events from Router Syslogs. In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: Association for Computing Machinery, 2010, s. 472–484. IMC '10. DOI: 10.1145/1879141.1879202. ISBN 9781450304832. Dostupné z: https://doi.org/10.1145/1879141.1879202.

[14] SLAVICEK, K., LEDVINKA, J., JAVORNIK, M. a DOSTAL, O. Mathematical Processing of Syslog Messages from Routers and Switches. In: *2008 4th International Conference on Information and Automation for Sustainability*. 2008, s. 463–468. DOI: 10.1109/ICIAFS.2008.4783957.

[15] YAMANISHI, K. a MARUYAMA, Y. Dynamic Syslog Mining for Network Failure Monitoring. In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. New York, NY, USA: Association for Computing Machinery, 2005, s. 499–508. KDD '05. DOI: 10.1145/1081870.1081927. ISBN 159593135X. Dostupné z: https://doi.org/10.1145/1081870.1081927.

Příloha A

Obsah přiloženého paměťového média

Přiložené médium obsahuje:

- tento dokument ve formátu PDF,
- zdrojové soubory této práce ve složce `text source`,
- zabalený modul DSM ve složce `dsm`,
- zabalené rozšíření typu Add-on ve složce `addon`,
- zdrojové soubory rozšíření ve složce `addon/source`,
- qid mappings ve souboru `qid_mappings.csv`,
- qid records ve souboru `qid_records.csv`,
- záznamy z analýzy zpráv v souboru `dataset.log`,
- skripty použité pro kategorizaci zpráv ve složce `log extraction scripts`,
- instalační manuál a návod k použití v souboru `manual.md`.