VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÁ APLIKACE PRO VÝVOJ A ÚDRŽBU KORELAČNÍCH PRAVIDEL SIEM SYSTÉMŮ

WEB APPLICATION FOR DEVELOPMENT AND MAINTENANCE OF SIEM SYSTEM CORRELATION RULES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Oliver Bielik

AUTHOR

VEDOUCÍ PRÁCE — Ing. Yehor Safonov

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Oliver Bielik ***ID:*** 231229

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Webová aplikace pro vývoj a údržbu korelačních pravidel SIEM systémů

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je návrh a implementace webové aplikace, která umožní efektivní vývoj korelačních pravidel pro účely dohledu nad počítačovou infrastrukturou. Aplikace taktéž poskytne možnost integrace s vybraným systémem typu SIEM. Funkcionalita aplikace bude zahrnovat správu verzí zdrojového kódu korelačních pravidel (např. pomocí integrace s Git platformou) a disponovat možností ukládání sad pravidel pro různé počítačové infrastruktury. Grafické uživatelské rozhraní aplikace umožní správci SIEM systému kontrolovat změny pravidel, vracet se ke starším pravidlům a snadno opravovat chyby. Aplikace bude propojena s jedním SIEM systémem (např. Netwitness XDR, ELK, QRadar), a to pomocí API rozhraní. Vzhledem k tomu, že pravidla obsahují citlivé informace, bude aplikace splňovat základní bezpečnostní požadavky. V teoretické části nastudujte problematiku bezpečnostního monitoringu, srovnejte existující SIEM řešení a jejich přístupy pro vývoj korelačních pravidel. Nastudujte syntaxi pravidel a funkcionalitu vybraného API. Definujte funkční, nefunkční a kritické požadavky kladené na webovou aplikaci. Výsledné řešení otestujte na vývoji třech korelačních pravidel.

DOPORUČENÁ LITERATURA:

[1] MIKOWSKI, Michael; POWELL, Josh. Single page web applications: JavaScript end-to-end. Manning Publications Co., 2013.

[2] KOTENKO, Igor; CHECHULIN, Andrey. Attack modeling and security evaluation in SIEM systems. International Transactions on Systems Science and Applications, 2012, 8: 129-147.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 26.5.2023

Vedoucí práce: Ing. Yehor Safonov

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Dnešný svet technológií sa rýchlo a neustále rozvíja. Rovnako rýchlo sa tvoria nové riziká, ktoré ohrozujú túto sféru. Z tohto dôvodu je potrebné technológie monitorovať a brániť vniknutiu nebezpečenstvám do systémov. Jednou z technológií, ktorá pomáha tejto ochrane je SIEM systém. Tento systém slúži ako investigačný nástroj, ktorý umožňuje vykonávať bezpečnostný monitoring a vyšetrovanie. Bezpečnostný monitoring sa vykonáva na základe korelačných pravidiel, ktoré sú vyvíjané v bezpečnostných operačných centrách (SOC). Ich úlohou je vyhľadávať potencionálne nebezpečenstvá a ohlasovať ich. Hlavným cieľom predkladanej bakalárskej práce je vytvorenie nástroju, ktorý umožní vývojárom v SOC jednoduchý vývoj korelačných pravidiel. Cieľom aplikácie je zjednodušiť vývoj a zabezpečiť lepší prehlad nad jednotlivými korelačnými pravidlami. Teoretická časť bakalárskej práce sa zameriava na problematiku bezpečnostného monitoringu, ktorý sa snaží čitateľovi priblížiť. Bližšie opisuje fungovanie systému a prácu SOC operátorov, ktorých náplňou práce je taktiež vývoj korelačných pravidiel. Praktická časť bakalárskej práce je zameraná na uľahčenie vývoju týchto pravidiel. Bakalárska práca je ukončená záverom, čitateľovi stručne popisuje zistené skutočnosti a spracovanie požiadaviek na bakalársku prácu.

KĽÚČOVÉ SLOVÁ

API, Flask, GitLab, SIEM, SOC, bezpečnostný monitoring, framework, git, korelačné pravidlo, požiadavok, webová aplikácia.

ABSTRACT

Today's world of technology is developing rapidly and constantly. Just as quickly, new risks are forming that threaten this sphere. For this reason, technologies need to be monitored and hazards prevented from entering systems. One of the technologies that helps this protection is a system called SIEM. This system serves as an investigative tool that allows security monitoring and investigations to be carried out. Security monitoring is carried out based on the correlation rules that are developed in security operations centers (SOC). Their task is to look for the potential dangers and report them. The main goal of the presented bachelor thesis is to create a tool that allows developers in SOC to easily develop correlation rules. The aim of the application is to simplify development and ensure a better overview of individual correlation rules. The theoretical part of the bachelor thesis focuses on the issue of security monitoring and explains it to the reader. It describes in more detail the functioning of the system and the work of SOC operators, whose job is the development of correlation rules as well. The practical part of the bachelor thesis is aimed at facilitating the development of these rules. The last part of the bachelor thesis is a conclusion, it briefly describes to the reader the observed facts and processing of the requirements for the bachelor thesis.

KEYWORDS

API, Flask, GitLab, SIEM, SOC, crelation rule, framework, git, request, security monitoring, web application.

Vysadené pomocou balíčku `thesis` verzie 4.07; `http://latex.feec.vutbr.cz`

BIELIK, Oliver. *Webová aplikace pro vývoj a údržbu korelačních pravidel SIEM systémů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 108 s. Bakalárska práca. Vedúci práce: Ing. Yehor Safonov

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Oliver Bielik
VUT ID autora:	231229
Typ práce:	Bakalárska práca
Akademický rok:	2022/23
Téma závěrečnej práce:	Webová aplikace pro vývoj a údržbu korelačních pravidel SIEM systémů

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Veľmi rád by som poďakoval vedúcemu mojej bakalárskej práce pánovi Ing. Yehorovi Safonovi za podporu, časté konzultácie, za odborné vedenie a vhodné podnety na úpravu. Za veľa trpezlivosti pri riešení problémov, ktoré sa počas bakalárskej práce vyskytovali.

Obsah

Zoznam obrázkov

Úvod

Každým rokom sa zrýchľuje rozvoj informačných technológií. Rovnakým tempom sa k nim pridávajú rôzne hrozby, ktoré na tieto technológie útočia. Napríklad medziročný nárast DDoS útokov na rôzne infraštruktúry je 30 % [1]. Spoločnosti, ktoré čelia takýmto útokom, môžu byť vyradené z prevádzky niekoľko hodín a stratiť veľké množstvo financií. Voči tomuto a iným útokom je možné sa chrániť rôznymi technológiami. Jedným z často využívaných riešení je bezpečnostný monitoring alebo SIEM.

Využíva dve základné časti správu bezpečnostných udalostí (SEM) a správu bezpečnostných informácií (SIM). Ich kombinácia tvorí silný nástroj na monitorovanie siete a zachytávanie podozrivej aktivity. SIEM funguje na princípe zbierania logov a ich korelácie pomocou korelačných pravidiel. Tieto pravidlá sú vytvárané bezpečnostnými analytikmi v bezpečnostných operačných centrách alebo aj SOC.

Pri vývoji pravidiel môže dochádzať k rôznym chybám a potrebám ich úpravy, alebo vracaniu sa k staršej verzii pravidla. Nakoľko SIEM neposkytuje takúto funkcionalitu, je potrebné vytvorenie takéhoto nástroja. Aplikácia takéhoto typu by výrazne uľahčila vývoj a opravu pravidiel a ich jednoduchú revíziu.

Cieľom tejto bakalárskej práce je vytvoriť systém, ktorý bude vývojárom umožňovať jednoduchú tvorbu a správu korelačných pravidiel. Vývoj týchto pravidiel bude užívateľovi jednoducho prístupný pomocou webovej aplikácie. Aplikácia bude integrovať systém GitLab, ktorý jej umožni jednoducho a bezpečne ukladať a sledovať vývoj pravidiel.

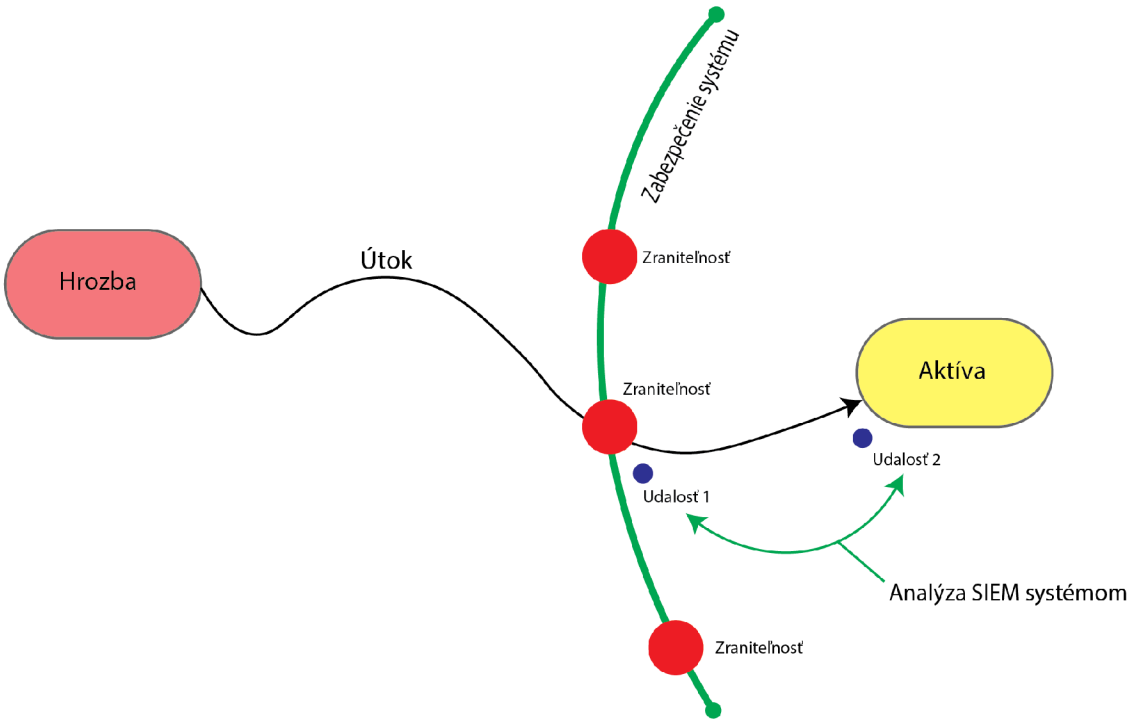
Logika tohto ukladania je postavená na tom, že užívateľ odošle pravidlo pomocou mikroservisy a tá ho následne spracuje a uloží do príslušnej zložky v systéme GitLab. Pravidlá budú triedené podľa názvu SIEM systémov, z ktorých pochádzajú. Každý systém má vlastnú štruktúru korelačného pravidla, preto je nesmierne dôležité ich triediť do vlastných zložiek, aby bola ich evidencia prehľadnejšia.

V rámci teoretickej časti bude priblížená problematika bezpečnostného monitoringu, proces vývoju korelačných pravidiel a analýza potrebných technológií k vývoju aplikácie, zľahčujúcej vývoj korelačných pravidiel. Praktická časť bude zameraná na implementáciu analyzovaných technológií a samotný vývoj mikroservís, ktoré sa budú starať o spracovávanie prichádzajúcich dát z webovej aplikácie.

Na konci dokumentu sa nachádza záver, ktorý stručne zhrnie splnené požiadavky stanovené bakalárskou prácou. Zhrnie čo bolo dosiahnuté v rámci bakalárskej práce.

1 Bezpečnostný monitoring počítačovej infraštruktúry

V dnešnej digitálnej dobe sa väčšina našej práce odohráva v kyber priestore, preto je nesmierne dôležité tento priestor chrániť. Rozvoj informačných technológií priniesol viaceré výhody, ale aj nevýhody. Uľahčujú nám život tým, že si my, ako používatelia, nemusíme pamätať obrovské množstvo dát, ktoré dennodenne využívame - ako príklad môžeme uviesť heslá, bankové účty, platobné karty, informácie o bydlisku alebo zdravotné záznamy. Tieto informácie sú zväčša uložené na serveroch rôznych spoločností. Ak si teda vytvoríme bankový účet, banka naše osobné údaje spoločne s účtom uloží na svoje servery. Nakoľko takáto spoločnosť spravuje citlivé údaje a narušenie ich integrity by mohlo napáchať vysoké škody, je potrebné takúto infraštruktúru chrániť. Škoda môže byť spôsobená rôznymi spôsobmi. Útočník môže ukradnúť osobné informácie a prihlasovacie údaje, čo môže viesť k odcudzeniu financií používateľa. Ďalšou škodou, ktorú môže páchateľ spôsobiť, je vyradenie prevádzky spoločnosti. Ak takúto ujmu spôsobí, zabráni užívateľom prístup k poskytovaným službám a spoločnosť začne strácať peniaze. [2]



Obr. 1.1: Demonštrácia útoku a vytvorenie bezpečnostnej udalosti. [3]

Druhy útokov, ktoré môže útočník využívať:

- **Útok škodlivým softvérom** – je to bežný útok, ktorým útočník podvrhne škodlivý softvér užívateľovi, aby nadobudol prístup k jeho zariadeniu [5]. Pojem

škodlivý softvér zahŕňa všetky druhy škodlivého softvéru, vrátane najznámejších foriem, ako sú trójske kone, ransomware, vírusy a červy [4].

- **SQL injection útok** – tento útok využíva SQL príkazy k napadnutiu databáz. Na základe tohto útoku je útočník schopný získať neautorizovaný prístup do databázových súborov. To mu umožňuje úpravu, alebo zmazanie dát, následne je schopný vážne poškodiť funkcionalitu systému. [5]
- **Phishingové útoky** – zámer tohto útoku je získať dôverné informácie ako napríklad číslo platobnej karty alebo prihlasovacie údaje. Útočník sa tieto údaje snaží získať podvrhnutím falošného e-mailu, ktorý však vyzerá dôveryhodne. Použiť môže taktiež falošnú webstránku, tá vyzerá rovnako ako tá, kde sa užívateľ chce prihlásiť. [5]
- **Botnets** – pod týmto pojmom sa rozumie sieť napadnutých počítačov. Útočníci ich môžu využiť na rozsiahle DDoS útoky. Následne nad nimi môžu prevziať kontrolu alebo vykonávať automatizované úlohy - zasielanie škodlivého softvéru, alebo nevyžiadanej pošty do e-mailových schránok používateľov.[6].
- **DoS a DDoS útoky** – útočník týmto útokom dokáže zabrániť používateľovi prístup k aplikácii. Aby útočník dokázal zabrániť používateľom k pripojeniu, posiela veľké množstvo falošných požiadaviek na server. Týmito požiadavkami sa server zahltí, čo spôsobí zastavenie odpovedí na požiadavky alebo serveru veľmi dlho trvajú odpovede. [5]

Jedným z najčastejších útokov je práve DDoS (*Distributed Denial of Service*). Priemerná dĺžka takéhoto útoku v druhom štvrťroku 2022 bola 3000 minút, v prepočte sú to približne 2 dni [7]. Strata spoločnosti sa v priemere pohybuje v sume $5 600 za minútu, čo predstavuje približne $336 000 za hodinu [8].

Keďže útoky môžu vážne poškodiť prevádzku spoločností je potrebná náležitá ochrana voči útokom. V informačných technológiách sa tejto ochrane hovorí aj kyber bezpečnosť. Cieľom je chrániť online poskytované služby pred ich prerušením, exploitáciou, alebo dáta pred ich krádežou. [9].

Kyber bezpečnosť je rozdelená do 5 častí. Sieťová bezpečnosť, internetová bezpečnosť, bezpečnosť koncových zariadení, cloudová bezpečnosť a bezpečnosť aplikácií. Tieto časti budú rozobrané v texte nižšie. [10]

Sieťová bezpečnosť

Je to súbor technológií, ktoré chránia určitú infraštruktúru, bránením pred vstupom alebo šírením potencionálneho nebezpečenstva v sieti. Efektívna sieťová bezpečnosť sa skladá z viacerých vrstiev zabezpečenia. K tomu, aby bola ochrana účinná sú potrebné aj nasledovné nástroje. [11]

- **Firewall** – je software alebo zariadenie, ktoré sleduje prichádzajúci a odchádzajúci tok dát. Tieto dáta sú sledované na základe nakonfigurovaných pravidiel a filtrov [11].
- **Load Balancer** – slúži k rovnomernému rozmiestneniu záťaže na servre. Pokiaľ teda máme k dispozícii viacero serverov, tak sa požiadavky rovnomerne rozdelia medzi ne [11].
- **IDS/IPS** – je zariadenie umiestnené v ceste prichádzajúcich paketov. Toto zariadenie je schopné blokovať nežiadúce prvky už na hranici siete [12].
- **Sandbox** – je software, ktorý zachytáva neznáme súbory vstupujúce do siete. Vykonáva ich analýzu a spúšťa ich vo virtuálnom prostredí. Analýzou vie do niekoľkých minút správne identifikovať nový alebo upravený malware. [13]
- **NTA** – využíva kombináciu strojového učenia, behaviorálneho modelovania a detekovania založeného na rôznych pravidlách a lokalizácií anomálií a podozrivých aktivít. [14]
- **NDR** – riešenie, ktorého účelom je využívať pokročilé analytické techniky napríklad strojové učenie na odhalenie podozrivej aktivity. [15]
- **Proxy** – sprostredkúva bezpečné pripojenie na internet. Funguje ako brána, ktorá stojí medzi užívateľom a navštevovanými online webovými stránkami. Bráni napríklad pred škodlivým softwarom, ktorý by sa mohol v online svete nachádzať. [16]

Internetová bezpečnosť

Pozostáva z ochrany informácií ktoré sú posielané a prijímané pomocou internetu. Taktiež je zameraná na zabezpečenie webových aplikácií. Tieto typy ochrany slúžia ako prevencia pred vniknutím škodlivého softvéru do zariadení používateľov. [10] Chránia nás pred:

- **Počítačovými červami** – sú programy, ktoré sa samovoľne dokážu rozmiestniť po sieti, rozmnožiť sa alebo je možné ich prenášať pomocou prenosových médií. [17]
- **Botnetmi** – sieť napadnutých počítačov nad, ktorými dokáže mať útočník kontrolu. Viacej informácii je sa nachádza na začiatku kapitoly 1.
- **Škodlivým softvérom** – je taký software, ktorý dokáže poškodiť zariadenie alebo na ňom vykonávať nežiadúce aktivity [18]. Viac informácií na začiatku kapitoly 1.

Bezpečnosť koncových zariadení

Je to ochrana zameraná na chránenie koncových zariadení. Ako sú napríklad počítače alebo telefóny. Koncové zariadenie môže vytvárať bránu do siete spoločnosti

pre útočníka, ktorý ju môže následne exploitovať. Bezpečnosť koncových zariadení chráni vstupy pred ich zneužitím. [19] Technológie, ktoré pomáhajú zlepšiť bezpečnosť koncových zariadení sú nasledovné:

- **EDR** – Detekcia a odozva koncového bodu, je integrovaný koncový bod bezpečnostného riešenia, ktorý monitoruje a zbiera dáta na koncovom bode. Dokáže ich zanalyzovať a vykonávať potrebné operácie [20].
- **Antivírus** – Je to bezpečnostný software, ktorý vyhľadáva a detekuje kybernetické hrozby. Po detekcii ich odstraňuje dokáže aj brániť ich rozšíreniu sa v systéme [27].
- **Firewall operačného systému** – Firewall je bezpečnostné zariadenie v prípade OS firewallu ide o software, ktorého hlavným účelom je chrániť zariadenie pred nežiadúcim neautorizovaným prístupom do zariadenia [28].

Cloudová bezpečnosť

Je časť kyberbezpečnosti určená k zabezpečeniu cloudových výpočtových systémov. Zahŕňa držanie dát v súkromí naprieč celou infraštruktúrou, ktorá môže pozostávať z rôznych serverov, aplikácií a platforiem.[21]. Bezpečnosť cloudu sa skladá z dvoch pilierov. Pozostáva zo zodpovednosti poskytovateľa a klienta[26].

- **Poskytovateľ** - má za úlohu chrániť fyzickú infraštruktúru, aby nedošlo k nepovolenej manipulácii so serverom. Konfigurovať fyzickú sieť a vydávať pravidelné updaty systému, aby bol zabezpečený pred novými hrozbami.
- **Klient** - spravuje užívateľov a ich oprávnenia. Zabezpečuje chránenie cloudových účtov pred nepovoleným prístupom a šifrovanie ukladaných dát.

Nasledujúca časť popíše technológie, ktoré sa využívajú pri cloudovej bezpečnosti. Tieto technológie sú:

- **Správa identít a prístupu (IAM)** – tento systém poskytuje napríklad zamestnancom firmy bezpečný prístup k citlivým firemným údajom. IAM sa začal využívať s prestupom ľudí z kancelárií do domácností. Vznikla tu potreba chrániť dáta, ktoré chcú užívatelia získať. Systém overuje identitu užívateľa, ktorý žiada prístup k dátam. [22]
- **Bezpečnostný monitoring (SIEM)** – riešenie, ktoré pomáha organizáciám zisťovať a reagovať na bezpečnostné hrozby a riziká skôr ako napáchajú škodu v infraštruktúre. [23]
- **Agent zabezpečenia prístupu do cloudu (CASB)** – tento agent je umiestený medzi používateľmi a poskytovateľom cloudových riešení. Môže zabezpečovať viacero rôznych služieb, napríklad mapovanie a overovanie prihlasovacích údajov, zisťovanie malvéru alebo aj šifrovanie. [24]
- **Prevencia úniku údajov (DLP)** – citlivé dáta môžu uniknúť zo spoločnosti

úmyselne alebo neúmyselne. Dáta môžu byť odcudzené napríklad prostredníctvom vírusu, ktorý sa dostal do zariadení užívateľov. DLP technológia slúži na identifikáciu a ochranu citlivých dát chráni ich pred odcudzením aj stratou. [25]

Bezpečnosť aplikácií

Celosvetovo sa dnes používa viac ako 14 miliárd zariadení, schopných pripojenia do IoT (Internet of things) [29]. Tieto zariadenia majú dostatočnú výpočtovú silu aby mohli vykonávať zložitejšie úlohy. Na tieto zariadenia je možné inštalovať rôzne aplikácie prístupné na internete. Na internete sa nachádza približne 8,9 milióna aplikácií [30]. Mnohé z týchto aplikácií nemusia byť dostatočne zabezpečené, alebo môžu byť vytvorené s nekalím úmyslom ohroziť ich používateľov. Z toho dôvodu je dôležité dbať na to, aké aplikácie sú sťahované do našich zariadení. Druhy aplikácií:

- **Mobilné aplikácie**
- **Počítačové aplikácie**
- **Webové aplikácie**

Zvýšiť bezpečnosť aplikácii a tak chrániť ich používateľov je možné rôznym testovaním vyvýjaných aplikácií. Podľa existujúcich štandardov alebo softwareom na to určeným. Jeden z používaných štandardov je od spoločnosti OWASP s názvom OWASP MASVS viac si je možné prečítať na webovej stránke.[31]

Koncové zariadenia, smerovače, prepínače servery všetky tvoria počas ich práce logové záznamy alebo logy. Tieto záznamy sa taktiež vytvárajú aj aplikáciami počas ich používania, alebo samotným operačným systémom, ktorý je používaný zariadením.

Zápisy logov sú nevyhnutné pre počítačové systémy a aplikácie. Pomáhajú im splniť bezpečnostné nároky a požiadavky na spoľahlivosť. Tvorba týchto záznamov vytvára históriu diania určitého systému, aplikácie alebo zariadenia. Ak nastane chyba v nejakom procese, môže byť história zápisov použitá na analýzu toho, čo chybu vyvolalo. [33]

Tieto záznamy sa ďalej využívajú k detekcii potencionálnych útokov na infraštruktúru. Pre zabezpečenie ochrany infraštruktúry je potrebné, aby sa logy rýchlo spracovali, analyzovali a vyhodnotili. Takéto úkony človek nie je schopný robiť sám, nakoľko by musel v reálnom čase kontrolovať tisícky záznamov, čo možné nie je. K uľahčeniu tejto práce slúžia SIEM (*Security Information and Event Management*) systémy, ktoré je možné nasadiť na rozsiahlu infraštruktúru. SIEM systémy sú ďalej prebraté v kapitole (1.3).

The Open Worldwide Application Security Project

OWASP je svetová nezisková organizácia, ktorá sa zaoberá zlepšovaním bezpečnosti softwareu. Snaží sa toho dosiahnuť pomocou open-source projektov, ktoré sú pod vedením ich komunity. Skoro 20 rokov ich podporujú korporácie, developeri a dobrovoľníci. Snažia sa rozširovať znalosti medzi ľudí rôznymi konferenciami, ktoré uskutočňujú. [34]

Jedným zo známych projektov je OWASP Top Ten. Je to dokument obsahujúci najkritickejšie bezpečnostné riziká webových aplikácií. Tento dokument je globálne uznávaný medzi vývojármi a je to jeden z prvých krokov k bezpečnému vývoju kódu. Najnovšie dáta sú z roku 2021, k tomuto roku OWASP vyhlásil nasledovné kategórie bezpečnostných rizík: [35]

- **Broken Access Control**;
- **Cryptographic Failures**;
- **Injection**;
- **Insecure Design**;
- **Security Misconfiguration**;
- **Vulnerable and Outdated Components**;
- **Identification and Authentication Failures**;
- **Software and Data Integrity Failures**;
- **Security Logging and Monitoring Failures**;
- **Server-Side Request Forgery**;

Vývojári webových aplikácií by sa mali riadiť týmito kategóriami aby sa predišlo rizikám, pri používaní webovej aplikácie. OWASP aktualizuje tento zoznam pravidelne každé 3 až 4 roky. [36]

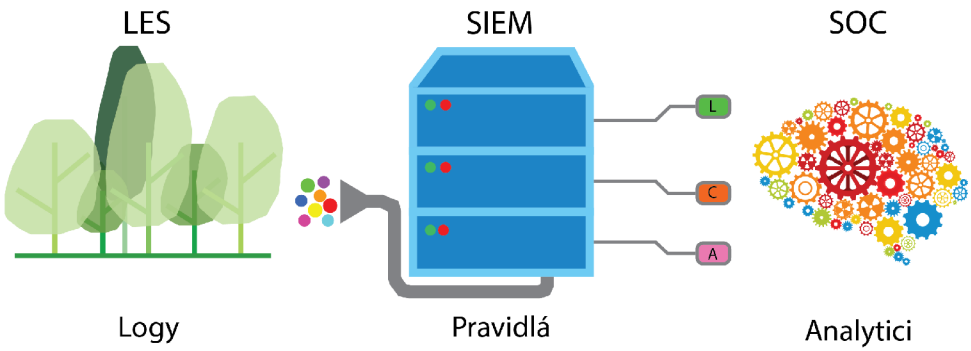
1.1 Charakteristika logových záznamov

Logy sú súbory, ktoré uchovávajú dáta, vytvorené operačným systémom a aplikáciami týkajúce sa vykonávaných aktivít. Z týchto záznamov je možné vytvoriť udalosť, ktorá odpovedá určitému dianiu systéme alebo sieti. [32]

Sú dôležitou súčasťou operačných systémov. Na základe logov je možné vyhodnocovať správnu funkcionalitu systému, či vyčítať rôzne anomálie. V operačnom systéme sa vytvára história logových zápisov. Tieto záznamy je možné použiť napríklad pri poruche systému, aby sa vrátil do svojo normálneho stavu. [33]

Logy z rôznych systémov majú rôznu štruktúru príklad logu zo systému Windows je možné vidieť na obrázku 1.3. Tento log pochádza z webovej stránky [1]. Tento

[1]Viacero logov je možné získať z: `https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/`



Obr. 1.2: Schéma cesty logov od ich zdroja po analýzu v bezpečnostnom operačnom centre.

logový záznam sa vyskytuje pri spustená systému. Logon proces ktorý je možné vidieť na obrázku 1.3 je dôveryhodnou súčasťou operačného systému. Zabezpečuje funkciu rôznych logon funkcií ako napríklad RSA pripojenie. [37] Logy ktoré prejdu cez SIEM a sú v ňom spracované sa ďalej posunú do bezpečnostného centra kde ich analyzujú analytici viac o bezpečnostnom centre v kapitole 2.1. To ako logy cestujú z ich zdrojov až po ich analýzu zobrazuje obrázok 1.2.

```
A trusted logon process has been registered with the Local Security Authority.
This logon process will be trusted to submit logon requests.

Subject:

Security ID:  SYSTEM
Account Name:  MS4$
Account Domain:  WORKGROUP
Logon ID:  0x3e7
Logon Process Name:  IKE
```

Obr. 1.3: Príklad windows logu.

Jednotlivé logové záznamy sú kritickou súčasťou SIEM systémov. Delíme ich na: [32]

- **IOS Event Logy** – zariadenia s operačným systémom IOS logové záznamy nezbierajú avšak zbierajú zápisy o pádoch aplikácií. IOS zariadenia majú vlastné bezpečnostné funkcie, ktoré používajú API (*Application Programming Interface*) Pomocou tohto softvérového riešenia dokážu aplikácie tretích strán pristupovať k dátam z nasledujúcich služieb: bezpečnosť aplikácií, sieťová bezpečnosť, internetové služby, šifrovanie dát, správca hesiel, kontrola zariadenia a kontrola súkromia.

- **Android Event Logy** – narozdiel od IOS zariadení, android poskytuje logy z aplikácií, kernelu a C, C++ a Java záznamy.
- **Linux Event Logy** – pri systémoch, ktoré používajú Linux, je možné zostaviť časovú os. Tá sa skladá z udalostí týkajúcich sa kernelu, serveru a aplikácií.
- **Windows Event Logy** – windows, vytvára logy aktivít na základe hardvéru a softvéru pripojených k počítaču. Používa šesť základných kategórií tými sú aplikačné, systémové, bezpečnostné, adresárové logy a logy z DNS serveru a serveru aplikácií a súborov.

Následne sa logy delia na 5 častí, delenie podľa [38]:

- **Informačné** – hovoria len o stave udalostí.
- **Varovné** – označujú chýbajúcu funkciu alebo časť systému.
- **Chybové** – označujú chybu, ktorá môže ohroziť správne fungovanie systému.
- **Ladiace** – používajú sa pri vývoji korelačných pravidiel. Viac v kapitole 2.
- **Pohotovostné** – označujú udalosti spojené s bezpečnosťou.

1.2 Popis korelačných pravidiel

Korelovať znamená spracovávať prichádzajúci tok udalostí. Korelovaním sa snažíme identifikovať vzory udalostí v obrovskom objeme prichádzajúcich dát. Logika tohto spracovania sa nachádza v korelačných pravidlách. [39]

Korelačné pravidlo je mechanizmus, ktorý umožňuje automatizovať akcie, ktoré sledujú udalosti v reálnom čase. Toto pravidlo dozerá na dianie v sieti, pokiaľ sú splnené podmienky stanovené v pravidle, tak je spustený alarm. Písanie korelačného pravidla je komplexný proces, ktorý si vyžaduje odbornú znalosť. [39]

Táto časť bude demonštrovať tvorbu korelačného pravidla. Korelačné pravidla je tvorené na základe získaných logov. Na obrázku 1.4 je možné vidieť výpis 5 z týchto logov sú 4 o neúspešnom prihlásení a 1 o úspešnom prihlásení študenta. [39] Takýto

```
May 19 2023 09:24:13 server1 sshd[1234]: Failed password for student from 192.168.0.1 port 54321 ssh2
May 19 2023 09:25:07 server1 sshd[1234]: Failed password for student from 192.168.0.1 port 54321 ssh2
May 19 2023 09:26:00 server1 sshd[1234]: Failed password for student from 192.168.0.1 port 54321 ssh2
May 19 2023 09:26:54 server1 sshd[1234]: Failed password for student from 192.168.0.1 port 54321 ssh2
May 19 2023 09:27:48 server1 sshd[1234]: Accepted password for student from 192.168.0.1 port 54321 ssh2
```

Obr. 1.4: Ukážka výpisu logov o neúspešnom a následne úspešnom prihlásení.

výskyt môže indikovať pokus o neoprávnený pristúp do systému. Z tohto dôvodu by sa malo vytvoriť korelačné pravidlo, ktoré by dokázalo detektovať takýto jav. Obrázok 1.5 znázorňuje jednoduché pravidlo, ktoré detekuje neúspešné a následne úspešné prihlásenie študenta.

```
(Event.Source = "sshd" AND Event.Description CONTAINS "Failed password for" AND Source.IPAddress = "192.168.0.1")
FOLLOWED BY
(Event.Source = "sshd" AND Event.Description CONTAINS "Accepted password for" AND Source.IPAddress = "192.168.0.1")
```

Obr. 1.5: Ukážka výpisu korelačného pravidla.

1.3 Základná charakteristika SIEM systému

Pri obrovskom množstve IoT zariadení, nie je pre človeka možné sledovať všetky vytvárané logy a analyzovať ich v reálnom čase. V oblasti analýzy a vyhodnocovania logových záznamov sa používajú robustné SIEM systémy. Sú schopné vyhodnocovať dianie v infraštruktúre a podávať hlásenie SOC (*Security Operations Center*) tímu, podrobnejšie rozobrané v časti 2.1. SOC má na starosti ich správu, čiže tvorbu korelačných pravidiel a vyhodnocovanie alertov. Uľahčuje im prácu vďaka možnosti vytvárať automatizované bezpečnostné reporty a audity. [23]

SIEM (*Security Information and Event Management*) riešenie je unikátne v analyzovaní logov, prichádzajúcich zo všetkých častí IT (*Information technology*) infraštruktúry. Ktorá sa skladá z firewallov, IPS/IDS, serverov ako napríklad webových alebo proxy serverov. Všetky dôležité časti softvéru a hardvéru vytvárajú logové záznamy. Neexistujú logy, ktoré by SIEM systém nedokázal spracovať. Otázne však je ako efektívne budú jednotlivé záznamy analyzované. Pretože nie všetky logové zápisy sú dôležité alebo vyhodnotené správne. K vyhodnocovaniu logov slúžia korelačné pravidlá, viac o pravidlách v kapitole 2. Tento systém neslúži k odstraňovaniu hrozby, ale primárne k zbieraniu a korelovaniu bezpečnostných udalostí avšak niektoré SIEM riešenia ponúkajú aj možnosť eliminovať [40]. Odporúčané je začať do systému pridávať menšie množstvo zdrojov, ktoré vytvárajú logy. Ako napríklad hlavné firewally, Active Directory controller security event logs. Neodporúča sa analyzovať všetky prichádzajúce logy, nakoľko by to spôsobilo zahltenie a zlú analýzu. Vzhľadom na to, že cena SIEM systémov sa odráža aj na počte spracovaných logov, treba ich zdroje vyberať precízne a požívať tie najdôležitejšie.[41] Napríklad Microsoft uvádza viac ako 370 bezpečnostných udalostí ale z nich len 11 považuje za kritické [42].

SIEM sa skladá z dvoch hlavných častí. SIM (*Security Information Management*) a SEM (*Security Event Management*).

- **SIM** – je technológia, ktorá sa zaoberá zbieraním informácií z logov, môžu sa skladať z rôznych typov dát.[43] Toto umožňuje administrátorom vytvárať vlastné korelačné nástroje, ktoré špecifikujú dátové vzory a incidenty. Na základe týchto vzorov je možné spúšťať automatizovanú odpoveď.[44]
- **SEM** – proces zaoberajúci sa identifikovaním, zbieraním, monitorovaním a hlá-

senším bezpečnostných udalostí v sledovanej infraštruktúre. SEM umožňuje nahrávanie a hodnotenie udalostí, pomáha tým správcom riadiť vývoj bezpečnostnej architektúry.[45]

V roku 2018 uskutočnil Manfred Vielberth a Günther Pernul výskum. Ktorý priniesol 6 hlavných schopností, ktoré by mal každý SIEM systém spĺňať. Sú nimi zbieranie, normalizovanie, obohacovanie, korelovanie a analyzovanie, varovanie a odozva, hlásanie a výmena hrozieb. [46] Jednotlivé časti sú prebrané v nasledujúcej časti.

Zbieranie

Existuje viacero techník zbierania dát. Môžu byť vytiahnuté priamo zo zdroju dát alebo získané z externých zdrojov. Taktiež môžeme rozlišovať medzi centralizovaným a distribuovaným zbieraním informácii. [46]

Normalizovanie

Uľahčuje alebo umožňuje následné spracovanie dát. Normalizovaním sa takzvané surové dáta prekladajú do jednotného formátu. Je tu veľmi dôležitá synchronizácia s časom. [46]

Obohacovanie

Súvislosti medzi dátami hrajú veľkú rolu pri detekcii útokov. Preto predtým zozbierané logové informácie je potrebné obohatiť o kontextové dáta z rôznych zdrojov. [46]

Korelovanie a analyzovanie

Podľa Bračevaca sú korelácie mierené k vyvodeniu stavu prostredia (stavu zabezpečenia infraštruktúry, ktorú SIEM spravuje) s pomocou pozorovaných udalostí z viacerých zdrojov. Tým môžu byť útoky alebo všeobecne incidenty detekované automaticky alebo expertami na bezpečnosť. [46]

Varovanie a Odozva

Pokiaľ je detekovaný incident, musia byť informované všetky strany, ktorých sa týka daná udalosť. Následne musia byť uskutočnené vhodné opatrenia buď automatické alebo manuálne, pre zabezpečenie minimálnych škôd. [46]

Hlásenie a výmena hrozieb

Účel hlásenia je dvojaký. V prvom rade hlásenie bezpečnostných incidentov môže byť z povinnosti. Napríklad Európsky parlament a Kongres Spojených štátov amerických vytvorili zákony, ktoré požadujú hlásenie vzniknutých bezpečnostných incidentov v konkrétnych prípadoch. Po druhé väčšina SIEM riešení je napojená na platformy výmeny hrozieb, kde sú incidenty hlásené a zdielané s inými organizáciami. [46]

Obr. 1.6: Schéma SIEM systému.

Vďaka týmto schopnostiam je SIEM, schopný ponúknuť rozsiahlu ochranu pred útokmi. Obrázok 1.6 popisu ako jednotlivé časti na seba nadväzujú.

1.4 Porovnanie SIEM systémov

Ceny za software sa pohybujú medzi 20 000 - 1 000 000 dolárov, v priemere je to však okolo 50 000 dolárov. Čo teda predstavuje významnú investíciu aj pre veľkú spoločnosť.[47] Existuje viacero SIEM riešení, zaoberajú sa nimi aj rôzne svetoznáme podniky ako napríklad IBM alebo McAfee. Pre porovnanie bolo vybratých 13 riešení, ktoré si priblížime v nasledujúcej časti.

Datadog cloud SIEM

Vytvorený spoločnosťou Datadog je bezpečnostný monitoring na báze cloudu. Ponúka analyzovanie bezpečnostných logov v reálnom čase, bez ohľadu na ich množstve. Vývojári alebo bezpečnostný tím môžu využiť detailné údaje o nazbieraných dátach pre urýchlenie vyšetrovania na jednej zjednotenej platforme. Datadog ponúka viac ako 600 vstavaných funkcionalít pre lepšiu viditeľnosť sldeovanej siete. K dispozícii sú aj vstavané pravidlá, ktoré umožňuju rýchlo detekovať anomálie a nebezpečenstvá v sieti. Dokáže identifikovať bežné hrozby alebo útoky pomocou MITRE ATT&CK®

rozhraním. Umožňuje písanie vlastných pravidiel bez potreby učiť sa špeciálny jazyk. Zabezpečuje rýchlu detekciu pri komplexných pravidlách a skúmaní veľkého množstva prichádzajúcich dát. [48]

SolarWinds Security Event Manager

Ponúka rýchlejšiu detekciu podozrivej aktivity, pomocou detekovania koncovej aktivity používateľov. Pomáha agregovať logy z pfSense firewallu (viac o pfSense [49]). Ktoré zabezpečujú efektívnu prácu pri riadení zabezpečenia. Obrovské množstvo zbieraných dát sa ťažko spravuje a SolarWinds používa nástroj SIEMu – SEM, ktorý poskytuje hĺbkovú analýzu, kotrá aktívne sleduje pfSense logy a detekuje akúkoľvek podozrvú aktivitu. Riešenie od SolarWinds umožňuje korelovanie dát s inými udalosťami, ktoré sa vyskytujú v sledovanej sieti. Redukuje čas strávený auditom, keďže umožňuje určiť prípady, kedy napríklad je zvýšená návštevnosť, čo zabráni vyhodnoteniu udalosti ako falošne pozitívnej. SolarWinds umožňuje okamité liečenie vzniknutej bezpečnostnej sytuácie. Môže napríklad blokovať IP adresu, odhlásiť užívateľa alebo prerušiť proces. Vie taktiež zistiť prítomnosť BotNetu a eliminovať ho.[50]

LogPoint

Tradičné SIEM systémy sú drahé, vyžadujú si veľa skúseností s ich nasadením a udržiavaním. Logpoint ponúka riešenie založené na cloude, ktoré je možné rýchlo implementovať do siete podniku. O produkt sa starajú výskumníci zameraní na bezpečnosť, pravidelne aktualizujú schopnosti detekcie a odozvy, čo zvyšuje schopnosť reagovať včas pri vyskytnutom riziku. [51]

Graylog Security

Kombinuje kľúčové vlastnosti bezpečnostnej analýzy a detekovania anomálií. Umožňuje rýchle riešenie kritických situácií. Snaží sa eliminovať falošne pozitívne hlásenia a uprednostniť tie skutočné. Je schopný prehľadať terabajty dát behom pár milisekúnd a znížiť čas strávený vyšetrovaním udalostí. [52]

Microsoft Sentinel

Nová generácia bezpečnostného monitoringu postavená na cloude prepojenom so strojovým učením. Vidí a zneškodní hrozby skôr ako napáchajú škodu. Tento zrenovovaný SIEM je určený pre dnešný moderný svet. Sentinel detekuje a odpovedná

na hrozby ráýchlejšie a inteligentnejšie vďaka strojovému učeniu. Ponúka zníženie nákladov na prevádzku až o 48 % oproti konkurencii. [53]

ManageEngine Log360

Je obsiahle SIEM reišenie, ktoré detekuje hrozby snažiace sa penetrovať do siete a snaží sa ich eliminovať v ich zárodku. Ponúka automatizovanú správu logov, monitoruje Active Direcotry alebo napríklad aj Microsoft 365. Generuje veľké množstvo auditových správ a oznamuje potencionálne hrozby v reálnom čase. Jeho sila spočíva v kombinácii 5 najsilnejších nástrojov od spoločnosti ManageEngine.[54]

- ADAudit PLus;
- EventLog Analyzer;
- M365 Manager PLus;
- Exchange Reporter Plus;
- Cloud Security Plus;

Exabeam Fusion

Poskytuje najsilnejší a najvyspelejší SIEM nástroj v tomto odvetví. Je postavený na cloude a kombinuje schopnosti všetkých produktov od spoločnosti Exabeam ktorými sú:

- Cloudové úložisko dát;
- Rýchle spracovanie prichádzajúcich údajov;
- Hyper rýchle spracovanie dotazov;
- Analýza správania;

Umožňuje analytikom prácu z jedného kontrolného panelu, ktorý automatizuje väčšinu manuálnych úloh. Poskytuje centralizované úložisko a inteligentné vyhľadávanie naprieč celou sledovanou infraštruktúrou. Exabeam má možnosť integrovania známych hrozieb z volne dostupných zdrojov alebo aj z plarených zdrojov. Nie je potrebné sa báť o nedostatok miesta na ukladanie všetkých logov pravidiel a pod. nakoľko Exabeam poskytuje cloudové úložisko až do 100 PetaBytov pre každé SIEM riešenie. [55]

Splunk Enterprise Security

Prináša viditeľnosť celého sledovaného systému do jednej platformy. Umožňuje sledovať všetky zlomyselné hrozby, ktoré sa môžu ocitnúť v prostredí na, ktoré je SIEM nasadený. Zabezpečuje rýchlu detekciu hrozieb, o ktorú sa stará pokročilá dátová analýza a strojové učenie. Platforma, na ktorej je postavený zvyšuje produktivitu a znižuje čas strávený vyšetrovaním udalostí, ktoré nastali. Splunk je schopný sa

rýchlo prispôsobiť rozrastajúcemu prostrediu, ktoré sleduje. K hľadaniu hrozieb využíva strojové učenie a napríklad maticu hrozieb MITRE ATT&CK. [56]

OSSEC

Je platforma pre monitorovanie a kontrolovanie systému. Mieša všetky aspekty narušenia detekcie, monitorovania logov a bezpečnostného monitoringu do kopy v jednom silnom a open source riešení. Tento systém je možné používať na rôznych platformách ako napríklad Linux, Windows, MacOS alebo Solaris. Poskytuje sledovanie diania v reálnom čase. [57]

McAfee Enterprise Security Manager

Efektívna bezpečnosť začína s viditeľnosťou diania v reálnom čase. Toto SIEM riešenie umožňuje sledovanie používateľov systému, siete, databáz a aplikácii v reálnom čase vďaka čomu je možné odhaliť hrzbu včas. Riešenie od spoločnosti McAfee dokonale spolupracuje s už existujúcim biznis modelom a môže mu pomôcť rozšíriť ho do nových oblastí. Tento produkt je zameraný na to, aby poskytol:[58]

- Škálovateľnosť;
- Flexibilitu a zníženie nákladov nasadenia;
- Bezpečnosť;
- Centralizované riadenie;

AT&T Cybersecurity AlienVault

SIEM software ponúka cenné bezpečnostné informácie ale tiež si žiada drahú a časovo náročnú integráciu do systému a vyžadujú si veľa znalostí pri obsluhe. AlienVault centralizuje všetky potrebné bezpečnostné schopnosti a znižuje vynaložené úsilie pri obsluhe, poskytnutím intuitívneho dizajnu, ktorý je zameraný na najvážnejšie hrozby. Tento dizajn zobrazuje evidenciu hrozieb, metódy útoku, cieľovú IP adresu útoku a IP adresu zdroju. Toto umožňuje obsluhe včasnú reakciu. Systém je schopný spracovať viac ako 1800 detekčných pravidiel a viac ako 750 modelov správania.[59]

RSA NetWitness

Poskytuje vysokovýkonnú správu systému postavenú na cloudovom riešení. Eliminuje tradičné nasadenie a administratívu potrebnú pri inštalácii riešenia. Čím zabezpečuje rýchly vysoko kvalitný a pomerne jednoducho nasaditeľný systém. Tieto výhody zabezpečujú jednoduchú správu systému naprieč rozsiahlou infraštruktúrou.

NetWitness parsuje logy k čomu používa patentovanú technológiu, obohacuje ich indexuje ich na základe času, čím dramaticky zrýchľuje oznámenie hrozby a taktiež urýchľuje analýzu zozbieraných dát. Dokáže spracovať logy z viac ako 360 zdrojov udalostí.[60]

IBM QRadar

QRadar od spoločnosti IBM zameraný na sieťovú bezpečnosť poskytuje informácie, o tom čo sa deje v daných častiach siete. Využíva kombináciu znalostí fungovania siete, korelácie bezpečnostných udalostí. QRadar umožňuje simulovanie diania v sieti, ponúka možnosť nahrania vlastných logov, na základe čoho sa obsluha môže rýchlo naučiť, ako vyzerá pravá hrozba. Ponúka sledovanie udalostí v reálnom čase automaticky oskenuje sieť a zistí, čo presne sa v nej nachádza. [61]

2 Proces vývoju korelačných pravidiel

Spoločnosti, ktoré využívajú SIEM systémy, by sa mali vyhýbať neefektívnemu nastaveniu systému. V takomto prípade SIEM nemusí vôbec zachytávať závažné hrozby. K tomu, aby bola táto ochrana efektívna je potrebné mať dobre rozdelené využitie SIEM systému. Odporúča sa princíp 80/20, ktorý spočíva vo využití 20 % sily systému na odstránenie alebo, zmiernenie hrozieb a zvyšných 80 % by malo slúžiť k detekcii rizík. Využívať SIEM k tomu, aby bolo všetko logované a auditované je veľmi neafektívne a má to nízku výpovednú hodnotu. [67]

Pred nasadením SIEM systému by mala byť vedením spoločnosti zodpovedaná otázka: *„Aké sú 3 najväčšie priority ochrany v našej spoločnosti?“*. Na základe odpovede je teda možné rozdeliť ochranu do troch modelov. [67]

- **Zameraný na útočníka** – predvída spôsob útoku a zameriava sa na ochranu.
- **Zameraný na aktíva** – určuje hodnotu aktív v rámci siete, ktorú sleduje a vytvára im ochranu.
- **Zameraný na softvér** – účelom je zastaviť, identifikovať a odstrániť. zraniteľnosti skôr, ako sa stanú hrozbou.

O efektívnosť SIEMu sa stará práve, SOC viac v kapitole 2.1. Kde bezpečnostní technici vyvíjajú korelačné pravidlá, ktoré slúžia k detekcii hrozieb. [67]

Úlohou SOC je teda analyzovať udalosti a vyvíjať korelačné pravidlá. Pravidlá sú vyvíjané na základe analýzy zachytených logov.

Korelačné pravidlá pracujú na základe generovaných udalostí. Je nesmierne dôležité aby zdroje, ktoré tieto udalosti alebo logy generujú boli správne nakonfigurované. Vývoj takéhoto pravidla spočíva v piatich hlavných krokoch. [68]

- **Krok 1** – premyslieť logiku vyvíjaného pravidla a stanoviť, aké zdroje logov sú potrebné.
- **Krok 2** – uistiť sa, že najdôležitejšie zdroje informácií sú dostupné a je možné z nich získavať logové záznamy. Pokiaľ má spoločnosť zdroj, z ktorého nie je možné získať informácie, takéto pravidlo sa stane neužitočným a zbytočne zahltí systém. Je dôležité pochopiť, aké záznamy je nasadený SIEM schopný spracovať a aké nie.
- **Krok 3** – skontrolovať, či sú do systému pripojené všetky zdroje logov, je možné, že zlá konfigurácia zdroju alebo iný technický problém pripojenie neumožňuje. Toto by ohrozilo časť systému, ktorá by nemohla byť monitorovaná.
- **Krok 4** – správne auditovanie generovaných logov. Korelačné pravidlá sú nastavené tak, že čakajú na určitý druh udalosti, ktorú zdroje musia generovať.
- **Krok 5** – je potrebné pravidelne sledovať udalosti a ich zdroje, či udalosti produkujú. Tvorba udalostí je dôležitá pre fungovanie korelačných pravidiel, ak by sa nejaká udalosť prestala vyskytovať, tak korelačné pravidlá, ktoré sú

na ňu naviazané by mohli prestať fungovať.

Zbieranie logov, nie je jediná úloha SIEM systému, logy je taktiež potrebné pretriediť, pretože nie je v schopnosti tohto systému kvalitne analyzovať všetky prichádzajúce logy. Logy sa teda po zbieraní filtrujú, indexujú, analyzujú a korelujú. Po korelácii sa ešte vyhodnotia udalosti, ktoré po tomto procese ostali a z nich sa vyberú tie, ktoré sú považované za hrozby a budú sa ďalej riešiť.

2.1 Princíp fungovania bezpečnostného operačného centra

Úlohou SOC je monitorovanie, prevencia, vyšetrovanie a odpoveď na kyberbezpečnostné incidenty, ktoré sa môžu vyskytnúť. Tím, ktorý pracuje v SOC má za úlohu chrániť aktíva spoločnosti pred útočníkmi. Navrhuje hlavnú bezpečnostnú stratégiu pre spoločnosť.

Pri kyberbezpečnosti je vždy prevencia pred hrozbou efektívnejšia, ako riešenie následkov, ktoré môže spôsobiť. SOC tím monitoruje prostredie a snaží sa odhaliť podozrivú aktivitu pred tým, ako útočník vnikne do siete. Pokiaľ pracovník SOC detekuje nevhodnú aktivitu, zbiera o nej čo najviac údajov, ako je možné. Tieto údaje sú následne využité pri vyšetrovaní. [62]

Počas vyšetrovania incidentu tím analyzuje hrozbu a snaží sa zistiť jej pôvod. Bezpečnostný analytik sa pozerá na sieť organizácie ako útočník a snaží sa nájsť nedostatočne zabezpečené časti, tým že sa ich snaží exploitovať. Útokom sa analytik snaží vytvoriť hrozbu a zistiť ako proti nej najlepšie bojovať. Pomáha mu pri tom znalosť siete organizácie a najnovšie známe zraniteľnosti, ktoré je možné dohľadať napríklad v matici MITRE ATT&CK®[63].

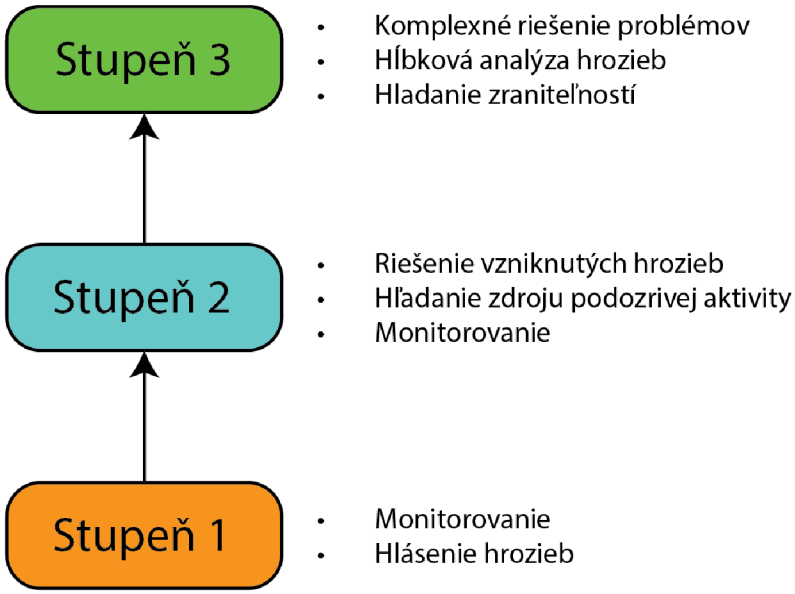
Po ukončení vyšetrovania, SOC tím začne prípravu na nápravu nájdených nedostatkov v sieti. Akonáhle začne útok na infraštruktúru SOC v prvom rade začne odpovedať na túto hrozbu napríklad tým, že izoluje koncové zariadenie, z ktorého prichádza útok, ukončí škodlivý proces alebo vymaže nechcené súbory. Po eliminovaní hrozby, tím rieši nápravu škody, ktorá bola spôsobená napríklad znovu zapne systémy, ktoré boli vypnuté alebo sa snaží obnoviť stratené dáta. Informácie boli čerpané z [64].

SOC okrem toho čo bolo spomenuté vyššie, pracuje aj na korelačných pravidlách, ktoré pomáhajú pri detekcii hrozieb a narušeniach integrity. Pri vývoji týchto pravidiel sa snažia o čo najmenej falošne pozitívnych hlásení, ktoré pravidlá môžu vyvolať. [65]

Štruktúra SOC

Analytici v SOC sa podľa ich skúseností a toho, za čo zodpovedajú delia do 3 kategórií, analytici prvého, druhého a tretieho stupňa. Informácie, ktoré zbierajú sú ďalej poskytované vedeniu SOC a vývojárskym tímom, aby mohli efektívnejšie pracovať na korelačných pravidlách. [66]

Analytici prvého stupňa sú začiatočníci, ktorí sú najmenej skúsení. Najväčšiu časť ich práce tvorí monitorovanie podozrivej aktivity v sieti alebo možných hrozieb. Nie sú zapojení do riešenia vzniknutých hrozieb. Po zistení hrozby, ju ohlásia druhému stupňu. [66]



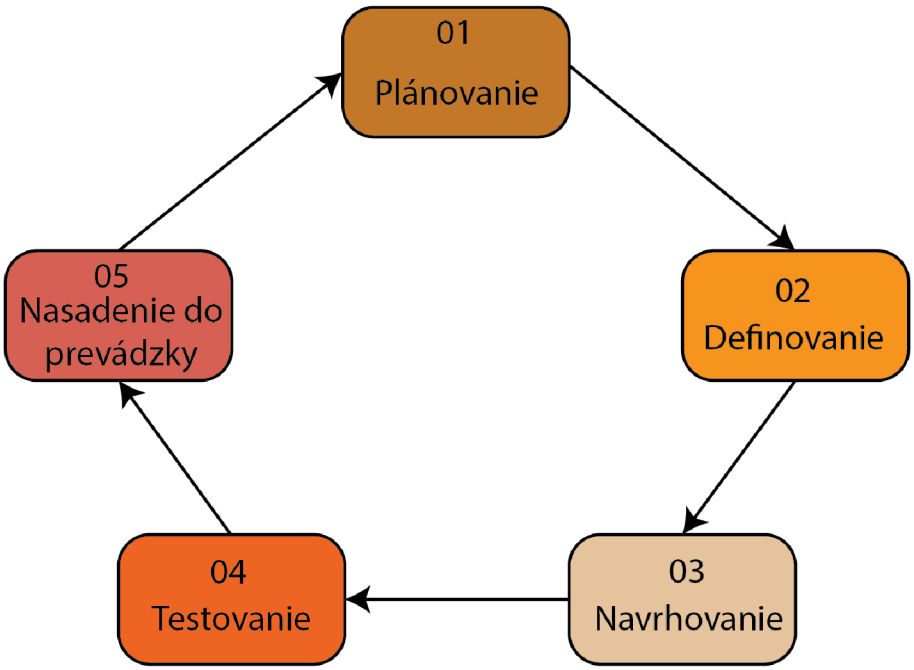
Obr. 2.1: Hierarchia SOC.

Druhý stupeň pracovníkov je viac skúsený, taktiež môžu vykonávať monitorovanie siete ako v prvom stupni, no ich hlavnou úlohou je hlbšie analyzovať podozrivú aktivitu, ktorú im ohlásili analytici prvého stupňa. Pokúšajú sa zistiť odkiaľ pochádza hrozba a pripraviť na ňu dostatočnú odpoveď. [66]

Tretí stupeň analytikov je naviac skúsený. Ich úlohou je využiť skúsenosti, ktoré majú na komplexné riešenie problémov, ktoré im predal druhý stupeň. Majú na starosť aj hľadanie hrozieb v sieti, ktoré sa mohli dostať cez pozornosť analytikov prvého a druhého stupňa. [66]

2.2 Životný cyklus korelačného pravidla

Vývoj korelačného pravidla je nekonečný proces, ktorý prebieha podobne ako vývoj softvéru. Práca na korelačnom pravidle si vyžaduje obrovské znalosti infraštruktúry a



Obr. 2.2: Kolobeh vývoju korelačného pravidla.

samotnej kyberbezpčnosti, vyvíjajú ich skúsení bezpečnostní technici. Títo technici sa v hierarchii SOC (viac v kapitole 2.1) nachádzajú nad analytikmi, od ktorých získavajú potrebné dáta pre vývoj a spätnú väzbu k pravidlám.

Vývoj sa skladá z 5 častí, ktorými sú plánovanie, definovanie, navrhovanie, testovanie a nasadenie do prevádzky. Po tom, ako sa prejde k poslednej časti sa celý proces znovu opakuje a už existujúce pravidlá sa vylepšujú o nové poznatky a zistené nedostatky sa odstraňujú. Priebeh vývoja je možné vidieť na obrázku 2.2. [69]

Plánovanie

Počas plánovania prebieha analýza prostredia, na ktoré má byť korelačné pravidlo použité. Ďalšou zložkou plánovania sú údaje od SOC analytikov, ktorí dávajú spätnú väzbu na už vyvinuté pravidlá. Po zozbieraní dostatočného množstva dát sa prechádza na ďalšiu časť vývoja.

Definovanie

Po analýze sa k zisteným informáciám pridajú jednotlivé požiadavky zákazníka. Prihliada sa aj na existujúce štandardy v oblasti bezpečnosti, využíva sa napríklad matica MITRE ATT&CK®.

Následne po pridaní dodatočných informácií sú definované jednotlivé potrebné korelačné pravidlá. Všetky tieto informácie sa posunú do časti navrhovania, ktorú majú na starosť skúsení vývojári.

Navrhovanie

Z dodaných informácií vývojári navrhnú, ako budú pravidlá vyzerať. Po tomto návrhu začne proces vývoja, kde sú jednotlivé požiadavky spracované a premenené do korelačného pravidla.

Testovanie

Táto časť slúži k otestovaniu správnosti fungovania pravidiel. Sú testované na už existujúcich logoch, ktoré sa objavili v infraštruktúre.

Proces prebieha, tak že sa vyberie log a použije sa naň korelačné pravidlo. Z výstupu, ktorý vývojár dostane vie zistiť, či funkcia pravidla bola správna a či je ho možné posunúť do prevádzky.

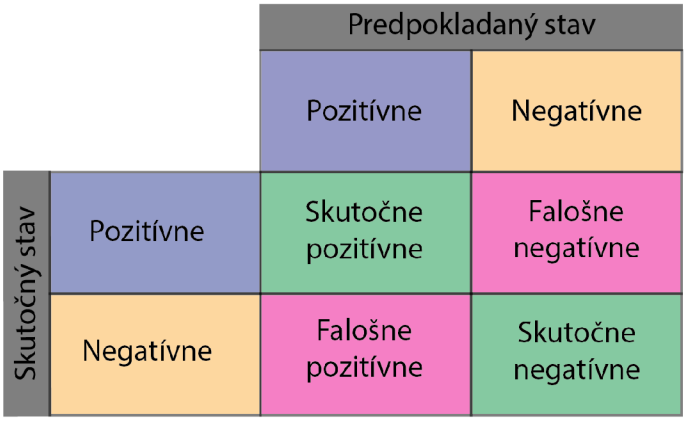
Nasadenie do prevádzky

Počas toho, ako je pravidlo v prevádzke nasadené na reálne incidenty, je sledované analytikmi SOC, ktorí okrem skúmania upozornení prichádzajúcich zo SIEM systému sledujú aj správanie pravidla. Sledujú to, ako efektívne dohliada nad prevádzkou siete a či správne vyhodnocuje vzniknuté udalosti. Pokiaľ je v pravidle nejaký nedostatok, sú tieto informácie posunuté vývojárom, aby znovu pravidlo prerobili a zlepšili jeho funkčnosť. Pravidlá môžu generovať množstvo falošne pozitívnych hlásení. Tato skutočnosť je zistená až po nasadení pravidla do prevádzky. Vývojári sa následne snažia minimalizovať falošne pozitívne hlásania nakoľko tie zahlcujú SOC analytikov a tí sa nemôžu sústrediť na reálne hrozby.

Falošne pozitívne a falošne negatívne hlásenia

Pri SIEM systémoch je nesmierne dôležité dbať na minimalizáciu falošne negatívnych a pozitívnych hlásení. Týmto je možné zefektívniť systém a SOC (*Security Operations Center*), ktoré spravuje systém a vyhodnocuje upozornenia.

Falošne negatívne vyhodnotenie nastáva vtedy, keď je rizikové dianie vyhodnotené ako dianie, ktoré je normálne. Falošne pozitívne vyhodnotenia sú opakom a vyskytujú sa vtedy, keď systém vyhodnocuje normálne správanie ako podozrivé. [70]



Obr. 2.3: Vyhodnotenie falošne negatívneho a falošne pozitívneho stavu.

2.3 Problematika udržovania verzie korelačných pravidiel

Informácie, ktoré sú nazbierané od analytikov sa následne využívajú pri tvorbe korelačných pravidiel. Proces tvorby týchto pravidiel sa stále opakuje, jednotlivé pravidlá sa stále počas fungovania systému menia. Je vysoko pravdepodobné, že na jednom pravidle bude pracovať viacero vývojárov. Pri vývoji sa môže stať, že v pravidle je chyba a je ju potrebné opraviť.

Nakoľko pravidlo nemusí opravovať človek, ktorý ho vyvíjal, alebo ak ho opravuje rovnaký vývojár nemusí si pamätať, ako pravidlo vyzeralo pred tým, ako v ňom bola chyba. SIEM systémy neposkytujú žiadnu možnosť sledovania vývoja korelačných pravidiel. Čiže, ak sa po nejakom čase zistí chyba je zložitejšie a časovo náročnejšie ju opraviť.

Túto prácu by dokázal uľahčiť nástroj, ktorý by bol prepojený so SIEM systémom a dokázal by snímať vývoj pravidiel a zaznamenávať ich podobu v čase. Umožňovalo by to vývojárovi jednoduchšie hľadanie chýb a vykonávanie potrebných opráv.

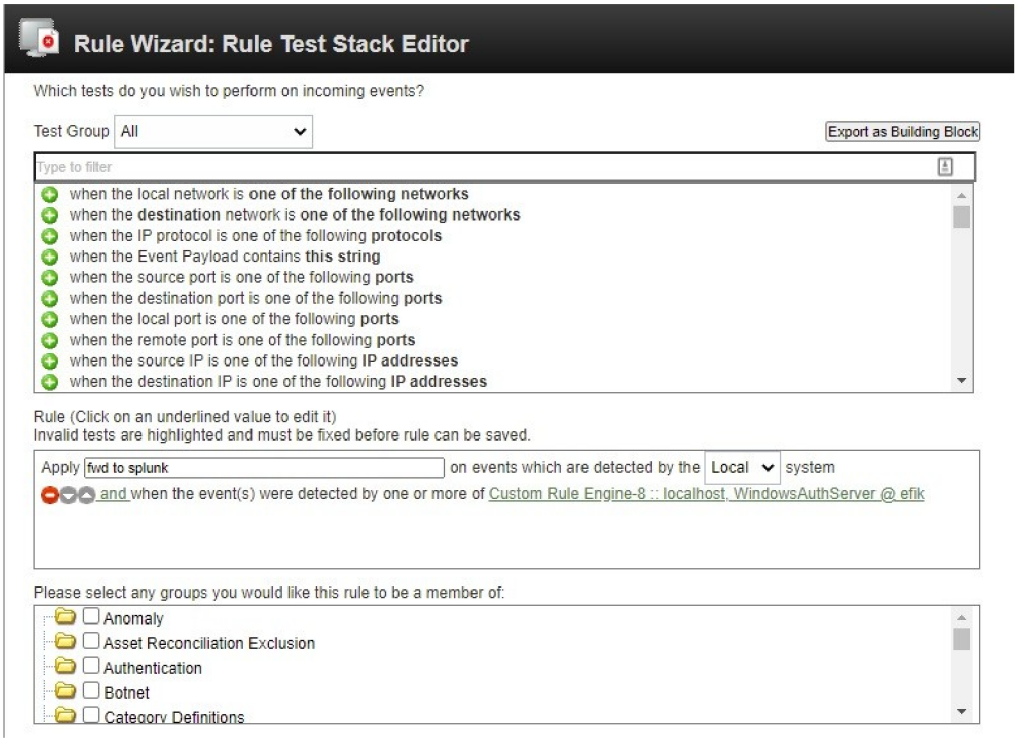
Vývoj takéhoto systému je predmetom tejto bakalárskej práce. Analýza potrebných komponentov, ktoré sú dôležité pri vývoji takejto aplikácie sú rozobraté v nasledovnej kapitole 3.

2.4 Prístup SIEM systémov k tvorbe korelačných pravidiel

V rámci práce budú porovnané dve SIEM riešenia. QRadar[1] od spoločnosti IBM a NetWitness[2] od spoločnosti RSA Securty. Porovnanie bude zamerané na tvorbu korelačných pravidiel. Ďalšie SIEM riešenia budú vypracované v rámci bakalárskej práce.

QRadar

Tvorba vlastných pravidiel umožňuje korelovať medzi rôznymi zdrojmi a udalosťami nachádzajúcimi sa v sieti. Tvorba týchto pravidiel prebieha pomocou GUI, ktoré poskytuje QRadar. Tvorba spočíva v tom, že vývojár si vyberá z poskytnutých možností, čo má pravidlo sledovať, tento výber sa uskutočňuje v najväčšom okne, okno je možné vidieť na obrázku 2.4. [71]

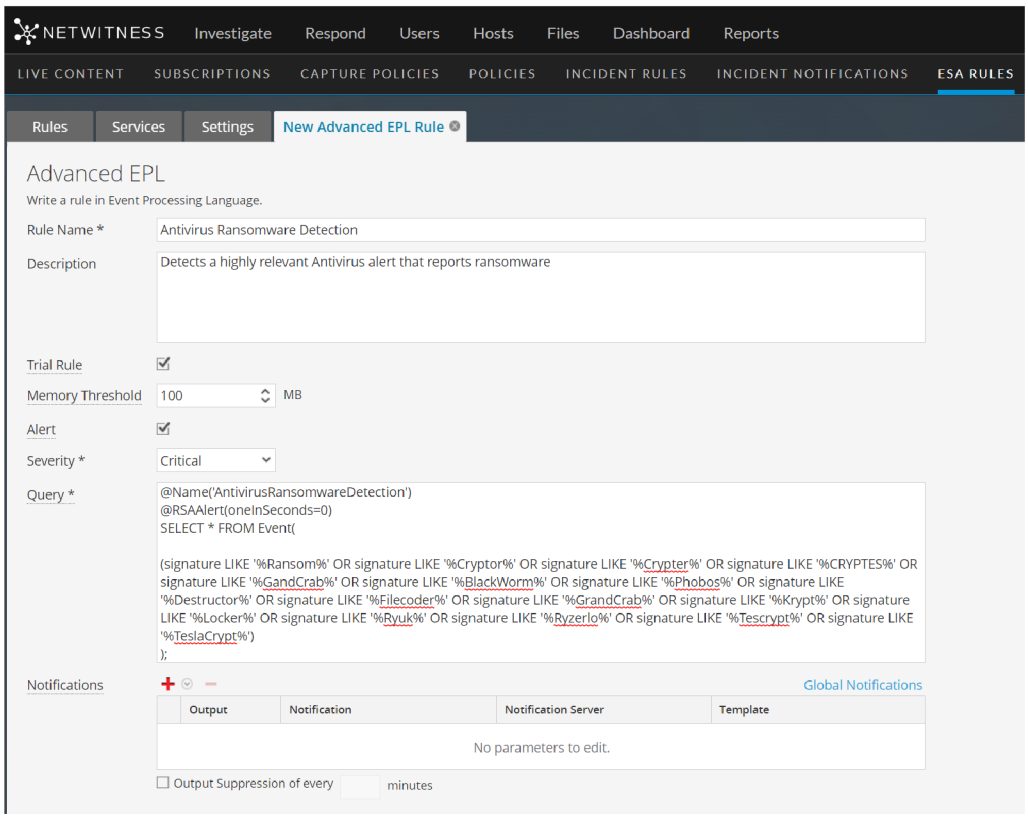


Obr. 2.4: Grafické rozhranie tvorby korelačného pravidla QRadar. [72]

[1]Viac o informácií o aplikácii je možné získať z: `https://www.ibm.com/cz-en/qradar`
[2]Bližšie informácie sú dostupné z: `https://www.netwitness.com/`

NetWitness

Na rozdiel od QRadaru, tvorba pravidiel v SIEM NetWitness prebieha písaním pravidla. Pravidlo sa vpisuje do GUI, ktoré táto aplikácie poskytuje. Príklad je možné vidieť na obrázku 2.5. Tento prístup, ktorý používa NetWitness umožňuje vývojárovi mať väčšiu voľnosť v pravidle a vie si ho podla potreby rýchlo upraviť.



Obr. 2.5: Grafické rozhranie tvorby korelačného pravidla NetWitness. [73]

3 Analýza aplikácie

K vytvoreniu požadovanej aplikácie bude potrebné použiť niekoľko technológií. Nakoľko sa jedná o webovú aplikáciu je potrebné vybrať z niekoľkých možností pre tvorbu webu. Vývoj aplikácie bude prebiehať v jazyku Python viac o jazyku [74], ktorý bol z pomedzi iných jazykov vybratý, z dôvodu veľkej škály knižníc, ktoré je možné pri vývoji použiť.

Aplikácia bude postavená na architektúre mikroservís. Čo jej umožní v prípade potreby rýchlu možnosť opravy. Problematika je bližšie priblížená v kapitole 3.3

Python ponúka viacero frameworkov, s ktorými je možné vyvíjať webové aplikácie. Sú nimi napríklad Django, Flask, Web2Py, Bottle [75]. Aplikácia bude vyvíjaná s pomocou frameworku Flask. Tento framework je oproti ostatným viac rozšírenejší a je napríklad jednoduchší na implementáciu ako Django. Tým, že sa jednoduchšie implementuje však neznamená, že by strácal na svojej sile, alebo mu chýbali určité funkcionality. Užívateľovi bude aplikácia sprostredkovaná cez GUI, ktoré bude vytvorené pomocou Vue.js. Tento framework je bližšie priblížený v 3.3.

Jednou z najdôležitejších súčastí aplikácie je možnosť zálohovať vývoj korelačných pravidiel. K tomuto slúži Git viac c kapitole 3.4. Jednou z najdôležitejších súčastí aplikácie je GitLab viac v kapitole 3.4.2. GitLab bude využitý k zálohovaniu vyvíjaných korelačných pravidiel.

V aplikácii je potrebné uchovávať rôzne informácie o používateľoch ako napríklad prihlasovacie údaje a údaje, ktoré slúžia k vývoju pravidiel. Pre správu týchto informácií je potrebné využívať databázový systém. Existuje mnoho databázových systémov, ktoré je možné využívať. Napríklad MongoDB, PostgreSQL, MySQL, Oracle SQL Developer. Z týchto technológií bola vybratá práve PostgreSQL databáza. Je to kvôli jej rýchlemu rozširovaniu v IT svete a taktiež z toho, že je to open source projekt. Je ju teda možné využívať kýmkoľvek, za akýmkoľvek účelom bez kúpi licencie. Databáza je viac opásaná v 3.3.

K pripojeniu relačnej databázy a Flask framweorku bude slúžiť knižnica SQLAlchemy. Bližšie ku knižnici je napísane v časti 3.2.

Webová aplikácie funguje na základe komunikácie frontendu s backendom. K tejto komunikácii, slúži API (*Application programming interface*) rozhranie. Táto technológia zabezpečuje plynulú komunikáciu všetkých častí GUI (*Graphical user interface*) so serverom na pozadí, ktorý spracováva informácie. API je priblížené v kapitole 3.3.

Najdôležitejšou časťou aplikácie je server, ktorý bude sprostredkúvať všetku komunikáciu medzi klientom a aplikáciou. Ako základ pre server bude operačný systém CentOS. Je to systém založený na Linuxe. Bol zvolený z dôvodu veľkého rozšírenia Linuxových serverov a ich spoľahlivosti a aj tým, že je to opensource systém.

3.1 Požiadavky kladené na aplikáciu

Úlohou tejto kapitoly je je jasne popísať požiadavky, ktoré sú na systém kladené. Tento krok je potrebné uskutočniť ešte pred návrhom aplikácie, aby bolo zaručené že všetky požiadavky budú v návrhu obsiahnuté.Požiadavky kladené na systém by si nemali odporovať a mali by byť realistické aby ich bolo možné splniť. Základné požiadavky vznikli na základe skúmania vývoju korelačných pravidiel. Vďaka požiadavkám je možné vytvoriť funkčný systém, ktorý je zameraný na vykonávanie potrebných funkcií.

3.1.1 Model prípadov užitia aplikácie

Model prípadov použitia alebo aj UML diagram slúži ako predloha k vývoju aplikácie. Tento model na rozdiel od kódu jednoducho znázorňuje funkcionalitu aplikácie, kde bude medzi užívateľom a aplikáciou vznikať interakcia. Tento diagram sa vytvára ako jedna z prvých vecí pred štartom vývoja projektu. [76] Diagram na obrázku 3.1 zaznamenáva úlohu aplikácie alebo aj UC (*Use Case*), opisuje hlavnú funkcionalitu softvéru s ktorou bude vytváraná interakcia. Ďalšou častou diagramu 3.1 vývojár. Vývojár reprezentuje úlohu užívateľa, ktorý bude ovládať systém. [76]

Prípady užitia, ktoré sú stanovené v diagrame budú zohľadnené pri budúcom vývoji aplikácie. Tomuto vývoju sa bude venovať kapitola 5, kde bude možné vidieť implementáciu jednotlivých častí aplikácie.

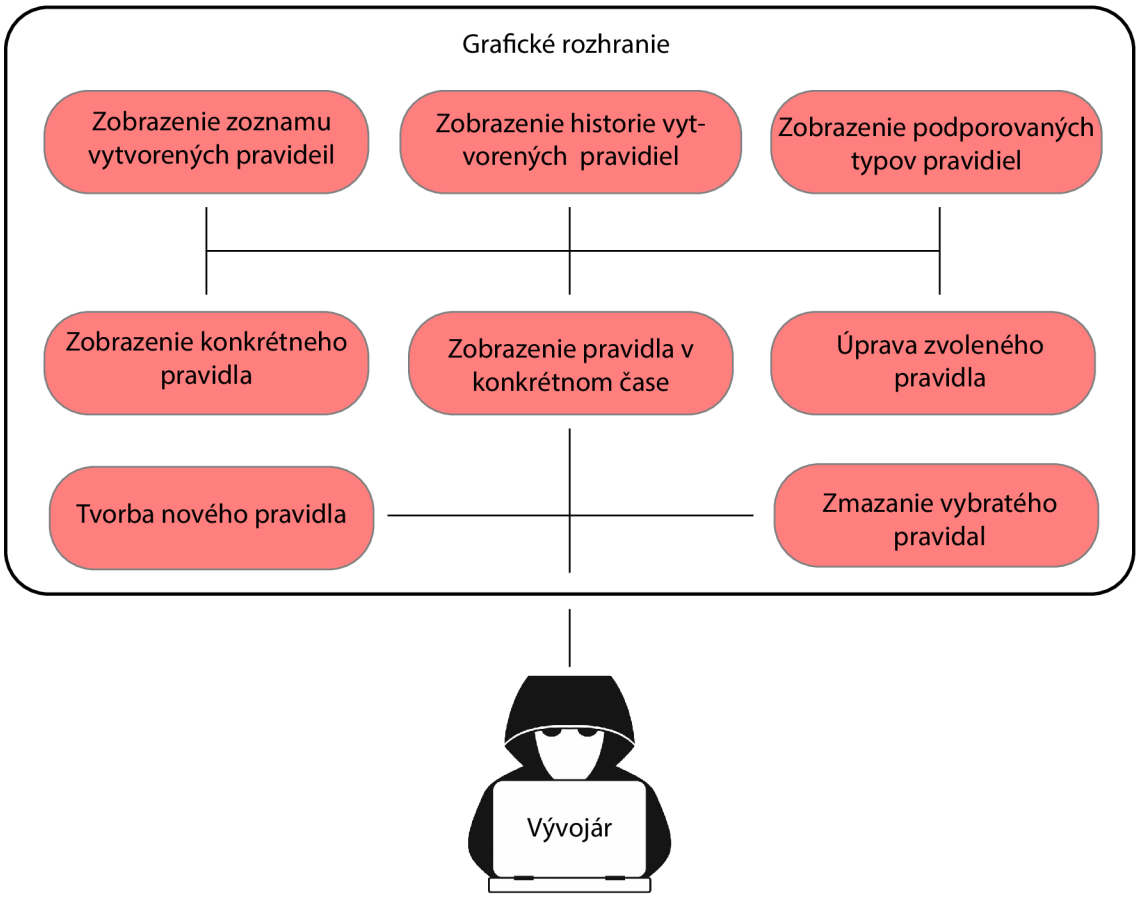
3.1.2 Funkčné požiadavky

Funkčné požiadavky systému predstavujú základnú skupinu operácií, ktoré má byť aplikáciou schopné vykonať [77]. Tieto požiadavky vznikli pri štúdiu danej problematiky, ktorú má aplikácia riešiť. Následne boli vytvorené nasledujúce požiadavky:

1. Aplikácia bude schopná umožňovať vývojárovi jednoduchú tvorbu korelačných pravidiel pre zvolený typ pravidla.
2. Aplikácia bude umožňovať správu vytvorených korelačných pravidiel a to ich aktualizáciu a mazanie.
3. Aplikácia bude zaznamenávať úkony vytvorené na konkrétnom pravidle.
4. Vývojárovi bude umožnené zobrazovať si pravidlo po jednotlivých úpravách.
5. Vývojár si bude môcť zobraziť svoje korelačné pravidlá.

3.1.3 Nefunkčné požiadavky

Nefunkčné požiadavky predstavujú skupinu obmedzení, ktoré sú kladené na daný vyvíjaný systém. Tieto požiadavky sa môžu týkať výkonnosti, bezpečnosti alebo



Obr. 3.1: Ukážka modelu prípadov užitia.

nejakých druhov obmedzení. [77]

1. Aplikácia bude naprogramovaná v jazyku Python.
2. Aplikácia bude využívať API na backende.
3. Bude zabezpečená základná bezpečnosť aplikácie.
4. Aplikácia bude spustená pomocou technológie Docker [1].
5. Aplikáciu bude možné používať vo webovom prehliadači Mozilla Firefox [2]
6. Zdrojový kód musí byť dostatočne okomentovaný pre zabezpečenie jeho údržby.
7. Grafické rozhranie bude vytvorené pomocou moderného frameworku Vue.js

3.1.4 Kritické požiadavky

Tieto požiadavky sa zameriavajú na používateľa. Snažia sa určiť čo je nevyhnutná funkcionalita aplikácie, ktorú bude užívateľ používať. Určenie kritických požiadaviek je vhodné na dobré nastavenie funkčnosti softvéru. Pri aplikáciách je dôležité aby boli

[1] Viacej informácií o technológii dostupné z: `https://www.docker.com/`
[2] Viacej informácií o prehliadači dostupné z: `https://www.mozilla.org/sk/firefox/new/`

schopné fungovať za ich primárnym účelom a je pre ne zbytočné používať množstvo iných funkcionalít. [78]

Po analýze naštudovaných požiadaviek bolo určené, že kritickými funkcionalitami budú práve tvorba korelačných pravidiel, ich aktualizácia a mazanie a zobrazovanie histórie zmien na pravidle.

3.2 Technológie technológie umožňujúce splniť ciele aplikácie

Vývoj aplikácie si vyžaduje pomerne veľké množstvo technológií. Využívajú sa rôzne programovacie jazyky, frameworky, databázové systémy rôzne druhy spracovania dát verzovacie systémy a mnohé ďalšie.

V nasledujúcich kapitolách budú bližšie popísané využité technológie k vývoju aplikácie. Všetky technológie sú verejne dostupné a môžu byť použité kýmkoľvek.

Framework pre tvorbu webu – Flask

Tento framework vyniká spomedzi ostatných frameworkov. Je tomu tak z dôvodu toho, že je navrhnutý tak, aby vývojár rozhodoval o tom, akým smerom sa má jeho aplikácia uberať. Flask podporuje celú škálu databáz, či už je potrebné používať relačnú databázu, NoSQL alebo žiadnu databázu. Záleží to len na vývojárovi a charakteristike aplikácie. Framework podporuje všetky známe databázové riešenia a je ich možné jednoducho implementovať. Flask je malý framework, ktorý podporuje všetky potrebné funkcionality pre tvorbu webových aplikácií.[79]

Ak chce vývojár používať nejakú funkciu, ktorú Flask neobsahuje, môže ju jednoducho získať zo zdrojov tretích strán. Alebo si funkcie môže vývojár vytvoriť sám.

Tieto vlastnosti v konečnom dôsledku umožňujú Flasku byť výkonnejším a robustnejším ako iné frameworky. [79]

Flask bude použitý na vytvorenie API, ktoré slúži na komunikáciu webstránky so serverom, kde sa uchovávajú a spracúvajú dáta.

Na obrázok 3.2 je naznačená jednoduchá aplikácia vytvorená pomocou Flasku. Po vpísaní IP adresy, na ktorej aplikácia počúva do ľubovolného prehliadaču je možné vidieť výpis HelloWorld. K tomu slúži funkcia `HelloWorld` túto funkciu je možné rozšíriť a prispôsobiť podľa vlastnej predstavy. Výpis slúži len pre predstavu toho, ako sa vytvára základ aplikácie. Zložitejšie výpisy s rôznymi funkciami budú demonštrované v časti 4

```python
app = Flask("SimpleApp")
api = Api(app)

class HelloWorld(Resource):
  return "HelloWorld!"

api.add_resource(HelloWorld, "/")

if __name__ == "__main__":
  app.run(host = "localhost", port = "5555")
```

Obr. 3.2: Príklad jednoduchej Flask aplikácie v jazyku Python.

Objektovo relačné mapovanie – ORM

Táto architektúra je založená na prepojení relačnej databázy a objektovo orientovaného programovania. Vývojárovi to uľahčuje prácu tým, že nemusí vytvárať zložité SQL zápisy v databáze. Taktiež táto technika zvyšuje bezpečnosť aplikácií, keďže príkazy do SQL databázy sa nenachádzajú priamo v písanom kóde.

Pre porovnanie je na obrázku číslo 3.3 a 3.4. vidieť rozdiel medzi SQL príkazom a príkazom pomocou ORM. Po vykonaní príkazov sú vrátené informácie o užívateľovi s číslom 13. [80]

```sql
SELECT id, first_name, second_name, email, phone_number
FROM users WHERE id = 13
```

Obr. 3.3: Príkaz vo formáte SQL.

```python
stmt = select(user_table).where(user_table.c.id == 13)
```

Obr. 3.4: Príkaz v jazyku Python s využitím ORM.

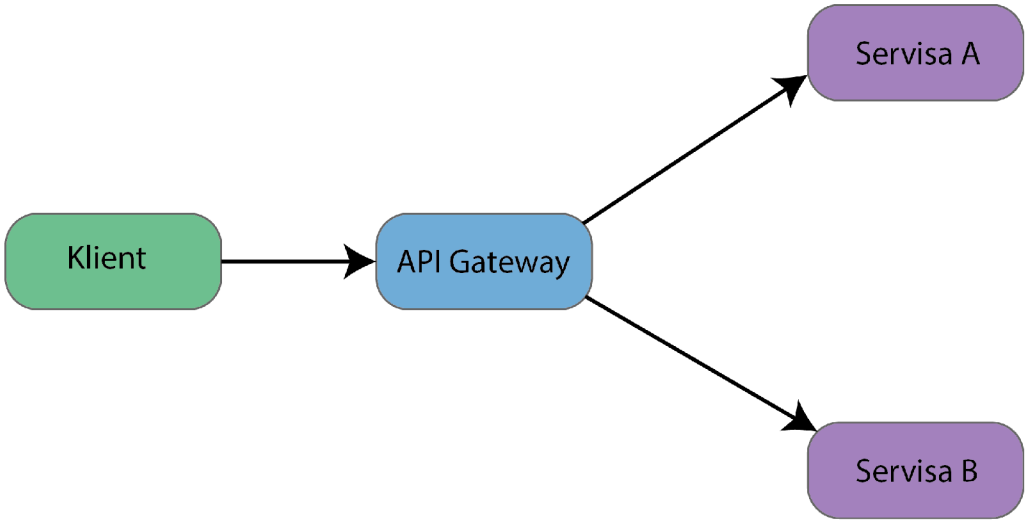
Nástroj využitý k objektovo relačnému mapovaniu – SQLAlchemy

Je to nástroj vytvorený pre Python, ktorý sa používa pri objektovo relačnom mapovaní. Poskytuje sadu prostriedkov navrhnutých pre efektívne pristupovanie k databáze, ktoré sú iplmenetované do jazyka Python.

Tento nástroj je používaný spoločnosťami ako napríklad Yelp!, reddit, Survey Monkey alebo DropBox. [81]

3.3 Architektúra webových aplikácií pomocou mikroservisy

Architektúra mikroservís umožňuje aplikáciam byť viac flexibilné. Aplikácie sú tvorené z viacerých častí, ktoré dokážu perfektne fungovať bez závislosti na inej. Jedna aplikácia je teda tvorená z viacerých mikroservís. Každú mikroservisu môže mať na starosti iný tím ľudí. Takáto aplikácia je jednoduchá na úpravu, pretože pridanie novej funkcie nevyradí z prevádzky celú aplikáciu, ale len dočasne nejakú jej jednu časť.



Obr. 3.5: Schéma mikroservisy.

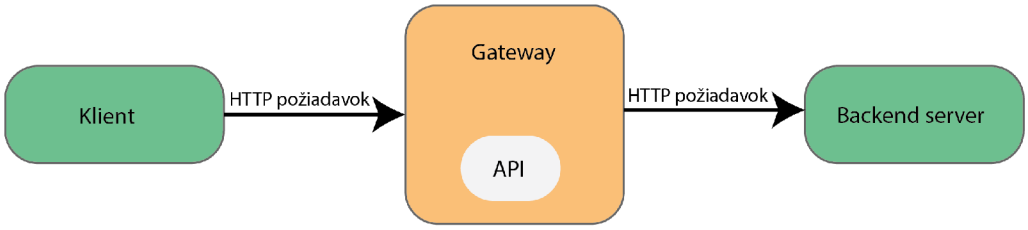
Pred tým, ako vznikla architektúra mikroservís existovali takzvané monolitické aplikácie. Kde bola údržba veľmi nákladná a často si vyžiadala veľký tím ľudí. Tieto aplikácie museli byť písané len v jednom jazyku, čo pri mikroservisách nie je potrebné. Každá mikroservisa môže byť napísaná v inom jazyku. Taktiež každá mikroservisa môže používať iné závislosti z rôznych verzií technológií,ktoré používa. Pri monolitických aplikáciach to nebolo možné a bolo potrebné vybrať jednu verziu

technológie, na ktorej je celá aplikácia závislá. Mikroservisy komunikujú pomocou API rozhrania.[82]

API rozhranie

API je mechanizmus, ktorý umožňuje dvom alebo viacerým softvérom komunikovať medzi sebou pomocou predpripravených protokolov. Druhy protokolov podľa [83]:

- **SOAP (*Simple object access protocol*)**
- **REST (*Representational state transfer*)**
- **gRPC (*Google remote procedure call*)**
- **JSON-EPC (*JavaScript object notation–remote procedure call*)**
- **GraphQL (*Graph query language*)**
- **Apache Thrift**



Obr. 3.6: Schéma API.

Pri weobých aplikáciách je najviac rozšírený práve protokol REST. Funguje na základe toho, že klient posiela požiadavky na server pomocou API endpointu. Táto požiadavka je odoslaná serveru, ktorý po jej spracovaní odošle spätne odpoveď. Odpoveď je typicky v JSON, HTML, XLT formáte alebo v obyčajnom textovom formáte [84]. Požiadavky a odpovede sa na server a zo serveru posielajú pomocou HTTP protokolu. Príklad API endpointu `/users/get/1`, po zadaní endpointu do webového prehliadača by nám server poslal v odpovedi na požiadavku informácie o užívateľovi `1`. Príklad takéhoto výpisu je možné vidieť na obrázku 3.7.

API umožňuje byť firmám konkurencie schopné a v rýchlom čase sa prispôsobiť novým trendom a aplikáciam.

Z takéhoto výpisu je potom vývojár schopný získať informácie pre jeho aplikáciu. Napríklad vyvíja časť aplikácie, ktorá zobrazuje informácie o danom užívateľovi. Z odpovede si vie vybrať jednotlivé potrebné položky, ktoré chce zobraziť napríklad v GUI.

Následne, keď sa užívateľ prihlási do aplikácie, odošlú sa informácie na server a server vráti požadované informácie aplikácii, ktorý mu ich zobrazí.

```
{
  "Id":           "1",
  "FirstName":    "Jon",
  "LastName":     "Doe",
  "EmailAddress": "jondoe@mail.com"
}
```

Obr. 3.7: Odpoveď na požiadavok vo formáte JSON.

Framework využitý k tvorbe frontendu – Vue.js

Vue je framework, ktorý rozširuje štandardný JavaScript. Je určený pre tvorbu UI (*User Interface*). Je postavený nad štandardným HTML (*HyperText Markup Language*) a CSS (*Cascading Style Sheets*) a JavaScriptom. Poskytuje vývojový model, ktorý pomáha efektávne vyvíjať UI, či už jednoduché alebo komplexné. Na obrázku 3.8 a 3.9 je zobrazené vytvorenie jednoduchej aplikácie pomocou Vue frameworku. Táto aplikácia užívateľovi zobrazí klikajúce tlačítko, ktoré počíta každý klik a zobrazuje počet kliknutí. [85]

```
{
  import { createApp } from 'vue'

  createApp({
    data(){
      return {
        count: 0
      }
    }
  }).mount('#app')
}
```

Obr. 3.8: Jednoduchá Vue aplikácia.

Vue ponúka:

- **Deklaratívne vykresľovanie** – Vue rozširuje štandardné HTML so syntaxou, ktorá umožňuje deklaratívne popísať HTML výstup založený na JavaScripte.
- **Reaktivitu** – Vue automaticky sleduje JavaScript a efektívne aktualizuje výstup zobrazený pomocou HTML, po každej zmene.

Tento framework je nadizajnovaný tak, aby bol flexibilný a aby sa jednoducho adaptoval zväčšujúcemu sa prostrediu. V závislosti od použitia môže byť aplikovaný na:

```
{
<div id='app'>
  <button @click='count++'>
    Count is: {{ count }}
  </button>
</div>
}
```

Obr. 3.9: Html výpis pre zobrazenie textu v prehliadači.

- **Vylepšenie statického HTML**;
- **Vkladanie ľubovolných komponentov do rôznych častí aplikácie**;
- **Jednostránková aplikácia**;
- **Vykresľovanie na strane serveru**;
- **Generovanie statických stránok**;
- **Použitie na počítači, smartfóne, alebo aj v termináli**;

Informácie sú čerpané z [85]. Vyvíjaná aplikácia bude používať Vue k rýchlemu jednoduchému a responzívnemu dizajnu.

PostgreSQL

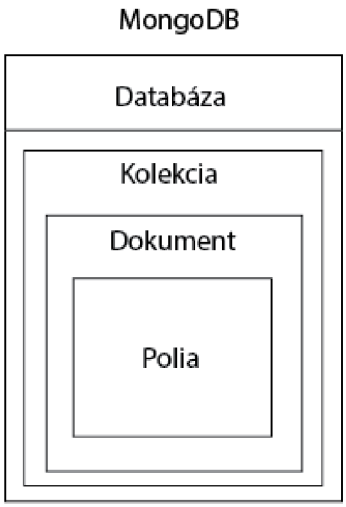
PostgreSQL je open source relačná databáza, ktorá využíva a rozširuje štandardný SQL jazyk. Je skombinovaná s rôznymi funkciami, ktoré jej umožňujú ukladať a spracovať najkomplikovanejšie úlohy. PostgreSQL nadobudlo silnú reputáciu vďaka jej dobrej architektúre, spoľahlivosti, dátovej integrite, robustným funkcionalitám a možnostiam rozšíriť databázu o funkcie vývojárov tretích strán. Vývojárom umožňuje definovať vlastné dátové typy, vytvoriť vlastné funkcie.

Tento systém je možné používať na všetkých dôležitých operačných systémoch. Databázu je možné nainštalovať ako server a ovládať ju buď pomocou terminálu alebo rôznych GUI, ktoré sú dostupné na webe. Jedným z najpoužívanejších GUI je práve PgAdmin, ktorý bude použitý na správu databázy. Toto rozšírenie umožňuje pomocou SQL vytvárať rôzne tabuľky, stĺpce a pridávať informácie do databázy. [86]

MongoDB

MongoDB je na rozdiel od PostgreSQL nerelačná databáza. Hlavným rozdielom je ukladanie a správa dát. Tradičné databázy využívajú SQL jazyk na tvorbu dotazov. Nerelačná databáza využíva silu programovacieho jazyku na ukladanie a správu dát. [89]

Hlavným účelom MongoDB je spracovanie veľkého distribuovaných dát, ktoré sú dokumentovo orientované. Táto databáza zvláda ukladanie veľkého množstva dát a následnú prácu s nimi vo veľkej rýchlosti. Napríklad sa využíva pri vybavovaní JavaScriptových príkazov, indexovaní či load balancingu. Databáza nevyužíva tabuľky ako to je pri relačných databázach ale jej architektúra sa skladá z kolekcií dokumentov a polí. Databázové riešenie je podporované medzi mnohými programovacími jazykmi napríklad: C, C++, Python, Ruby a iné. [90] Štruktúru tejto databázy je možné vidieť na obrázku 3.10.



Obr. 3.10: Štruktúra MongDB.

3.4 Systém kontroly verzií

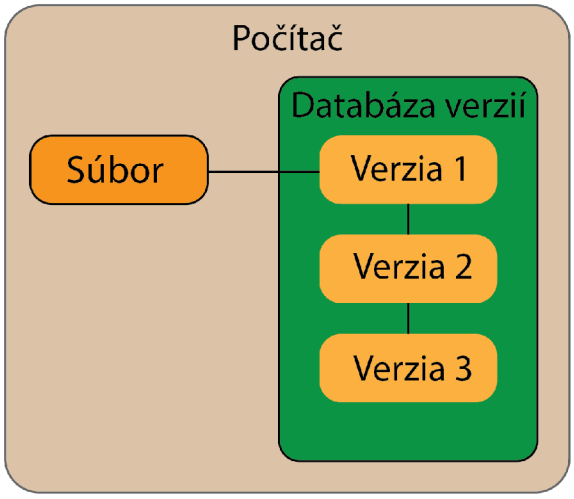
Zásadnou časťou vyvíjanej aplikácie je verzovací nástroj. Existujú mnohé nástroje na kontrolu verzií systému. Ako napríklad Git, Mercurial, Bazaar a mnohé ďalšie. Najrozšírenejším je však práve Git. Je to open source projekt, ktorý jej schopný komunikovať pomocou protokolov ako napríklad HTTP, FTP, ssh. Umožňuje efektívne spravovať malé ale aj veľké projekty. Git sa v dnešnej dobe najčastejšie používa pomocou webových aplikácií ako je GitHub alebo GitLab. Pre projekt bol zvolený GitLab, o ktorom je viac napísané v nasledujúcej kapitole 3.4.2.

Kontrola verzií je dôležitým nástrojom pri tvorbe elektronického diela. Umožňuje zaznamenávať stav vytváraného projektu v čase a vracať sa späť k tomu, ako dielo vyzeralo pred určitými zmenami.

Napríklad pri tvorbe aplikácie je možné v prípade nejakej chyby dohľadať, kedy program ešte fungoval správne a kto urobil poslednú zmenu. Nakoľko pri vývoji aplikácie na nej zväčša pracuje skupina ľudí a je nesmierne dôležité vedieť, kto a kedy vykonal zmenu.

Existujú 3 typy verzovacieho systém, lokálny verzovací systém, centralizovaný verzovací systém, distribuovaný verzovací systém. Bližšie budú popísane v nasledujúcej časti.

Lokálny verzovací systém

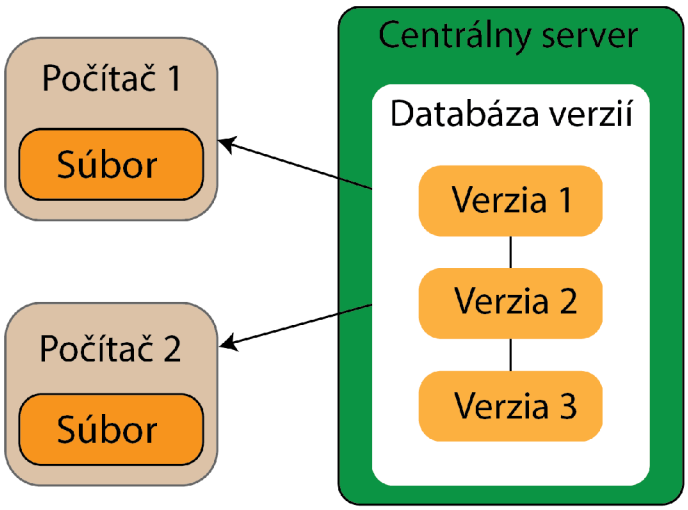


Obr. 3.11: Lokálny verzovací systém.

Systém funguje na základe jednoduchej databázy, ktorá je uschovaná lokálne na počítači. Táto databáza ukladá každú zmenu vykonanú v súbore, ktorý je do nej pridaný. Jedným z najpopulárnejších nástrojov je RCS, ktorý zaznamenával rozdiely medzi sledovanými súbormi a ukladal ich v špeciálnom formáte na disk. Užívateľ mal potom možnosť znovu vytvoriť súbor v stave, v akom bol pred nejakým časom.

Centralizovaný verzovací systém

Problém predchádzajúceho systému nastal vtedy keď na jednom projekte chcelo pracovať viacero ľudí. Tento problém vyriešil centralizovaný verzovací systém, ktorý používal jeden server, na ktorom bola práca zálohovaná. Uchovával taktiež evidenciu o počte ľudí, ktorí na projekte pracovali. Umožňoval sledovať do určitého bodu, kto na čom pracuje. Problémom tohto systému však bolo, že všetky dáta sa rovnako ako



Obr. 3.12: Centralizovaný verzovací systém.

aj pri lokálnom verzovacom systéme nachádzali na jednom mieste. Čiže pri vyradení serveru, alebo poruche bol znemožnený prístup k projektu, alebo bol stratený.

Distribuovaný verzovací systém

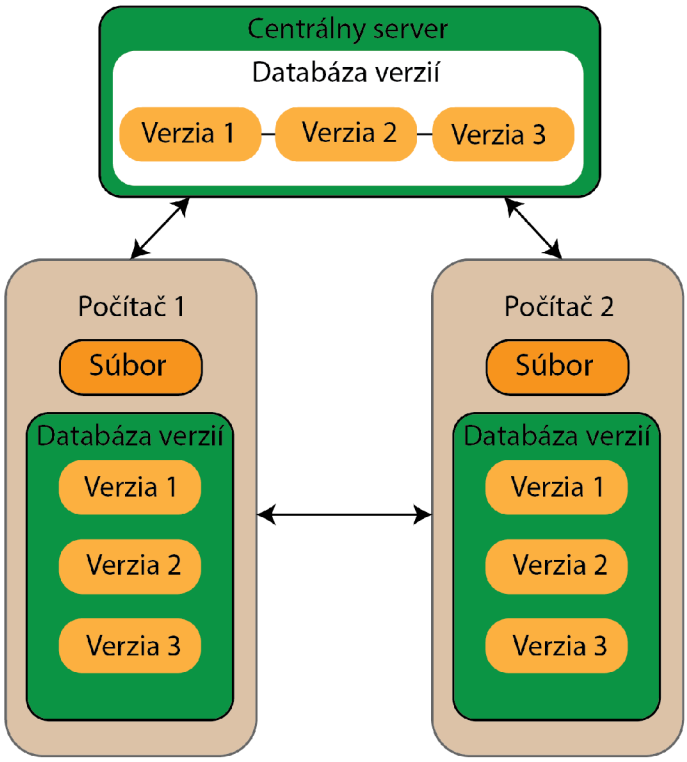
Dnes je najviac používaným systémom. Distribuovaný verzovací systém je základom práve pre Git. Používatelia nemajú k dispozícii len najnovší záznam projektu, na ktorom pracujú, ale naopak majú k dispozícii celú históriu vývoja súborov, na ktorých pracujú. Tento systém umožňuje vzdialene pracovať rôznym ľudom na rôznych projektoch. Funkcionalita je založená na tom, že užívateľ má lokálnu databázu kam sa ukladajú súbory, ktoré sú ďalej posielané do databázy na vzdialenom servery.

3.4.1 Verzovací systém Git

Najväčším rozdielom Gitu oproti iným verzovacím systémom je ten, akým štýlom sa pozerá na uschovávané dáta. Väčšina týchto systémov ukladá informácie ako zoznam zmien týkajúcich sa súboru. Pozerajú sa na ukladané informácie ako na súbor záznamov, ktoré sa týkajú jednotlivých súborov počas celej doby zaznamenávania.

Git sa však na dáta pozerá inak. Zaznamenáva dáta ako momenty malého súborového systému. Akoby urobil fotku toho, v akom stave sa momentálne nachádza súbor, túto fotku následne uloží a s ňou aj referenciu, pod ktorou sa dá dohľadať. Ak sa súbor nezmenil, ale znovu ho chceme uložiť, tak Git neukladá ďalšiu fotku, ale vytvorí len novú referenciu, na ktorú sa odkáže.

Väčšinu úkonov, ktoré Git ponúka je možné robiť lokálne. Čo Gitu umožňuje, aby nebol odkázaný na iný počítač alebo server. Čo pri niektorých verzovacích systémoch mohlo spôsobovať oneskorenie. Informácie v kapitole 3.4 sú čerpané z [87, 88]



Obr. 3.13: Distribuovaný verzovací systém.

3.4.2 Webová aplikácia GitLab

GitLab je webová aplikácia založená na Gite, ktorá spravuje repozitáre. Užívatelia si môžu klonovať projekty na svoje lokálne počítače a zálohovať ich späť na GitHub pomocou príkazov, ktoré používa Git. Umožňuje sledovanie vývoju kódu v čase. Umožňuje tvorbu wiki, je na ňom vedené diskusné fórum, kde vývojári môžu rozoberať rôzne otázky. [91]

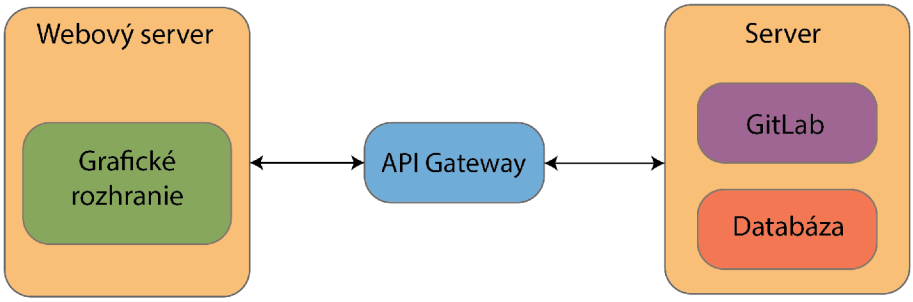
GitLab je používaný viac ako 50 000 organizáciami. Počas uplynulých rokov zaznamenal GitLab rýchly rast a vytvorenie silnej komunity.

Pre aplikáciu, ktorá bude vyvíjaná je GitLab nesmierne dôležitou súčasťou. Keďže GitLab umožňuje vďaka svojej opensource licencii využitie kýmkoľvek. Bude nainštalovaný na server, kde bude slúžiť k správe vyvíjaných korelačných pravidiel. Bude ho možné používať aj na prideľovanie práv rôznym užívateľom a vedenie evidencie o užívateľoch. [91]

Aplikácia bude komunikovať so serverom, na ktorom je nainštalovaný GitLab pomocou API rozhrania. Kde budú z webového GUI vytvárané a upravované pravidlá a následne budú posielané na server.

3.5 Stabilná prevádzka aplikácie

K spracovávaniu bude slúžiť server, je to počítač, ktorý slúži ako poskytovateľ služieb viacerým užívateľom. Server slúži ako prostredie, v ktorom je možné inštalovať rôzne aplikácie ako napríklad databázu, mikroservisy či webové rozhranie.



Obr. 3.14: Schéma serverov.

Vyvíjaná aplikácia bude používať dva servery. Jeden webový kde bude umiestnené GUI a druhý, ktorý bude obsahovať GitLab a PostgreSQL databázu, schému je možné vidieť na obrázku 3.14. Komunikáciu medzi servermi bude zabezpečovať API.

Jedným z najčastejšie používaných operačných systémov pre server je Linux. Pri príprave serveru bude použitá distribúcia Linuxu s názvom CentOS 7 viac o operačnom systéme v nasledujúcej kapitole. Webový server bude používať distribúciu Ubuntu.

Serverový operačný systém CentOS 7

CentOS 7 je postavený na RHEL (Red Hat Enterprise Linux) open source kóde, ktorý je zadarmo a je primárne mierený pre podniky. Nakoľko nové verzie sú menej frekventované ako pri Ubuntu alebo Debiane je podstatne stabilnejší a vhodnejší na použitie, ako serverové riešenie v podniku. Jeho stabilita bola dôvod vybrať ho ako operačný systém, na ktorom funguje server s GitLabom. [92]

4 Návrh aplikácie

V tejto časti sa bude nachádzať návrh aplikácie. V čom spočíva navrhnutie frontendovej a backendovej časti a komunikácie medzi nimi. Aplikácia bude nasadená v experimentálnom prostredí, ktoré má za úlohu simulovať reálnu prevádzku na internete. Návrh tohto prostredia je možné vidieť na obrázku 5.2. Nasadeniu aplikácie do tohto prostredia sa bude venovať kapitola 5.4. Toto prostredie bude využité na otestovanie funkcionality a správania sa aplikácie.

4.1 Špecifiká ukladania dát

K zabezpečeniu fungovania aplikácie bolo potrebné vymyslieť ako budú ukladané dáta, ktoré sú vytvárané počas používania aplikácie užívateľom. Budú použité dva databázové systémy a to MongoDB a PostgreSQL.

PostgreSQL

Táto databáza bude využívaná na ukladanie dát ohľadom užívateľa. Dáta, ktoré tu budú uložené sú vytvorené pri tvorbe užívateľa po zavolaní funkcie `CreateUser`, ktorú je možné vidieť na obrázku 4.4. Na nasledujúcom obrázku pod číslom 4.1, je možné vidieť návrh tabuľky, do ktorej sú ukladané dáta o užívateľoch. Jednotlivé položky sú podrobnejšie opísané v kapitole 5.3.2.

MongoDB

Pre uspokojenie požiadaviek aplikácie, kde je potrebné ukladať commit hashe vytvorených súborov, bolo potrebné vymyslieť logiku pomocou ktorej budú tieto hashe uložené. Táto logika je podrobnejšie vysvetlená v kapitole 5.3.3. Na základe vymysleného postupu bolo potrebné dáta ukladať.

Pre ukladanie týchto dát bol zvolený práve databázový systém MongoDB, ktorý veľmi dobre dokáže spracovávať užívateľmi generované dáta. Umožňuje napríklad uskladňovať komentáre, statusy, užívateľské hodnotenia a mnohé iné [94]. Aj z tohto dôvodu bolo zvolená pre uskladňovanie hashov práve táto databáza. Nakoľko tieto hashe bude užívateľ pravidelne a často generovať.

MongoDB ukladá dáta do štruktúry ktorá je podobná ako JSON súbor. Toto umožňuje jednoduché spracovanie dát vo webovej aplikácií kam sú dáta z tejto databázy odosielané. Viacej o tom ako sú dáta využité je opísané v 4.4. Toto jednoduché spracovanie dát pomocou frontendovej aplikácie bol ďalší dôvod k výberu MongoDB pre ukladanie commit hashov.

Obr. 4.1: Štruktúra PostgreSQL tabuľky.

Štruktúru MongoDB je možné vidieť na obrázku 4.2. Tento systém sa skladá z databázového serveru, kde sa nachádzajú jednotlivé databázy. Každá databáza obsahuje kolekcie čo je súbor dokumentov, ktoré obsahujú dáta. Dokument je štruktúrou podobný JSON súboru, sú ňom uložené jednotlivé dáta. Tento dokument je vytvorený na základe MongoDB validátoru, ktorý určuje, to akú štruktúru bude dokument mať. Validátor slúži ako zabezpečenie, toho že do databázy sa pridajú len správne naformátované dáta [1]. [95]

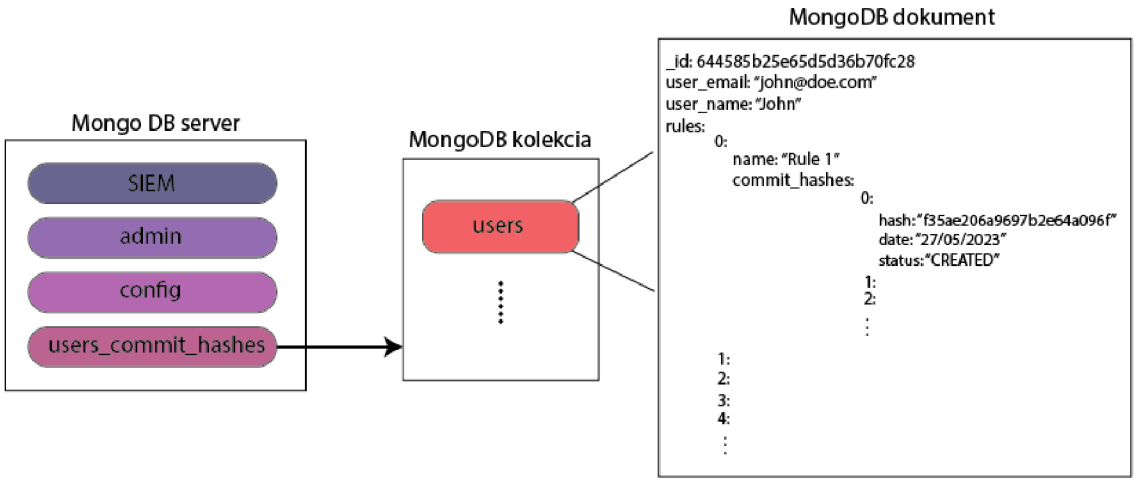
Obrázok 4.2 zobrazuje databázový server MongDB. Na tomto servery sa nachádzajú 4 databázy. Databáza **admin**, ktorá obsahuje dáta spojené s autorizáciou, autentizáciou [2] a **config**, kde sa nachádzajú konfiguračné súbory [3]. Databáza s názvom **SIEM** je rozobratá v nasledovnej kapitole pod číslom 4.2.

Časť s názvom **users_commit_hashes** ukladá commit hashe tvorených resp. upravovaných súborov užívateľom. Nachádza sa tú kolekcia users, ktorá obsahuje

[1] Viacej informácií o validátore dostupné z: `https://www.mongodb.com/docs/manual/core/schema-validation/`

[2] Bližšie informácie k tejto databáze dostupné z: `https://www.mongodb.com/docs/manual/tutorial/manage-users-and-roles/`

[3] Viac o tejto databáze dostupné z: `https://www.mongodb.com/docs/manual/reference/config-database/`



Obr. 4.2: Štruktúra databázy ukladajúcej hashe.

dokument podobný JSON súboru. Štruktúra tohto dokumentu pozostáva z nasledovných častí:

- **_id** – id je automaticky generované systémom MongoDb.
- **user_email** – táto položka ukladá užívateľov email, v prípade potreby to umožňuje lepšiu orientáciu v systéme.
- **user_name** – meno užívateľa sa tu nachádza z rovnakého dôvodu ako e-mail.
- **rules** – v tejto časti sa nachádza pole, ktorého číslovanie začína od 0 až po *n*. Každé pole obsahuje časť **name** a časť **commit_hashes**.
  - **name** – položka obsahuje názov vytvoreného pravidla
  - **commit_hashes** – ďalšou častou poľa je ďalšie pole, ktoré obsahuje položky **hash**, **date** a **status**. Toto pole je znovu číslované od 1 až po *n*.
    - **hash** – sem sa ukladá commit hash daného pravidla.
    - **date** – dátum obsahuje informáciu o tom kedy bolo pravidlo vytvorené alebo upravené.
    - **status** – tento segment indikuje operáciu, ktorá bola na pravidle vykonaná. Môže obsahovať dve hodnoty a to `CREATED` alebo `UPDATED`.

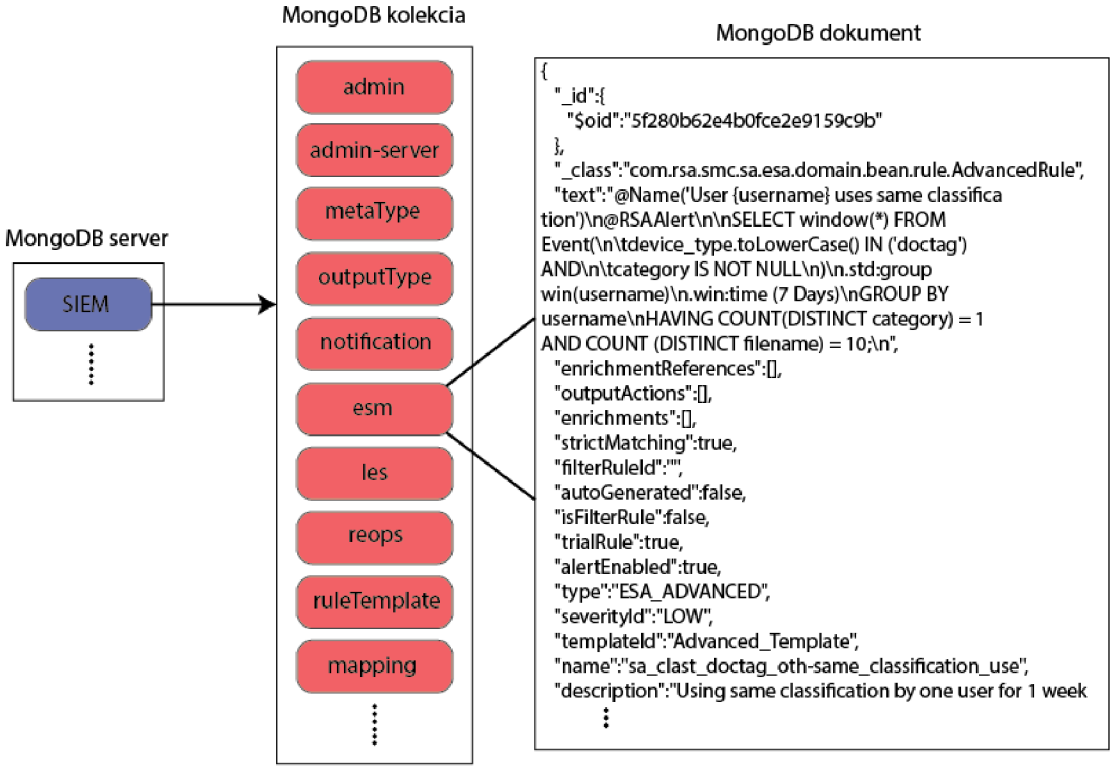
4.2 Databáza SIEM aplikácie

Vyvíjaná aplikácia je určená na vývoj a správu korelačných pravidiel SIEM systémov. Je teda potrebné vytvoriť spojenie s danou technológiou. Prepojenie bude uskutočnené so SIEM riešením s názvom *RSA Net Witness*.

SIEM sa bude s vyvíjanou aplikáciou prepájať pomocou databázového systému MongoDB. Je to najlepší spôsob, ako aplikáciu a technológiu SIEM prepojiť, nakoľko SIEM si všetky svoje dáta berie z databázových súborov. Najjednoduchšou

cestou je teda pridávanie a úprava korelačných pravidiel priamo v databáze s ktorou komunikuje SIEM.

Štruktúru databázy SIEM je možné vidieť na obrázku 4.3. Databáza obsahuje veľké množstvo kolekcií, niektoré z nich sú zobrazené na obrázku 4.3. Každá kolekcia obsahuje dokumenty. Dokumenty všetkých kolekcií majú podobnú štruktúru ako je vidieť na obrázku 4.3. Všetky časti súboru sú generované technológiou SIEM. Najdôležitejšou časťou, ktorá je potrebná pre správnu funkciu vyvíjanej aplikácie je časť s názvom **text**. V tomto segmente sa nachádza samotný text korelačného pravidla, ktoré je pomocou webovej aplikácie vyvíjané alebo upravované.



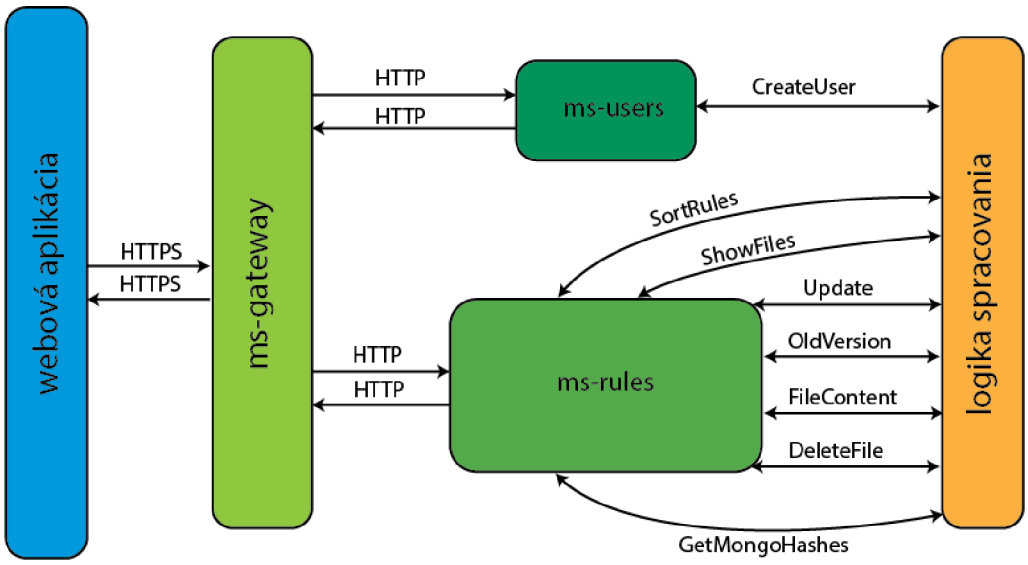
Obr. 4.3: Štruktúra databázy SIEM.

4.3 Návrh backendu

Backendová časť aplikácie má za úlohu využívať naprogramovanú logiku spracovania dát. Tieto dáta sú prijímané pomocou HTTP požiadaviek, ktoré prichádzajú z webovej aplikácie. Ku spracovaniu týchto požiadaviek, je potrebné vytvoriť určitý mechanizmus, ktorý sa volá API.

API umožňuje jednoduché spracovanie dát a ich následné posúvanie ďalej do rôznych funkcií, ktoré vykonávajú rôzne operácie.

Štruktúra backendu pozostáva z 3 častí. Prvá časť sa nazýva `ms-gateway`, ktorá slúži ako brána pred vstupom do citlivej časti mikroservís a preposiela im dáta z webovej aplikácie. V segmente citlivých mikroservís sa nachádza `ms-user` a `ms-rules`. Na rozdiel od `ms-gateway` v týchto častiach dochádza už k reálnemu spracovaniu dát. Bližší popis ku každej z mikroservís sa nachádza v kapitole 5.3.



Obr. 4.4: Návrh backendu aplikácie.

Na obrázku 4.4, je možné vidieť časť webovej aplikácie, ktorá bude bližšie rozobraná v kapitole 4.4. Z tejto časti putujú HTTPS požiadavky smerom na `ms-gateway`. Kde sa zistí, pre ktorú z dvoch mikroservís patrí daná požiadavka. Následne je požiadavka odoslaná k správnej mikroservise pomocou HTTP požiadavky. Rozdielnosť medzi požiadavkami je z dôvodu toho, že požiadavky medzi webovou aplikáciou a `ms-gateway` je potrebné chrániť pred potencionálnym útokom, keďže táto časť aplikácie je dostupná útočníkovi. Komunikáciu medzi mikroservisami nie je už potrebné chrániť, nakoľko do tejto časti sa útočník nedostane. Ak požiadavka dorazí k mikroservise `ms-users`, ktorá sa venuje tvorbe nových užívateľov, je zavolaná funkcia `CreateUser`. Po zavolaní, funkcia začne vykonávať naprogramovanú logiku, ktorá je popísaná v 5.3.2 a odošle spracované dáta späť najprv pomocou HTTP požiadavky k bráne a potom pomocou HTTPS požiadavky k webovej aplikácii.

Druhou mikroservisou, na ktorú môže byť požiadavka presmerovaná je `ms-rules`. Tento segment spracúva všetky úkony, ktoré sa týkajú zabezpečenia plynulého vývoju korelačných pravidiel. Funkcie, ktoré si môže mikroservisa zavolať sú nasledovné:

- `SortRules`;
- `ShowFiles`;

- `Update`;
- `OldVersion`;
- `FileContent`;
- `DeleteFile`;
- `GetMongoHashes`;

Po zavolaní ktorejkoľvek funkcie sa začne vykonávať jej predpripravená logika. Každá z týchto funkcií je podrobne popísaná v kapitole 5.3.3. Funkcie po ich zavolaní fungujú na rovnakom princípe ako funkcia z `ms-users`, čiže sú spätne odoslané požadované dáta pomocou HTTP a HTTPS požiadavky.

4.4 Návrh frontendu

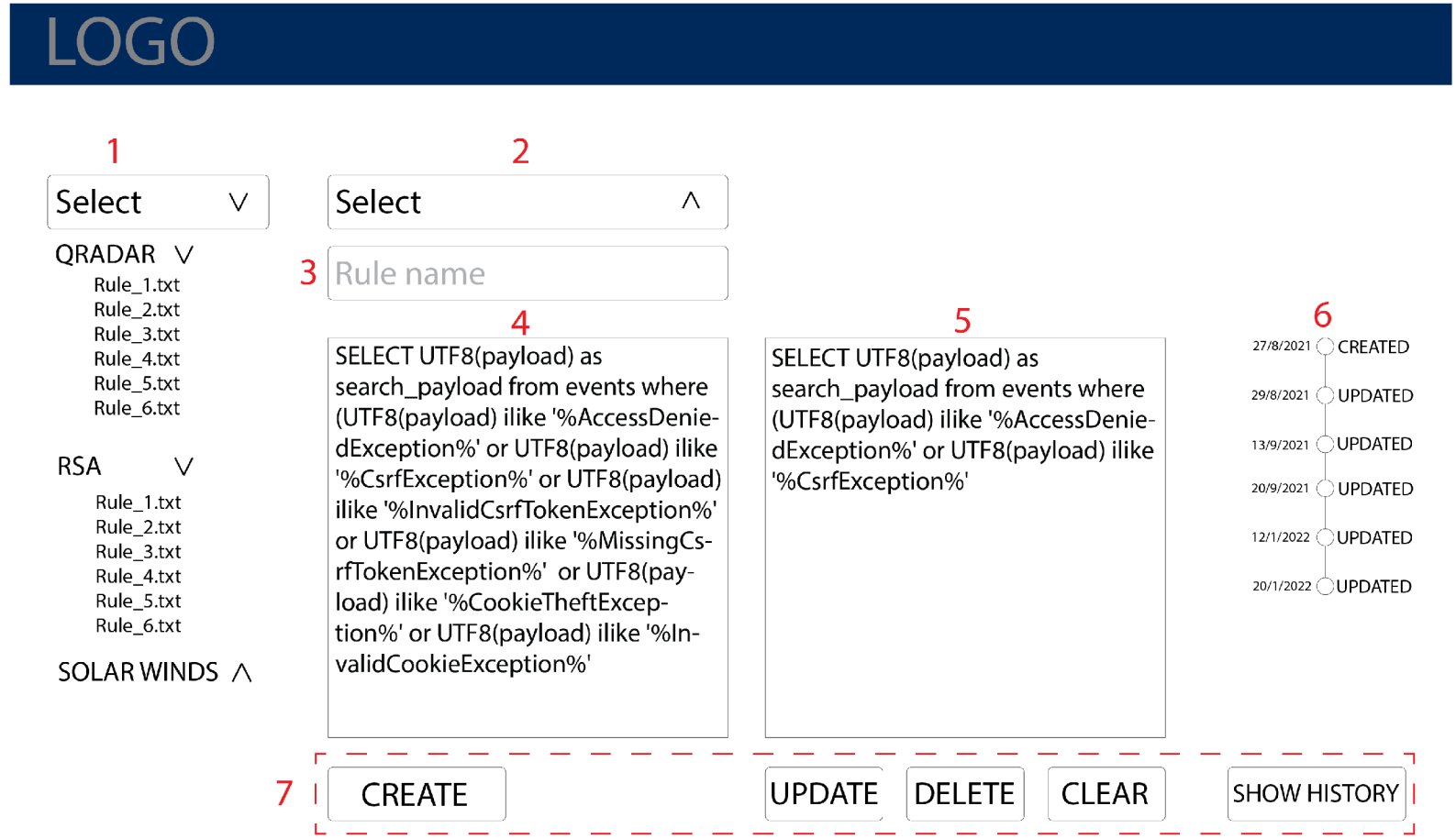
Webová aplikácia má slúžiť ako jednoduchý nástroj k správe korelačných pravidiel. Užívateľovi má uľahčiť prácu na pravidlách a vykonávanie jednotlivých úkonov, ktoré sa v backende vykonávajú na základe funkcií, ktoré boli popísané v kapitolách 5.3.2 a 5.3.3.

Aplikácia bude teda umožňovať jednoduchú tvorbu, úpravu a mazanie pravidiel. Taktiež bude schopná zobraziť históriu zmien jednotlivých pravidiel. Každý vykonaný úkon vo webovej aplikácií je poslaný na backendový server pomocou HTTPS požiadavky. Na základe požiadavky sú spätne poslané dáta v podobe JSON súboru. Tieto dáta sú následne v aplikácii spracované a zobrazené užívateľovi v grafickom rozhraní.

Hlavná časť aplikácie bude disponovať dvoma rolovacími komponentami, ktoré po kliknutí zobrazia obsah. Prvý komponent je možné vidieť na obrázku 4.5 pod číslom 1. Tento komponent má za úlohu poslať HTTPS požiadavku na backendový server na základe zvolenej zložky. Názvy zložiek z obrázka sú `QRADAR`, `RSA`, `SOLAR WINDS`. Po kliknutí na zložku je teda odoslaná požiadavka, ktorá pošle naspäť názvy súborov. Tieto názvy budú následne zobrazené pod sebou vo webovej stránke. Po kliknutí na ľubovolný názov pravidla je poslaný nová požiadavka, v ktorej odpovedi príde obsah konkrétneho pravidla. Tento obsah bude následne zobrazený do textového okna, ktoré je na obrázku 4.5 možné vidieť pod číslom 5.

V okne s číslom 5 je následne možné pravidlo upraviť a aktualizovať pomocou tlačítka `UPDATE` v sekcii s číslom 7. Po stlačení tlačítka je poslaná ďalšia požiadavka na server, ktorý má za úlohu aktualizovať obsah pravidla a uložiť hash tejto akcie do databázy.

Tlačítko `SHOW HISTORY` zo sekcie s číslom 7 slúži na zobrazenie histórie vybraného pravidla. Po kliknutí na tlačítko sa vo webovej aplikácii zobrazí časová os, ktorú je možné vidieť na obrázku 4.5 pod číslom 6.



Obr. 4.5: Návrh webovej aplikácie.

Časová os obsahuje údaje, ako dátum kedy bolo pravidlo vytvorené respektíve aktualizované a údaj o tom aká akcia bola vykonaná. Pokiaľ si užívateľ chce pozrieť, ako pravidlo vyzeralo v minulosti, stačí ak klikne na dátum, ktorý ho zaujíma. Po kliknutí je vytvorená nová požiadavka, ktorá spätne vráti obsah pravidla z jeho histórie.

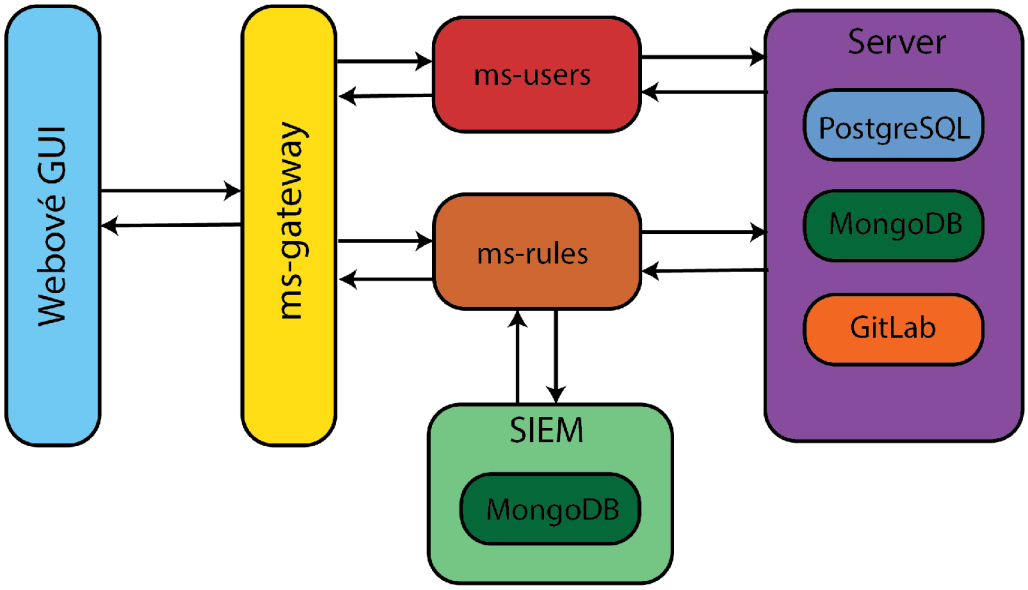
V sekcii s číslom 7 sa nachádzajú ešte dve tlačítka, ktoré majú názov `DELETE` a `CLEAR`. Tlačítkom `DELETE` užívateľ môže zmazať dané pravidlo, ktoré si zobrazil a tlačítkom `CLEAR` užívateľ vyčistí textové pole, v ktorom sa nachádza zobrazené pravidlo.

Rolovacia komponenta z obrázku 4.5 s číslom 2, slúži na tvorbu nových pravidiel. Po kliknutí na ňu sa zobrazia názvy zložiek do ktorých je možné pravidlo uložiť. Názvy zložiek sú `QRADAR`, `RSA`, `SOLAR WINDS`. Po vybratí je dôležité, aby užívateľ napísal názov tvoreného pravidla, ktorý môže napísať do textového poľa pod číslom 3. Akonáhle je užívateľ spokojný so svojim pravidlom môže si ho uložiť, k tomuto slúži tlačidlo `CREATE`, ktoré sa nachádza v sekcii 7. Implementáciu aplikácie je možné vidieť na obrázku 5.20.

5 Implementácia aplikácie

Implementácia aplikácie bude realizovaná v dvoch fázach. Prvá fáza sa bude venovať príprave prostredia, v čom spočíva príprava serveru, inštalácia a konfigurácia potrebných aplikácií. Medzi tieto aplikácie patria databázové systémy PostgreSQL, MongoDB a aplikácia GitLab.

Úlohou GitLabu je sledovať a spravovať vývoj korelačných pravidiel, ktoré bude užívateľ vytvárať pomocou webového rozhrania. Rozhranie bude spolupracovať s mikroservisami, ktoré sa budú nachádzať na servery Miri viď. tabuľka číslo 5.2.



Obr. 5.1: Návrh systému.

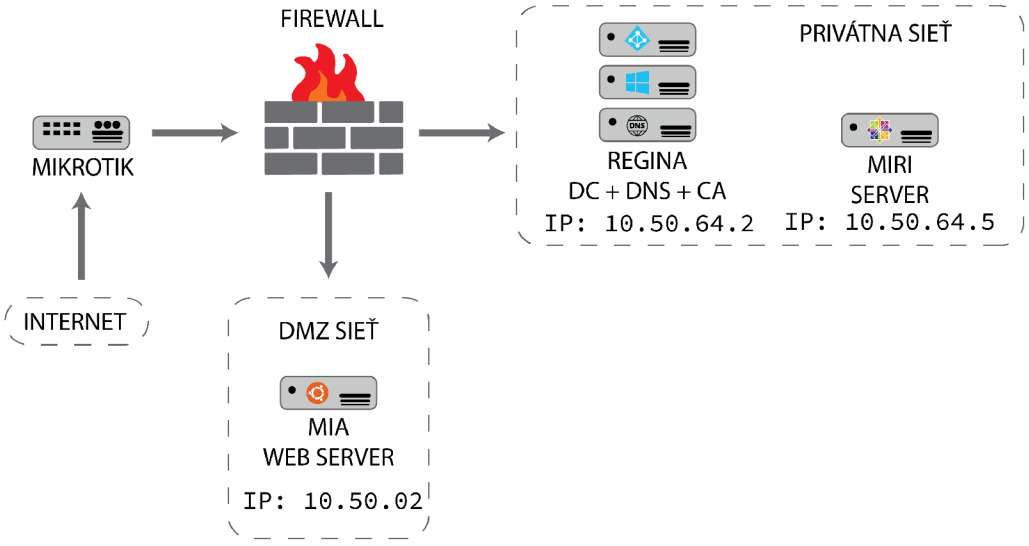
Druhá fáza bude zameraná na tvorbu samotných mikroservís a ich implementaciu. Pre označenie mikroservís bude použitá skratka a názov `ms-názov`. Aplikácia bude využívať 3 mikroservisy, a to `ms-user`, `ms-rules`, `ms-gateway`. Na obrázku 5.1 je možné vidieť podrobnejší návrh systému. Prvá mikroservisa je `ms-gateway`, má za úlohu spracovávať prichádzajúce požiadavky a preposielať ich ďalej príslušným mikroservisám. Spracovávanie požiadaviek, ktoré sa týkajú informácií o užívateľoch má na za úlohu `ms-user`. Najdôležitejšou mikroservisou je `ms-rules`, ktorá spracováva všetky požiadavky týkajúce sa tvorby pravidiel a zabezpečuje ich plynulú tvorbu. Táto mikroservisa je napojená na SIEM databázu v ktorej sa ukladajú korelačné pravidlá. Mikroservisy budú bližšie popísané v kapitole 5.3. V druhej fáze bude taktiež vytvorené webové rozhranie, ktoré umožní užívateľovi prácu na korelačných pravidlách a ich tvorbu.

5.1 Fáza I

Prvá fáza bude popisovať experimentálne prostredia pre spustenie aplikácie a jej správnu funkcionalitu. Postupne bude opísaný postup spustenia serveru, následnej inštalácie potrebných technológií ako Gitlab, PostgreSQL a MongoDB.

Experimentálne prostredie, ktoré je možné vidieť na obrázku 5.2 bude využívať server, ktorý slúži ako sprostredkovateľ služieb ukladania informácií. Mikroservisa `ms-user` a `ms-rules` bude využívať server na ukladanie informácií o užívateľoch do PostgreSQL databázy a správu korelačných pravidiel pomocou GitLabu. Server bude mať za úlohu spracovať veľké množstvo požiadaviek, ktoré mu budú prichádzať z mikroservís. K tomu aby boli dáta správne spracované je potrebné správne nainštalovať a nakonfigurovať dôležité aplikácie, ktorými sú PostgreSQL, MongoDB a GitLab.

5.2 Popis experimentálneho prostredia



Obr. 5.2: Architektúra prostredia.

Prostredie je zložené zo smerovaču Mikrotik, Firewallu, webového serveru, ďalšou časťou prostredia je privátna sieť v ktorej sa nachádza certifikačná autorita (CA), DNS server a radiču domény (DC) a DMZ sieť v ktorej je umiestnený server starajúci sa o backend aplikácie. Do prostredia je možné sa pripojiť cez internet. K vstupu je však potrebné mať nakonfigurovanú VPN. Prichádzajúcu premávku riadi smerovač mikrotik, ktorý je chránený firewallom. Pomocou smerovaču je možné sa dostať

k webovému serveru, kde sa bude nachádzať GUI, alebo do privátnej siete, v ktorej bude pripravený server s náležitými funkcionalitami a aplikáciami, ktoré boli prebraté v kapitole 3.

Tab. 5.1: Zoznam serverov

Názov serveru	IP adresa	Operačný systém	Úloha
miri.vmware.fekt.cz	10.50.64.5	CentOS	Backend
mia.vmware.fekt.cz	10.50.0.2	Ubuntu	Frontend
regina.vmware.fekt.cz	10.50.64.2	Widows	DC + DNS + CA

5.2.1 Inštalácia serveru GitLab

GitLab je hlavnou súčasťou navrhnutej aplikácie. Zabezpečuje jej plynulý chod a umožňuje správu korelačných pravidiel. Z tohto dôvodu bude návod jeho inštalácie priložený v prílohe C. V návode je vysvetlený každý potrebný krok k správnej inštalácii aplikácie na server CentOS 7.

5.2.2 Inštalácia PostgreSQL

Pre inštaláciu databázy PostgreSQL bude využitá technológia Docker. Docker umožňuje jednoduchú inštaláciu aplikácie a následnú prácu s ňou. [96]. Po nainštalovaní databázového serveru je možné sa vzdialene pripojiť na nainštalovaný server pomocou IP adresy serveru a portu na ktorom je spustený proces. Po pripojení sa, je potrebné do príkazového riadku zadať príkaz `psql` [97]. Následne je možné vytvárať databázy.

5.2.3 Inštalácia MongoDB

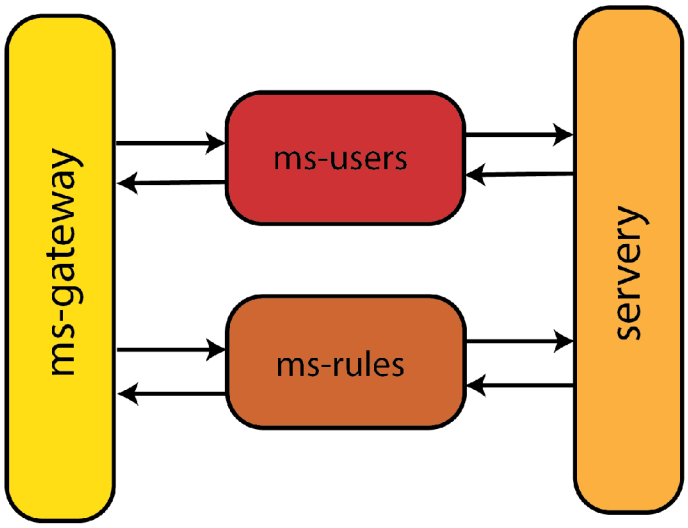
Na inštaláciu MongoDB pomocou technológie Docker je potrebné stiahnuť obraz databázového serveru. Tento obraz je možné nájsť na webovej stránke DockerHub [1]. Rovnako ako pri inštalácii PostgreSQL je možné sa vzdialene pripojiť na databázový server. Na pripojenie k databázovému serveru je možné využiť aplikáciu MongoDB Compass [2], ktorá umožňuje jednoduché pripojenie sa k serveru a poskytuje užívateľovi prehľadné GUI. [104]

[1] Rôzne obrazy MongoDB dostupné z: `https://hub.docker.com/r/mongodb/mongodb-community-server/tags`

[2] Viac informácii o aplikácii MongoDB Compass dostupné z `https://www.mongodb.com/products/compass`

5.3 Fáza II

V tejto fáze budú opísané jednotlivé mikroservisy, ich funkcionalita a hlavný účel. Na obrázku 5.3 je možné vidieť štruktúru mikroservís, ktoré sú využívané na backende aplikácie a skladajú sa z `ms-gateway`, `ms-user` a `ms-rules`. Všetky funkcie, ktoré sa nachádzajú v jednotlivých mikroservisách sú volané pomocou HTTP požiadavkou, ktoré prechádzajú cez `ms-gateway`.



Obr. 5.3: Štruktúra mikroservís.

Aby bol zabezpečený plynulý chod aplikácie a v prípade potreby vykonávania rôznej údržby bolo rozhodnuté, že mikroservisy budú práve tri. Pri tejto architektúre sú rôzne funkcionality rozdelené do logických celkov a v prípade potrebnej úpravy je možné sa jednoduchšie orientovať v tejto štruktúre.

5.3.1 Vývoj komponenty `ms-gateway`

Mikroservisa `ms-gateway` slúži ako ochranná brána pred vstupom k citlivým informáciám. Architektúra, ktorá zahŕňa túto bránu umožňuje skryť pred užívateľom porty na ktorých sú spojazdnené zvyšné mikroservisy. Tento spôsob zvyšuje celkovú bezpečnosť aplikácie pred možným útokom. Taktiež zjednodušuje tvorbu webového rozhrania kde nie je potrebné preposlať požiadavky na rôzne IP adresy alebo porty. Všetky požiadavky sú posielané cez bránu, ktorá ich na základe naprogramovanej logiky roztriedi a prepošle na správnu mikroservisu.

Logika triedenia spočíva v tom, že mikroservisa má preddefinované endpointy, ktoré sa uvádzajú za IP adresu alebo názov webovej stránky. IP adresa s endpointom môže vyzerať nasledovne `http://127.0.0.0/<endpoint>` príklad zobrazuje lokalhost IP adresu za, ktorú je možné dosadiť ľubovolný názov endpointu avšak tento

názov musí byť zadefinovaný v samotnej mikroservise v opačnom prípade by bol do webového prehliadača vrátený chybový kód `404 Not Found`. Ukážku definovania endpointov v aplikácii je možné vidieť na obrázku 5.4.

```
api.add_resource(UserLogin, '/v1/login')
api.add_resource(SortRules, '/v1/postRule')
api.add_resource(ShowFiles, '/v1/showFiles')
api.add_resource(UpdateFiles, '/v1/updateFile')
api.add_resource(CreateUser, '/v1/createUser')
api.add_resource(GetOldVersion, '/v1/oldVersion')
api.add_resource(GetContent, '/v1/GetContent')
api.add_resource(GetMongoData, '/v1/GetMongo')
api.add_resource(DeletFileFromRepo, '/v1/DeletFile')
```

Obr. 5.4: Ukážka definovanie endpointov v aplikácii.

Každý riadok v tomto výpise definuje jeden endpoint a funkciu ktorá sa zavolá po prijatí požiadavky z webovej aplikácie. V zátvorke je teda na prvom mieste názov funkcie a na druhom mieste ľubovolný názov endpointu. Endpoint musí vždy začínať s `/`.

Mikroservisa obsahuje špeciálny endpoint, ktorým je `/v1/login`. Tento endpoint je odlišný od ostatných a jeho úlohou je zabezpečenie prihlásenia užívateľa do aplikácie. Prihlásenie funguje tak, že po zadaní prihlasovacích údajov užívateľom do prihlasovacieho okna sú tieto údaje poslané práve na endpoint `/v1/login`. Tu sa spracujú a prepošlú do databázového systému, kde sa zistí, či daný užívateľ existuje alebo nie. Ak užívateľ neexistuje pošle sa mu hláška o zadaní nesprávnych údajov.

Pri úspešnom prihlásení táto funkcia vygeneruje JWT (JSON Web Token), ktorý umožňuje užívateľovi pracovať v aplikácii bez toho, aby sa pri každom úkone musel znovu prihlasovať. Tento token obsahuje časovú známku po ktorej jeho platnosť vyprší a už ho nie je možné používať pri práci s aplikáciou a užívateľ sa na vytvorenie nového bude musieť opätovne prihlásiť. Druhou informáciou, ktorú obsahuje je verejné ID užívateľa na základe, ktorého aplikácia vie o akého užívateľa ide. Toto ID je používané pri komunikácii s databázou pokiaľ je potrebné z nej brať údaje. Ukážku JWT je možné vidieť na obrázku. K čítaniu JWT sa na bakcende používa parsovacia funkcia ktorá ho dokáže rozdeliť a jednotlivé časti rozšifrovať a získať potrebné údaje.

Endpoint s názvom `/v1/postRule` po prijatí požiadavky prepošle dáta konkrétnej mikroservise, ktorá je definovaná vo funkcii. Funkciu je možne vidieť na obrázku 5.6.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwi
aWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Obr. 5.5: Ukážka JSON Web Tokenu.

```
class SortRules(Resource):
    def post(self):
        data = request.data
        req = requests.post(ms_rule_url + config_gateway.MS_RULES_API_ENDPOINT_DAT, data)
        return req.json()
```

Obr. 5.6: Funkcia endpointu.

Premenná `req` ukladá v sebe požiadavku v ktorom je definovaná URL na ktorú má byť požiadavka s dátami, ktoré sú uložené v premennej `data` preposlané. URL mikroservisy je zložená z dvoch premenných `ms_rule_url` v ktorej je uložená adresa s portom serveru mikroservisy `ms-rules` a `MS_RULES_API_ENDPOINT`, ktorý ukladá endpoint príslušný danej požiadavke. Z bezpečnostných dôvodov je tento endpoint v uložený v konfiguračnom súbore. Ukladanie do konfiguračného súboru umožňuje aj jednoduchú zmenu parametrov pokiaľ ich je potrebné meniť.

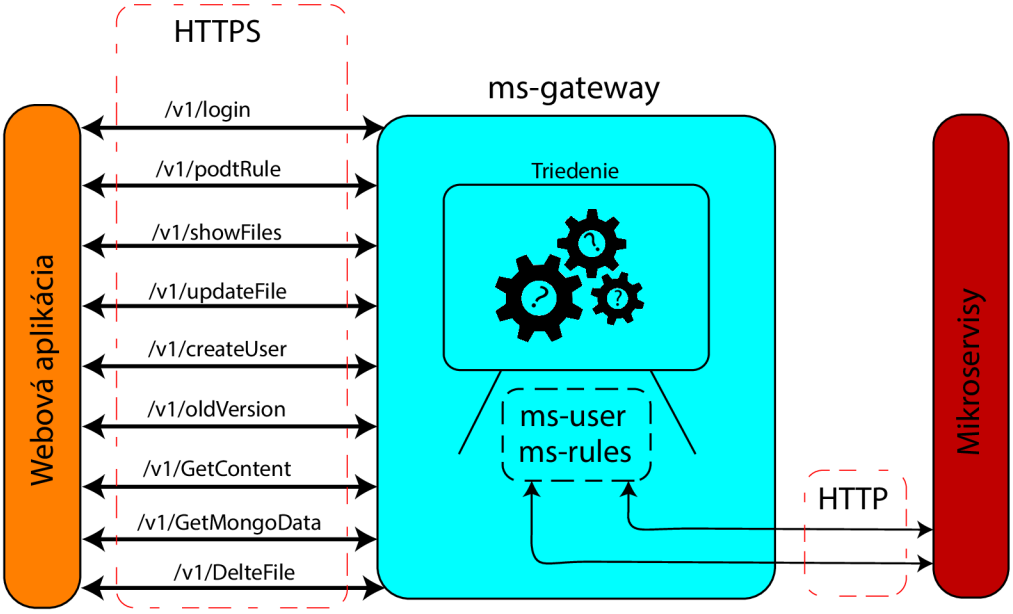
Ostatné funkcie, ktoré sa nachádzajú v `ms-gateway` majú rovnakú štruktúru ako funkcia z obrázku 5.6. Rozdiely sú len v definovaní URL a ich názve.

Podrobná ukážka toho ako funguje mikroservisa `ms-gateway` je možné si obhliadnuť na obrázku 5.7. Znázornená je komunikácia webovej aplikácie s bránou, ktorá ďalej komunikuje s ostatnými mikroservisami.

5.3.2 Vývoj komponenty `ms-user`

Mikroservisa je rozdelená do dvoch vlákien. Toto rozdelenie bolo potrebné z toho dôvodu, že je nutné, aby spustený kód vykonával dve funkcionality súčasne. Prvá je správa a odpovedanie na prichádzajúce požiadavky a druhá je obnovovanie GitLab prístupových tokenov užívateľov. Táto funkcia bude opísaná neskôr v tejto kapitole.

Prvá funkcia tejto mikroservisy je určená na tvorbu nových užívateľov v systéme. Po prijatí požiadavky z webovej platformy je zavolaná funkcia `create()`. Táto funkcia pomocou ďalších funkcií vytvára užívateľa v databázovom systéme PostgreSQL. Do databázy sú následne uložené tieto dáta:



Obr. 5.7: Podrobný nákres ms-gateway.

- **id** – vytvára sa automaticky pomocou databázovej funkcionality toto ID začína od 1 a postupne sa navyšuje tvorbou užívateľov.
- **public_id** – verejné ID je tvorené funkciou na backende, využíva sa pre bezpečnú komunikáciu medzi backendom a databázovým systémom. Je náhodne generované a nezačína sa od čísla 1 s postupným navyšovaním, čo znamená, že ho nie je ľahké uhádnuť a v prípade útoku má útočník veľmi malú šancu na získanie informácií z databázy pod daným ID. Takto náhodne generované ID môže vyzerať nasledovne `6d3d05ec-78d1-4fe4-a715-3fbb351e97ec`.
- **repository_id** – pri tvorbe nového užívateľa, je zavolaná funkcia, ktorá mu vytvára repozitár v GitLabe. V tomto repozitáry sa automaticky vytvoria preddefinované zložky v ktorých sa budú ukladať pravidlá. Názvy zložiek sú definované v konfiguračnom súbore. V každej z týchto zložiek je vytvorený inicializačný súbor bez ktorého by nebolo možné vytvoriť zložku nakoľko GitLab nedovoľuje vytvárať prázdne zložky. Po vytvorení týchto častí GitLab server vráti backendu ID daného repozitáru, ktoré sa tu uloží do databázy. Toto ID je využívané pri práci na pravidlách aby aplikácia vedela s akými pravidlami má pracovať.
- **gitlab_id** – toto ID patrí vytvorenému užívateľovi v GitLabe.
- **mongo_db_id** – užívateľovi je vytváraný aj súbor v MongoDB databázovom systéme, tu sa jeho ID ukladá a využíva sa pri v mikroservise `ms-rules` kde bude bližšie popísaná jeho využiteľnosť.
- **access_token** – je to token, ktorý sa využíva pri komunikácii s GitLab serve-

rom. Tento token je určitý druh zabezpečenia komunikácie s GitLabom a užívateľom, aby nebolo možné neoprávnene pristupovať k jeho dátam.
- **e-mail** – tu je uložený email užívateľa, ktorý si zvolil pri registrácií.
- **name** – užívateľ zadáva svoje meno.
- **surname** – tu sa nachádza užívateľove priezvisko, ktoré zadal pri registrácii.
- **nick** – je vybraná prezývka, ktorú chce užívateľ v systéme používať.
- **password** – zadané heslo užívateľom je pomocou knižnice `bcrypt` zhashované a je k nemu pridaná kryptografická soľ aby bola zvýšená bezpečnosť hashovania a heslo bolo odolné voči slovníkovým útokom.

Druhou z funkcií je už spomínané obnovovanie tokenov užívateľov. Toto obnovovanie funguje v druhom vlákne do, ktorých je mikroservisa rozdelená. Tieto tokeny sú dôležitou súčasťou bezpečnej komunikácie medzi užívateľom a GitLab serverom. Prístupový token môže vyzerať nasledovne `glpat-QY_G8Tu-qsys2KX7v7DZ`. Tento token sa vytvára na obmedzený čas, čiže po uplynutí doby jeho platnosť vyprší a už ho nie je možné použiť na komunikáciu. Práve z tohto dôvodu bolo potrebné vytvoriť funkciu, ktorá by tokeny pravidelne obnovovala.

Volaná funkcia v tomto vlákne sa volá `renewAllTokens()`. Jej úlohou je každý týždeň vytvoriť nové tokeny pre všetkých užívateľov. Tvorba tokenov prebieha v sobotu v skorých ranných hodinách. Tento čas bol vybratý z dôvodu žiadnej alebo minimálnej premávky na serveroch. Keďže táto funkcia zaberie určitý čas v závislosti na množstve užívateľov je pravdepodobné, že by bolo na nejaký čas obmedzené používanie aplikácie, práve preto je potrebné tokeny vytvárať v čase kedy sa predpokladá minimálne využívanie aplikácie.

Ako bolo spomenuté, tokeny sa obnovujú každý týždeň no ich platnosť je nastavená na dva týždne. Dvoj týždňová platnosť bola vybratá z toho dôvodu že v prípade poruchy a neobnovenia tokenov po týždni je v rezerve ešte 7 dní na opravu poruchy a vytvorenie nových platných tokenov.

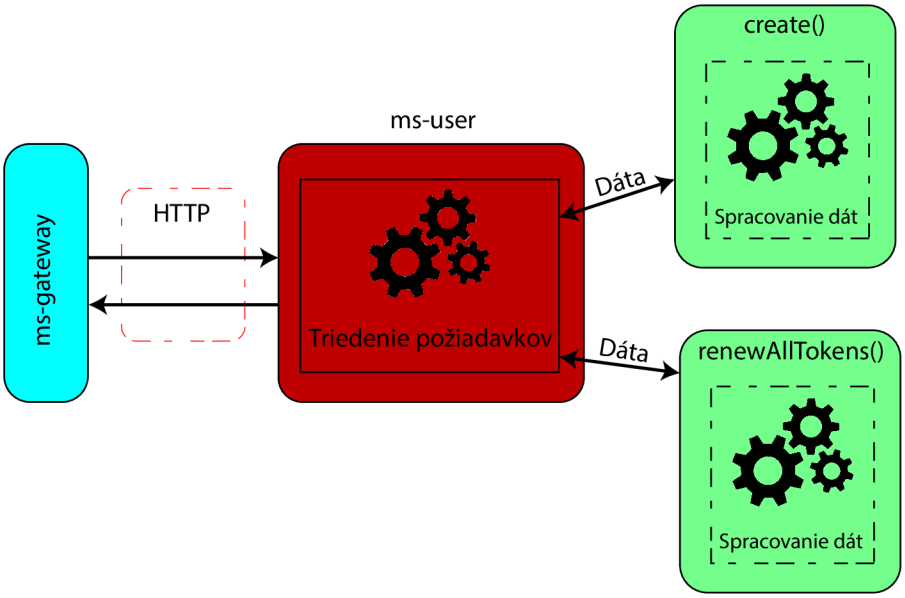
Obrázok 5.8 zobrazuje priebeh funkcionality vyvíjanej mikroservisy `ms-user`.

5.3.3 Vývoj komponenty `ms-rules`

V tejto mikroservise prebiehajú všetky úkony, ktoré sa týkajú vývoju korelačných pravidiel. Vykonáva sa tu ich tvorba, spravovanie, ukladanie triedenie a získavanie ich histórie. Budú tu popísané jednotlivé funkcie a ich význam a logika za ich fungovaním.

Nasledujúca časť práce bude rozdelená do menších častí podla názvov funkcii mikroservisy.

Všetky funkcie posielajú po vykonaní daných úkonov spätnú odpoveď webovej aplikácií, kde sa následne spracúvajú do výslednej podoby a sú zobrazené užívateľovi.



Obr. 5.8: Podrobný nákres `ms-user`.

Funkcionalite webovej aplikácie sa bude venovať kapitola 5.3.4.

SortRules – triedenie pravidiel

Prvá funkcia v mikroservise slúži k triedeniu prichádzajúcich pravidiel, ktoré si užívateľ vytvára. Pravidlá sú preposielané z webového rozhrania. Ukážku kódu je možné vidieť na obrázku 5.9.

Do premennej `data` sa ukladajú údaje, ktoré prichádzajú z požiadavky preposlanej cez `ms-gateway`. Následne je pomocou knižnice `jsno` táto požiadavka rozparsovaná do premenných `type`, `text`, `name`, `public_id`. štruktúru prijímaného požiadavku je možné vidieť na obrázku 5.10. Následne sú dáta z premenných na základe podmienok, ktoré sa nachádzajú vo switchi roztriedené. Ak sa názov v premennej `type` rovná podmienke, ktorá sa nachádza za slovom `case`, tak sa vykoná príslušná funkcia s názvom `pushFile`.

Funkcia s názvom `pushFile` po zavolaní ako prvé vytvára spojenie s MongoDB databázou. Následne sú zavolané funkcie `getAccessToken` a `getRepoId`, tieto funkcie sú štruktúrou veľmi podobné, obe získavajú dáta pomocou ORM z databázy PostgreSQL. Tieto dáta sú užívateľov prístupový token a ID jeho repozitáru, aby funkcia vedela do akého repozitáru má ukladať pravidlo. Informácie sa následné uložia do premenných `acc_token` a `project_id`. Tieto premenné budú neskôr použité na pripojenie a tvorbu súboru.

Po získaní týchto informácií je nadviazaný kontakt s GitLab serverom. K správnemu pripojeniu sa na server je potrebný práve prístupový token, ktorý bol získaný

```
data = request.data
json_data = json.loads(data)

type = json_data['type']
type = type.upper()
text = json_data['data']
name = json_data['name']
pub_id = json_data['public_id']

match type:
    case "RSA":
        stemp = pushFile(type, text, name, pub_id)
        return stemp

    case "QRADAR":
        stemp = pushFile(type, text, name, pub_id)
        return stemp

    case "SOLARWINDS":
        stemp = pushFile(type, text, name, pub_id)
        return stemp

    case _:
        stemp = pushFile(type, text, name, pub_id)
        return stemp
```

Obr. 5.9: Triedenie prichádzajúcich pravidiel.

vyššie. Na základe tohto tokenu server vie o akého užívateľa sa jedná. Nadviazanie spojenia prebieha pomocou funkcie na obrázku 5.11 riadok 1 a 2. V prvom riadku je možné vidieť funkciu pomocou, ktorej sa aplikácia pripája k serveru. V prvej časti je možné vidieť IP adresu serveru, za ktorou nasleduje užívateľov prístupový token. V druhom riadku je táto funkcia zavolaná a sú uložené dáta o užívateľovi do premennej `project`.

Tretí riadok zobrazuje tvorbu samotného pravidla. Do funkcie sa uvádza premenná `compered_type`, v ktorej je uložený typ pravidla a následne meno pravidla. Poslednou súčasťou je vloženie samotného textu pravidla, ktorý je uložený v premennej `text`. Po vložení dát do premenných je zavolaná funkcia `commit`, ktorá je súčasťou knižnice `python-gitlab`. Táto funkcia odošle dáta na GitLab server a uloží ich.

Po uložení súboru sa volá funkcia `getCommitHash`. Táto funkcia zistí aktuálny commit hash vytvoreného pravidla, ktorý uloží do premennej. Hash sa z premennej preposiela do funkcie z knižnice `pymongo`, ktorá tento hash spolu s dátumom a názvom pravidla uloží do MongoDB databázy. K hashu je tiež priradený `status` ku ktorému sa ukladá slovo `CREATED` alebo `UPDATED` na základe toho aký úkon bol na

```
{
 "type" :"solarwinds",
 "name": "Testovacie pravidlo",
  "public_id": "6d3d05ec-78d1-4fe4-a715-3fbb351e97ec"
 "data" : "SELECT UTF8(payload) as search_payload from
    events where (UTF8(payload) ilike
'%AccessDeniedException%' or UTF8(payload) ilike
%CsrfException%' or UTF8(payload) ilike
%InvalidCsrfTokenException%' or UTF8(payload) ilike
'%MissingCsrfTokenException%' or UTF8(payload) ilike
'%CookieTheftException%' or UTF8(payload) ilike
'%InvalidCookieException%' or UTF8(payload) ilike
'%RequestRejectedException%')"
}
```

Obr. 5.10: Štruktúra prijímaného požiadavku.

```
1. gl_user = gitlab.Gitlab(url = "http://10.50.64.5", private_token = f"{acc_token}",
keep_base_url=True)
2. project = gl_user.projects.get(project_id)
3. f = project.files.create({'file_path': f'{compared_type}'+'_RULES/'+f'{name}'+'.txt',
                             'branch': 'main',
                             'content': f'{text}',
                             'author_email': 'test@example.com',
                             'author_name': 'yourname',
                             'commit_message': 'Create testfile'})
```

Obr. 5.11: Vytvorenie spojenia a pridanie pravidla na server.

pravidle vykonaný. Uložený hash je možné vidieť na obrázku 5.12.

ShowFiles – zobrazenie súborov

V druhej funkcii mikroservisy sú spracované dáta z požiadavky a prepísané do funkcie s názvom `showMyFiles`. Táto funkcia slúži na zobrazenie súborov ktoré sa nachádzajú v repozitáry užívateľa. Rovnako ako pri predošlej funkcii prijaté dáta sú rozparsované a uložené do premenných. Príklad obsahu požiadavky je možné vidieť na obrázku 5.13.

Po rozparsovaní sú zavolané funkcie `getAccessToken` a `getRepoId`. Následne je vytvorené spojenie s GitLab serverom rovnako ako v obrázku 5.11 riadok 1. Ďalším krokom je získanie súborov z repozitáru, ktorého názov je uložený v premennej

Obr. 5.12: Ukážka uloženého hashu v MongoDB.

```
{
    "rule_dir": "QRADAR_RULES",
    "public_id": "6d3d05ec-78d1-4fe4-a715-3fbb351e97ec"
}
```

Obr. 5.13: Ukážka požiadavku pre zobrazenie pravidiel.

`rule_dir`. Na základe tohto názvu a funkcie `get` a `repository_tree`, ktoré pochádzajú z knižnice `python-gitlab`, sú získané jednotlivé súbory, ktoré sa spätne pošlú webovej aplikácii na ďalšie spracovanie. Funkciu je možné vidieť na obrázku 5.14.

```
project = gl_user.projects.get(project_id)
items = project.repository_tree(path=str(dir_name.upper()), get_all=True)
```

Obr. 5.14: Funkcia, ktorá získa súbory z GitLabu.

Update – aktualizácia

Rovnako ako pri predošlej funkcii, sa tu len spracujú prijaté dáta z požiadavku a prepošlú sa do ďalšej funkcie s názvom `updateFiles`. Prvým krokom funkcie je rozparsovanie prijatých dát. Dáta sú následne uložené do premenných `file_path`, ktorý obsahuje cestu k pravidlu, `new_content`, v ktorej je uložený nový obsah pravidla. `Message` správa obsahuje ľubovolnú informáciu odosielanú s aktualizáciou a `public_id`. Prichádzajúce dáta z požiadavky je možné vidieť na obrázku 5.15.

Po rozparsovaní sú znovu zavolané funkcie pre získanie prístupového tokenu, ID repozitáru užívateľa a funkcia, ktorá získa ID užívateľa v MongoDB. Táto funkcia je jednoduchý ORM príkaz, ktorý je preložený do SQL jazyku a vytiahne z databázy

```
{
    "file_path": "QRADAR_RULES/QRADAR_ff06f44c-beb0-11ed-a2b8-acde48001122.txt",
    "content": "Skusam WEB API GATEWAY FUNGUJE ? asi hej",
    "commit_message": "Test branch",
    "public_id:" "6d3d05ec-78d1-4fe4-a715-3fbb351e97ec"
}
```

Obr. 5.15: Ukážka dát pri aktualizácii pravidla.

potrebný údaj, ktorý vráti naspäť. Ďalším krokom funkcie je nadviazanie spojenia s GitLab serverom. Po nadviazaní spojenia je zavolaná funkcia `get` a `save`, pomocou týchto funkcií je nový obsah pravidla uložený na miesto pôvodného pravidla.

```
file = project.files.get(file_path=file_path, ref="main")
file.content = new_content
file.save(branch="main", commit_message=message)
```

Obr. 5.16: Aktualizácia pravidla.

Poslednou časťou tejto funkcie je pridanie hashu aktualizovaného pravidla do databázy MongoDB. Sem sa pridá nový hash spolu s dátumom vykonanej aktualizácie a slovom `UPDATED`, ktoré napovedá o vykonanej akcii.

OldVersion – získanie starej verzie

Samotný GitLab a jeho API nepodporujú možnosť vyhľadať staršie verzie súborov. Pre naplnenie požiadavky o získavaní starších verzii pravidiel bolo potrebné vymyslieť spôsob akým bude možné získať staršiu verziu pravidla.

Vytvorenie pravidla a každá jeho zmena produkuje `commit hash`. Tento hash je ukladaný do databázy MongoDB. Na základe hashu z tejto databázy sa vytvorí dočasná `brench` v GitLabe. Vďaka tomuto spôsobu je možné vytvoriť nový súbor, ktorý je vytvorený na základe hashu pravidla. GitLab dokáže pomocou tohto hashu nájsť konkrétnu verziu súboru a uložiť ju. Následne je tento obsah zo súboru získaný pomocou funkcie `get` a preposlaný do webovej aplikácie. Následne po vykonaní týchto úkonov je dočasná brench zmazaná z dôvodu, aby nezahlťovala pamäť v repozitáry.

Samotný postup funkcie je podobný ako pri ostatných. Je použitý parser, ktorý vytiahne z požiadavky potrebné dáta, ktoré sú uložené do premenných `name`, `commit-`

`_hash`, `file_path`, `public_id`. Následne je získaný prístupový token a ID repozitáru. Ukážku prichádzajúcej požiadavky je možné vidieť na obrázku 5.17.

```
{
    "name": "Test",
    "commit_hash": "e32a44ff6012d70534d2575098f200a9a32baa36",
    "file_path": "QRADAR_RULES/AlohaRule_8411a354-e5c0-11ed-a7bf-acde48001122.txt",
    "public_id": "6d3d05ec-78d1-4fe4-a715-3fbb351e97ec"
}
```

Obr. 5.17: Požiadavok pre tvorbu dočasnej branch.

FileContent – obsah pravidla

Získavanie obsahu pravidla je dôležitou súčasťou aplikácie. Užívateľovi umožňuje náhľad na to ako pravidlo vyzerá a v prípade potreby ho môže upraviť alebo zmazať pokiaľ ho už nepotrebuje.

Pri poslaní požiadavky sa zavolá funkcia `getCertainFile`. Tu sa prijatá požiadavka rozparsuje a potrebné údaje sa uložia do premenných. `file_folder` sem je uložený názov zložky, v ktorej sa daný súbor nachádza, `file_name` meno konkrétneho súboru, `public_id`. Znovu sa získa prístupový token a ID repozitáru, následne sa vytvorí spojenie s GitLab serverom. Získavanie obsahu súboru je pomerne jednoduché, sú k tomu využité dve funkcie `get` a `raw`. Funkcie postupne získajú umiestnenie súboru a následne z neho zoberú jeho obsah, ktorý je vrátený webovej aplikácii na ďalšie spracovanie. Ukážku týchto funkcií je možné vidieť na obrázku 5.18.

```
project = gl_user.projects.get(project_id)
raw_content =   project.files.raw(file_path=f'{file_folder.upper()}/{file_name}', ref='main')
```

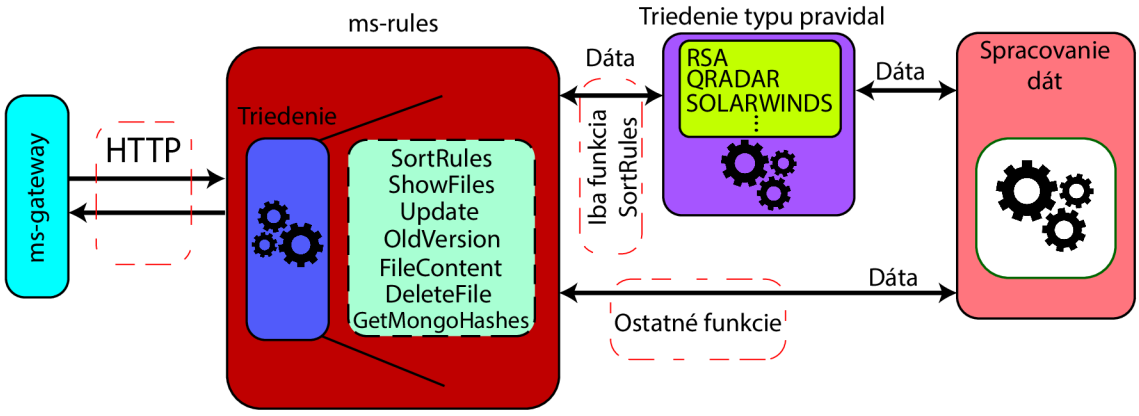
Obr. 5.18: Získanie obsahu súboru.

GetMongoHashes – získanie hashov z MongoDB databázy

V tejto funkcii je v požiadavke poslaný len jeden údaj, ktorý je `public_id`. Využije sa na získanie ID, ktoré patri užívateľovi a jeho súboru v databázovom systéme MongDB. Pomocou tohto ID sa z databázy získajú hashe z uložených pravidiel. Tieto hashe sú ďalej preposlané webovej aplikácií, ktorá ich následne spracuje.

DeleteFile – zmazanie súboru

Poslednou z funkcií tejto mikroservisy je zmazanie súboru. V požiadavku sa pošle `public_id` a `file_path`, tieto údaje sú po rozparsovaní uložené do premenných. K zmazaniu sa využije cesta k súboru z premennej a súbor sa vymaže pomocou funkcie `delete`. Obrázok 5.19 zobrazuje priebeh funkcionality vyššie popísanej mirkoserivsy a jej funkcií.



Obr. 5.19: Podrobný priebeh funkcionality `ms-rules`.

5.3.4 Implementácia webovej aplikácie

Vytvorená webová aplikácia má mierne odlišný dizajn ako bolo možné vidieť v jej návrhu na obrázku 4.5. Tieto rozdiely sú len v spôsobe zobrazenia komponenty číslo 1 z obrázku 4.5 a tlačítka `SHOW HISTORY` v sekcii 7 obrázok 4.5

Aplikácia na komunikáciu s backnedovým serverom využíva HTTPS požiadavky. Komunikácia prebieha z bezpečnostných dôvodov len medzi webovou aplikáciou a mikroservisou `ms-gateway`. V prípade útoku je možné sa dopátrať len k IP adrese a portu brány, samotné mikroservisy, ktoré spracúvajú dáta sú teda chránené.

Požiadavky sú z webovej aplikácie posielané pomocou technológie axios [3]. K využitiu tejto technológie je potrebné ju nainštalovať a následne je potrebné vytvoriť funkciu v jazyku TypeScript, ktorá bude spracovávať požiadavky. Príklad toho ako môže takáto funkcia vyzerať je na obrázku 5.21.

Funkcia, ktorá sa nachádza na obrázku 5.21 je určená pre získanie obsahu vybraného pravidla. Funkcia príjme parametre na základe výberu z komponenty 1 na obrázku 4.5. Tieto parametre sú následne uložené do premennej `rul`, kde sa vytvorí JSON súbor, ktorý bude odoslaný pomocou technológie axios. Poslednou časťou

[3] Viac o technológii dostupné z: `https://axios-http.com/docs/intro`

Obr. 5.20: Implementovaný vzhľad webovej aplikácie.

```
getRule(selectedRule: string, ruleName: string, public_id: string) {
    let rul = {
      file_folder: selectedRule + "_RULES",
      file_name: ruleName,
      public_id: public_id,
    };
    return http.post(`v1/GetContent`, rul);
  }
```

Obr. 5.21: Príklad funkcie pre poslanie požiadavku.

funkcie je poslanie požiadavky, definuje sa tu endpoint, na ktorý má byť požiadavka poslaná a obsah, ktorý má byť odoslaný. Ostatné funkcie požiadaviek majú rovnakú štruktúru, rozdiel je v dátach, ktoré sa do nich vkladajú a v inom JSON súbore ktorý je odosielaný.

5.4 Nasadenie aplikácie do experimentálneho prostredia

Aplikácia bola vyvíjaná a testovaná v lokálnom prostredí. V rámci sprístupnenia aplikácie rôznym užívateľom je potrebné aby bola premiestnená z lokálneho prostredia do toho reálneho. Toto premiestnenie bude obsahovať tvorbu Docker súborov, ktoré budú umiestnené na server Miri. Tento server bude určený pre riadenie backendu aplikácie. Frontendová časť bude uložená na server Mia. Celkový počet súborov pre mikroservisy bude 3. Jeden súbor pre každú z mikroservís. Následne budú na server premiestnené zdrojové kódy jednotlivých mikroservís. A pomocou Dockeru budú spustené. Nasledujúca časť bude popisovať tvorbu Docker súborov a ich obsah. Ukážka Docker súboru sa nachádza na obrázku 5.22.

Tvorba docker súboru pre backend

V súbore sa nachádza 6 hlavných príkazov, z ktorých každý má za úlohu vykonať pridelenú úlohu.

- **FROM** – táto inštrukcia určuje základný obraz podľa, ktorého bude vytvorený obraz pre aplikáciu. Konkrétny obraz ktorý sa používa je možné získať z webovej stránky DockerHub[4]. Docker vytvorí na základe inštrukcie kontajner ktorý bude obsahovať operačný systém Linux Alpine[5]. V tomto operačnom

[4]Viac informácií o DockerHube dostupné z:`https://hub.docker.com/`

[5]Viac informácií o operačnom systéme je možné získať z: `https://www.alpinelinux.org/about/`

```
FROM python:3.10.0-alpine

WORKDIR /app

COPY ./requirements.txt /app
COPY <názov_súboru>.py /app

RUN pip install -r requirements.txt
COPY . .

ENV FLASK_APP=<názov_súboru>.py

CMD ["flask", "run", "--host", "IP ADRESA SERVERU"]
```

Obr. 5.22: Docker súbor.

systéme bude spustená aplikácia. [98]

- **WORKDIR** – týmto príkazom je v docker kontajnery vytvorený pracovný priečinok s názvom `app`, do ktorého budú neskôr umiestnené potrebné súbory. [99]
- **COPY** – tento príkaz je určený ku kopírovaniu súborov a zložiek do vybraného priečinku. V súbore tento príkaz kopíruje súbor s názvom `requrements.txt` a `view.py` do pracovného priečinku `app`. [100]
- **RUN** – touto inštrukciou je spustený príkaz ktorý je za slovom `RUN` napísaný. Týmto spôsobom je možné spúšťať ľubovolné príkazy. V aplikácií sa spúšťa príkaz s funkciou `pip`, ktorá má za úlohu nainštalovať potrebné knižnice, ktoré sa nachádzajú v súbore `requrements.txt`. [101]
- **ENV** – pomocou tohto príkazu je možné nastaviť premenné prostredia v docker kontajnery. V aplikácii sa nastavuje premenná `FLASAK_APP`, ktorá kontajneru hovorí ako sa volá hlavný súbor, ktorý má kontajner spustiť. [102]
- **CMD** – inštrukciou je docker kontajneru povedané čo je jeho základný príkaz, ktorý má vykonať. V tomto súbore je príkaz využitý na zapnutie Flask aplikácie, ktorá počúva na zvolenej IP adrese. [103]

Týmto spôsobom budú pripravené všetky docker súbor. Ich rozdiel bude spočívať v inom porte pri IP adrese a názve, ktorým bude nahradená časť `názov_súboru`, ktorý je vidieť na obrázku 5.22. Aplikácia bude využívať porty v rozmedzí 5101-5150. Názvy hlavných súborov, v ktorých je spúšťaná Flask aplikácia sú:

- `view.py` – súbor obsahuje mikroservisu `ms-rules`.
- `gateway_view.py` – v súbore sa nachádza mikroservisa `ms-gateway`.
- `view_users.py` – tento súbor obsahuje mikroservisu `ms-user`.

6 Testovanie aplikácie

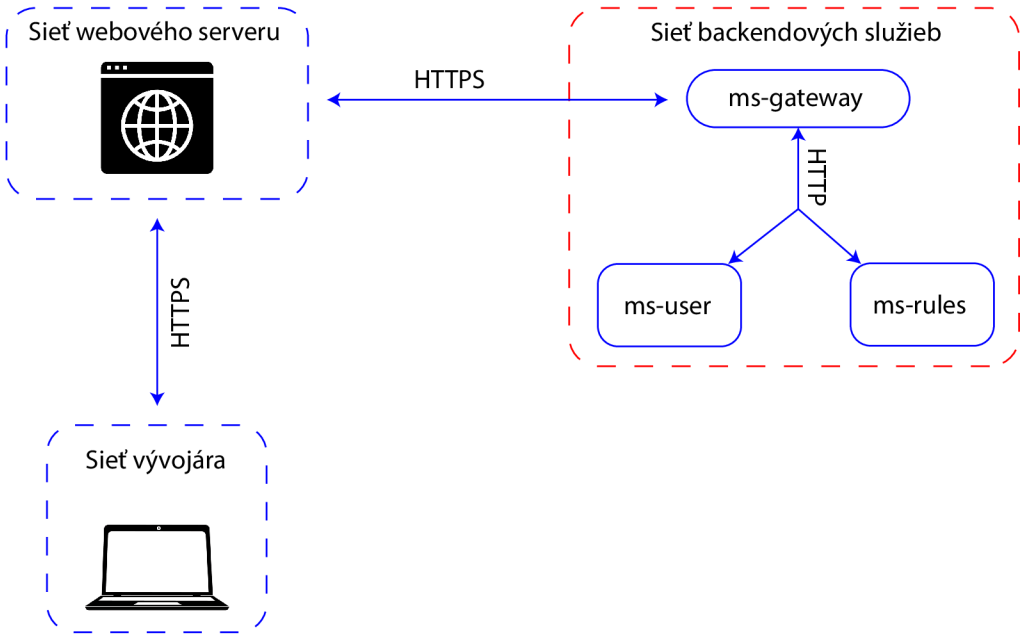
Posledná kapitola práce je venovaná testovaniu aplikácie.

Prvá úloha má zistiť funkčnosť aplikácie ktorej vývoj bol popísaný v kapitolách vyššie.

Druhou úlohou kapitoly je priblížiť čitateľovi reálnu funkcionalitu aplikácie. Kapitola 6.2, bude demonštrovať reálnu funkcionalitu aplikácie a ukáže jednotlivé možnosti jej používania. Jednotlivé pasáže budú obsahovať obrázky, na ktorých si čitateľ môže pozrieť vizuálne výsledky testovania webovej aplikácie.

Testovanie tejto aplikácie prebiehalo počas celého vývoja, táto kapitola bude popisovať len finálny test, ktorý slúži ako demonštrácia funkcionalít. Počas celej doby vývoju bol na testovanie používaný software Postman [1].

Obrázok 6.1 demonštruje možné spojenie medzi webovou aplikáciou a vývojárom, ktoré môže v reálnom nasadení nastať.



Obr. 6.1: Možné pripojenie sa k aplikácii.

6.1 Postup testovania

Každý obrázok je číselne označený pre jednoduchšiu orientáciu a pochopenie, čo sa odohráva v jednotlivých častiach obrázku. Obrázky obsahujú číselné značky, ktoré sú nižšie v texte popísané.

[1] Viac o tomto software dostupné z: `https://www.postman.com/`

Obrázok číslo 6.2 znázorňuje tvorbu pravidla vývojárom. Prvým krokom je výber kategórie pravidla. Výber sa uskutočňuje v okne pri čísle 1. Toto okno je rolovací element, ktorý sa po kliknutí zobrazí. Obsahuje zoznam SIEM systémov pre, ktoré môže vývojár tvoriť pravidlo. V okne pri čísle 2 vývojár napíše názov pravidla. Ďalšie okno sa nachádza pri čísle 3, do tohto textového poľa môže vývojár napísať text pravidla. Po tom ako je pravidlo dopísané, vývojár klikne na tlačítko pri čísle 4. Toto tlačítko pošle požiadavku smerom na backend, kde sa uloží pravidlo.

Obrázok číslo 6.3 demonštruje zobrazenie pravidla, ktoré sa nachádza v zozname vývojárových pravidiel. Prvým krokom je kliknutie na pravidlo, ktoré chce vývojár zobraziť, môže tak urobiť v zozname pri čísle 1. Po kliknutí sa odošle požiadavka, ktorá vráti obsah vybratého pravidla, tento obsah sa zobrazí v okne pri čísle 2.

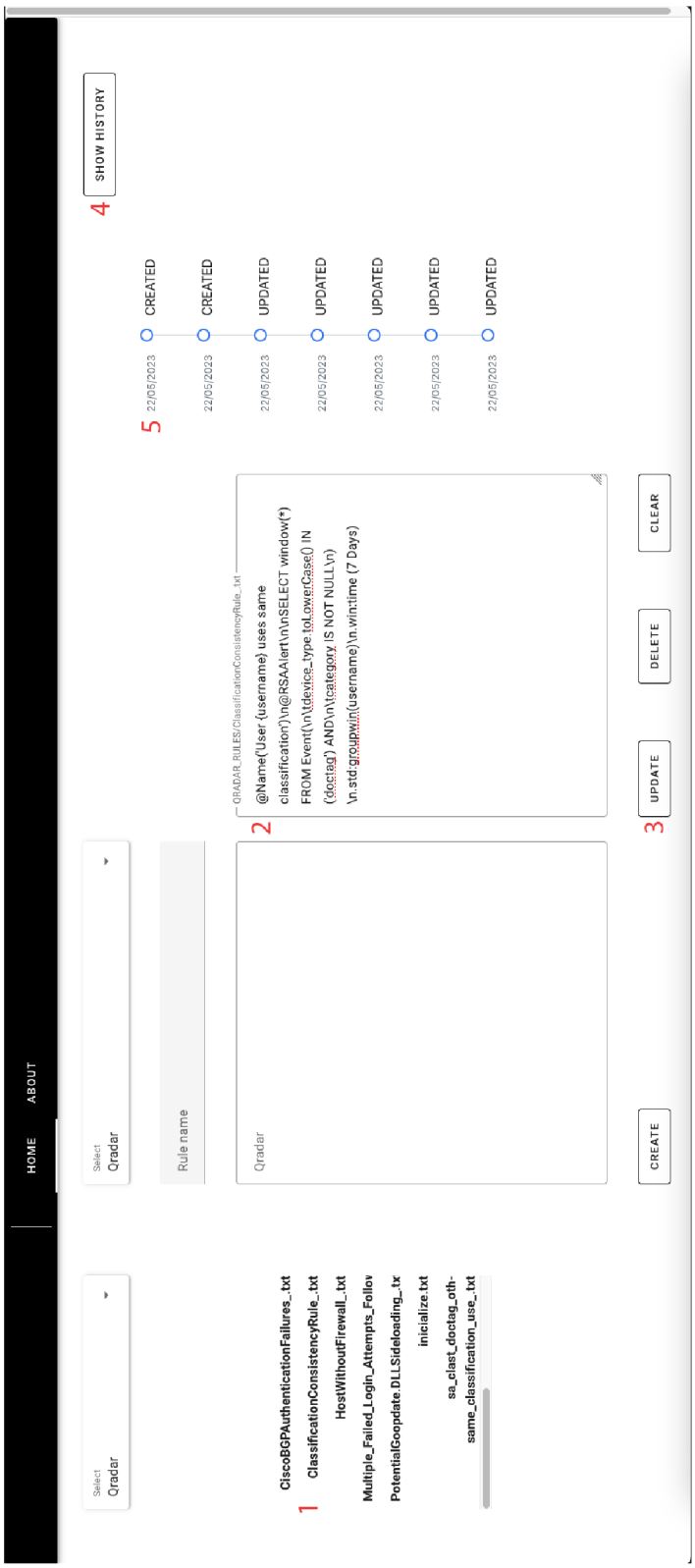
Obrázok pod číslom 6.4 zobrazuje možnosť aktualizácie zvoleného pravidla a zobrazenie histórie úprav. Vývojár si vyberie pravidlo, ktoré che upraviť, toto urobí v zozname pri čísle 1. Po výbere sa pravidlo zobrazí v textovom poli pri čísle 2. Tu môže vývojár uskutočniť potrebné zmeny na pravidle. Následne po ukončení úprav pravidla je potrebné stlačiť tlačítko pri čísle 3, týmto sa odošle požiadavka a vykonajú sa zmeny na pravidle, ktoré sa uložia. V prípade, že si che vývojár pozrieť históriu zmien tohto pravidla, tak klikne na tlačítko pri čísle 4. Toto tlačítko odošle požiadavku na server a vráti naspäť všetky úpravy, ktoré boli na pravidle vykonané. Následne sa pri čísle 5 zobrazí časová os, ktorá zobrazuje zmeny na pravidle. Zobrazuje sa tu dátum každej z úprav, ak si chce niektorú z úprav vývojár zobraziť, tak klikne na daný dátum. Po kliknutí sa odošle požiadavka, ktorá zobrazí v textovom poli pri čísle 2 obsah pravidla z vybraného dátumu.

Posledný obrázok 6.5 demonštruje mazanie pravidla. Vývojár si v zozname pri čísle 1 vyberie pravidlo, ktoré chce zmazať. Toto pravidlo sa následne zobrazí v okne číslo 2, kde sa môže vývojár uistiť, že sa jedná o dané pravidlo, ktoré chce zmazať. Ak si je vývojár istý, že pravidlo chce zmazať, tak klikne na tlačítko pri čísle 3, týmto je pravidlo trvalo zmazané.

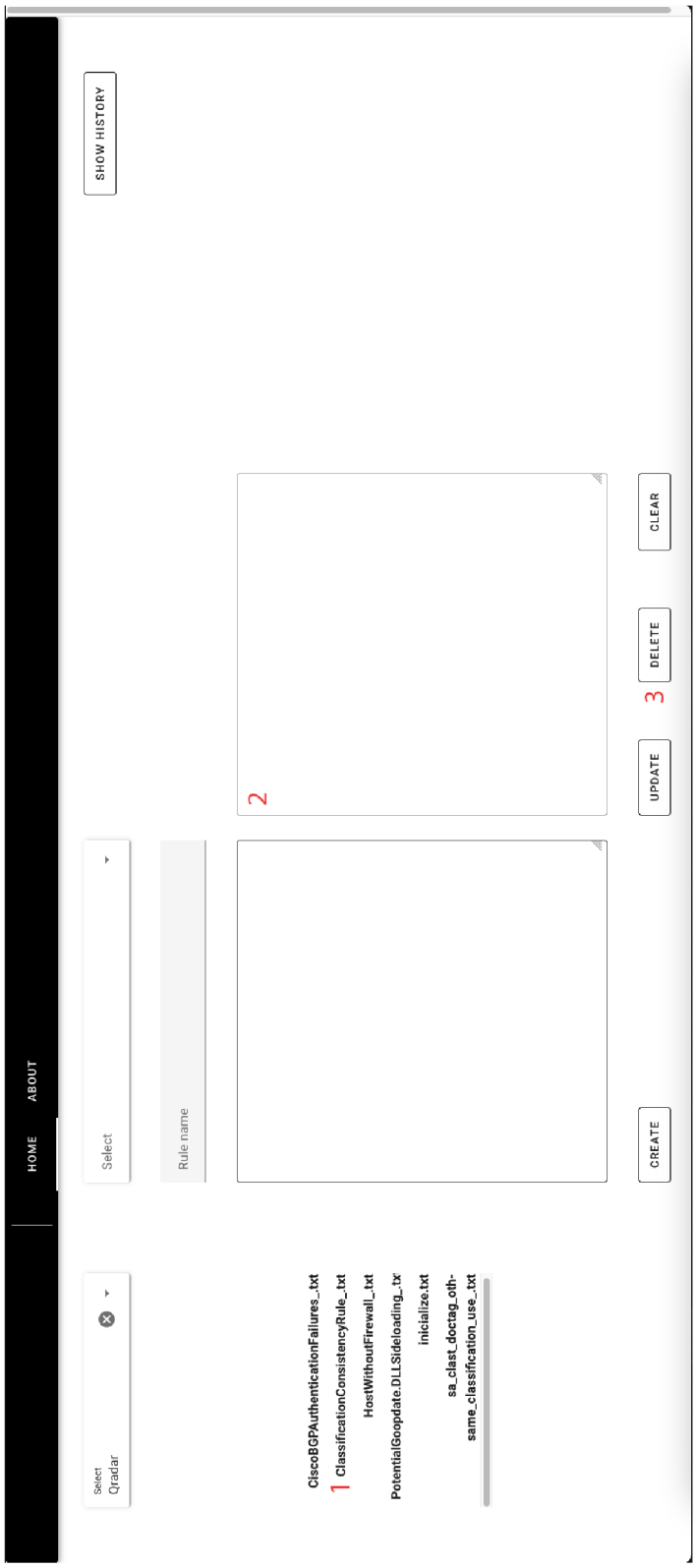
Na všetkých obrázkoch sa nachádza tlačítko s názvom `CLEAR`, toto tlačítko slúži na vyčistenie textového poľa, v ktorom sa zobrazujú pravidlá. Zabezpečuje to, aby vývojár pri prezeraní pravidiel omylom nenechal v textovom poli otvorené pravidlo, ktoré by mohol nechtiac upraviť. Po kliknutí na tlačítko `CLEAR` ostane textové pole prázdne.

Obr. 6.2: Demonštrácia tvorby pravidla vývojárom.

Obr. 6.3: Zobrazenie vybraného pravidla zo zoznamu.



Obr. 6.4: Aktualizácia pravidla a zobrazenie, histórie zmien.



Obr. 6.5: Zmazanie vybraného pravidla.

Obr. 6.6: Vývoj prvého pravidla.

Obr. 6.7: Vývoj druhého pravidla.

Obr. 6.8: Vývoj tretieho pravidla.

6.2 Záver testovania

Na základe informácií z kapitoly , je možné usúdiť že, aplikácia bola riadne otestovaná a predstavená čitateľovi. Kapitola jasne a stručne popísala všetky dôležité časti funkcionality aplikácie, s ktorými sa vývojár môže stretnúť. Ako bolo možné vidieť na obrázkoch v predošlej kapitole všetky jednotlivé funkcie sú plne funkčné a teda umožnia vývojárovi plynulý vývoj korelačných pravidiel. Obrázky 6.6, 6.7, 6.8 demonštrujú vývoj troch korelačných pravidiel.

Záver

V závere je možné povedať, že všetky základné požiadavky a ciele definované na začiatku bakalárskej práce boli úspešne splnené. Pomocou vypracovanej teoretickej časti, ktorej je venovaná prvá časť bakalárskej práce, boli následne vypracované praktické súčasti. Pri tvorbe praktickej časti aplikácie sa využili znalosti z kapitoly 3, kde boli vybraté vhodné technológie pre tvorbu samotnej aplikácie.

Cieľom praktickej časti bolo vytvorenie nástroja, ktorý by umožňoval vývojárom v bezpečnostných operačných centrách jednoduchý a stabilný vývoj korelačných pravidiel a ich kontrolu. Nástroj považujem za úspešne vytvorený. Ako bolo možné vidieť v kapitole 6.2, nástroj bol poriadne otestovaný a bola demonštrovaná jeho funkcionalita. Toto testovanie prebiehalo na tvorbe reálneho korelačného pravidla, ktoré by mohol vývojár v skutočnej situácii vyvíjať. Bol zobrazený každý prípad použitia aplikácie, ktorý by mohol nastať v rámci toho čo aplikácia umožňuje.

Teoretická časť bakalárskej práce podrobne vysvetľuje čitateľovi problematiku bezpečnostného monitoringu. Zároveň vysvetľuje základné pojmy a koncepcie pri ochrane sieťovej infraštruktúry. Následne sú v tejto časti popísané postupy pri vývoji korelačných pravidiel. Pre operátorov SOC a vývojárov sú tieto pravidlá veľkou výzvou. Predkladaná bakalárska práca sa snaží vysvetliť problematiku pri ich tvorbe a odôvodniť potrebu nového nástroja, ktorý by vývoj pravidiel uľahčoval. Obsahuje aj samotnú tvorbu potrebného nástroju.

Po oboznámení čitateľa s danou problematikou nasleduje v bakalárskej práci časť analýzy. Vysvetľuje pojmy potrebné pri porozumení jednotlivých technológií. Opisuje jednotlivé mechanizmy, ktoré je možné využiť pri tvorbe programu. Vyberá z nich tie najvhodnejšie, ktoré vyhovujú charakteru aplikácie a poskytujú potrebné funkcie.

Po analýze, ktorá sa nachádza v kapitole 3 je čitateľovi predložený návrh aplikácie. Popísané sú funkčné, nefunkčné a kritické požiadavky, ktoré boli pri návrhu zohľadnené.

Návrh v kapitole 4 podrobne opisuje možnú funkcionalitu aplikácie a jednotlivé prepojenia medzi jej časťami. Táto kapitola je rozdelená na dve časti, prvá časť sa venuje návrhu backendu, kde je navrhnutá logika spracovania dát. Druhou časťou je návrh frontendu, čo predstavuje webovú aplikáciu, pomocou ktorej bude užívateľ interagovať s backendom.

Po návrhu nasleduje implementačná časť v kapitole 5, ktorá čitateľovi prednesie ako bola aplikácia implementovaná, vysvetlí mu jednotlivé funkcie, ako sa v nich spracúvajú dáta a ako medzi sebou spolupracujú. Taktiež prednáša architektúru experimentálneho prostredia, v ktorom bola aplikácia testovaná.

Záverečnou časťou je kapitola 6 venovaná testovaniu, kde je aplikácia otestovaná

v experimentálnom prostredí na tvorbe reálnych korelačných pravidiel. V tejto kapitole je možné si prehliadnuť obrázky, ktoré testovanie podrobne opisujú. V tejto časti bol taktiež demonštrovaný vývoj 3 korelačných pravidiel.

Na záver je teda možné povedať, že všetky nároky kladené na aplikáciu v zadaní bakalárskej práce boli splnené. Bola vysvetlená problematika bezpečnostného monitoringu, určené funkčné, nefunkčné a kritické požiadavky. Aplikácia pre uľahčenie vývoja korelačných pravidiel bola úspešne vytvorená a bol podrobne popísaný postup jej vývoja. V závere bola demonštrovaná jej funkcionalita.

Literatúra

[1] JUREK, Michael a Lukáš MALINA. Prednáška Bezpečnosť ICT 2: *Útoky DDoS a testování bezpečnosti a výkonnosti sítě* [online]. Vysoké učení technické v Brně, 2022 [cit. 2022-12-11]. Dostupné z: interného systému.

[2] MAAYAN., Gilad David. What is SIEM and Why is it So Important?. *Dataversity.net* [online]. Los Angeles: Dataversity Digital, 2023, Október 1 2019 [cit. 2023-05-23]. Dostupné z: `https://www.dataversity.net/what-is-siem-and-why-is-it-so-important/`

[3] MARTINÁSEK, Zdeněk. Prednáška Bezpečnosť ICT 2: *Úvod do síťové bezpečnosti* [online]. Vysoké učení technické v Brně, 2022 [cit. 2022-12-12]. Dostupné z: interného systému.

[4] Malvér. *Eset.com* [online]. Bratislava: ESET, 2022 [cit. 2022-12-11]. Dostupné z: `https://www.eset.com/sk/malver/`.

[5] SUDAR, K. Muthamil, P. DEEPALAKSHIM, P. NAGARAJ a V. MUNEESWARAN. Analysis of Cyberattacks and its Detection Mechanisms. *Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)* [online]. Bengaluru: Christ University, 2020, 2020(5), 12-16 [cit. 2022-12-11]. Dostupné z: `doi:10.1109/ICRCICN50933.2020.9296178`.

[6] Botnet. *Eset.com* [online]. Bratislava: ESET, 2022 [cit. 2022-12-11]. Dostupné z: `https://help.eset.com/glossary/sk-SK/botnet.html`.

[7] GUTNIKOV, Alexander, Oleg KURPEEV a Yaroslav SHMELEV. DDoS attacks in Q2 2022. *Securelist.com* [online]. Moskva: AO Kaspersky Lab, 2022, 3 August 2022 [cit. 2022-12-11]. Dostupné z: `https://securelist.com/ddos-attacks-in-q2-2022/107025/`.

[8] ROBINSON, Seth. DDoS attacks on the rise: A closer look at the data. *Crayondata.com* [online]. Singapore: Crayon Data, 2020, 10 December 2020 [cit. 2022-12-11]. Dostupné z: `https://www.crayondata.com/ddos-attacks-on-the-rise-a-closer-look-at-the-data/`.

[9] BACON, Madelyn. Security. *Techtarget.com* [online]. Newton: TechTarget, 2022, Jún 2021 [cit. 2022-12-11]. Dostupné z: `https://www.techtarget.com/searchsecurity/definition/security`.

[10] What Is IT Security?. *Cisco.com* [online]. San José: Cisco Systems, 2022 [cit. 2022-12-11]. Dostupné z: https://www.cisco.com/c/en/us/products/security/what-is-it-security.html#~related-topics.

[11] What is network security?. *Vmware.com* [online]. Palo Alto: VMware, 2022 [cit. 2022-12-11]. Dostupné z: https://www.vmware.com/topics/glossary/content/network-security.html.

[12] Ids/ips. *Sectec.sk* [online]. Bratislava: SecTec, 2022 [cit. 2022-12-11]. Dostupné z: https://www.sectec.sk/bezpecnost/ids-ips.

[13] Sandbox. *Sectec.sk* [online]. Bratislava: SecTec, 2022 [cit. 2022-12-11]. Dostupné z: https://www.sectec.sk/bezpecnost/sandbox.

[14] What Is Network Traffic Analysis?. *Cisco.com* [online]. San José: Cisco Systems, 2022 [cit. 2022-12-11]. Dostupné z: https://www.cisco.com/c/en/us/products/security/what-is-network-traffic-analysis.html.

[15] What Is Network Detection and Response? *Cisco.com* [online]. San José: Cisco Systems, 2022 [cit. 2022-12-07]. Dostupné z: https://www.cisco.com/c/en/us/products/security/what-is-network-detection-response.html.

[16] What is a Proxy Server? How does it work?. *Fortinet.com* [online]. Sunnyvale: Fortinet, 2022 [cit. 2022-12-11]. Dostupné z: https://www.fortinet.com/resources/cyberglossary/proxy-server.

[17] Počítačový červ. *Avast.com* [online]. Praha: Avast, 2022 [cit. 2022-12-11]. Dostupné z: https://www.avast.com/cs-cz/c-computer-worm.

[18] Čo je škodlivý softvér? *Slps.sk* [online]. Bratislava: Slovenská sporiteľňa, 2023 [cit. 2023-05-20]. Dostupné z: https://www.slsp.sk/sk/ludia/otazky-a-odpovede/co-je-skodlivy-softver.

[19] What is endpoint security and how does it work?. *Kaspersky.com* [online]. Moskva: AO Kaspersky Lab, 2022 [cit. 2022-12-11]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/what-is-endpoint-security.

[20] What Is Endpoint Detection and Response (EDR)? *Trellix.com* [online]. Musarubra US, 2023 [cit. 2023-05-20]. Dostupné z: https://www.trellix.com/en-us/security-awareness/endpoint/what-is-endpoint-detection-and-response.html.

[21] What is Cloud Security?. *Kaspersky.com* [online]. Moskva: AO Kaspersky Lab, 2022 [cit. 2022-12-11]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/what-is-cloud-security.

[22] Čo je správa identít a prístupu (IAM)?: Čo je IAM a čo robí. *Microsoft.com* [online]. Redmond: Microsoft, 2023 [cit. 2023-05-20]. Dostupné z: https://www.microsoft.com/sk-sk/security/business/security-101/what-is-identity-access-management-iam.

[23] Co je SIEM?. *Microsoft.com* [online]. Redmond: Microsoft, 2022 [cit. 2022-12-07]. Dostupné z: https://www.microsoft.com/cs-cz/security/business/security-101/what-is-siem.

[24] Čo je agent zabezpečenia prístupu do cloudu (CASB)?: Definovanie agenta zabezpečenia prístupu do cloudu (CASB). *Microsoft.com* [online]. Redmond: Microsoft, 2023 [cit. 2023-05-20]. Dostupné z: https://www.microsoft.com/sk-sk/security/business/security-101/what-is-a-cloud-access-security-broker-casb.

[25] Data Loss Prevention. *Aec.cz* [online]. Brno: AEC, 2023 [cit. 2023-05-20]. Dostupné z: https://www.aec.cz/sk/dlp.

[26] What is Cloud Security?. Checkpoint.com [online]. Tel Aviv-Jafo: Check Point Software Technologies, 2022 [cit. 2022-12-11]. Dostupné z: https://www.checkpoint.com/cyber-hub/cloud-security/what-is-cloud-security/.

[27] Antivirus *Eset.com* [online]. Bratislava: ESET, 2022 [cit. 2022-05-18]. Dostupné z: https://www.eset.com/cz/antivirus-software/.

[28] JOHANSEN, Alison Grace. What is a firewall? Firewalls explained and why you need one: Firewall defined. *Norton.com* [online]. Sunnyvale: NortonLifeLock, 2023, 17 Jun 2021 [cit. 2023-05-20]. Dostupné z: https://us.norton.com/blog/emerging-threats/what-is-firewall#.

[29] HASAN, Mohammad. State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally. *Iot-analytics.com* [online]. Hamburg: IoT Analytics, 2022, 18 Máj 2022 [cit. 2022-12-11]. Dostupné z: https://iot-analytics.com/number-connected-iot-devices/.

[30] KOETSIER John. There Are Now 8.9 Million Mobile Apps, And China Is 40% Of Mobile App Spending. *Forbes.com* [online] 2020. New York: Forbes, 2022 [cit. 2022-11-20]. Dostupné z https://www.forbes.com/sites/johnkoetsier/2020/02/28/

there-are-now-89-million-mobile-apps-and-china-is-40-of-mobile-\
app-spending/?sh=447a186b21dd.

[31] OWASP Mobile Application Security. *Owasp.org* [online]. Maryland: OWASP Foundation, 2022 [cit. 2022-12-11]. Dostupné z: https://owasp.org/www-project-mobile-app-security/.

[32] Logging of security events in SIEM. *Logsign.com* [online]. İstanbul: Logsign, 2020, 10 Február 2020 [cit. 2022-12-11]. Dostupné z: https://www.logsign.com/blog/logging-of-security-events-in-siem/.

[33] FINLAYSON, R. a D. CHERITON. Log files: an extended file service exploiting write-once storage. *ACM SIGOPS Operating Systems Review* [online]. ACM SIGOPS, 1 November 1987, 21(5), 139-148 [cit. 2022-12-11]. Dostupné z: doi:https://doi.org/10.1145/37499.37516.

[34] Who is the OWASP®Foundation? *Owasp.org* [online]. Maryland: OWASP Foundation, 2022 [cit. 2022-12-11]. Dostupné z: https://owasp.org/.

[35] OWASP Top Ten. *Owasp.org* [online]. Maryland: OWASP Foundation, 2022 [cit. 2022-12-11]. Dostupné z: https://owasp.org/www-project-top-ten/.

[36] OWASP Top 10 Vulnerabilities: What Are the Top 10 OWASP Vulnerabilities?. *Snyk.io* [online]. Boston: Snyk Limited, 2023 [cit. 2023-05-20]. Dostupné z: https://snyk.io/learn/owasp-top-10-vulnerabilities/.

[37] Windows Security Log Event ID 4611: A trusted logon process has been registered with the Local Security Authority. *Ultimatewindowssecurity.com* [online]. Monterey: Monterey Technology Group, 2023 [cit. 2023-05-20]. Dostupné z: https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventid=4611.

[38] MARTINÁSEK, Zdeněk. Prednáška Bezpečnosť ICT 2: *Problematika logování, systémy IDS a IPS* [online]. Vysoké učení technické v Brně, 2022 [cit. 2022-12-12]. Dostupné z: interného systému.

[39] Correlation Rules. *Cybersecurity.att.com* [online]. Dallas: AT&T [cit. 2022-12-01]. Dostupné z: https://cybersecurity.att.com/documentation/usm-anywhere/user-guide/rules-management/correlation-rules.htm.

[40] SIEM. *Digitalnipevnost.cz* [online]. Digitální pevnost, 2018 [cit. 2022-12-11]. Dostupné z: https://www.digitalnipevnost.cz/viki/siem.

[41] PODZINS, O. a A. Romanovs. Why SIEM is Irreplaceable in a Secure IT Environment?. O*pen Conference of Electrical, Electronic and Information Sciences: eStream* [online]. Vilnius: IEEE, 2019, 2019, 1-5 [cit. 2022-12-11]. Dostupné z: doi:10.1109/eStream.2019.8732173.

[42] Appendix L: Events to Monitor. *Learn.microsoft.com* [online]. Redmond: Microsoft, 2022, 8 Jún 2022 [cit. 2022-12-11]. Dostupné z: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/appendix-l--events-to-monitor.

[43] STOLTZFUS, Justin. What's the difference between SEM, SIM and SIEM?. *Techopedia.com* [online]. Techopedia, 2022, 31 marec 2022 [cit. 2022-12-07]. Dostupné z: https://www.techopedia.com/7/31201/security/whats-the-difference-between-sem-sim-and-siem.

[44] Security Information Management. *Solarwinds.com* [online]. Tulsa: SolarWinds Worldwide, 2022 [cit. 2022-12-07]. Dostupné z: https://www.solarwinds.com/security-event-manager/use-cases/sim-security-information-management.

[45] Security Event Management. *Techopedia.com* [online]. Techopedia, 2022, 8 Apríl 2015 [cit. 2022-12-07]. Dostupné z: https://www.techopedia.com/definition/25763/security-event-management.

[46] Security Information and Event Management (SIEM). In: VIELBERTH, Manfred. *Encyclopedia of Cryptography, Security and Privacy* [online]. Springer: Springer, 2021, s. 1-3 [cit. 2022-12-07]. ISBN 978-3-642-27739-9. Dostupné z: https://doi.org/10.1007/978-3-642-27739-9_1681-1.

[47] SIEM Cost Breakdown and Tips. *Peerspot.com* [online]. New Yorku: PeerSpot, 2022, 11 September 2022 [cit. 2022-10-11]. Dostupné z: https://www.peerspot.com/articles/siem-cost-breakdown-and-tips.

[48] Datadog Cloud SIEM. *Datadoghq.com* [online]. New York: Datadog, 2022 [cit. 2022-12-07]. Dostupné z: https://www.datadoghq.com/product/cloud-security-management/cloud-siem/.

[49] Get to Know pfSense Plus. *Netgate.com* [online]. Austin: Rubicon Communications, 2022 [cit. 2022-12-07]. Dostupné z: https://www.netgate.com/pfsense-plus-software.

[50] SolarWinds Security Event Manager Features. *Solarwinds.com* [online]. Austin: SolarWinds Worldwide, 2022 [cit. 2022-12-07]. Dostupné z: https://www.solarwinds.com/security-event-manager/use-cases.

[51] Converged SIEM. *Logpoint.com* [online]. København: Logpoint, 2022 [cit. 2022-12-07]. Dostupné z: https://www.logpoint.com/en/product/comprehensive-threat-detection-and-response-with-converged-siem/.

[52] Graylog Security. *Graylog.org* [online]. Houston: Graylog, 2022 [cit. 2022-12-07]. Dostupné z: https://www.graylog.org/products/security/.

[53] Microsoft Sentinel. *Azure.microsoft.com* [online]. Redmond: Microsoft, 2022 [cit. 2022-12-07]. Dostupné z: https://azure.microsoft.com/en-us/products/microsoft-sentinel/#features.

[54] Log360, the one-stop solution for all you SIEM needs. *Manageengine.com* [online]. Chennai: Zoho, 2022 [cit. 2022-12-07]. Dostupné z: https://www.manageengine.com/log-management/integrated-log-management-and-siem-solution.html.

[55] Exabeam Fusion. *Exabeam.com* [online]. Foster City: Exabeam, 2022 [cit. 2022-12-07]. Dostupné z: https://www.exabeam.com/product/fusion/.

[56] Splunk Enterprise Security. *Splunk.com* [online]. San Francisco: Splunk, 2022 [cit. 2022-12-07]. Dostupné z: https://www.splunk.com/en_us/products/enterprise-security.html.

[57] Getting started with OSSEC. *Ossec.net* [online]. OSSEC Project Team, 2021 [cit. 2022-12-07]. Dostupné z:https://www.ossec.net/docs/docs/manual/non-technical-overview.html.

[58] McAfee Enterprise Security Manager: Security monitoring solutions for Managed Service Providers (MSPs). *Mcafee.com* [online]. Santa Clara: McAfee, 2017, Máj 2015 [cit. 2022-12-12]. Dostupné z: https://www.mcafee.com/enterprise/en-us/assets/solution-briefs/sb-enterprise-security-manager.pdf.

[59] SHAKEEL, Irfan. SIEM software & solutions. *Cybersecurity.att.com* [online]. Dallas: AT&T, 2022, 7 December 2022 [cit. 2022-12-07]. Dostupné z: https://cybersecurity.att.com/solutions/siem-platform-solutions.

[60] NetWitness®Cloud SIEM. *Netwitness.com* [online]. Bedford: RSA Security, 2022 [cit. 2022-12-07]. Dostupné z: https://www.netwitness.com/products/cloud-siem/.

[61] QRadar overview. *Ibm.com* [online]. Endicott: IBM Corporation, 2021, 2 September 2022 [cit. 2022-12-07]. Dostupné z: https://www.ibm.com/docs/en/qsip/7.4?topic=started-qradar-overview.

[62] Cyber Security: Why Prevention Is Better Than Recovery. *Riversafe.co.uk* [online]. Londýn: RiverSafe, 2023 [cit. 2023-05-23]. Dostupné z: https://riversafe.co.uk/tech-blog/cybersecurity-why-prevention-is-better-than-recovery/

[63] *MITTRE ATT&CK* [online]. Bedford: The MITRE Corporation, 2023 [cit. 2023-05-21]. Dostupné z: https://attack.mitre.org/.

[64] What is a Security Operations Center (SOC)?. *Checkpoint.com* [online]. Tel Aviv-Jafo: Check Point Software Technologies [cit. 2022-11-30]. Dostupné z: https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-soc/.

[65] OLSZEWSK, Alan. How do correlation strategies work within an SOC?. *Post.lu* [online]. Luxembursko: POST Group, 2023, 20 December 2020 [cit. 2023-05-21]. Dostupné z: https://www.post.lu/en/business/blog/articles/cybersecurite/soc/strategies-de-correlation.

[66] SOC Analysts: What They Are, What They Do + Salary. *Secureframe.com* [online]. San Francisco: Secureframe, 25 Október 2022 [cit. 2022-12-04]. Dostupné z: https://secureframe.com/blog/what-is-a-soc-analyst.

[67] VISSER, Jurgen. What's Your Approach to Building SIEM Use Cases?. *Correlatedsecurity.com* [online]. Corelated Security, 22 November 2014 [cit. 2022-11-28]. Dostupné z: http://correlatedsecurity.com/risk-driven-siem-use-case-development-methods-2/.

[68] SIEM Depths: Out-of-Box Correlation. Part 5. Methodology for developing correlation rules. *Sudonull.com* [online]. Berlin: Sudo Null [cit. 2022-11-29]. Dostupné z: https://sudonull.com/post/30656-SIEM-Depths-Out-of-Box-Correlation-Part-5-Methodology-for-\developing-correlation-rules.

[69] Co je vývoj softwaru?. *Cs.education-wiki.com* [online]. [cit. 2022-12-02]. Dostupné z: https://cs.education-wiki.com/9575463-what-is-software-development.

[70] Practices of Science: False Positives and False Negatives. *Manoa.hawaii.edu* [online]. Manoa: University of Hawaii, 2022 [cit. 2022-12-11]. Dostupné z: https://manoa.hawaii.edu/exploringourfluidearth/chemical/matter/properties-matter/practices-science-false-positives-and-false-negatives.

[71] TKACHENKO, Eugene. Creating Rules in IBM QRadar. *Socprime.com* [online]. Boston: SOC Prime, 2 November 2017 [cit. 2022-12-06]. Dostupné z: https://socprime.com/blog/creating-rules-in-ibm-qradar/.

[72] KAUFMAN, Efi. Side-by-Side SIEMs, Part 2: Forwarding QRadar to Splunk. In: *Linkedin.com* [online]. Mountain View: LinkedIn Corporation, 4 Marec 2021 [cit. 2022-12-06]. Dostupné z: https://www.linkedin.com/pulse/side-by-side-siems-part-2-forwarding-qradar-splunk-efi-kaufman/.

[73] *Alerting with ESA Correlation Rules User Guid: for RSA NetWitness®Platform 11.5* [online]. RSA Security, 2020 [cit. 2022-12-06]. Dostupné z: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwj14svS0-X7AhVLq6QKHTthAyYQFnoECA4QAQ&url=https%3A%2F%2Fcommunity.rsa.com%2Fyfcdo34327%2Fattachments%2Fyfcdo34327%2Fnetwitness-online-documentation%2F3191%2F1%2Frsa_nw_11.5_esa_alerting_user_guide.pdf&usg=AOvVaw1g1Ocz6TomotYW_8LC6Eji.

[74] Python 3.11.1 documentation. *Docs.python.org* [online]. Wilmington: Python Software Foundation, 2022, 11 December 2022 [cit. 2022-12-11]. Dostupné z: https://docs.python.org/3/.

[75] JAVIN, Paul. 5 Best Python Frameworks for Web Development in 2022. In: *Medium.com* [online]. 25 Január 2022 [cit. 2022-11-25]. Dostupné z: https://medium.com/javarevisited/top-5-python-frameworks-for-web-development-e034ebe85574.

[76] Use-case diagrams. *Ibm.com* [online]. Armonk: IBM Corporation, 2023, 3 Apríl 2021 [cit. 2023-05-22]. Dostupné z: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

[77] CHREN, Stanislav. Prednáška Softwarové inženýrství I: *Špecifikácia požiadaviek, Diagramy prípadov užitia* [online]. Masarykova Univerzita, 26 September 2012 [cit. 2023-05-22]. Dostupné z: https://is.muni.cz/el/1433/podzim2012/PB007/um/35424437/35424457/pb007-cvicenie-02.pdf

[78] SAFONOV, Yehor. *Aplikace pro grafickou reprezentaci průběhu útoku.* Brno, 2018, 95 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikač- ních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

[79] GRINBERG, Miguel. *Flask Web Development: Developing Web Applications with Python.* 2. Portland: O'Reilly Media, 2018. ISBN 978-1491991732.

[80] ABBA, Ihechikara Vincent. What is an ORM – The Meaning of Object Relational Mapping Database Tools. *Freecodecamp.org* [online]. San Francisco: Free Code Camp, 21 Október 2022 [cit. 2022-11-28]. Dostupné z: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database\-tools/.

[81] The Python SQL Toolkit and Object Relational Mapper. *Sqlalchemy.org* [online]. SQLAlchemy [cit. 2022-11-28]. Dostupné z: https://www.sqlalchemy.org/.

[82] RICHARDSON, Chris. What are microservices?. *Microservice Architecture* [online]. [cit. 2022-11-26]. Dostupné z: https://microservices.io/.

[83] HOSSEIN, Ashtari. What Is an API (Application Programming Interface)? Meaning, Working, Types, Protocols, and Examples. *Spiceworks.com* [online]. 12 August 2022 [cit. 2022-11-26]. Dostupné z: https://www.spiceworks.com/tech/devops/articles/application-programming-interface/.

[84] What is a REST API?. *Redhat.com* [online]. Raleigh: Red Hat, 2022, 8 Máj 2020 [cit. 2022-12-12]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[85] Introduction. *Vuejs.org* [online]. Sydney: Evan You, 2022 [cit. 2022-12-12]. Dostupné z: https://vuejs.org/guide/introduction.html.

[86] About. *Postgresql.org* [online]. Berkeley: The PostgreSQL Global Development Group, 2022 [cit. 2022-12-12]. Dostupné z: https://www.postgresql.org/about/.

[87] CHACEON, Scott a Ben STRAUB. *Pro Git: Everything you neeed to know about git* [online]. 2 vyd. apress, 2014 [cit. 2022-11-27]. ISBN 978-1484200773. Dostupné z: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control.

[88] CORTUK, Derya Cortuk. Version Control Software Comparison: Git, Mercurial,CVS, SVN. *Medium.com* [online]. 28 október 2019 [cit. 2022-11-27]. Dostupné z: https://medium.com/@derya.cortuk/version-control-software-comparison-git-mercurial-cvs-svn-\21b2a71226e4.

[89] Non-relational data and NoSQL. *Microsoft.com* [online]. Redmond, Washington: Microsoft 2023 [cit. 2023-05-20]. Dostupné z:

https://learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data.

[90] GILLIS, Alexander. MongoDB. *Techtarget.com* [online]. Newton: TechTarget, 2022, Jún 2021 [cit. 2023-05-20]. Dostupné z: https://www.techtarget.com/searchdatamanagement/definition/MongoDB.

[91] *Gitlab.com* [online]. San Francisco: GitLab B.V., 2023 [cit. 2023-05-23]. Dostupné z: https://about.gitlab.com/

[92] *Centos.org* [online]. Raleigh: The CentOS Project, 2023 [cit. 2023-05-23]. Dostupné z: https://www.centos.org/

[93] API Endpoints - What Are They? Why Do They Matter?: What is an API Endpoint?. *Smartbear.com* [online]. Greater Boston Area: SmartBear Software, 2022 [cit. 2022-12-09]. Dostupné z: https://smartbear.com/learn/performance-monitoring/api-endpoints/.

[94] Content Management. *Mongodb.com* [online]. New York City: MongoDB, Inc. 2023 [cit. 2023-05-19]. Dostupné z: https://www.mongodb.com/use-cases/content-management.

[95] MongoDB - Data Modelling. *Tutorialspoint.com* [online]. Hyderabad: Tutorials Point India Private Limited, 2023 [cit. 2023-05-23]. Dostupné z: https://www.tutorialspoint.com/mongodb/mongodb_data_modeling.htm

[96] M, Ishwarya. How to Install Docker PostgreSQL Environment? : 3 Easy Steps. *Hevodata.com* [online]. Bangalúr: Hevo Technologies India, 2023, 7 Február 2022 [cit. 2023-05-20]. Dostupné z: https://hevodata.com/learn/docker-postgresql/.

[97] DAS, Aveek. Getting started with PostgreSQL on Docker: Running the PostgreSQL image. *Sqlhacks.com* [online]. Newport Beach: Quest Software, 2023, 12 August 2022 [cit. 2023-05-20]. Dostupné z: https://www.sqlshack.com/getting-started-with-postgresql-on-docker/.

[98] Dockerfile reference: FROM. *Docker.com* [online]. Palo Alto: Docker, 2023 [cit. 2023-05-21]. Dostupné z: https://docs.docker.com/engine/reference/builder/.

[99] Dockerfile reference: WORKDIR. *Docker.com* [online]. Palo Alto: Docker, 2023 [cit. 2023-05-21]. Dostupné z: https://docs.docker.com/engine/reference/builder/.

[100] Dockerfile reference: COPY. *Docker.com* [online]. Palo Alto: Docker, 2023 [cit. 2023-05-21]. Dostupné z: https://docs.docker.com/engine/reference/builder/.

[101] Dockerfile reference: RUN. *Docker.com* [online]. Palo Alto: Docker, 2023 [cit. 2023-05-21]. Dostupné z: https://docs.docker.com/engine/reference/builder/.

[102] Dockerfile reference: ENV. *Docker.com* [online]. Palo Alto: Docker, 2023 [cit. 2023-05-21]. Dostupné z: https://docs.docker.com/engine/reference/builder/.

[103] Dockerfile reference: CMD. *Docker.com* [online]. Palo Alto: Docker, 2023 [cit. 2023-05-21]. Dostupné z: https://docs.docker.com/engine/reference/builder/.

[104] Install MongoDB Community Edition on Red Hat or CentOS. *Mongodb.com* [online]. New York: MongoDB, 2023 [cit. 2023-05-22]. Dostupné z: https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-red-hat/

[105] Containerize an application. *Docker.com* [online]. Palo Alto: Docker, 2023 [cit. 2023-05-23]. Dostupné z: https://docs.docker.com/get-started/02_our_app/

[106] How to Install and Configure GitLab CE on CentOS 7. *HowToForge.com* [online]. HowToForge [cit. 2023-05-19]. Dostupné z: https://www.howtoforge.com/tutorial/how-to-install-and-configure-gitlab-ce-on-centos-7/.

Zoznam symbolov a skratiek

API *Application Programming Interface* – Rozhranie pre programovanie

CASB *Cloud Access Security Broker* – Agent zabezpečenia prístupu do cloudu

CSS *Cascading Style Sheets* – Kaskádový štýl

DDoS *Distributed Denial of Service* – Distribuované odmietnutie služieb

DLP *Data Leakage Prevention* – Prevencia úniku údajov

DNS *Domain Name System* – Systém doménových mien

DoS *Denial of Service* – Odmietnutie služby

EDR *Endpoint Detection and Response* – Detekcia a odozva koncového bodu

FTP *File Transfer Protocol* – Protokol prenosu súborov

GRAPHQL *Graph Query Language* – Jazyk dotazu na graf

GRPC *Google Remote Procedure Call* – Vzdialené volanie procedúry Google

GUI *Graphic User Interface* – Grafické používateľské rozhranie

HTML *HyperText Markup Language* – Hypertextový značkovací jazyk

HTTP *Hypertext Transfer Protocol* – Hypertextový prenosový protokol

IAM *Identity and Access Management* – Správa identít

IDS *Intrusion Detection System* – Systém detekcie narušenia

IOS *iPhone Operating System* – Operačný systém iPhone

IoT *Internet of Things* – Internet vecí

IP *Internet Protocol* – Internetový protokol

IPS *Intrusion Prevention System* – Systém prevencie vniknutia

IT *Information Technology* – Informačné technológie

JSON *JavaScript Object Notation* – Zápis objektov JavaScript

JSON-EPC *JavaScript Object Notation Remote Rrocedure Call* – Vzdialené volanie procedúry so zápisom objektov JavaScript

NDR *Network Detection and Response* – Detekcia a odozva siete

NTA *Network Address Translation* – Preklad sieťových adries

ORM *Object Relational Mapping* – Objektovo relačné mapovanie

OWASP *Open Web Application Security Project* – Otvorený projekt zabezpečenia webových aplikácií

RCS *Revision Control System* – Systém kontroly revízií

REST *Representational State Transfer* – Prevod reprezentačného stavu

RHEL *Red Hat Enterprise Linux* – Red Hat podnikový Linux

SEM *Security Event Manage* – Správa bezpečnostných udalostí

SIEM *Security Information and Event Management* – Bezpečnostné informácie a správa udalostí

SIM *Security Information Management* – Správa bezpečnostných informácií

SOAP *Simple Object Access Protocol* – Jednoduchý protokol prístupu k objektu

SOC *Security Operations Center* – Bezpečnostné operačné centrum

SQL *Structured Query Language* – Štruktúrovaný dopytovací jazyk

SSH *Secure Shell* – Bezpečný shell

UC *Use Case* – Prípad Použitia

UI *User Interface* – Používateľské rozhranie

XTL *Microsoft Excel Template* – Šablóna programu Microsoft Excel

JWT *JSON Web Token* – Webový token JSON

Zoznam príloh

A Obsah priloženého média – zdrojové kódy vyvíjanej aplikácie

Toto médium obsahuje všetky zdrojové kódy a konfiguračné kódy ktoré boli potrebné k vytvoreniu aplikácie. Taktiež obsahom priloženého média sú aj súbory obsahujúce textové súbory s potrebnými knižnicami. Tieto knižnice je potrebné pred spúšťaním aplikácie stiahnuť.

A.1 Zdrojový kód – mikroservisa `ms-user`

Priložené médium obsahuje zdrojový kód a potrebné konfiguračné súbory mikroservisy `ms-user`. Funkcie mikroservisy sú bližšie popísané v kapitole 5.3.2.

```
/ ..... koreňový adresár priložených súborov
└── ms-user ..... zdrojové kódy mikroservisy ms-user
    ├── database ..... zdrojové kódy k tvorbe databázy
    │   ├── create_database.py ..... ORM tvorba databázy
    │   └── database_config-py ..... konfiguračný súbor databázy
    ├── config_user.py ..... konfiguračný súbor užívateľov
    ├── functions_user.py ..... súbor obsahujúci funkcie mikroservisy
    ├── view_user.py ..... hlavná časť kódu Flask API užívateľov
    ├── requirements.txt ..... balíčky potrebné k spusteniu kódu
    └── dockerfile ..... Súbor docker na tvorbu kontajneru mikroservisy
```

A.2 Zdrojový kód – mikroservisa `ms-gateway`

Priložené médium obsahuje zdrojový kód a potrebné konfiguračné súbory mikroservisy `ms-gateway`. Funkcie mikroservisy sú bližšie popísané v kapitole 5.3.1.

```
/ ..... koreňový adresár priložených súborov
└── ms-gateway ..... zdrojové kódy mikroservisy ms-gateway
    ├── database ..... zdrojové kódy k tvorbe databázy
    │   ├── create_database.py ..... ORM tvorba databázy
    │   └── database_config-py ..... konfiguračný súbor databázy
    ├── config_gateway.py ..... konfiguračný súbor brány
    ├── gateway_view.py ..... hlavná časť kódu Flask API brány
    ├── requirements.txt ..... balíčky potrebné k spusteniu kódu
    └── dockerfile ..... Súbor docker na tvorbu kontajneru mikroservisy
```

A.3 Zdrojový kód – mikroservisa `ms-rules`

Priložené médium obsahuje zdrojový kód a potrebné konfiguračné súbory mikroservisy `ms-rules`. Funkcie mikroservisy sú bližšie popísané v kapitole 5.3.3.

```
/ ..... koreňový adresár priložených súborov
└── ms-rules ..... zdrojové kódy mikroservisy ms-rules
    ├── database ..... zdrojové kódy k tvorbe databázy
    │   ├── create_database.py ..... ORM tvorba databázy
    │   └── database_config.py ..... konfiguračný súbor databázy
    ├── config.py ..... konfiguračný súbor pravidiel
    ├── view.py ..... hlavná časť kódu Flask API pravidiel
    ├── functions.py ..... súbor obsahujúci funkcie mikroservisy
    ├── requirements.txt ..... balíčky potrebné k spusteniu kódu
    └── dockerfile ..... Súbor docker na tvorbu kontajneru mikroservisy
```

A.4 Zdrojový kód – ORM tvorba databázy

Súbory, ktoré umožňujú tvorbu PostgreSQl databázy na servery.

```
/ ..... koreňový adresár priložených súborov
└── database ..... zdrojové ORM kódy pre tvorbu databázy
    ├── create_database.py ..... súbor obsahujúci tvorbu databázy
    └── database_config.py ..... konfiguračný súbor databázy
```

A.5 Zdrojový kód – tvorba databázy MongoDB

Súbory obsahujú všetky potrebné zdrojové kódy na vytvorenie funkčnej databázy na MongoDb servery.

```
/ ..... koreňový adresár priložených súborov
└── database ..... zdrojové kódy pre tvorbu databázy
    ├── create_mongoDB.py ..... súbor obsahujúci tvorbu databázy
    ├── mongo_config.py ..... konfiguračný súbor databázy MongoDB
    └── requirements.txt ..... potrebné balíčky pre funkčnosť kódu
```

A.6 Zdrojový kód – tvorba SIEM databázy

Súbory obsahujúce tvorbu konektoru na databázu SIEM.

```
/ ..... koreňový adresár priložených súborov
└── SIEM ..... zdrojové kódy pre connector SIEM databázy
    ├── connector.py ..... súbor obsahujúci connector databázy
    └── conn_config.py ..... konfiguračný súbor databázy MongoDB
```

A.7 Zdrojový kód webovej aplikácie

Súbory obsahujú všetky potrebné časti na tvorbu webovej aplikácie, ktorá je napojená na backendový systém.

```
/ ..... koreňový adresár priložených súborov
├── node_modules
├── scr ..... zdrojove komponenty aplikácie
│   ├── assets
│   ├── components ..... jednotlivé komponenty
│   │   ├── DropMenu.vue ..... zdrojový kód DropMenu
│   │   ├── NotFound.vue ..... zdrojový kód NotFound
│   │   ├── TextArea.vue ..... zdrojový kód TextArea
│   │   └── TimeLine.vue ..... zdrojový kód TimeLine
│   ├── interfaces
│   ├── plugins
│   ├── router
│   ├── services
│   │   ├── AxiosService.ts ..... zdrojový kód definovania http requestov
│   │   └── RuleService.tx ..... zdrojový kód jednotlivých http requestov
│   ├── stores
│   │   └── rule.ts ..... ukladanie dát naprieč aplikáciou
│   ├── styles ..... štýly aplikacie
│   ├── views
│   │   └── HomeView.vue ..... zobrazenie HomeView
│   ├── App.vue ..... hlavný spúštaný súbor aplikácie
│   ├── main.ts
│   └── shims-vuetify.d.ts
├── .eslintrc.cjs
├── Dockerfile
├── env.d.ts
├── index.html
├── cpackage-lock.json
├── package.json
├── tsconfig.config.json
├── tsconfig.json
└── vite.config.tx
```

B Návod spustenia aplikácie

K spusteniu aplikácie je potrebný nástroj Docker. V každej zložke v častiach A, A.1, A.2 sa nachádza `Dockerfile` pomocou, ktorého bude aplikácia spustená. V prvom kroku je potrebné vytvoriť kontajner, čo sa vykoná pomocou príkazu na riadku 1 vo výpise nižšie.

```
1. docker build -t alpine /cesta/k/súboru
2. docker run -d -p <port>:<port> --name <názov>
   alpine:latest
```

Na základe každého súboru je potrebné napísať jeho cestu. V každom spúšťanom kontajneri je potrebné nastaviť odlišné porty na, ktorých bude aplikácia počúvať. Časť príkazu s názvom `alpine`, je operačný systém na ktorom bude vo vnútri kontajneru spustená aplikácia.

ďalším krokom je tvorba databázových súborov ktoré sa nachádzajú v častiach A.4 a A.5. Prvým krokom je nastavenie konfiguračných súborov kde je potrebné zmeniť IP adresu, port a názov databázy na databázovom servery. Následne je potrebné použiť príkaz `python3 <názov súboru začínajúci na create_>.py>`. Týmto sú pripravené databázy a je možné sa presunúť na posledný krok. Ktorým je spúšťanie webovej aplikácie, ktoré je možné vidieť na nasledujúcom výpise. `Dockerfile` sa nachádza v časti A.7. Následne je potrebné vložiť do webového prehliadaču `localhost:<port>`.

Informácie boli čerpané z [105]

```
1. docker build -t /ceste/k/súboru .
2. docker run -d -p <port>:<port> --name <názov>
```

B.1 Alternatívne spustenie aplikácie

Aplikáciu je možné spustiť aj pomocou jazyku Python. Pomocou pythonu je potrebné spustiť nasledovné súbory.

- `view_user.py`
- `gateway_view.py`
- `view.py`
- `create_database.py`
- `create_mongoDB.py`

Tieto súbory je potrebné spustiť nasledovným príkazom:

```
python3 <názov_súboru>.py
```

Posledný krok je spustenie frontendovej aplikácie, je potrebné sa premiestniť do zložky z prílohy A.7. Spúšťa sa nasledovným príkazom.

```
npm run dev
```

C Návod inštalácie aplikácie GitLab

Tento návod jednoducho popíše ako nainštalovať GitLab na server CentOS 7. Pre správne fungovanie aplikácie je potrebné v prvom rade nainštalovať 4 nasledovné balíčky:

- `policycoreutils`;
- `openssh-server`;
- `openssh-clients`;
- `postfix`;

Postup inštalácie je uvedený v nasledujúcich príkazoch. Riadok číslo 1 popisuje inštaláciu. Na riadku 2 až 3 sa nachádza postup spustenia nainštalovaných aplikácií. Riadok 4 až 5 nastavuje aby sa tieto aplikácie naštartovali naraz s operačným systémom.

```
1. yum -y install curl policycoreutils openssh-server
   openssh-clients postfix
2. systemctl start sshd
3. systemctl start postfix
4. systemctl enable sshd
5. systemctl enable postfix
```

Ďalším krokom inštalácie je stiahnutie samotnej aplikácie GitLab. Postup je možné vidieť na nasledujúcom výpise riadok 1. Riadok číslo 2 popisuje inštaláciu aplikácie.

```
1. curl -sS https://packages.gitlab.com/install/
   repositories/gitlab/gitlab-ce/script.rpm.sh
   | sudo bash
2. yum -y install gitlab-ce
```

Tretím krokom je konfigurácia URL, túto konfiguráciu je možné vykonať v súbore s názvom `gitlab.rm`. Tento súbor sa nachádza v `/etc/gitlab/`. Je v ňom potrebné upraviť položku s názvom `external_url`. Informácie k tomuto návodu boli čerpané z [106].

D Link na zdielané úložisko

Na tomto linku sa nachádza video ukážka funkcionalít aplikácie.
https://drive.google.com/drive/folders/1sfQDKR281tuMNrCHDRw3TweTKswG3o_q?usp=share_link