VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO VIZUALIZACI TRENDŮ V ŠÍŘENÍ MALWARU

SYSTEM FOR MALWARE SPREADING VISUALIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DAŠA NOSKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. DOMINIKA REGÉCIOVÁ**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání bakalářské práce

Studentka: **Nosková Daša**
Program: Informační technologie
Název: **Systém pro vizualizaci trendů v šíření malwaru**
System for Malware Spreading Visualization
Kategorie: Bezpečnost

Zadání:

1. Studujte problematiku škodlivého kódu (malwaru), jeho typů a rodin. Seznamte se se systémy pro detekci malwaru ve společnosti Avast a metodami, jakými tyto systémy informují o nalezených detekcích.
2. Seznamte se s existujícími způsoby vizualizace malwarových detekcí a agregací výsledků za účelem sledování malwarových rodin. Analyzujte slabiny vizualizace a získejte od uživatelů těchto systémů zpětnou vazbu s ohledem na možná vylepšení.
3. Navrhněte novou platformu pro sledování, agregování, analyzování a vizualizaci malwarových detekcí. Adresujte nedostatky a požadavky z bodu 2. Zaměřte se na schopnost sledování detekcí z různých pohledů a metrik. Sledujte například verze detekcí, vztahy mezi nimi, trendy v čase atd. Generujte pravidelné reporty a umožněte uživatelům definovat si vlastní podmínky, za kterých budou upozorněny na potenciální anomálie.
4. Implementujte systém navržený v bodě 3. Vhodně použijte principy SOA (Architektury orientované na služby), DevOps, a TDD (Vývoj řízený testy).
5. Nasaďte vytvořený systém do praxe, sledujte jeho chování na reálných datech.
6. Svoji práci zhodnoťte, diskutujte případné problémy a možný budoucí vývoj.

Literatura:

- P. Szor: The Art of Computer Virus Research and Defense, Addison-Wesley Professional (2005), ISBN 978-0321304544
- Recorded Future: The Threat Intelligence Handbook, CyberEdge Group (2018), ISBN 978-0999035467
- B. Ellis: Real-Time Analytics: Techniques to Analyze and Visualize Streaming Data, Wiley (2014), ISBN 978-1118837917
- Interní dokumentace společnosti Avast
- Dále dle doporučení vedoucího či konzultanta

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů a rozpracování bodu čtvrtého.

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

Vedoucí práce: **Regéciová Dominika, Ing.**
Konzultant: Matula Peter, Ing., Avast
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 14. října 2021

Abstrakt

Táto bakalárska práca rieši návrh a tvorbu služby pre spracovávanie, ukladanie a vizualizáciu časových dát, ktorá informuje o aktuálnej malwarovej hrozbe. Časové dáta sú ukladané do time-series databázy a vizualizované na vytvorenej webovej stránke vo forme grafov a tabuliek. V práci sú preskúmané jednoduché štatistické metódy pre detekciu extrémov na reálne nazbieraných dátach. Je vytvorená služba poskytujúca, okrem prehľadu informácií, aj možnosť upozornení na užívateľom definované anomálne správanie. Služba poskytuje analytikom vizuálny pohľad na šírenie a detekciu malwaru a spôsob pre detekciu anomálií.

Abstract

This bachelor's thesis deals with the design and implementation of a service for processing, storing, and visualization of time-series data, which informs about actual malware threat. Time-series data are stored in a time-series database, and visualized in the form of charts and tables on a created website. The work examines simple statistical methods for detecting extremes on real system data. In addition to an overview of information, the created service provides the possibility to define alerts for the user set anomalous behavior of malware. The service provides analysts with a visual view of malware spreading and its detection, and provides a way to detect anomalies.

Kľúčové slová

yara, škodlivý kód, šírenie malwaru, vizualizácia trendov, časové dáta, time-series databáza, detekcia anomálií

Keywords

yara, malware, spreading of malware, visualization of trends, time-series data, time-series database, anomaly detection

Citácia

NOSKOVÁ, Daša. *Systém pro vizualizaci trendů v šíření malwaru*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Dominika Regéciová

Systém pro vizualizaci trendů v šíření malwaru

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením Ing. Dominiky Regéciovej. Ďalšie informácie mi poskytli Ing. Marek Milkovič a Ing. Peter Matula. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Daša Nosková
9. mája 2022

Poďakovanie

Rada by som poďakovala mojej vedúcej práce Ing. Dominike Regéciovej za jej rýchle reakcie a prínosné komentáre. Zároveň sa chcem poďakovať aj Ing. Marekovi Milkovičovi za jeho trpezlivosť, usmerňovanie a odbornú pomoc. Na koniec by som sa rada poďakovala môjmu priateľovi, kamarátom a rodine za ich podporu počas písania tejto práce.

Obsah

Kapitola 1

Úvod

Čím viac sa náš život prenáša do digitálneho sveta a internetový obsah sa zväčšuje, tým viac naberajú na intenzite aj kybernetické útoky, vyskytujúce sa v rôznych podobách a formách. Kybernetické útoky môžu spôsobiť stratu firemných dát, nefunkčnosť počítačov a serverov alebo narušiť ochranu osobných údajov. S rastúcou digitalizáciou sa zvyšuje záujem o tvorbu škodlivého softwaru (ang. *malware, malicous software*) pre cielené vydieranie a osobný zisk. Škodlivé programy nezmiznú, kým neprestanú byť vytvárané, keďže neexistuje program, ktorý by vedel predpovedať, ako bude vyzerať a správať sa program vytvorený v budúcnosti, keďže tvorba programu je kreatívny proces závislý od človeka [26].

Dôležitú úlohu v boji proti škodlivému kódu a zvyšovaní informačnej bezpečnosti zohrávajú antivírusové spoločnosti, vyvýjajúce antivírusové programy, ktorých hlavným cieľom je poskytnúť čo najvyššiu ochranu proti škodlivým programom, poškodeniu a strate dát či úniku osobných informácií. Úlohou antivírusových spoločností je okrem detekcie a odstránenia škodlivých programov aj zber a analýza relevantných informácií, napríklad ohľadom vývoja, šírenia a správania sa škodlivých programov. Na zmeny v správaní malwaru treba reagovať okamžite. Takéto zmeny sú pokladané za anomálie, na ktoré treba upozorniť odborníkov.

Cieľom práce je vytvoriť informačný systém pre firmu Avast s webovým užívateľským rozhraním. Systém má za úlohu poskytovať analytikom prehľad o šírení škodlivého softwaru, pochopiť, ako sa škodlivé softwary vyvíjajú, a zároveň upozorňovať na zmeny v správaní jednotlivých skupín škodlivého softwaru.

Práca začína kapitolou 2, ktorá je úvodom do témy škodlivého kódu a jeho detekcie. V podkapitolách sa čitateľ zoznámi s metódami a nástrojmi využívanými pri detekcii kódu. V sekcii nájde 2.5 odpoveď na otázku, prečo je v boji proti škodlivému kódu dôležitá vizualizácia a v kapitole 3 sa nachádza úvod k detekcii anomálií. Práca pokračuje kapitolou 4, obsahujúcou krátky úvod do problematiky práce s dátami závislými od času (ang. *time-series data*) a potreby využívania nových databáz. V kapitole 5 je čitateľovi priblížená koncepcia služby, v kapitole 6 opísaná jeho programová realizácia. Na konci práce čitateľ nájde informácie o rýchlosti systému a užívateľskej spätnej väzbe.

Kapitola 2

Šírenie a detekcia malwaru

Malware, po slovensky škodlivý software, je typ softwaru vytvorený so zámerom poškodenia počítačových systémov alebo získať citlivé dáta zo zariadení alebo sietí o užívateľoch, spoločnostiach či štátnej správy. Dôvodov tvorby malwaru je mnoho, väčšinou sa dôvody zhodujú na zlomyseľných úmysloch páchateľa, ktorého cieľom je vydieranie alebo priamy zisk informácií, a tým poškodenie napadnutej osoby, inštitúcie alebo spoločnosti. Malware sa šíri automaticky bez zásahu užívateľa a kopíruje sa na ďalšie zariadenia. Malware môže napadnúť počítačové zariadenie skrz iné napadnuté infekčné aplikácie, reklamy, e-mail, v dnešnej dobe aj cez komunikačné aplikácie, napr. cez službu Messenger, alebo sa do počítačového zariadenia môže dostať pri inštalovaní iného obsahu. Malware môže byť súčasťou napr. stiahnutého ilegálneho obsahu, v ktorom sa tvári ako potrebná aplikácia, a preto je dôležité sťahovať a inštalovať obsah iba z overených a bezpečných zdrojov. Malware je všeobecný pojem na označenie skupiny softwaru s cieľom hocijakým spôsobom škodiť. Všeobecný pojem sa ďalej delí na rôzne druhy malwaru na základe jeho funkcionality a zámeru. Konkrétny malware však môže spadať do viacerých kategórií, a teda plniť viaceré úlohy. Nižšie sa nachádza zoznam najznámejších malwarových typov, avšak ich zoznam je oveľa dlhší [26].

- Vírus
- Červ
- Trójske kone
- Rootkits
- Ransomware
- Spyware
- Adware
- Phishing
- Flooders (DDoS útoky)

Každý počítačový program je jedinečný a pri škodlivom programe, to nie je inak, preto sa malwarové typy ďalej delia do malwarových skupín, tzv. malwarových rodín (ang. *strains, families*) na základe spoločného jadra kódu. Každá rodina má svoj názov, ale názvy malwarových rodín nie sú vo všeobecnosti žiadnym spôsobom štandarizované. Mená rodín sú vymýšľané analytikmi v antivírusových firmách, čo spôsobuje problémy, ako napríklad používanie rôznych mien pre rovnakú rodinu alebo nastáva opačný prípad, kedy je jedno meno používané pre rôzne rodiny [13].

Dnešné systémy pre detekciu škodlivého kódu sú založené na znalosti statickej časti kódu alebo na znalosti o správaní malwaru. Systémy deklarujú software za škodlivý na základe porovnania s databázou spomenutých vzorcov tzv. signatúr [22].
Na detekciu a analýzu malwaru sa využívajú metódy opísané v nasledujúcich podkapitolách.

2.1 Statická analýza

Statická analýza je proces skúmania škodlivého kódu bez nutnosti jeho spustenia. Hlavný bod skúmania tvorí zdrojový kód. V prípade, ak zdrojový kód nie je k dispozícii, skúma sa binárny kód cez metódu reverzného inžinierstva [18]. Jedna zo základných metód reverzného inžinierstva spočíva v hľadaní vzorov (ang. *pattern matching*) [26], ktoré prebieha automaticky pomocou nástrojov, ako napr. YARA. Medzi hľadané vzorce patria časti textových alebo bytových reťazcov, mnohokrát sa objavujúcich v škodlivých súboroch.

Statickou analýzou je možné preskúmať hlavičkové súbory a časti kódu, a tým získať informácie, napríklad o štruktúre škodlivého programu, používaných knižniciach a funkciách, alebo je možné odhaliť chybové správy, IP adresy, alebo relevantné komentáre cez skúmanie reťazcov. Nevýhodou statickej analýzy je, že tvorcovia malwaru využívajú *obfuskáciu* zdrojového kódu, čo znamená, že zdrojový kód skrývajú, šifrujú a snažia sa analýzu malwaru sťažiť mnohými metódami, ktorými sa stáva zdrojový kód nejasným [22]. Statická analýza je rýchly spôsob detekcie škodlivého programu, ak sú škodlivé sekvencie kódu už známe, avšak pri skúmaní nových programov môže byť odhaľovanie škodlivých sekvencií časovo náročná úloha [18].

2.2 Dynamická analýza

Dynamická analýza sa zameriava na skúmanie správania malwaru. Správanie malwaru je možné sledovať v špeciálnom prostredí, sandboxe alebo emulátore. Sandbox je zabezpečené virtuálne prostredie s vlastným operačným systémom, oddelené od zvyšku počítačového zariadenia, v ktorom je možné analyzovaný program bezpečne spustiť a ďalej analyzovať [12].

Emulátor plní podobnú funkciu ako sandbox. Ide o software napodobňujúci funkcionalitu a dizajn iného zariadenia. Pomocou emulátoru môžu byť spustené a analyzované aj programy navrhnuté pre iný typ počítačovej architektúry alebo iný typ systému. Na rozdiel od sandboxu, ktorý simuluje kompletné virtuálne prostredie, v ktorom je vzorka skúmaná, emulátor simuluje iba spustenie konkrétnej vzorky a iba tie časti počítačového prostredia, ktoré sú relevantné k spusteniu skúmaného súboru, napr. heslá, systémové registre alebo antivírusové programy. Medzi úlohy, ktoré emulátor vykonáva, patrí napr. skúmanie správania podozrivej vzorky, rozbaľovanie vzoriek šifrovaných alebo kompresovaných neznámym algoritmom, alebo skúma shell kód v dokumentoch. Emulátory je pomerne jednoduché oklamať, pretože je nemožné, aby boli implementované všetky inštrukcie poskytované skutočnými CPU, popr. OS. Z tohto dôvodu musia byť emulátory pravidelne aktualizované [1], [18].

Skúmanému programu sú vo virtuálnom prostredí poskytnuté plné administratívne práva a sleduje sa jeho nasledujúce správanie, napr. cez debugger. Objektom skúmania dynamickej analýzy sú programom vykonané zmeny v registri, zmeny súborov a spôsob týchto zmien [17].

Uprednostňované sú behaviorálne pravidlá, teda pravidlá založené na chovaní, pretože sú neraz viac spoľahlivé ako statické časti kódu v súboroch, ktoré môžu byť veľa ráz zašifrované. Na druhej strane dynamická analýza nie je v dnešnej dobe vždy jednoduchý proces, keďže mnoho škodlivých programov využíva metódy k zamedzeniu analýzy jeho kódu spôsobom, že dokážu rozpoznať, keď sa nachádzajú v kontrolovanom prostredí a nespustia sa, kým nie sú splnené definované podmienky [16].

2.3 YARA

YARA[1] je open-source nástroj, vyvinutý firmou VirusTotal, využívaný na identifikáciu a klasifikáciu malwaru na základe charakteristických znakov pre malwarovú rodinu. YARA je nezávislá od druhu súboru, takže poskytuje univerzálne pokrytie pre všetky typy formátov, v ktorých môže rozpoznávať škodlivé znaky kódu. YARA poskytuje možnosť rozšírenia svojej funkcionality skrz moduly. Medzi takéto rozšírenia patria možnosti definície vlastných dátových štruktúr a funkcií, ktoré prispievajú k písaniu cielenejších pravidiel a následne k presnejšej a viac vyjadrujúcej analýze súborov. YARA je nástroj v podobe deklaratívneho jazyka, využívaný na vyhľadávanie vzorov. Na vyhľadávanie vzorov v súboroch využíva celky nazývané pravidlá (ang. *rule*), ktoré sú tvorené z bytových sekvencií a booleanovských podmienok. YARA pracuje s bytovými sekvenciami, ktoré môžu byť zapísané vo forme textových reťazcov, reťazcov v hexadecimálnej sústave alebo v tvare regulárnych výrazov. Tieto definované sekvencie vzorov plnia funkciu premenných, ktoré sú použité pri definícií booleanovských podmienok. Keď súbor obsahuje hľadané reťazce alebo obsahuje definované charakteristické črty, podmienka definovaná v pravidle sa vyhodnotí ako pravdivá a vytvorí sa tzv. hit. YARA, ktorá podporuje hľadanie charakteristických znakov malwaru v skúmanom súbore na základe viacerých charakteristík. Hľadanie znakov prebieha, buď na základe meta informácií daného súboru, alebo podľa jeho statických vlastností, napríklad na základe obsahu už známych sekvencií znakov, alebo naopak, podľa jeho dynamického prejavu správania [5].

Keďže YARA pravidlá sú nezávislé časti kódu a pravidlo samotné je platnou časťou, tak sa dajú YARA pravidlá jednoducho zdieľať, opätovne využívať a meniť. K dispozícii sú aj verejné YARA pravidlá pre voľné použitie[2].

Syntax pravidla

YARA pravidlo vždy začína kľúčovým slovom `rule` a nasleduje case-sensitive identifikátorom. Telo pravidla sa skladá z 3 častí, z toho prvé dve nie sú pokladané za nevyhnutný prvok:

1. Meta informácie o pravidle
2. Ľubovoľný počet reťazcov
3. Aspoň jedna podmienka booleanovského typu

Ako bolo už spomenuté, reťazce môžu byť reprezentované ako text, hexadecimálne číslo alebo regulárne výrazy. Každý reťazec obsahuje na ľavej strane identifikátor, začínajúci znakom $ nasledujúcimi alfanumerickými znakmi a podčiarkovníkmi. Na pravej strane je definovaná konkrétna sekvencia hľadaných znakov, uzatvorená špeciálnymi znakmi na základe svojho typu, aby mohli byť rôzne typy sekvencií pri interpretácii kódu rozlíšiteľné. Textové reťazce sú ohraničené štandardne dvojitými úvodzovkami a sú kódované v ASCII. YARA však podporuje aj vyhľadávanie reťazcov s iným kódovaním pomocou pridania modifikátora za reťazec. Napríklad pridaním kľúčového slova `base64` za textovým reťazcom bude YARA vyhľadávať definovaný reťazec v kódovaní base64. Textové reťazce sú predvolene vyhľadávané ako case-sensitive, ale YARA povoľuje vyhľadávanie definovaných reťazcov aj ako case-insensitive cez modifikátor `nocase`. Používanie vyhľadávania case-insensitive reťazcov sa je vhodné pri reťazcoch, ktoré môžu využívať rovnaké slová ale iný štýl písma [5].

[1] http://virustotal.github.io/yara/
[2] https://github.com/Yara-Rules/rules

Pri hľadaní vzorov v reťazcoch sa dajú aplikovať rôzne spôsoby rozširujúce hľadanú oblasť. Nástroj YARA využíva wildcards, skoky a alternatívy, ktoré sa definujú pri definícii konkrétneho reťazca. Wildcards poskytujú možnosť využitia regulárnych výrazov, možnosť vynechávania bytov a tiež umožňujú prácu s nibble bytami. Hodia sa pri reťazcoch fixnej veľkosti, pri ktorých sa mení iba ich obsah. Naopak, pri reťazcoch variabilnej dĺžky sa využívajú skoky, vďaka ktorým sa dá v určitom reťazci definovať miesto a rozsah, v ktorom sa môže nachádzať ľubovoľná postupnosť znakov v danej medzi [26].

Regulárne výrazy sú v jazyku YARA uzatvorené lomítkami /, inak je ich definícia obdobná ako pri zvyšných typoch reťazcov.

Esenciálnu časť pravidla tvoria podmienky vyhodnocujúce booleanovské výrazy. Podmienky môžu byť zložené z rôznych typov operátorov ako booleanovské, relačné, aritmetické a bitové operátory. Podmienky poskytujú špecifikáciu miesta, kde hľadať definované reťazové sekvencie, ich kombinácie, množstvo a offsety. Pravidlá umožňujú prácu s meta dátami súborov, čo je vhodné napr. pre kontrolu veľkosti súboru, a ďalej sa dokážu odkazovať na iné pravidlá, čím aplikujú abstrakciu a prispievajú k čitateľnosti pravidiel.

K pravidlám sa môžu pomocou sekcie s názvom „meta“ pridať dodatočné informácie slúžiace pre nasledujúcu analýzu. Tagy prispievajú k lepšej filtrácii pravidiel. Nachádzajú sa napravo od identifikátoru pravidla, od ktorého sú oddelené dvojbodkou [5].

```
rule example : random_tag
{
    meta:
        author = "John Snow"
        description = "This is an example"
    strings:
        $hex_string = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $text_string = "this is a text string"
        $reg_exp = /md5: [0-9a-fA-F]{32}/
    condition:
        $hex_string or $text_string or $reg_exp
}
```

Kód 2.1: Príklad štruktúry YARA pravidla, v ktorom musí analyzovaný súbor obsahovať aspoň jeden z definovaných reťazcov.

2.4 YARA v Avaste

YARA je využívaná vo firme Avast mnohými systémami na identifikáciu a klasifikáciu škodlivého kódu. YARA je rozšíriteľná cez moduly napísané v programovacom jazyku C, čo prináša široké možnosti využitia nástroja YARA pre skenovanie rôznych typov súborov a integráciu s inými systémami a nástrojmi v boji proti malwaru. Vo firme Avast sú využívané viaceré moduly nástroja, napr. moduly pre prácu so súbormi typu PE, .NET a Mach-O. Aby boli všetky moduly integrované do jedného celku, používa sa prispôsobený interný skener. Pravidlá sa delia do dvoch hlavných kategórií na základe účelu. Podľa mena sa dá na prvý pohľad rozpoznať príslušnosť pravidla do spomenutých kategórií typu: *hunting* alebo *classification*. Kategórie pravidiel typu hunting sa líšia od pravidiel typu classification v meta

parametroch. Pravidlá, patriace do kategórie pravidiel klasifikačných, obsahujú vždy meta informácie: `reliability, strain, type` a `severity`. Pravidlá, patriace do druhej kategórie, naopak, obsahujú `hunting_tag` a neobsahujú informácie o `strain, type, severity`, keďže to sú v prípade pravidiel kategórie hunting nadbytočné informácie, pretože hunting pravidlá sa neviažu na žiadnu špecifickú malwarovú rodinu a sú používané pre skúmanie všeobecného škodlivého správania. Zatiaľ čo, klasifikačné pravidlá sa používajú pre klasifikáciu špecifických malwarových rodín. Meta parameter `reliability` hovorí o spoľahlivosti pravidla v identifikácii malwaru. Pravidlá sú používané pri statickej aj dynamickej analýze a rozlišujú sa podľa meta parametru, `rule_type`, ktorý rozdeľuje pravidlá na behaviorálne a statické, popr. zložené. Behaviorálne pravidlá sú založené na výsledkoch zo systému Cuckoo, kým statické pravidlá sa spoliehajú iba na súbor a nie na správanie alebo meta informácie. Zmiešané pravidlá vyžadujú pre identifikáciu aj statické, aj behaviorálne informácie. Meta parameter `severity` rozdeľuje identifikovaný malware do kategórií podľa ich závažnosti, napr. malware, pup, tool, damaged atď. Parameter `severity` sa bližšie špecifikuje cez `type`.

Cuckoo sandbox Jeden z najznámejších open-source systémov pre analýzu malwaru, používaným aj vo firme Avast, je Cuckoo sandbox[3]. Je to automatizovaný systém pre dynamickú analýzu, zbierajúci dôležité informácie o tom, ako sa správa škodlivý program po jeho spustení v izolovanom počítačovom prostredí. Cuckoo sandbox sleduje procesy, zmeny súborov, sieťovú komunikáciu, urobené screenshoty a výpisy pamäte. Podporuje analýzu súborov vo viacerých formátoch, vrátane binárnych súborov, dokumentov vo formáte pdf, archívov vo formáte zip a webových stránok. Analýza súborov je podporovaná viacerými platformami, medzi ktoré patrí Windows, macOS, Linux a Android. Architektúra Cuckoo sandboxu je zložená z dvoch častí: 1) z hostiteľského systému, (ang. *host*), zodpovedného za riadenie procesu analýzy, a generovanie správ a 2) z izolovaných prostredí zodpovedných za bezpečné vykonanie analýzy, hostí (ang. *guests*). Po skončení procesu analýzy, hlásia hostia výsledok naspäť hostiteľskému systému [3].

2.5 Vizualizácia

Vizualizácia dát poskytuje nové spôsoby, ako sa dá na dáta pozerať. Vizualizácia pomáha objaviť vzťahy a trendy, ktoré by z nespracovaných dát alebo iba s použitým štatistických metód nemuseli byť zjavné, preto je vizualizácia dôležitým nástrojom pre malwarových analytikov, poskytujúcich prehľad o aktuálnych trendoch v šírení malwaru. Cez vizualizáciu sú sledované metriky ako: percentuálne zastúpenie typov šíriaceho sa malwaru, šírenie v systémoch, šírenie podľa závažnosti malwaru a agregované hodnoty hitov ako: maximum, minimum, priemer a ďalšie.

Na základe analýzy je následne prispôsobená tvorba nových pravidiel alebo editácia už existujúcich pravidiel. Pomocou vizualizácie trendov šírenia malwaru je možné získať tiež informácie o najrušnejších časoch šírenia typov alebo rodín malwaru, napr. aký malware sa najviac šíri počas pracovných hodín,alebo naopak, aký druh malwaru sa šíri cez víkendy alebo počas sviatkov. Cez vizualizáciu sa dá sledovať šírenie rôznych malwarov na rôznych platformách (pozn. rôzne systémy sú zamerané na analýzu iných operačných systémov, viď 2.4) a získať tak znalosti o náchylnosti operačného systému na malwarovú rodinu alebo zistiť, či bol škodlivý software odhalený pri statickej alebo pri dynamickej analýze.

[3] https://cuckoosandbox.org/

Grafana Súčasné riešenie vizualizácie vo firme Avast je vytvorené pomocou open source multi-platformy Grafana[4]. Grafana poskytuje možnosť interaktívnej vizualizácie dát v podobe širokého výberu grafov a tvorby panelov (ang. *dashboards*), zobrazujúcich prehľady o definovaných dátach. Platforma Grafana poskytuje taktiež možnosť nastavenia upozornení na užívateľom definované situácie, pričom upozornenia môžu byť spravované aj priamo pomocou nastavení v grafe. Upozornenia sú tzv. dashboard panel-driven, teda ich používanie je viazané na jednotlivé panely grafov, ktoré sa nachádzajú vo väčšom súhrnnom panely (dashboard). Upozornenia sú posielané v prípade prekročenia nastavenej hodnoty (ang. *threshold*) v pridruženom paneli s grafom. Z toho vyplýva vzťah 1-1 medzi entitou upozornenia a panelom s grafom. Upozornenia v Grafane sú vhodné pre menšie tímy, ktorých súhrnné panely (dashboards) majú medzi sebou minimálne závislosti, a nevadí im tvorba nového grafu pre každé upozornenie. Grafana je rozšíriteľná pomocou pluginov, vďaka ktorým sa dajú v systéme meniť grafické prvky alebo využívať externé služby [6].

Obr. 2.1: Ukážka súhrnného panelu, ktorý obsahuje menšie panely vizualizujúce grafy[4].

[4] https://grafana.com/

Kapitola 3

Anomálie a možnosti ich detekcie

„Anomálie sú vzory v dátach, ktoré nezodpovedajú definovej norme pre normálne správanie [10].” *Detekcia anomálií* je dátová analýza riešiaca problém identifikácie takýchto vzorov. Pojem anomália a líšiaci sa vzor sú viacvýznamové pojmy a dajú sa bližšie rozdeliť následovne [7] :

1. **Bodovou anomáliou** označujeme konkrétnu jednu vzorku dát s inými vlastnosťami ako zvyšok datasetu.
2. **Kontextová anomália** je vzorka dát s nezvyčajnými vlastnosťami, objavujúca sa iba za špeciálnych podmienok.
3. **Kolektívna anomália** je jav, kedy skupina dát s podobnými dátami nadobudla nezvyčajné vlastnosti.

Metódami detekcie anomálií sa zaoberajú prevažne oblasti štatistiky a strojového učenia. Pre detekciu anomálií v dátach existuje mnoho metód, pričom niektoré z metód sú vytvorené pre konkrétne prípady alebo domény, kým iné sú navrhnuté pre širšiu vzorku dátových súborov [7] [10].

Detekcia anomálií sa dá definovať ako funkcia ϕ:

$$\phi : \mathbb{R}^n \longrightarrow \mathbb{R} \tag{3.1}$$

$$\phi(x) \mapsto \gamma \tag{3.2}$$

kde γ je skóre anomálie, $x \in X \subseteq \mathbb{R}^n$ a X je dataset. Po konvertovaní hodnoty γ na binárnu hodnotu - normálna vs. anomálna hodnota, sa definuje funkcia ϕ_{binary} ako:

$$\phi_{binary} : \mathbb{R}^n \longrightarrow \{\text{normálna, anomália}\} \tag{3.3}$$

$$\phi_{binary}(x) \mapsto \begin{cases} \text{anomália, ak } \phi_{score}(x) > \delta \\ \text{normálna, inak} \end{cases} \tag{3.4}$$

kde $\delta \in \mathbb{R}$ a δ značí hraničnú hodnotu, tzv. threshold, od ktorého bude hodnota pokladaná za anomálnu [9].

Inak formulované v článku [8], bod v čase t sa dá považovať za anomálnu hodnotu, ak je jeho vzdialenosť od prepokladanej hodnoty väčšia ako hraničná hodnota δ. Tento vzťah sa dá matematicky zapísať ako rovnica 3.5.

$$|x_t - \hat{x}_t| > \delta \tag{3.5}$$

V kontexte tejto práce sa pracuje s time-series dátami, ktoré autori článku [10] zaradili do skupiny kontextových anomálií. Na tieto dáta sa dá ale pozerať z dvoch pohľadov, makro a mikro pohľadu:

a) Berúc do úvahy všetky rodiny a pravidlá sa vo všeobecnosti jedná o kontextové anomálie, pretože napr. hodnota 500 hitov za minútu sa dá interpretovať dvomi spôsobmi. Prvá možná interpretácia je pre malwarovú rodinu, ktorá má v priemere za minútu 200-1000 hitov, je 500 hitov/min nepodstatnou hodnotou, zatiaľ čo druhá interpretácia pre rodinu, ktorá má v priemere za minútu 0 hitov, tvorí 500 hitov za minútu značne anomálnu hodnotu.

b) Z pohľadu iba jednotlivých entít, môžeme hovoriť o bodových anomáliách, keď bude mať entita v nejakom čase neprimerane nízku alebo vysokú hodnotu hitov oproti ostatným hodnotám.

O kolektívnych anomáliách by sa dalo hovoriť v prípade, kedy by malo viac entít v rovnakom čase rovnaký trend.

3.1 Algoritmy pre detekciu anomálií

Výber metódy pre detekciu anomálií závisí od charakteru skúmaných dát. V prípade, ak sa skúmajú dáta z jedného systému, ide o dáta, pri ktorých budeme pracovať s jednou premennou, po anglicky nazvané ako *univariate data*. Na druhej strane, pri dátach z viac systémov budeme pracovať s dátami s viacerými premennými, ang. *multivariate data*. Okrem systémov môžeme v rámci vzorky dát pre túto bakalársku prácu takto kategorizovať popr. aj pohľad na typ pravidiel. Pri sledovaní iba trendov pri konkrétnom jednom type môžeme hovoriť o dátach s jednou premennou (univariate), zatiaľ čo pri sledovaní závislosti medzi statickými a behaviorálnymi pravidlami sa už jedná o prácu s viacerými premennými, teda multivariate. Okrem závislosti na premenných sa pri time-series dátach dajú sledovať aj nasledujúce vlastnosti,ktoré sú opísané v literatúre [9], [14].

Trend Time-series dáta majú trend, ak priemer μ nie je konštantný, ale v čase sa zvyšuje alebo znižuje. Trend v dátach môže byť lineárny alebo nelineárny a dá sa zistiť cez aplikáciu metód kĺzavých priemerov, ktoré z dát odstránia sezónne vplyvy a výsledkom je vyhladená verzia originálneho časového radu. Vo všeobecnosti platí, čím je väčšia dĺžka kĺzavého priemeru, tým bude vyhladenie časovej rady väčšie. Pre vyhladenie sezónneho vplyvu, napríklad kolísanie hodnôt behom hodiny, dňa alebo týždňa, je vhodné zvoliť dĺžku kĺzavého priemeru rovnú dĺžke sezónneho vplyvu. Ďalšou možnosťou je využitie vážených kĺzavých priemerov.

Sezónnosť Sezónnosť v dátach je každý typ opakujúceho sa správania v pravidelnom časovom intervale, napr. každý deň, týždeň, mesiac alebo rok. Dáta môžu obsahovať sezónnosť aj vo viacerých frekvenciách.

Level Level time-series dát sa rovná priemeru datasetu. Level sa mení, ak pre time-series dáta existuje trend.

Stacionarita Stacionarita dát nastáva v prípade, keď majú time-series dáta rovnaké správanie v každom časovom intervale, tj. sú bez trendu a majú konštantný priemer, rozptyl a autokoreláciu v čase.

Pre detekciu anomálií v dátach sa využíva mnoho algoritmov, ktoré autori článku [25] rozčlenili do nižšie spomenutých siedmich kategórií. Dodatočné informácie pochádzajú z článku [10].

1. **Algoritmy založené na štatistických metódach** Medzi výhody štatistických metód patrí ich jednoduchosť a intuitívnosť. Na druhej strane väčšina metód je vhodná iba pre dáta pracujúce s jednou premennou a pracujú na základe predpokladu o distribúcii dát.

2. **Algoritmy založené na hustote**
   Algoritmy založené na hustote vyhodnocujú hodnotu za anomálnu, ak jej hustota je zreteľne väčšia ako hustota normálnej hodnoty. Tieto algoritmy sú veľmi intuitívne a majú lepšie výsledky ako metódy založené na štatistike alebo vzdialenosti. Algoritmy založené na hustote sú ale výpočetne náročné a nie vhodné pre veľké datasety alebo datasety citlivé na veľkosť najbližších susedov.

3. **Algoritmy založené na vzdialenosti** Skóre anomálie pre jeden bod je vypočítaný ako suma vzdialenosti medzi bodmi a vzdialenosti najbližších susedov, hodnota sa označí za anomálnu, ak sa nachádza ďaleko od najbližšieho suseda. Algoritmy založené na vzdialenosti sú ako algoritmy spomínané vyššie tiež intuitívne. Okrem toho sú ľahko implementovateľné a nezávislé od distribúcie dát. Medzi nevýhody týchto metód patrí vysoká časová náročnosť, pohybujúca sa okolo $O(n^2)$. Časová náročnosť týchto metód sa dá znížiť využitým vhodných štruktúr na časovú náročnosť $O(nlog(n))$.

4. **Algoritmy založené na združovaní (ang. *clustering*)** Tieto metódy pracujú na základe združovania hodnôt do kategórií a dajú sa rozdeliť na ďalšie podkategórie. Jedna z kategórií pracuje na základe binárneho skóre, kedy je každý bod datasetu buď súčasť clusteru, alebo anomália. Ďalšia kategória pracuje s myšlienkou, že väčšie clustery sú normálne hodnoty a menšie clustery anomálne. Posledná kategória označuje za normálne hodnoty dáta bližšie k stredu clusteru a hodnoty vzdialenejšie od stredu označuje za anomálne. Algoritmy, patriace do tejto kategórie majú vysokú časovú zložitosť, sú citlivé na parametre a detekujú abnormálne hodnoty, iba v prípade ak ich nie je príliš veľa.

5. **Algoritmy založené na izolácii** Algoritmy založené na izolácii pracujú s myšlienkou, že dataset obsahuje iba malé množstvo anomálií a teda je ich jednoduché odizolovať od normálnych hodnôt. Tieto algoritmy sú vysoko škáľovateľné a majú relatívne nízku časovú zložitosť a väčšiu úspešnosť detekcie správnych anomálií. Všeobecne sú algoritmy vhodné pre dáta, obsahujúce lokálne ako aj globálne anomálie, ale závisí od typu algoritmu.

6. **Algoritmy založené na podpriestore** Algoritmus rozdelí dáta na podpriestory s určenými vlastnostami. V týchto podpriestoroch je potom jednoduché rozoznať anomáliu od normálnej hodnoty. Algoritmy, patriace do tejto kategórie, dokážu odhaľovať skryté anomálie a sú vhodné aj pre vysoko rozmerné datasety, ale na úkor vysokej časovej zložitosti.

7. **Algoritmy založené na skladaní** Tieto algoritmy fungujú na princípe využitia viacerých detektorov. Algoritmy, patriace do tejto kategórie, sú veľmi stabilné, ale zatiaľ existuje iba veľmi málo metód, patriacich k týmto algoritmom.

V rámci tejto práce bola pozornosť skúmania zameraná prevažne na štatistické metódy, ktoré boli vybraté po dohode s vedúcim práce, keďže výhodou týchto metód je jednoduchosť, intuitívnosť a užívateľovi poskytujú možnosť voľby viacerých parametrov.

Štatistické metódy Štatistické metódy sú založené na predpoklade, že normálne hodnoty sa nachádzajú v regiónoch s vysokou pravdepodobnosťou, zatiaľ čo abnormálne hodnoty sa nachádzajú v regiónoch s nízkou pravdepodobnosťou [10]. Štatistické metódy sa na základe princípu, na ktorom sú založené, delia na predpovedajúce a odhadujúce. Metódy na základe odhadu (ang. *estimation model-based methods*) berú do úvahy hodnoty v čase okolo x_t, vrátane hodnôt budúcich. Modely založené na predpovedi (ang. *prediction model-based methods*) vyhodnocujú hodnotu x_t, iba vo vzťahu k minulým hodnotám, čo prináša možnosť vyhodnotenia novej vzorky dát hneď ako sa objaví. Preto sa využívajú estimujúce modely viac pre dáta statického charakteru a modely predikujúce na real-time dátach. Najzákladnejšie modely založené na odhade sú založené na mediáne alebo absolútnej odchýlke od mediánu (MAD). Niektoré modely, založené na predikcii, používajú ako základ fixné modely, čo má za následok neschopnosť adaptácie na zmeny dát v čase. Medzi predikujúce modely, ktoré využívajú fixné modely, patria napríklad autoregresívne modely alebo ARIMA model. Obidva spôsoby využívajú implicitne definovanú hraničnú hodnotu δ. Na adaptáciu zmien v čase sa využívajú napr. kĺzavé okná [8].

Kapitola 4

Dáta zavislé na čase

V dnešnom svete sú dáta cenným zdrojom informácií pre vedcov a analytikov v rôznych pracovných odvetviach a čoskoro bude mať všetko, vrátane ľudského tela, senzor, monitorujúci jeho stav. S narastajúcim záujmom o sledovanie rôznorodých dát, prichádza dopyt po zaznamenávaní širšieho spektra informácií a skúmanie ich vzťahov. Dáta poskytujú podstatné informácie v kombinácii s časovým údajom, avšak veľkosť prínosu takejto informácie spojenej s časom sa odvíja od spôsobu ukladania časovej informácie. Je rozdiel, či sa do databázovej tabuľky ukladá vždy iba užívateľove posledné prihlásenie alebo sa každé prihlásenie ukladá ako nový záznam spolu s inými informáciami. V prvom príklade, spomenutom vyššie, sa časový údaj neustále prepisuje, čím sa odstraňuje možnosť sledovania zmien v čase následkom kontinuálnej straty predchádzajúcich časových informácií. Ak sa však bude užívateľove každé nové prihlásenie zaznamenávať ako nový záznam v tabuľke spolu s typom prehliadača, zariadenia a polohou, tak sa časové informácie prestanú strácať a môžu byť ďalej analyzované vo vzťahu k ďalším atribútom. Z toho vyplýva, že je nutné do databázy vkladať každý nový časový údaj namiesto jeho aktualizácie. Takéto dáta sú dáta závislé na čase, ďalej iba time-series dáta. Sú to dáta, pri ktorých tvorí kľúčový údaj práve čas. Znalosti o tom, v akom čase boli konkrétne dáta zaznamenané, poskytujú nové možnosti dátovej analýzy, pretože s časovými údajmi sa dá sledovať zmena dát v minulosti, zmeny dejúce sa v prítomnosti, a zároveň predpovedať budúci vývoj na základe predchádzajúcich trendov [20]. Time-series dáta prinášajú cenné informácie a nové spôsoby analýzy, ale na úkor výkonnosti a zvýšenej veľkosti databázy, zaberajúcej na disku priestor, keďže vkladanie nového záznamu pri každej zmene spôsobuje veľkú redundanciu, čo vedie obzvlášť rýchlo k mohutnému nárastu objemu databázy. Problém pomalej výkonnosti a nadmernej veľkosti riešia databázy prispôsobené pre prácu s time-series dátami.

4.1 Time-series databáza

Relačné databázy nie sú prispôsobené pre time-series dáta a k nim patriacu redundanciu, preto vznikol dopyt po optimalizovaných databázach pre prácu s týmto typom dát. Time-series databázy (skrátene TSDB) sa líšia od relačných databáz v spôsobe ukladania dát, v retencii aj kompresii dát a v prvom rade v spôsobe v akom databázy prechádzajú množstvo záznamov. Time-series databázy sú zložené z dvojice času a zvyšných hodnôt, pričom čas je vždy kľúčovým indexom. Pozeraním sa na čas ako na kľúčový prvok sú databázy schopné poskytovať lepšiu škálovateľnosť a výkonnosť. Time-series databázy ako nástroj pre analýzu a monitorovanie rôznych metrík a agregovaných dát, poskytujú mnoho zabudovaných

funkcií, slúžiacich práve k spomenutým úlohám. S rastúcim záujmom o time-series dáta, rastie aj záujem o databázy, pracujúce s týmito dátami. Podľa stránky db-engines.com[1] vzrástol záujem o TSDB od januára 2016 po január 2022 o cca. 600%.

Všetky time-series databázy nie sú rovnaké a prispôsobené na rovnaký typ dát. Časové dáta v dnešnej dobe nie sú iba dáta zo senzorov a finančné dáta ale aj rôzne vzťahy, viažúce sa na časový údaj. Každá z databáz poskytuje výnimočný spôsob práce s time-series dátami. Zatiaľ čo niektoré databázy využívajú známy relačný model, iné využívajú na ukladanie dát vlastné dátové modely, pričom niektoré TSDB môžu požadovať definovanú schému, kým iné vytvárajú schému automaticky. Ďalší rozdiel medzi databázami tvorí ukladanie dát. Time-series dáta sú ukladané do databáz v krátkych intervaloch a mnoho ráz s redundanciou, čo má za dôsledok, že databázy veľmi rýchlo narastajú vo svojej veľkosti, preto je dôležitá kompresia a downsampling dát, pričom každá TSDB pracuje s inými algoritmami optimalizovanými pre svoj dátový model. Rozdiel medzi time-series databázami tvorí aj syntax jazyka a s ním spojená krivka učenia. Opäť TSDB poskytujú široké spektrum možností od syntaxe založenej na SQL jazyku po TSDB založené na NoSQL. Aktuálna ponuka time-series databáz poskytuje široké spektrum možností pre prácu s time-series dátami. Pri výbere správnej time-series databázy (TSDB) je dôležitý práve typ ukladaných dát spolu s veľkosťou kardinality medzi ukladanými dátami. Takisto treba zvážiť, aký dátový model poskytuje TSDB, a či je model vhodný pre navrhované riešenie.

V rebríčku nižšie je príklad prvých 10 najpopulárnejších TSDB v decembri 2021 podľa hodnotenia stránky db-engines.com[2].

1. InfluxDB
2. Kdb+
3. Prometheus
4. Graphite
5. TimescaleDB
6. Apache Druid
7. RRDtool
8. OpenTSDB
9. Fauna
10. GridDB

[1] https://db-engines.com/en/ranking/time+series+dbms
[2] https://db-engines.com/en/ranking_definition

Kapitola 5

Návrh systému pre vizualizáciu trendov

Cieľom tejto práce je vypracovanie systému, umožňujúce rýchle, intuitívne a moderné zobrazenie prehľadu o šírení malwaru, a poskytnúť možnosť detekcie anomálií. Táto kapitola sa zaoberá návrhom služby, na základe existujúcich požiadaviek od zamestnancov firmy Avast, a zároveň predstavuje možné spôsoby detekcie anomálií pomocou štatistických metód.

Predchádzajúce riešenie vizualizácie vo firme Avast je vytvorené pomocou služby Grafana (viď 2.5), ktorá je spustená na internom serveri firmy. Služba poskytuje pravidelné správy o stave a upozornenia na abnormálne správanie cez aplikáciu Slack. Napriek tomu, že platforma Grafana poskytuje široké možnosti rozšíriteľnosti, je toto rozširovanie zložitý proces, závislý od technológií platformy Grafana. Medzi zásadný nedostatok platformy Grafana patrí najmä spôsob vytvárania upozornení na anomálie, ktorý je viazaný na tvorbu grafu (viď 2.5), medzi menej závažný problém patrí aj napríklad obmedzená možnosť rozloženia grafických prvkov.

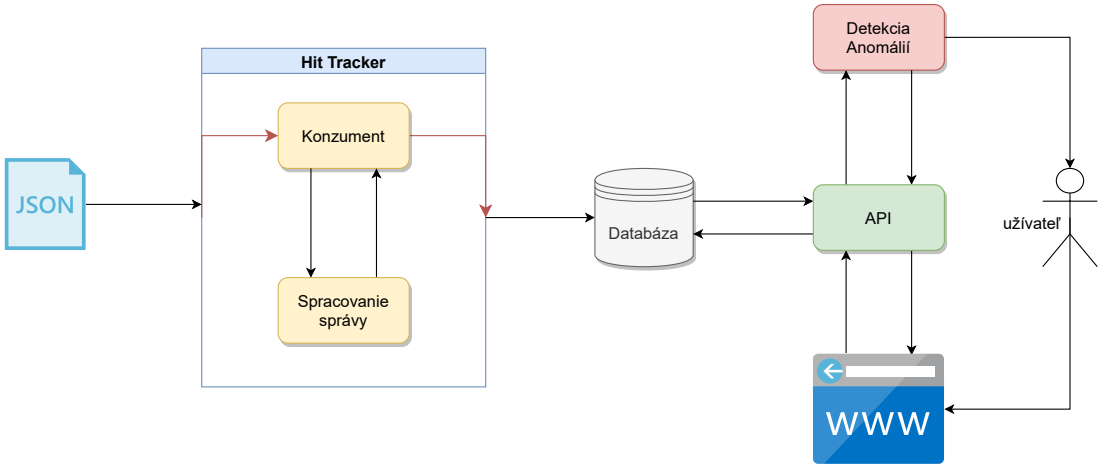
Okrem problému s detekciou anomálií je podľa užívateľov fundamentálnym nedostatkom aktuálneho riešenia aj pomalá odozva stránky pri výbere dát z dlhšieho časového intervalu, zapríčinená vysokým objemom dát a neprispôsobenou databázou pre rozsiahle časové dáta. Z týchto dôvodov bol pri tvorbe nového riešenia prvým dôležitým krokom výber databázy. Jedná sa o prácu s dátami závislými na čase, a preto bola pozornosť štúdie zameraná na rýchlosť práce s dátami a tomu odpovedajúce tzv. time-series databázy. Výber všetkých zvyšných technológií by mal byť taktiež zameraný na poskytnutie čo najvyššej výkonnosti.

5.1 Architektúra systému

Jednou z dôležitých úloh navrhovanej webovej služby je rýchla reakcia databázy na požadované dotazy a obratom rýchle zobrazenie dotazovaných dát. Žiadané dáta by mali byť znázornené pomocou prehľadných grafov, ale súčasne by mala vizualizácia slúžiť ako intuitívny nástroj, schopný zobrazovať rôzne metriky podľa zámeru užívateľa a dovoľovať mu definovať sledovanie anomálií.

Celkový systém je zložený z 5 hlavných modulov: Hit Tracker, time-series databáza, API, webové rozhranie a posledný modul tvorí modul detekcie anomálií. Prvý modul s menom Hit Tracker plní dve rôzne úlohy, a preto je rozdelený na dve časti. Úlohou prvej časti, ktorá je označená ako konzument, je zbierať dáta z výsledkov analýz z interných systémov firmy. Úlohou druhej časti je príprava dát na vloženie do databázy. Príprava dát

pre vloženie do databázy zahŕňa vytvorenie dátového modelu a extrahovanie iba relevantných dát. Druhý modul tvorí databáza, ukladajúca časové dáta. Tretím modulom systému je modul prezentujúcej vrstvy, ktorý realizuje komunikáciu medzi programom a používateľom systému cez užívateľské rozhranie. Sprostriedkovateľa komunikácie medzi databázou a webovým rozhraním tvorí aplikačné programové rozhranie (API), ktoré poskytuje možnosť využitia dát z databázy aj externým službám. Detekcia anomálií je posledný modul systému a jeho úlohou je priebežná detekcia definovaných anomálií v dátach v databáze. Detekcia anomálií je samostatný modul a komunikuje s databázou rovnako ako webové rozhranie cez modul API.



Obr. 5.1: Architektúra navrhovaného systému

5.2 Získavanie dát - Hit tracker

Navrhovaná aplikácia by mala spracovávať výsledné správy z vykonaných analýz z rôznych systémov. Správy sú rozposielané v jednotnom JSON formáte. Ukážka relevantnej časti správy je vidieť v kóde 5.1. Správa obsahuje všeobecné informácie o statuse správy a informácie o analýze ako čas, kedy bola analýza vykonaná, a názov systému, v ktorom bola vykonaná. Následne správa obsahuje výsledky analýzy v ktorej sa nachádza zoznam YARA pravidiel, ktoré našli pri analýze zhodu.

Získavanie dát by sa malo skladať z dvoch častí, v jednej časti správy prijímať a v druhej ich preformovať do štruktúry modelu a pripraviť pre vloženie do databázy. Prijímanie správ by malo pracovať asynchrónne a poskytovať možnosť spracovania správ vo väčších skupinách namiesto individuálneho spracovania. Každá správa by mala mať ďalej vytvorenú svoju inštanciu modelu danej triedy s priradenými hodnotami atribútov a následne byť vložená do databázy.

5.3 Databáza

Vo firme Avast sa objaví každý deň milión nových správ, ktoré obsahujú rôzny počet YARA pravidiel, ktoré vyvolali zhodu, preto musí mať navrhovaný systém schopnosť rýchlo pracovať s veľkým objemom dát. Doteraz používaná relačná databáza nedokáže spracovávať požiadavky dostatočne rýchlo. Potencionálne zlepšenie by mohla priniesť time-series data-

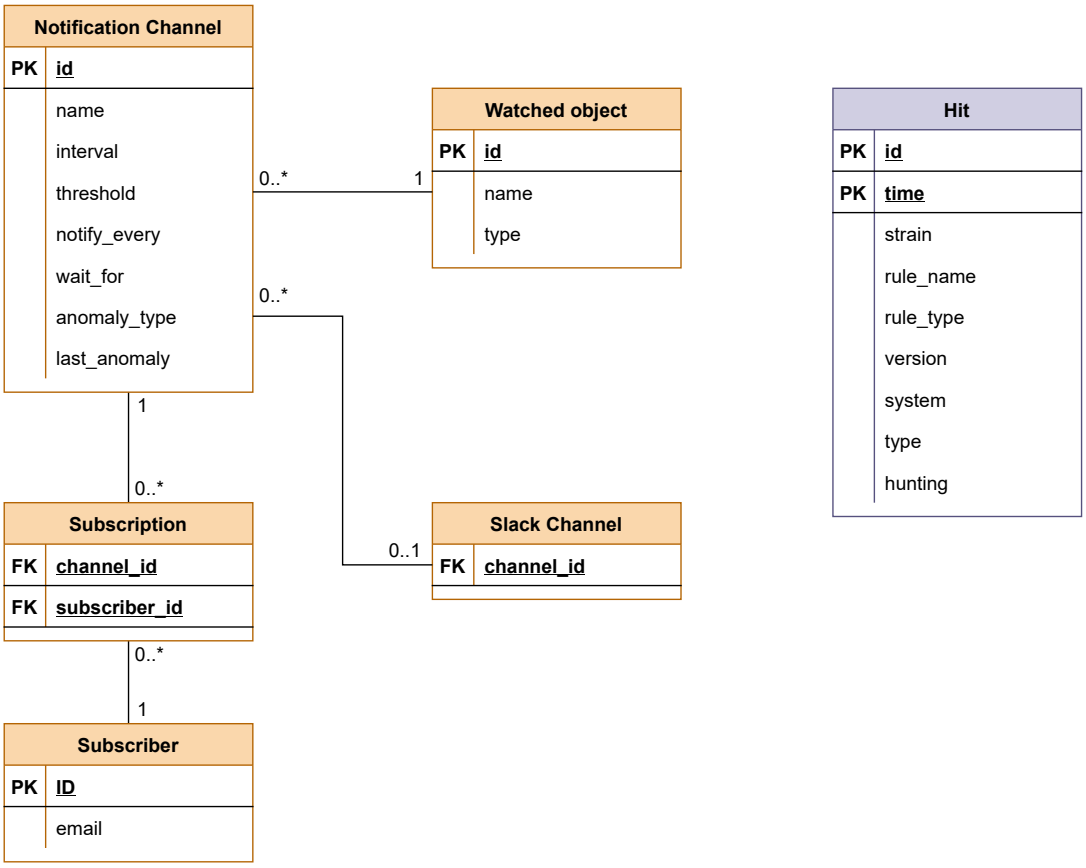
```
{
    "status": "success",
    "host": "cuckoo-26-50100",
    "task": {
    "timestamp": 1639855228,
    "system": "cuckoo",
    "result": {
        "yara": [{
            "meta": {
                "rule_type": "static",
                "description": "System discovery techniques.",
                "author": "Han Solo",
                "hunting_tag": "MITRE_T1010",
                "mitre_version": "1.1",
                "rule_version": 1,
                "attack_version": "ATT&CK-v8.2"
                    },
            "strings": [],
            "tags": [],
            "name": "random_name"
                  }
             ...
                ]
       ...}}}
```

Kód 5.1: Ukážka časti správy v JSON formáte

báza (sekcia 4.1). Nie všetky time-series databázy sú vhodné pre každý typ dát a rôzne time-series databázy sa narozdiel od relačných databáz s pevným relačným modelom môžu podstatne líšiť v modeli dát. Niektoré databázy sú zamerané na dáta pochádzajúce zo senzorov a zamerané na numerické hodnoty. Takýto typ databázy nie je vhodný pre navrhnutý systém, v ktorom je pozornosť sústredená na jednotlivé entity a ich agregované hodnoty, takže databáza optimalizovaná pre dáta zo senzorov nie je najlepšou voľbou. Faktorom, ktorý zohráva rolu pri výbere databázy, je aj skutočnosť, že komplexný systém bude pracovať aj s dátami, ktoré nie sú založené na čase. V takomto prípade sa poskytujú dve riešenia, a to výber dvoch databáz, kde jedna bude relačná a druhá time-series databáza, alebo druhé riešenie, kde time-series databáza bude poskytovať zároveň možnosť ukladania dát bez časového razítka.

Na ER diagrame 5.2 je navrhnutá štruktúra databázy celého systému. Tabuľka s názvom Hit tvorí základ systému, obsahuje časové záznamy a je určená na ukladanie do time-series databázy. Ostatné tabuľky, na obrázku oranžové, sú súčasťou aplikácie detekcie anomálií a majú vlastnosti normálnych relačných tabuliek.

Ukladané dáta Navrhovaný systém pracuje s reťazovými dátami získanými z meta informácií z vygenerovaných správ z vykonaných analýz v systémoch. Tieto správy obsahujú aj informácie o zhode s YARA pravidlami, z ktorých sa čerpajú ďalšie dáta, využívané ako zdroj dát v tomto systéme, viď 5.1. Niektoré dáta majú redundantný charakter a mohli by byť ukladané v samostatných relačných tabuľkách s využitým normalizácie, ale pri time-series dátach sa obvykle každý nový záznam ukladá do tabuľky ako celok, pretože referencova-



Obr. 5.2: Schéma databázy celkového systému

nie iných tabuliek by mohlo spomaľovať rýchlosť databázy [11]. Systém pracuje s tabuľkou hit napriek celým navrhovaným systémom. Dáta do tabuľky hit sa ukladajú v module Hit Tracker a sú vizualizované vo webovom rozhraní. Detekcia anomálií je založená taktiež na dátach z tejto tabuľky. Zvyšné tabuľky sú podstatné pre modul detekcie anomálií.

5.4 API

Databáza a klient by mali byť dve od seba nezávislé časti systému a dáta z logickej časti systému by mali poskytovať aj nezávislú možnosť ich využitia inými aplikáciami. Nezávislosť od webového rozhrania bude vytvorená cez REST (Representational state transfer) API založenom na princípoch rovnomenného (REST) internetového protokolu. API poskytuje lepšiu prenositeľnosť a flexibilitu aplikácie. Klient komunikuje s databázou prostredníctvom API v kombinácii s HTTP protokolom. Dáta z databázy budú v API spracované pre ďalšiu prácu. Vytvorené API bude podporovať nasledujúce základné operácie:

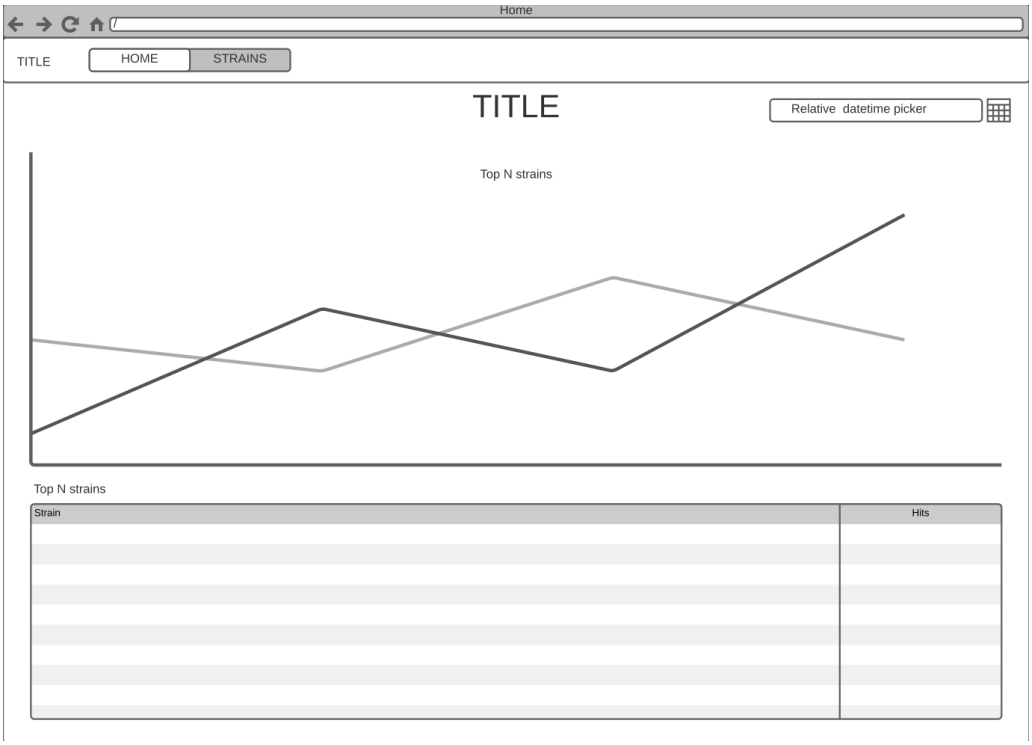
- Získanie agregované počtu objektu pre časový úsek.
- Získanie početnosti objektu rozloženého v čase.
- Získanie bližších detailov o objekte.
- Pridanie novej detekcie anomálie.

- Prihlásenie na odber upozornení pre detekciu.
- Odhlásenie z odberu upozornení pre detekciu.

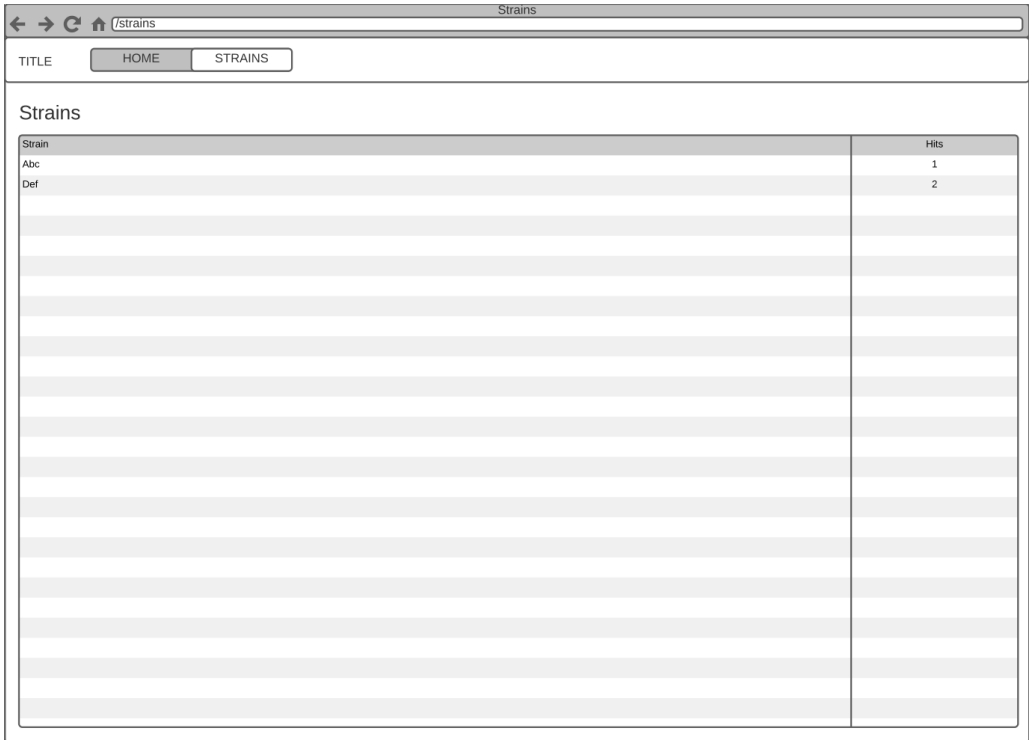
5.5 Webové rozhranie

Pri návrhu webového rozhrania bol kladený dôraz na jednoduchosť, intuitívnosť a zároveň poskytnutie žiadanej systémovej funkcionality. Základne prvky webového rozhrania budú tvorené informačnými tabuľkami a grafmi, zobrazujúcimi informácie o pravidlách a malwarových rodinách. Informácie budú poskytované aj vo forme všeobecných prehľadov ale aj konkrétne pre jednotlivé entity. Stránka bude taktiež obsahovať možnosť definície notifikačných kanálov pre detekciu anomálií (5.6) a prihlásenie alebo odhlásenie z odberu upozornení pre konkrétnu detekciu.

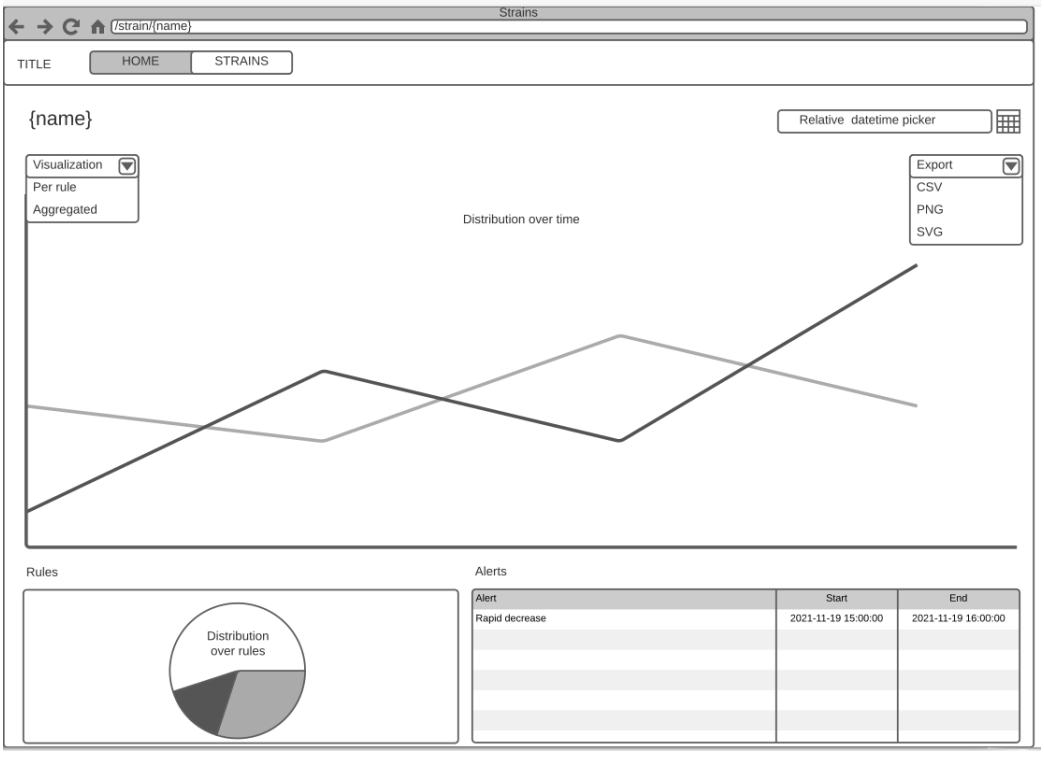
Ako je vidieť na obrázku 5.3, na hlavnej stránke sa budú nachádzať agregované počty o aktuálne najviac rozšírených malwarových rodinách vo forme tabuľky a k tomu príslušný graf. Užívateľ si bude môcť vybrať časový interval zobrazovaných dát. V hornom paneli budú umiestnené karty smerujúce na ďalšie stránky.



Obr. 5.3: Návrh úvodnej stránky.



Obr. 5.4: Návrh stránky zobrazujúcej všetky malwarové rodiny a ich početnosť. Po kliknutí na konkrétnu rodinu sa zobrazí podrobný profil rodiny, viď obrázok 5.5.



Obr. 5.5: Návrh detailu konkrétnej malwarovej rodiny.

Webová stránka bude okrem všeobecných prehľadov zobrazovať aj detailné pohľady na konkrétne entity pravidiel alebo malwarových rodín, obr. 5.5. Tento pohľad zobrazuje grafy agregovaných hitov konkrétnej rodiny alebo pravidla. Pre malwarové rodiny budú poskytnuté aj informácie o percentuálnom zastúpení v typoch pravidiel, distribúcia napriek internými systémami alebo zastúpenie naprieč dňami v týždni. Inštancie pravidiel aj rodín budú obsahovať tabuľku s informáciami o aktuálne registrovaných notifikačných kanáloch pre danú entitu. Užívateľovi bude poskytnutá možnosť stiahnuť si čiarový graf.

Stránky pre pravidlá budú vyzerať obdobne ako tie pre malwarové rodiny, zobrazené na obrázkoch 5.4 a 5.5.

Webové rozhranie bude poskytovať taktiež prehľad rôznych informácií na jednej stránke, kde si užívateľ bude môcť zvoliť časový interval zobrazovaných dát.

5.6 Návrh systému detekcie anomálií

Na základe užívateľských požiadaviek by mal systém podporovať detekciu anomálií a notifikovať užívateľov (zamestnancov firmy Avast) o abnormálnom správaní sledovaného objektu. Medzi príklady takéhoto správania patrí, napríklad rapídne stúpanie hitov od statických YARA pravidiel a klesanie pravidiel behaviorálnych, popr. nastáva klesanie hitov vyvolaných behaviorálnymi pravidlami, čo môže značiť, že sa daná malwarová rodina vytráca, alebo zmenila svoje správanie, z čoho plynie záver, že aktuálne YARA pravidlá už pri detekcii danej rodiny nie sú funkčné. Rapídne stúpanie konkrétnej rodiny môže značiť aktuálnu kampaň danej malwarovej rodiny. Okrem vyššie spomenutých typov anomálií sa za anomálie pokladajú aj prípady, kedy napr. objekt hitoval a zrazu prestane, popr. opačný prípad, kedy objekt nehitoval a zrazu začne. Po odhalení anomálneho správania vyhodnotí analytik ďalší potrebný postup.

Systém detekcie anomálií by mal byť oddelenou časťou od zvyšných častí programu a mal by komunikovať s ostatnými aplikáciami systému skrz rozhranie API. Tento vzťah je zobrazený na obr. 5.1. Detektor anomálií by mal byť založený na princípe bezstavovej aplikácie, tj. systém by nemal byť závislý od stavu predchádzajúcej operácie.

Modul detekcie anomálií by sa mal skladať minimálne z entity, definujúcej skúmanú anomáliu, nazveme ju notifikačný kanál, entity sledovaného objektu a entity sledovateľa danej anomálie. Kompletná schéma databázy, vrátane modulu detekcie anomálií, sa nachádza na obr. 5.2. Detekciu anomálií podporujú viaceré komerčné alebo open-source systémy, ku ktorým patria aj Grafana a Datadog[1], na základe ktorých bol návrh modulu detekcie anomálií inšpirovaný. Detektor bude pracovať v nekonečnej slučke, pričom pri každom priechode si vyžiada od API o zoznam aktuálnych notifikačných kanálov, ktoré bude postupne prechádzať. Každý notifikačný kanál bude niesť informáciu o type detekcie. Program si pre každý notifikačný kanál vyžiada od API potrebné dáta na základe typu detekcie.

Užívateľovi by mala byť poskytnutá možnosť výberu viacerých parametrov pri tvorbe notifikačného kanálu. Medzi tieto atribúty sa radí v prvom rade prahová hodnota, určujúca, kedy sa bude hodnota považovať už za anomálnu. Ďalej časový interval alebo tzv. time bucket, v akom sa budú dáta sledovať, pretože je rozdiel sledovať dáta agregované v 1 minútovom intervale v porovnaní s dátami agregovanými v hodinovom intervale. Väčšie časové intervaly prinášajú viac vyrovnané dáta ako intervaly menšie. Preto je pri detekcii dôležitá aj voľba intervalu. Pri príliš veľkých intervaloch sa môžu podstatné odchýlky od normálu ľahko stratiť. Na druhej strane pri príliš malých intervaloch môže nastať príliš veľa

[1] https://docs.datadoghq.com/

tzv. *false positives*, to znamená, že upozorňovanie na zmenu dát sa deje príliš často, čo nie je želaný jav. Pre príklady na reálnych dátach viď 6.6. Vhodným riešením pri sledovaní menších časových intervalov je využitie určitej doby čakania systému, skôr ako sa správanie označí za abnormálne. V rámci návrhu databázy je tento parameter označený ako `wait_for`. Podobný parameter reprezentuje hodnota označená v databázovej tabuľke, notifikačný kanál, ako `notify_every`. Táto hodnota značí dobu, akú by mal detektor medzi jednotlivými upozorneniami čakať pred poslaním nasledujúcej notifikácie. Entita notifikačného kanálu bude obsahovať aj informáciu o type detekcie a časové razítko posledného upozornenia, aby sa zabránilo spracovávaniu už spracovaných anomálií.

Notifikačný kanál bude posielať upozornenia cez aplikáciu Slack, ako vytvorený Slackbot, priamo užívateľovi prihlásenému na odber upozornení pre daný kanál alebo do dedikovaného komunikačného kanálu v aplikácii Slack. Sprostredkovateľ tejto komunikácie bude SlackBot, ktorý bude v rámci systému zastúpený ako vlastný objekt.

Kapitola 6

Implementácia navrhnutého systému

Obsahom tejto kapitoly je implementácia navrhnutého systému, opísaného v rámci kapitoly 5. Systém je implementovaný v jazykoch Python a JavaScript s využitím databázy TimeScaleDB. Bližšie priblíženie použitých technológií sa nachádza v nasledujúcej kapitole. V tejto práci sú surové dáta diskrétnymi hodnotami rozloženými v nepravidelných intervaloch, ale pracuje sa prevažne s agregovanými počtami dát, ktoré sú rozložené v rovnomerných časových úsekoch. Ak časový interval neobsahuje žiadne dáta, tak ich hodnota je reprezentovaná nulou, a teda sa dá povedať, že sa dáta javia ako spojité.

6.1 Použité technológie

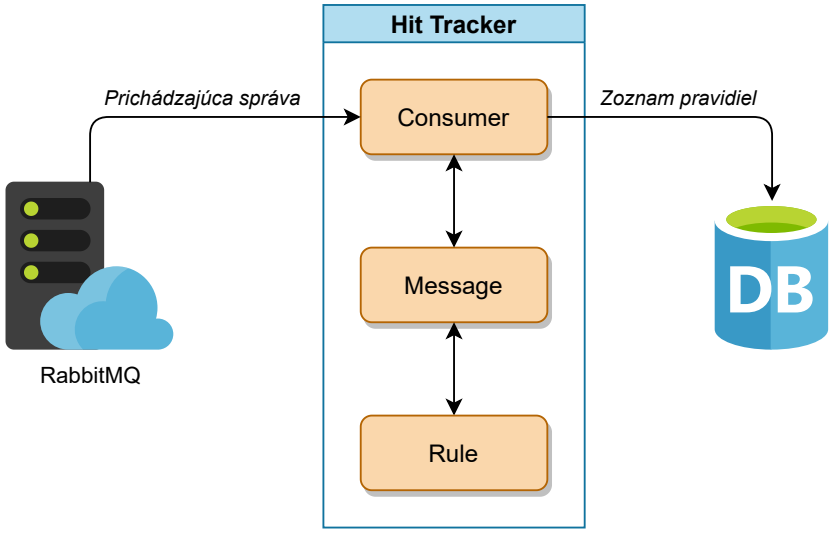
Databáza je sústredená na time-series dáta a po preskúmaní rôznych možností bola vybratá databáza TimeScaleDB[1]. Pre logickú časť aplikácie bol zvolený jazyk Python 3.9, v prezentujúcej časti sa pracuje s jazykom JavaScript, poskytujúcim široký výber dostupných knižníc pre tvorbu užívateľských rozhraní a prácu s dynamickými prvkami užívateľského rozhrania. Jazyk JavaScript je podporovaný mnohými prehliadačmi. Vďaka kompilácii kódu na klientovej strane prehliadača umožňuje rýchlu reakciu, čím sa šetrí čas potrebný na prenos dát z jedného zariadenia na druhé. Pre realizáciu webového rozhrania bol zvolený framework Vue.js[2] kvôli jednoduchšej budúcej integrácii s internými nástrojmi firmy Avast. Zvolený framework v kombinácii s frameworkom pre grafické komponenty Vuetifyjs[3] poskytuje moderné dynamické grafické prvky a prívetivé užívateľské rozhranie. Pre vizualizáciu grafov bol vybratý framework, dobre sa dopĺňajúci s Vue.js, Chart.js[4]. Pre realizáciu API bol vybratý webový framework FAST API[5], poskytujúci okrem svojej rýchlosti aj možnosť dátovej validácie pomocou nástroja Pydantic[6]. Kompletná aplikácia je zložená z Docker kontajnerov pre jednoduchú prenositeľnosť na iné systémy.

[1] https://www.timescale.com/
[2] https://vuejs.org/
[3] https://vuetifyjs.com/en/
[4] https://www.chartjs.org/
[5] https://fastapi.tiangolo.com/
[6] https://pydantic-docs.helpmanual.io/

6.2 Hit Tracker

Modul Hit Tracker sa skladá z dvoch častí. Prvá časť, nazvime ju konzument, správy prijíma, zatiaľ čo druhá časť modulu správy predspracováva pre vloženie do databázy.

Zo spomenutých správ získava navrhovaný systém dáta, s ktorými ďalej pracuje. Jedná sa o správy z výmenného systému služby RabbitMQ. Interné systémy spoločnosti Avast posielajú výsledky analýz vo forme JSON správy cez spomenutý výmenný systém RabbitMQ. Pre ukážku takejto správy viď 5.1. Výmenný systém správy ďalej distribuuje cez rady, ang. *queues*, z ktorých majú iné aplikácie možnosť správy prijímať. Aplikácia prijímajúca správy z výmenného systému sa nazýva štandarizovane konzument. Výhodou komunikovania cez výmenný systém je získanie iba potrebných správ na základe identifikácie pomocou smerovacieho kľúča, po anglicky pomocou *routing key*, a tým odfiltrovanie zbytočných správ pre daný systém. Dôležitou úlohou komunikácie cez výmenný systém je radenie správ do spomenutých radov, kde sú správy uskladnené až do času, kým ich konzument nebude schopný prijať, čo znamená, že napríklad v prípade výpadku konzumenta sa správy nestratia, ale budú čakať v rade, kým sa konzument opäť spustí.



Obr. 6.1: Hit Tracker komunikuje s RabbitMQ výmenným systémom, z ktorého dostane správu typu IncomingMessage. Hit Tracker ukladá do databázy zoznam pravidiel. Modul obsahuje triedy Consumer, Message a Rule.

Prvou časťou modulu Hit Tracker je konzument správ z RabbitMQ. Konzument pracuje asynchrónne s využitím knižnice AsyncIO, ktorá dovoľuje programu, pre maximálne využitie času, pokračovať na inú úlohu, kým sa dokončí IO operácia. Po prijatí správy odošle konzument každú správu najprv na spracovanie triede `Message`, ktorá má za úlohu zoskupiť všetky YARA pravidlá z jednej prijatej správy, aby sa mohli naraz vložiť do databázy. Ako je vidieť v kóde správy 5.1, niektoré údaje ako systém a čas analýzy sú zdieľanými atribútmi a sú spoločné pre všetky YARA pravidlá z danej správy. Spracovávajú sa iba správy s kódom „success", správy s iným kódom sú ignorované. YARA pravidlá sú v systéme reprezentované triedou `Rule`. V sekcii 2.4 o pravidlách YARA vo firme Avast boli spomenuté dve kategórie pravidiel: classification a hunting. Príslušnosť pravidla do kategórie je v triede `Rule` reprezentovaná parametrom `hunting` typu boolean. Parameter je nastavený na hodnotu `true`, ak je v spracovávanom slovníku pravidla prítomný meta para-

meter `'hunting_tag'`, typickým pre pravidlá kategórie hunting. Po spracovaní celej správy sa riadenie programu vráti konzumentovi, ktorý vloží správu do databázy. Trieda `Message` si ukladá jednotlivé inštancie triedy `Rule` do zoznamu a vloží všetky pravidlá do databázy naraz.

6.3 Databáza

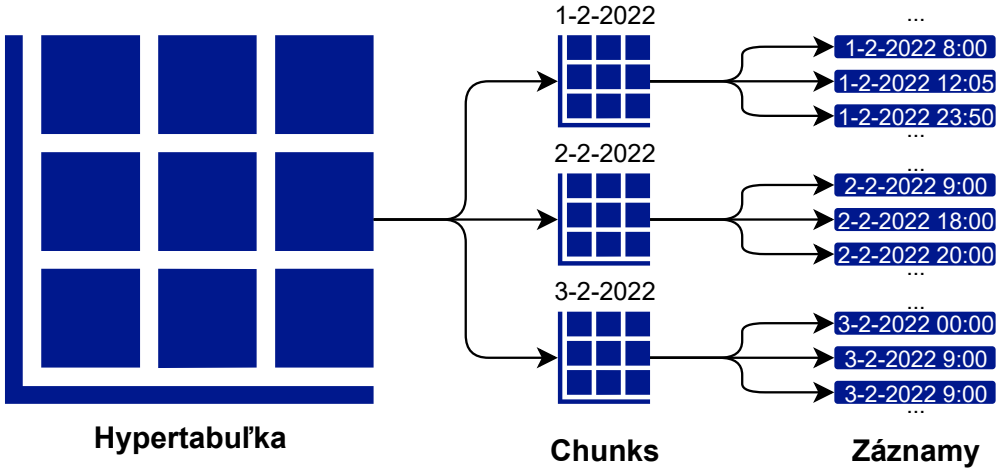
Pri výbere vhodných databáz hlavnou požiadavkou bola výkonnosť databázy a schopnosť rýchlo spracovávať dotazy nad uloženými dátami, preto bola pozornosť sústredená na time-series databázy. Ako bolo spomenuté v kapitole 4.1, nie všetky databázy sú vhodné pre všetky typy dát. Navrhovaný systém pracuje s textovými atribútmi (ang. *strings*), viď 5.3, a preto nie sú vhodnou voľbou databázy navrhnuté pre numerické hodnoty zo senzorov a rôznych meraní; medzi takéto databázy patria napríklad InfluxDB alebo GridDB. Nakoniec som sa rozhodla pre relačnú databázu TimeScaleDB, ktorá je rozšírením PostgreSQL databázy, aktívne využívanou vo firme Avast.

6.3.1 TimeScaleDB

TimeScaleDB je databáza optimalizovaná pre time-series dáta a je založená na princípe relačných databáz, vrátane dátového modelu. Každá hodnota má vlastný riadok v tabuľke, pričom časový údaj musí byť primárnym kľúčom alebo byť aspoň jeho súčasťou. TimescaleDB je rozšírením PostgreSQL databázy, takže databáza umožňuje zrýchlenie práce s relevantnými dátami, pričom si zachováva bežnú funkciu ukladania dát do relačných tabuliek. Ako rozšírenie PostgreSQL, TimeScaleDB využíva plne SQL syntax, takže ponúka užívateľovi intuitívne a známe prostredie bez nutnosti nového študovania syntaxe.

Databáza je založená na koncepte hypertables (v preklade hypertabuľky), ktoré sa ďalej delia na menšie chunks (vo voľnom preklade časti). Hypertabuľky poskytujú abstrakciu nad tabuľkami, ktoré dáta naozaj ukladajú, v koncepte tejto databázy sú nazývané „chunks". Hypertabuľky sprostredkujú všetku interakciu medzi užívateľom a jednotlivými tabuľkami s dátami. Hypertabuľka sa vytvorí najskôr ako normálna relačná tabuľka cez bežný SQL príkaz a nasleduje príkaz `SELECT create_hypertable()`, ktorý z nej urobí hypertabuľku. Hypertabuľky môžu byť iba tabuľky, v ktorých tvorí primárny kľúč časové razítko. Princíp hypertabuľky spočíva v rozdelení tabuľky na menšie štandardné databázové tabuľky, *chunks*, na základe špecifikovaného časového intervalu pri tvorbe hypertabuľky. To znamená, že záznamy sa zoskupia do častí na základe stĺpca s časovým údajom. Napríklad pri časovom intervale 1 deň budú v každej jednej časti (*chunk*) iba záznamy z tabuľky s rovnakým dňom. Proces rozdeľovania záznamov do častí je automatický, priebežný a bez zásahu užívateľa. Užívateľ databázy určí štandardným SQL príkazom časový interval, z akého by chcel získať dáta z hypertabuľky. Hypertabuľka pozná časové obmedzenia na jednotlivých chunks a na základe týchto nastavených obmedzení sa dotaz predá iba relevantným tabuľkám [4].

Hypertabuľky musia obsahovať, ako primárny kľúč alebo aspoň jeho časť, časové razítko. Databáza poskytuje možnosť tvorby aj „bežných " SQL tabuliek, kde čas nie je podstatným údajom. V rámci tohto systému je hypertabuľkou iba tabuľka hits. Hypertabuľky neumožňujú vzťahy many-to-many a pre zachovanie výkonnosti sa odporúča nechať aj napriek redundancii všetky stĺpce v jednej tabuľke [2].



Obr. 6.2: Delenie hypertabuľky na menšie tabuľky (chunks), ktoré obsahujú záznam s dátami.

SQLALchemy Keďže TimeScaleDB je rozšírením PostGreSQL databázy, je využiteľná aj Python knižnica SQLAlchemy[7], poskytujúca abstrakciu nad prácou s databázou. SQLAlchemy poskytuje dve možnosti práce s databázou a to cez 1) SQLAlchemy Core a 2) SQLAlchemy ORM. Pre túto prácu je lepšou voľbou práca s SQLALchemy Core, ktorá ponúka nižšiu abstrakciu a vyžaduje menšiu réžiu, preto oproti SQLAlchemy ORM, ktorá mapuje databázové vzťahy na objekty v jazyku Python, poskytuje SQLAlchemy Core vyššiu rýchlosť, a preto je vhodnejším riešením pre implementovaný systém.

Databáza je implementovaná s využitím knižnice SQLAlchemy Core a jej asynchrónnej verzie. Databáza využíva iba jednu hypertabuľku s názvom hit. Ostatné tabuľky patria k modelu detekcie anomálií a majú formu obyčajných relačných tabuliek. Pre schému databáze viď obrázok 5.2.

Práca s time-series dátami Všetky use-casy v systéme, ktoré využívajú časové intervaly, pracujú s dátami agregovanými do minimálnych časových okien o veľkosti 1 minúty. Databáza TimeScaleDB poskytuje dve funkcie na rozdelenie času do časových okien a to funkcie `time_bucket()` a `time_bucket_fill()`. Prvá funkcia využíva na mapovanie do časových okien iba dáta reálne prítomné v hypertabuľke. Naopak, druhá funkcia využíva tzv. *upsampling* a mapuje aj časové okná, kedy nie sú v tabuľke zaznamenané žiadne údaje a tým vypĺňa medzery medzi dátami. Uveďme príklad, kde malwarová rodina mala nasledujúce rozdelenie hitov agregovaných do časových okien o veľkosti 1 minúty cez funkciu `time_bucket()` a agregačnú funkciu `COUNT`:

1. 15.11.2021 8:00 5
2. 15:11.2021 9:00 1
3. 15.11.2021 9:02 1
4. 15.11.2021 9:30 5

[7] https://docs.sqlalchemy.org/en/14/core/

Na príklade je vidieť, že rodina nevyvolala hit každú minútu. Toto rozdelenie by neprinieslo objektívne premietnutie hodnôt do grafov v časti webového rozhrania a ani v časti detekcie anomálií. Medzi prvým a druhým záznamom by malo byť „10 jednotiek", medzi druhým a tretím iba „2 jednotky". Na grafe by sa avšak tento rozdiel buď neprejavil a obidva rozdiely by boli graficky reprezentované rovnako veľkým dielom, alebo by graf hodnoty medzi intervalmi spojil priamkou, a teda ani jedna reprezentácia hodnôt by nebola správna. Agregácia s využitým funkcie `time_bucket_fill()` rozdelí dáta na rovnomerné časové jednotky a počet pri časových štítkoch bez hitov označí hodnotou `null`.

6.4 API

API zabezpečuje komunikáciu medzi databázou a aplikáciami webového rozhrania a detekcie anomálií. Úlohami API je vkladanie a aktualizovanie získaných dát od iných aplikácií do databázy a zároveň získavanie požadovaných dát z databázy, popr. ich dodatočná úprava. API je implementované cez Python framework FAST API a protokol HTTP. Ako vyplýva z názvu, jedná sa o vysoko výkonný framework, ktorý je založený na type hint funkcii jazyka Python. Aby bol poskytnutý čas CPU využívaný na maximum, využíva sa asynchrónna práca s jednotlivými API endpointmi. API endpointy su tvorené ako štandardné funkcie v jazyku Python s dodatočným dekorátorom nad definíciou funkcie. Dekorátor špecifikuje hlavne typ HTTP operácie a URL adresu endpointu. Typová validácia rozširuje princíp tzv. type hintov poskytovaných knižnicou typing[8], cez knižnicu Pydantic, slúžiacu na dátovú validáciu. Knižnica Pydantic dohliada na správny typ a zároveň korektnú štruktúru žiadaných alebo odosielaných dát v endpointoch. API vracia vždy dáta reprezentované v odpovedajúcom dátovom modeli knižnice Pydantic. Pre dodatočnú prácu s dátami od databázy sa využíva knižnica Pandas.

Všetky endpointy, pracujúce s časovým intervalom, očakávajú časový údaj vo formáte `<int> <time-unit>` napr. *1 hour* alebo *1 minute*. Kombinácie časových jednotiek sú povolené, každá časová jednotka musí byť oddelená medzerou, ako napr. *1 day 1 hour 1 minute*. Medzi najzaujímavejšie API endpointy patria nasledujúce:

1. `/api/strains/count/` a `/api/rules/count/`, ktoré vracajú agregovaný počet hitov pre všetky alebo konkrétne entity.
2. `/api/strains/` a `/api/rules/`, vracajúce agregovaný počet hitov pre grafy. Funkcia endpointu získa z databázy agregované počty hitov pre 1 minútové intervaly, ktoré následne ďalej pomocou knižnice Pandas agreguje do rovnomerne rozložených X časových intervalov.
3. `/api/stats/`, vracajúci štatistické informácie o konkrétnom objekte.

Popis všetkých API endpointov sa nachádza v prílohe B.

6.5 Webové rozhranie

Webové rozhranie bolo implementované v jazyku JavaScript pomocou frameworku Vue.js s využitým grafických prvkov od frameworku Vuetify.js. Pre tvorbu grafov bol použitý framework Charts.js, konkrétne rozšírenie zamerané pre prácu s frameworkom Vue.js. Medzi

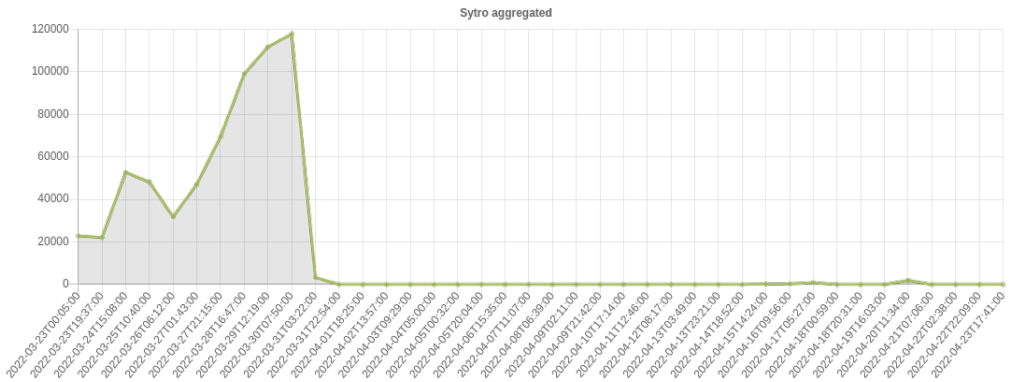
[8] https://docs.python.org/3/library/typing.html

využívané typy grafov v rámci tejto práce patria grafy čiarové a grafy v tvare tzv. doughnuts. Pre komunikáciu medzi API a webovým rozhraním bol využitý HTTP klient Axios[9].

Kód sa skladá z hlavných pohľadov (ang. *views*) webstránok, ktoré obsahujú komponenty, teda menšie časti kódu, ktoré môžu byť opakovane využité aj na inom mieste. Takéto rozdelenie znižuje redundanciu kódu a prináša prehľadnejší kód. Medzi komponenty sa radia napr. tabuľka alebo graf. Dáta v grafoch sú pre lepšiu čitateľnosť grafov agregované do 40 časových intervalov cez API endpointy spomenuté v 6.4.

Užívateľ nájde na hlavnej stránke prehľad o aktuálne piatych najrozšírenejších rodinách. Je poskytnutá aj možnosť zobrazenia viacerých rodín. Na každej stránke, vrátane hlavnej, je užívateľovi umožnená možnosť filtrácie dát podľa relatívneho času.

Stránky zobrazujúce agregované počty o malwarových rodinách a pravidlách obsahujú možnosť vyhľadania želaných výsledkov na základe kľúčového slova a obsahujú panely s možnosťou filtrácie výsledkov na základe systému. Pre pravidlá je implementovaná taktiež filtrácia na základe kategórie pravidla. Dvojitým kliknutým na príslušný riadok v tabuľke sa užívateľ dostane na detailnú stránku objektu. Na stránku inštancie objektu sa dá dostať aj cez URL endpoint `strain/{name}`, popr. `/rule/{name}`. Na stránke inštancie objektu pre pravidlá aj malwarové rodiny sa nachádza graf s agregovanými hodnotami v čase, obr. 6.3, ktorý je možné exportovať vo formáte jpeg, png alebo ako dáta vo formáte csv. Obrázky vo formáte png majú transparentné pozadie. Nižšie na stránke sa nachádza štatistická tabuľka zobrazená na obr. 6.4 a tabuľka s registrovanými detekciami pre daný objekt. Stránka inštancie malwarovej rodiny obsahuje okrem už spomenutých základných prvkov aj grafy distribúcie rodiny v systémoch, v typoch pravidlách alebo percentuálne zastúpenie šírenia podľa dní v týždni.

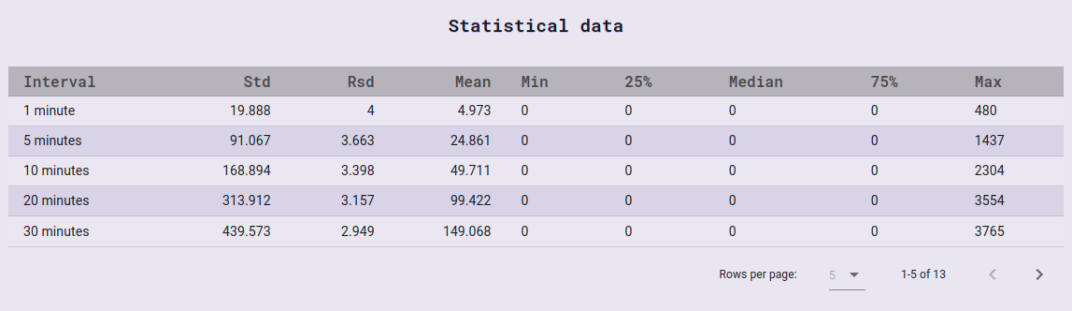


Obr. 6.3: Ukážka grafu s agregovanými počtami pre malwarovú rodinu Sytro

Stránka poskytuje aj prehľady rôznych informácií, ktoré užívateľ nájde pod URL endpointom `/rule/{name}` alebo kliknutým na tlačidlo `DASHBOARD` v navigačnej lište. Užívateľ si na stránke môže cez zadanie relatívneho času ľahko vygenerovať informácie pre žiadané obdobie, napr. posledný mesiac, týždeň alebo deň. Informácie zobrazené na stránke prehľadu zahŕňajú informácie o najviac rozšírených rodinách a pravidlách[10] a informácie o zastúpení detekovaného malwaru v systémoch a dňoch v týždni, o pomere detekcií podľa typu pravidiel alebo kategórií. Ako aj rozloženie *threat landscape*, v preklade ako rozloženie typov šíriaceho sa malwaru. Ukážka časti stránky, zobrazujúcej prehľady s informáciami na obr. 6.5.

[9] https://axios-http.com/

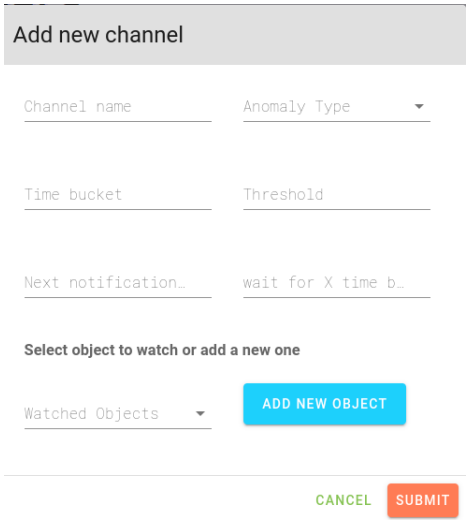
[10] pozn. pravidlá sa dajú filtrovať podľa kategórie



Obr. 6.4: Štatistická tabuľka zobrazujúca priemer, štandardnú odchýlku, variačný koeficient, priemer, minimálne a maximálne hodnoty a zároveň hodnoty kvartilov, vrátane mediánu, pre rôzne časové intervaly.

Obr. 6.5: Stránka zobrazujúca prehľady o najviac rozšírených pravidlách spolu s distribúciou malwaru v rámci rôznych vzťahov.

Webová stránka zahŕňa aj prehľad o aktuálnych detekciách a cez užívateľské rozhranie sprostredkováva definíciu nových detekcií, obr. 6.6. Kvôli návrhu databázy, 5.3, je nutné v prípade, ak ešte objekt neexistuje, definovať sledovaný objekt cez dodatočný formulár, ktorý sa zobrazí po kliknutí tlačidla `ADD NEW OBJECT`. V tabuľke s prehľadom o notifikačných kanáloch sa nachádzajú tlačidlá `SUBSCRIBE` a `UNSUBSCRIBE` pre správu upozornení pre daný kanál.



Obr. 6.6: Formulár pre definíciu nového notifikačného kanálu pre detekciu anomálií.

6.6 Detekcia anomálií

Pre detekciu zmien v dátach boli skúmané metódy pravidla 3 sigma, percentily, lineárna regresia, vzdialenosť od priemeru na základe relatívnej štandardnej odchýlky.

6.6.1 Experimenty

Pri overovaní metód popísaných nižšie v tejto podkapitole sa využívali reálne dáta, nazbierané v období 4 dní. Počty záznamov pre rôzne malwarové rodiny boli agregované do časových okien v intervaloch 5 minút, 20 minút, 1 hodina, 4 hodiny, 8 hodín a 12 hodín. Časové intervaly, ktoré neobsahovali žiadne záznamy, sú nahradené hodnotou 0 pre pravidelné rozdelenie intervalov. Viď 6.3, odsek **Práca s time-series dátami**. Pre spracovanie dát bola použitá knižnica pre analýzu dát, Pandas[11].

Pravidlo 3 sigma Pravidlo 3 sigma, označované ako 3σ, známe aj ako pravidlo 68-95-99,7, hovorí, že pri približne normálnom rozdelení štatistického súboru by sa mali skoro všetky relevantné hodnoty nachádzať do troch smerodajných odchýlok σ od aritmetického priemeru. V okolí pre 3σ od priemeru sa nachádza približne 99,73% hodnôt. Používajú sa aj vyššie násobky σ, pričom platí, čím vyšší násobok smerodajnej odchýlky σ, tým je väčšia tolerancia k extrémnym hodnotám. Štandardne sa na výpočet vzdialenosti hodnôt od priemeru využíva tzv. z-skóre, 6.1, ktoré hovorí, koľko štandardných odchýlok σ je určitá hodnota pod alebo nad priemerom. Z-skóre je vhodné pre porovnanie vzdialenosti od priemeru medzi viacerými premennými pre skúmanú hodnotu X [10], [21].

Pre túto prácu bol rámci skúmania vzorec 6.1 upravený na 6.2 a z-skóre z zvolené ako pevná hodnota.

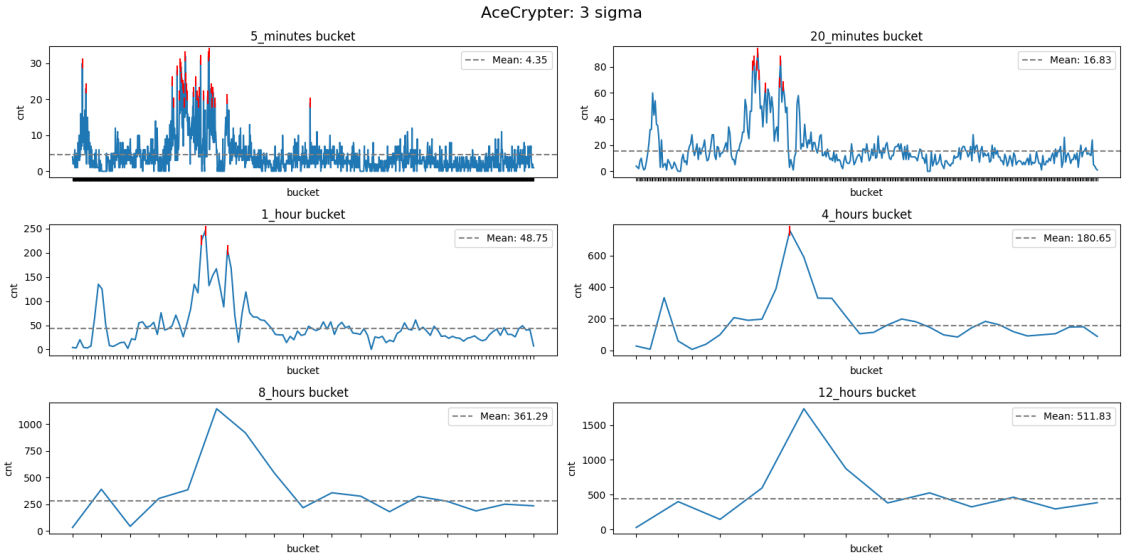
$$z = \frac{(X - \bar{x})}{\sigma} \tag{6.1}$$

[11] https://pandas.pydata.org/

kde X je hodnota znaku, $\bar{x}$ udáva priemer, σ smerodajnú odchýlku, z $\in \mathbb{R}$.

$$z \ \cdot \sigma \leq X - \bar{x} \tag{6.2}$$

Táto metóda sa ukázala ako učinná v detekovaní maxím v rámci dát, ale medzi jej nedostatky patrí predpoklad normálnej distribúcie dát a detekcia všetkých lokálnych maxím, čo pri entitách s malým priemerom prináša detekciu maxím aj tam, kde by to analytici nepovažovali za anomáliu. Takýto prípad je vidieť na grafe 6.7, kde metóda označila za anomálne všetky signifikantné vrcholy, ale z pohľadu analytika je anomália s hodnotou 30 nepodstatná. Pri širších časových intervaloch metóda vrcholy nedetekovala. Metóda je účinná pre detekciu abnormálne vysokých hodnôt, na druhej strane pre detekciu lokálnych miním nie je vhodnou voľbou, pretože v rámci skúmaného štatistického súboru sa lokálne minimá pohybujú okolo aritmetického priemeru. Vhodné riešenie pre detekciu abnormálne vysokých hodnôt by mohlo priniesť zvolenie $z \cdot \sigma$ pre $z \geq 5$ v kombinácii s inou metódou.

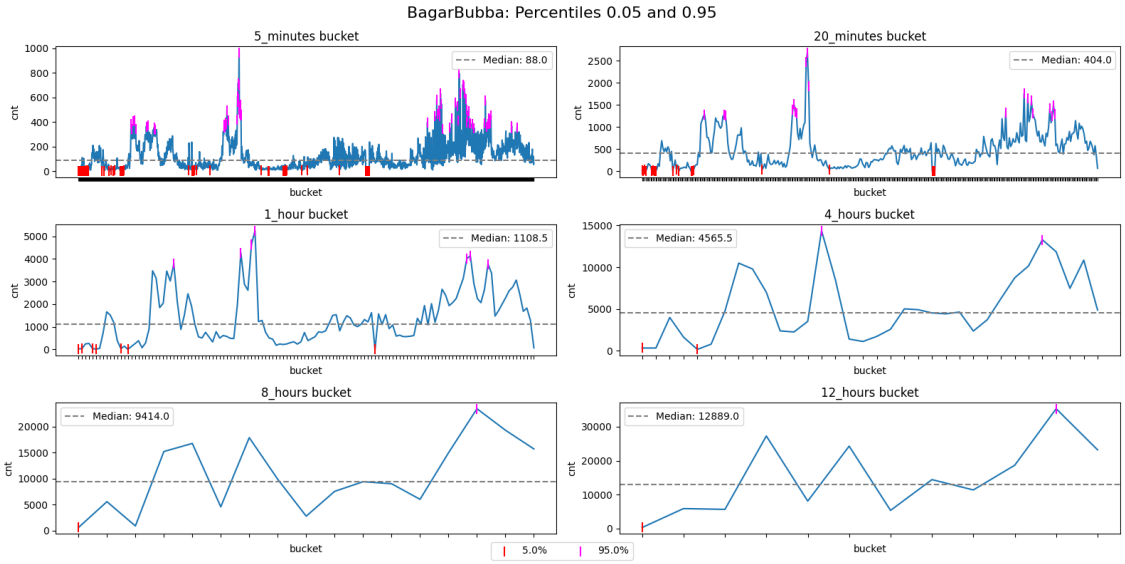


Obr. 6.7: Detekované anomálie pomocou pravidla 3 sigma.

Kvantily a percentily *„Kvantil x_p je hodnota znaku, pre ktorú platí, že najmenej p-percent prvkov má hodnotu menšiu alebo rovnú x_p*[14].“ Medián, označovaný ako $\widetilde{x}$, predstavuje strednú hodnotu štatistického súboru:

$$\widetilde{x} = x_{50} \tag{6.3}$$

Percentily sú kvantily, ktoré delia súbor na 100 častí a značia sa $x_1, x_2, \ldots, x_{98}, x_{99}$ [14]. Detekcia touto metódou dokázala detekovať abnormálne vysoké hodnoty, ale narozdiel od metódy 3σ, spomenutej vyššie, dokázala detekovať aj minimá. Rovnako ako metóda 3σ, aj táto metóda detekovala vrcholy, ktoré by nemali byť považované za abnormálne. Metóda založená na príslušnosti hodnôt do percentilov bola schopná detekovať lokálne extrémy aj v širších časových oknách. Metóda je vhodnejšia pre štatistické súbory s väčším mediánom, keďže medián ako stredná hodnota, určuje že 50% hodnôt bude $< \widetilde{x}$. Čo pri $\widetilde{x} = 0$, znamená, že minimálne všetky percentily $\leq x_{50}$ budú mať hodnotu 0.



Obr. 6.8: Detekované anomálie s hodnotou n: $n < x_5$ a $n > x_{95}$.

Jednoduchá lineárna regresia Tento paragraf bol napísaný na základe použitých zdrojov [15], [19], [24]. Úlohou regresnej analýzy je určiť vzťahy medzi dvomi alebo viacerými premennými a nájsť príslušnú regresnú funkciu, popisujúcu závislosti medzi premennými. Pri skúmaní vzťahu dvoch premenných sa jedná o jednoduchú regresiu, pri skúmaní vzťahov medzi viacerými premennými sa hovorí o viacnásobnej regresii.

Jednoduchá lineárna regresia určuje vzťah medzi dvomi premennými X a Y, pričom X je nezávislá premenná a Y premenná závislá od X. Táto závislosť je opísaná rovnicou 6.4, kde β_0, β_1 sú parametre regresnej priamky, označované aj ako regresné koeficienty. β_0 vyjadruje priesečník regresnej priamky s osou y. Koeficient β_1 vyjadruje sklon priamky k osy x alebo inak povedané smernicu priamky. ϵ vyjadruje chybu modelu.

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{6.4}$$

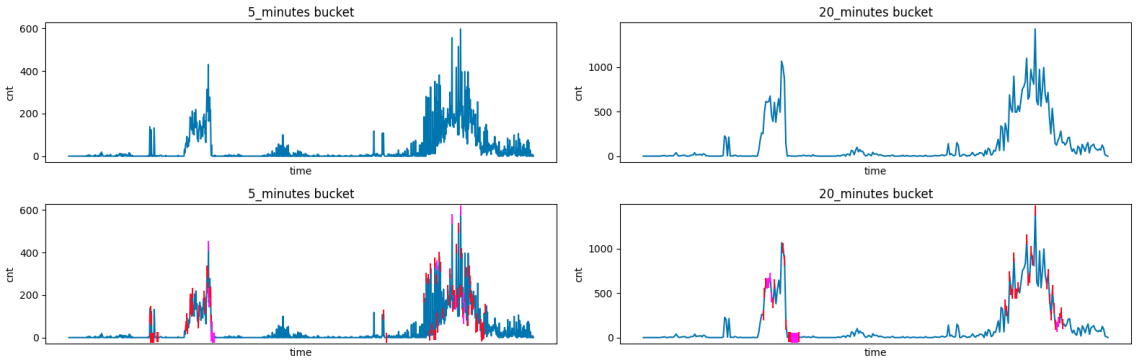
Použitie regresie pre časové okná > 4 hodiny sa ukázalo ako zbytočné, pravdepodobne vzhľadom na nízky počet obsiahnutých dát, a preto bolo rozhodnuté pre túto metódu preskúmať radšej okná v menších časových intervaloch, konkrétne okná o veľkosti 5 minút, 20 minút, 40 minút, 1 hodina, 90 minút a 4 hodiny. Pre túto metódu sa využila funkcia PostgreSQL databázy `REGR_SLOPE()`, pre výpočet smernice regresnej priamky. Smernica priamky sa počítala vždy pre časové úseky rozdelené do intervalov, pre lepšie zachytenie aktuálneho trendu namiesto generalizovaného trendu. Boli skúmané intervaly pre 3 a 10 predchádzajúcich okien. Časové razítka boli očíslované podľa poradia pre lepšiu prácu. Agregované počty pre každú malwarovú rodinu boli normalizované vzorcom 6.5.

$$m = n/\bar{x}(n) \tag{6.5}$$

kde n reprezentuje počet, m normalizované dáta a $\bar{x}$ značí priemer. V konečnom dôsledku sa získavali dáta funkciou v tvare `REGR_SLOPE(normalized, day_number)`. V knižnici pandas sa následne vybrali riadky, ktoré mali hodnotu smernice v skúmanom rozpätí.

Metóda dokázala dobre detekovať rapídne zmeny. Pri menej často vyskytujúcich sa vzorkách, označila metóda za anomálne pomerne veľa malých skokov. Pravdepodobne, to

súvisí s nepravidelnosťou dát, pretože pri sledovaní rozpätia 10 predchádzajúcich okien sa frekvencia detekovaných anomálnych hodnôt vo všeobecnosti pre prahové hodnoty > 1 znížila. Pri vzorkách rodín, pri ktorých sa nárast alebo pokles v čase dial v dlhšom časovom intervale, bolo zistené, že je potrebné nastaviť hraničnú hodnotu smernice priamky na menšiu hodnotu, pretože stúpanie nastáva pomaly, obr. 6.9. V takomto prípade by bolo vhodné využiť nižšiu prahovú hodnotu v kombinácii so sledovaním časovej dĺžky zmeny.



Obr. 6.9: Smernica regresnej priamky pre 10 predchádzajúcich časových okien. Na hornom obrázku sa nachádza detekcia pre smernicu k, $k > 1$, na dolnom obrázku pre $k > 0.2$.

Vzdialenosť od priemeru na základe variačného koeficientu Medzi možný spôsob detekcie anomálií patrí aj vzdialenosť od priemeru. Pri štatistických údajoch záleží na variabilite štatistického súboru a priemer nemusí byť veľavravný, preto sa využíva pri tejto metóde aj variačný koeficient c, inak relatívna štandardná odchýlka, 6.6.

$$c = \frac{\sigma}{\bar{x}} \tag{6.6}$$

kde σ je smerodajná odchýlka a $\bar{x}$ značí priemer. Variačný koeficient slúži k meraniu relatívnej variability a udáva z koľkých percent sa v priemere jednotlivé hodnoty odchyľujú od aritmetického priemeru. Čím je variačný koeficient menší, tým viac sú si dáta podobné. Naopak, vysoká hodnota variačného koeficientu značí, že priemer nevhodne vypovedá o skúmanom štatistickom súbore. Pre $c > 0.5$ sa dá povedať, že sú medzi hodnotami v súbore výrazné odchýlky [23]. Vzdialenosť d od priemeru $\bar{x}$ je vypočítaná vzorcom 6.7,

$$d = |1 - \frac{X}{\bar{x}}| \tag{6.7}$$

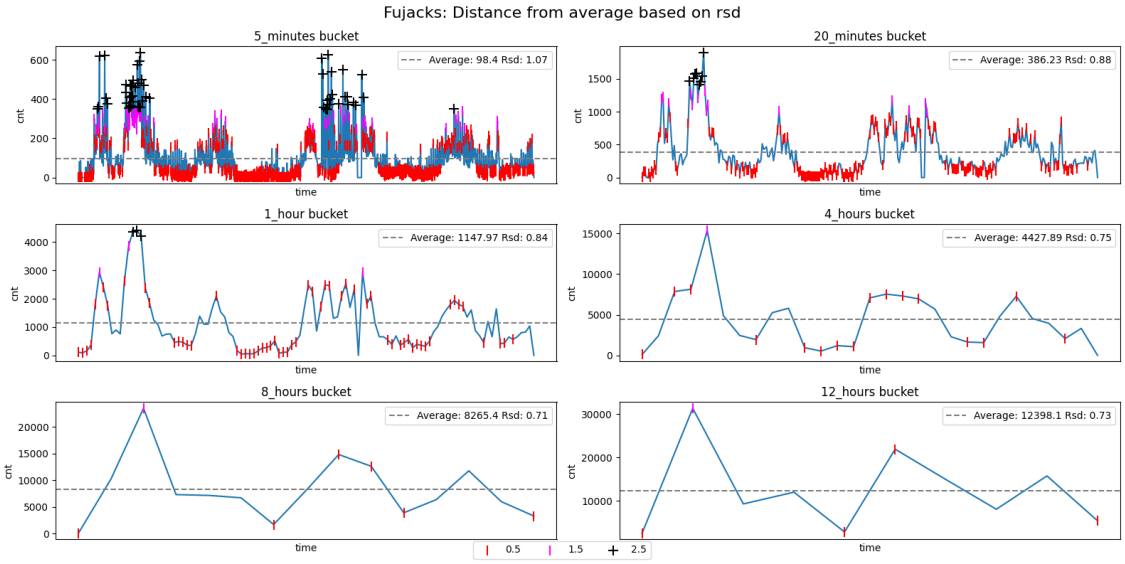
kde X značí skúmanú hodnotu.

Výsledky metódy závisia prevažne od priemeru skúmanej vzorky. Metóda dokázala dobre detekovať vyššie hodnoty od priemeru. Pri detekcii poklesov prinášala objektívnejšie výsledky skôr pre časové okná od 1 hodiny vyššie a hraničnej hodnoty h v rozmedzí $0.5 < h < 1.5$. Pri vzorkách s nižším priemerom by sa hraničná hodnota musela nastaviť pravdepodobne na vyššie hodnoty pre detekciu skutočne anomálnych hodnôt. Ako je vidieť v tabuľke 6.1, napríklad pre rodinu Allaple, s prierom 14 hitov za 5 minút, je hodnota 425 pokladaná za výrazne anomálnu, zatiaľ čo hodnota 420 pre rodinu BagarBubba s priemerom 130, nie je tak relevantnou odchýlkou. Pri vyberaní správneho parametru, sa musí zobrať do úvahy aj želaný výsledok. V tabuľke 6.1 je ďalej vidieť rozdiel medzi vzdialenosťou rodiny

Cosmu s hodnotou 10 a rodinou napr. AegisCrypter s hodnotou 3 alebo AceCrypter s hodnotou 17. Dve posledné spomenuté rodiny majú väčšiu vzdialenosť od priemeru ako rodina Cosmu so vzdialenosťou 0,9, pritom je zrejmé, že hodnota 10 pre priemer 158 hitov/min je viac anomálna ako hodnota 3 pre priemer 1 hit/min. Dá sa vyvodiť, že datasety sú vysoko variabilné v kontexte ich vlastností.

Názov	**Počet**	**Priemer**	**Variačný koeficient**	**Vzdialenosť**
Allaple	235	14.238	3.1707	15.505
Allaple	425	14.238	3.1707	28.8494
Cosmu	10	158.8643	1.3636	0.937
Cosmu	1174	158.8643	1.3636	6.3899
AceCrypter	30	4.8056	0.9695	5.2426
AceCrypter	17	4.8056	0.9695	2.5374
BagarBubba	420	130.764	0.9126	2.211
AegisCrypter	3	1.1724	0.3491	1.5588

Tabuľka 6.1: Vzdialenosť od priemeru pre vybraté malwarové rodiny pre 5 minútové časové okná.



Obr. 6.10: Detekcia na základe vzdialenosti pre rôzne hraničné intervaly.

Záver V prílohe A sa nachádzajú grafické výstupy pre zvyšné metódy pre rodiny AceCrypter a BagarBubba. Celkové výstupy pre ostatné testované rodiny sa nachádzajú na priloženom pamätovom médiu. Ukázalo sa, že metódy sú vhodnejšie pre detekciu v časových oknách v rozmedzí 20 minút - 4 hodiny. Menšie časové okná pomerne veľa oscilujú, čo môže mať za následok nadbytočné množstvo upozornení alebo naopak, že sa väčšie zmeny ľahko stratia. Obecne platí, čím dlhší časový interval, tým menej detekovaných anomálií.

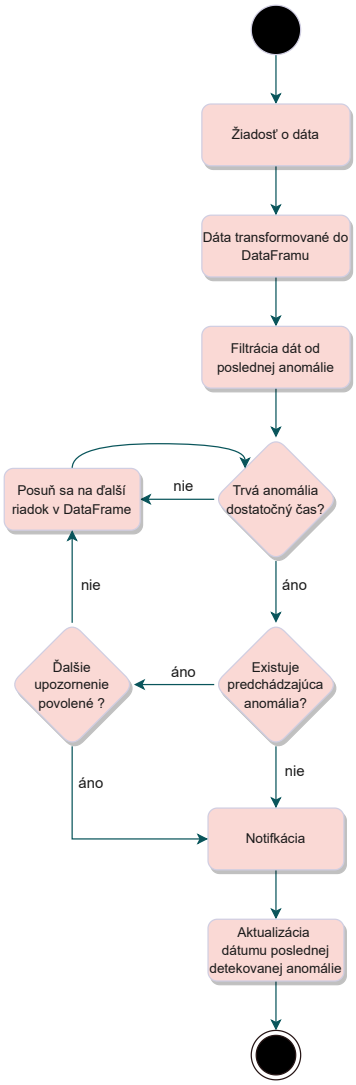
6.6.2 Modul detekcie anomálií

Modul detekcie anomálií je zastúpený v triede `AnomalyDetection` a komunikuje s databázou cez API. Detekcia konkrétnej anomálie je nazvaná ako notifikačný kanál. Návrh systému a využité parametre notifikačného kanálu sú opísané v časti 5.6. Možnosti voľby veľkosti sledovaného intervalu ako aj dodatočné parametre, určujúce hraničnú hodnotu, dĺžku časového intervalu nad hraničnou hodnotou a frekvenciu upozornení, sú prenechané užívateľovi systému. Pre posielanie upozornení cez aplikáciu Slack sa využíva samostatná trieda `SlackBot`, ktorá zastupuje vytvorenú aplikáciu v rámci komunikačnej platformy Slack. Systém pracuje s využitým knižníc jazyka Python, asyncio a pandas. Detekcia funguje na princípe nekonečného cyklu, v ktorom sa vždy na začiatku prechodu od API vyžiadajú definované notifikačné kanály, ktoré sa následne ďalej spracovávajú podľa typu detekcie. Systém podporuje 3 typy detekcie anomálií opísané nižšie v tejto sekcii. Všeobecne platí pre samostatné detekcie postup zobrazený na obr. 6.11. V prípade ak notifikačný kanál ešte nikdy nedetekoval žiadnu anomáliu, tak sa pri detekcii kontroluje iba prvá anomália v poradí. V opačnom prípade sa ako prvá označí najstaršia anomália. Tento spôsob detekcie anomálií umožňuje detekovať všetky anomálie aj v prípade nečakaného výpadku systému. Aby sa systém zbytočne nezahlcoval starými anomáliami, ktoré už nemajú význam, sú metódy obmedzené na detekciu iba za posledný deň. V prípade, ak je detekovaná anomália, systém vytvorí správu, ktorú odošle pomocou inštancie triedy `SlackBot` do súkromnej správy všetkým odberateľom pre daný notifikačný kanál alebo do dedikovaného komunikačného kanálu.

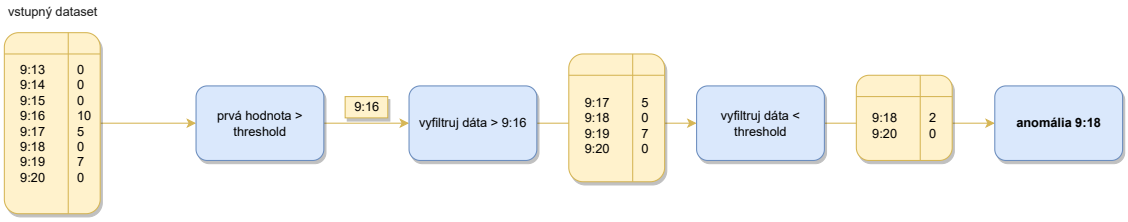
Detekcia - objekt sa dostal nad hodnotu X Metóda detekuje počet hitov nad zvolenou hraničnou hodnotou. Metóda si vyžiada cez API agregované počty v danom časovom okne za posledný deň a s využitým knižnice pandas sa vyfiltrujú dáta iba nad hraničnú hodnotu. Ďalej sa postupuje podľa obr. 6.11.

Detekcia - objekt sa dostal pod hodnotu X Metóda sleduje či objekt bol nad hraničnou hodnotou (vrátane). V prípade, ak áno, a zrazu sa jeho počty znížia pod hraničnú hodnotu, tak sa detekuje anomália. Bolo by bezvýznamné upozorňovať každých pár minút na anomáliu, poukazujúc na fakt, že sa objekt dostal pod hraničnú hodnotu, keď hraničnú hodnotu ani nikdy nepresiahol. Na obr. 6.12 je vidieť príklad, kde sa ako prvá anomália pre threshold 3, detekuje anomália s časom 9:17. Nasledujúca anomália by bola označená v čase 9:20.

Detekcia - vzdialenosť od priemeru Táto metóda bola bližšie preskúmaná v sekcii o experimentoch 6.6.1. Voľba vzdialenosti od priemeru sa prenecháva užívateľovi systému, ktorý môže pri voľbe hraničnej hodnoty vzdialenosti využiť pre voľbu štatistickú tabuľku zobrazujúcu mimo iné aj informácie o variačnom koeficiente a priemernej hodnote, 6.4. Detekcia získa hodnoty nad definovanou hraničnou hodnotou z databázy. Pretože sa rozdiel hodnôt odvíja od priemeru, ktorý sa môže v čase meniť, bolo rozhodnuté, že priemer bude počítaný vždy za posledných 5 dní a všetky anomálie budú najviac 5 dní staré.



Obr. 6.11: Postup detekcie anomálií.



Obr. 6.12: Detekcia poklesu hitov pod hraničnú hodnotu.

Kapitola 7

Testovanie systému a užívateľská spätná väzba

Systém bol otestovaný jednotkovými testami pre backendovú časť práce, ďalej bol systém otestovaný pomocou užívateľského testovania (ang. *user testing*) a nakoniec bola otestovaná aj odozva systému.

7.1 Jednotkové testy

Jednotkové testy testujú funkciu častí programu a každý test by mal byť nezávislý od iných testov. Jednotkové testy boli vytvorené pomocou knižnice pytest[1], ktorá umožňuje vytvárať tzv. *fixtures*, ktoré vytvárajú konzistentný kontext pre testy v podobe napr. databázy alebo testovacích dát. Vďaka parametrizovaniu týchto fixtures pomocou dekorátora `@pytest.mark.parametrize` môžu byť testy opakovane spustené s využitím rôznych parametrov. Keďže systém využíva vo veľkej časti aj prácu s asynchrónnymi funkciami, využila sa aj asynchrónna verzia knižnice pytest. Jednotkovými testami bola otestovaná funkcionalita backendovej časti, konkrétne funkcie modulu detekcie anomálií a API. Dokopy bolo vytvorených 139 testov. Testy sa nachádzajú v zložke `tests/`. Pre modul detekcie existuje dokopy 79 testov a nachádzajú sa v súbore `tests/test_detection.py`. Testy pre funkcie API, ktoré pracujú s časovými dátami, sa nachádzajú v súbore `tests/test_strains_rules.py` a obsahujú 29 testov. Funkcie v API, ktoré pracujú s inými dátami v databáze, sa nachádzajú v súbore `tests/test_database.py` a testy pre pomocné funkcie využívané modulom API zo súboru `api/utils.py` sú v súbore `tests/test_utils.py`. Pre tieto funkcie bolo vytvorených 9 testov.

7.2 Rýchlostné testy

Testy zamerané na skúmanie výkonnosti systému boli opakovane spustené päťdesiatkrát a pre účely testovania rýchlosti bol pozastavený Consumer zbierajúci dáta. Testovanie prebiehalo na 107 miliónoch záznamoch.

Prvá časť testovania zahŕňala testovanie rýchlosti zobrazenia dát na webovej stránke. Potom bola testovaná reakcia samotného API a nakoniec bola otestovaná aj rýchlosť detekcie anomálií.

[1] https://docs.pytest.org/en/7.1.x/

7.2.1 Výkonnosť webovej stránky

Testy, týkajúce sa zobrazenia dát na webovej stránke, boli merané pomocou nástroja lighthouse[2], ktorý vyhodnocuje výkon, dostupnosť a dokonca aj SEO (*Search Engine Optimization*). Sledovalo sa zobrazenie pre desktop a bola využitá metóda simulácie rýchlosti dátového prenosu `--throttling-method=simulate`. Na účely tohto skúmania bola sledovaná iba kategória `performance`, ktorá určuje, ako rýchlo sa stránka načíta. Pre testovanie systému bola pozornosť zameraná na parametre:

- **First Contentful Paint (FCP)** hodnotí, ako dlho trvalo prehliadaču načítať prvý DOM element.
- **Largest Contentful Paint (LCP)** reprezentuje rýchlosť zobrazenia najväčšieho obrázku alebo textového bloku.
- **Time to Interactive** hodnotí čas, kedy sa stránka stáva pre užívateľa interaktívnou, to znamená, že na stránke je zobrazený podstatný obsah, sú zaregistrovaní všetci správcovia udalostí (ang. *event handlers*) a stránka dokáže zareagovať na užívateľovu požiadavku do 50 ms.
- **Max Potential First Input Delay** vyjadruje trvanie maximálne najdlhšej úlohy po vykreslení prvého DOM elementu.

Skúmané boli konkrétne stránky HomePage, stránka rodiny Kraton a stránka Rules, zobrazujúca agregovanú tabuľku s pravidlami. Výsledné merania boli agregované pre priemer a medián, ako je vidieť v tabuľke 7.1.

	HomePage		**Rules**		**Kraton**	
	$\bar{x}$	$\widetilde{x}$	$\bar{x}$	$\widetilde{x}$	$\bar{x}$	$\widetilde{x}$
first-contentful-paint	0,389	0,4	0,366	0.37	0,364	0,38
largest-contentful-paint	0,692	0,7	0,01	0,01	0,204	0,135
interactive	0,947	0,97	0,111	0.11	0,945	0,96
max-potential-fid	0,965	0,96	0,948	0,95	0,9	0,91

Tabuľka 7.1: Priemer a medián pre výsledky meraní s nástrojom lighthouse. Hodnoty v tabuľke udávajú skóre podľa nástroja lighthouse. Skóre=1 je najlepšie možné.

	HomePage		**Rules**		**Kraton**	
	$\bar{x}$	$\widetilde{x}$	$\bar{x}$	$\widetilde{x}$	$\bar{x}$	$\widetilde{x}$
first-contentful-paint	1,8	1,78	1,86	1,83	1,86	1,8
largest-contentful-paint	1,84	1,82	8+	8+	5,45	6,05
interactive	3,34	2,79	13,69	13,69	3,34	2,55

Tabuľka 7.2: Prepočítané hodnoty skóre z tabuľky 7.1 na čas trvania v sekundách.

Pri načítávaní pravidiel je nízke skóre spôsobené tým, že hlavný element na stránke tvorí práve tabuľka, ktorá sa vykreslí až potom, čo získa dáta od API. Hodnoty skóre sa dajú pomocou nástroja[3] prepočítať aspoň na približný čas trvania načítavania prvkov v sekundách, tabuľka 7.2. Celkovo sa dá povedať, že nízke hodnoty skóre sú spôsobené čakaním na

[2] https://developers.google.com/web/tools/lighthouse
[3] https://googlechrome.github.io/lighthouse/scorecalc/

dáta z API a ich následné vykreslenie, pretože väčšina podstatných elementov na stránke je závislá od vyžiadaných dát. Najlepšie je vidieť závislosť medzi webovým rozhraním a API práve na stránke Rules, viď aj tab. 7.3 riadok 4, kde hodnota trvania interaktivity stránky Rules je spôsobená dlhším načítávaním dát z API. Parameter max-potential-fid má vysoké skóre, $0,9 - 0,96$, z čoho sa dá usúdiť, že stránka dokáže rýchlo reagovať na užívateľove žiadosti, ak si chce zobraziť iné dáta. Nástroj lighthouse poskytuje aj informácie o možných vylepšeniach. Pre webovú stránku navrhol nástroj kompresiu textových zdrojov, aby sa minimalizoval prenosový objem. Kompresia textu by mohla priniesť až 1,28 sekundové zlepšenie pre parametre FCP a LCP na stránke Rules. V rámci celej webovej stránky by bolo prínosné využívať pamäť cache pre statické webové prvky ako napr. firemné logo.

7.2.2 Výkonnosť API

Rýchlosť odpovede API bola meraná pomocou nástroja Apache Bench[4], ktorý slúži na porovnanie výkonnosti serveru. Všetky testy boli opäť spúšťané opakovane päťdesiatkrát. Testy boli spustené aj súbežne pre 10 procesov. Merané API žiadosti boli následovné:

1. `/api/strains/count/`
   Parametre: strain = Kraton
2. `/api/strains/count/`
   Parametre: limit = 5
3. `/api/strains/count/`
   Parametre: limit = 50
4. `/api/rules/count/`
   Parametre: system = cuckoo, gvma
5. `/api/strains/`
   Parametre: strain = BagarBubba, Sytro, Kraton, Fujacks, Cosmu
6. `/api/rules/distribution/`
7. `/api/stats/`
   Parametre: name = Kraton, type = strain
8. `/api/anomalies/distance/`
   Parametre: name = Kraton, typ = Strain, bucket_size = 1 minute, threshold = 0

	1 proces					**10 súbežných procesov**				
	min	$\bar{x}$	**std**	$\widetilde{x}$	**max**	**min**	$\bar{x}$	**std**	$\widetilde{x}$	**max**
1	2,01	2,39	0,48	2,34	5,20	2,76	12,93	11,09	12,00	31,02
2	4,81	4,99	0,25	4,92	6,32	5,14	6,21	0,49	6,11	7,14
3	4,88	4,99	0,18	4,92	5,83	4,91	6,03	0,24	6,09	6,31
4	71,33	83,68	5,98	82,29	98,31	73,71	575,00	85,71	586,21	658,48
5	6,3	8,36	1,52	9,35	10,13	6,74	13,24	2,68	13,02	19,15
6	3,55	3,71	0,19	3,68	4,67	3,79	4,63	0,36	4,62	5,23
7	4,23	4,38	0,11	4,40	4,77	4,64	8,9	0,98	9,11	9,75
8	2,42	2,69	0,37	2,63	4,72	2,87	6,36	0,72	6,42	8,87

Tabuľka 7.3: Výsledky meraní zo správy od Apache Bench v sekundách.

Ako je vidieť v tabuľke 7.3, systém má viditeľne nižšiu výkonnosť pri agregácii podľa systémov. Riešením tohto problému by mohlo byť pridanie indexov podľa systému. Pre zrýchlenie často využívaných API žiadostí sa dá využiť aj funkcia `continuous agreggates` od TimeScaleDB, ktorá vylepšuje funkciu `materialized views` od databázy PostGreSQL tým,

[4] https://httpd.apache.org/docs/2.4/programs/ab.html

že `continuous agreggates` obnovujú iba tú časť dát, ktorá bola zmenená. Z tabuľky taktiež vyplýva, že výkonnosť API sa mierne znižuje so zvyšujúcim sa počtom súbežne bežiacich žiadostí, čo je pravdepodobne spôsobené tým, že SQLAlchemy pri každej žiadosti vytvára nové pripojenie k databáze.

7.2.3 Výkonnosť detekcie anomálií

Skúmanie rýchlosti detekcie anomálií bolo rozdelené na dve skupiny, pretože detekcia na základe priemeru je implementovaná mierne odlišným spôsobom ako zvyšné dve detekcie, viď 6.6. Testovanie prebiehalo pomocou modulu jazyka Python, `timeit`[5] a bolo spustené pre každé meranie tisíckrát. Funkcia `timeit` vracia ako výsledok hodnotu pre sumu všetkých meraní, preto bola výsledná hodnota predelená číslom 1000, a tým sa získal priemerný čas. Pre všetky funkcie bol otestovaný čas exekúcie pre notifikačný kanál so žiadnymi dodatočnými parametrami a pre kanál, ktorý vyžadoval aspoň 3 za sebou idúce časové intervaly, tzn. kanál s parametrom `wait_for = 3`.

Testovanie prvej kategórie prebehlo na 272 628 záznamoch pre rodinu Kraton v čase 29-04-2022 00:00 – 29-04-2022 23:59 a bol testovaný iba čas exekúcie samostatných funkcií, nachádzajúcich sa v triedy `AnomalyDetector`. Konkrétne boli testované funkcie `ad_hits` a `ad_no_hits` už s dopredu stiahnutými dátami z API.

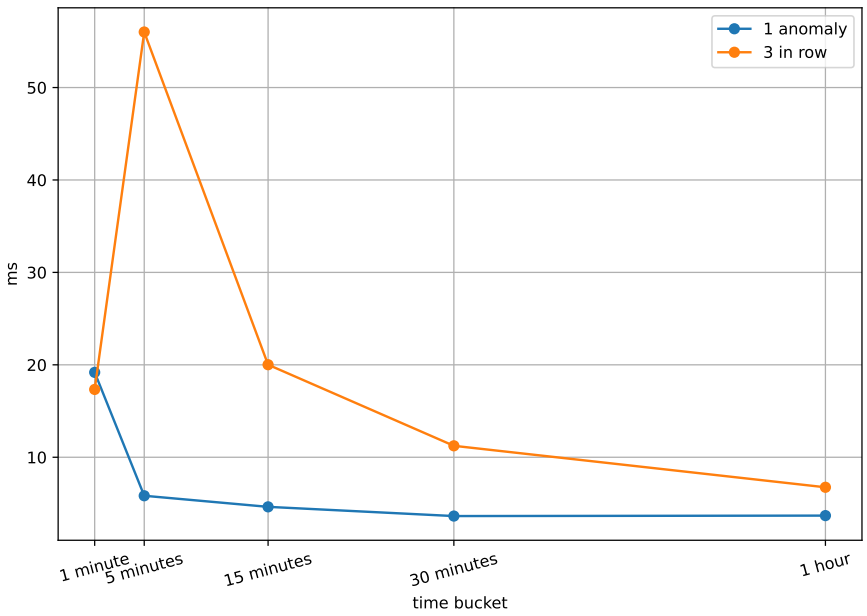
	NAD HODNOTOU		**POD HODNOTOU**	
	1. hodnota	**3 za sebou**	**1. hodnota**	**3 za sebou**
1 min	19,185	17,338	16,389	15,045
5 min	5,834	56,006	4,552	4,967
15 min	4,634	20,012	3,828	3,141
30 min	3,637	11,238	2,597	2,609
1 hod	3,686	6,751	2,392	2,390

Tabuľka 7.4: Výsledky meraní v ms pre nájdenie anomálie pre detekciu (NO HITS), kedy hodnota v danom časovom okne klesla pod polovicu priemernej hodnoty(okrem intervalu 1 hod, kedy bola hodnota nastavená na 7000) a pre detekciu (HITS), kedy hodnota dosiahla hodnotu nad medián počtu hitov pre dané časové okno.

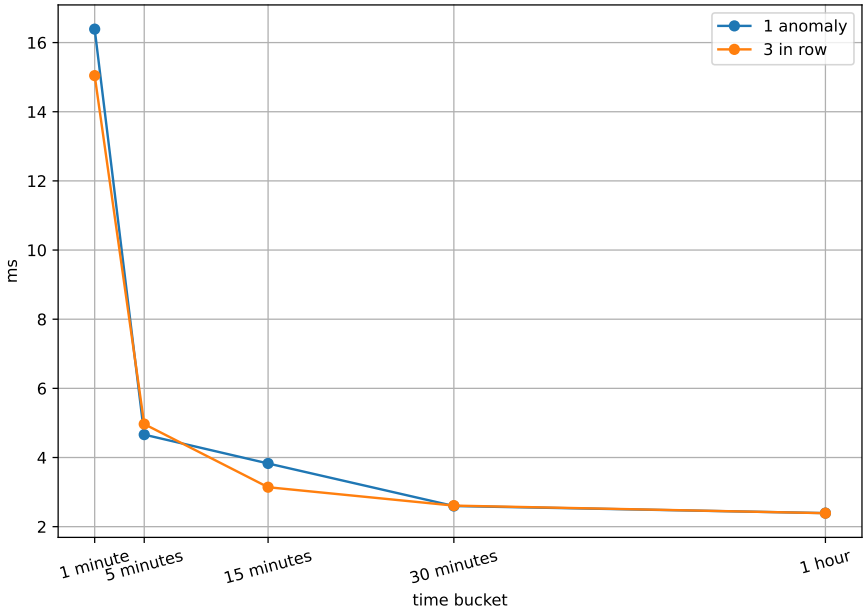
Testovanie druhej kategórie prebiehalo opäť pre rodinu Kraton na dátach v rozpätí 5 dní, konkrétne 26-04-2022 00:00 – 30-04-2022 23:59 a tvorilo ich 64 233 094 záznamov. Funkcia `ad_distance_from_avg` bola rozdelená v rámci testovania rýchlosti na dve časti, kde sa zvlášť merala rýchlosť reakcie API a rýchlosť exekúcie zvyšku funkcie. Priemerná rýchlosť reakcie API endpointu na žiadosť o dáta je cca. 15 sekúnd. Rýchlosť API bola otestovaná cez nástroj Apache benchmark. Testovanie času exekúcie kódu, ktorý spracováva už anomálie v triede `AnomalyDetector`, bolo vykonané pomocou funkcie `timeit`. Výsledky testovania funkciou `timeit` sa nachádzajú v tabuľke 7.5 a na obrázku 7.3.

Na obrázkoch 7.1, 7.2 je vidieť, že čas potrebný na detekciu anomálie sa znižuje s väčšími časovými oknami. Na obr. 7.1 a 7.3 je možno vidieť, že detekcia, skúmajúca, či anomália trvá určitý čas, t. j. viac časových okien po sebe, má lineárnu časovú zložitosť, pretože platí, že ak existuje n intervalov a je skúmané trvanie pre m intervalov, tak maximálne množstvo vykonaných priechodov bude $n - m + 1$, teda $O(n)$. Dlhší čas trvania detekcie pri

[5] https://docs.python.org/3/library/timeit.html



Obr. 7.1: Hodnoty merania z tabuľky 7.4 pre detekciu anomálií nad prahovou hodnotou.

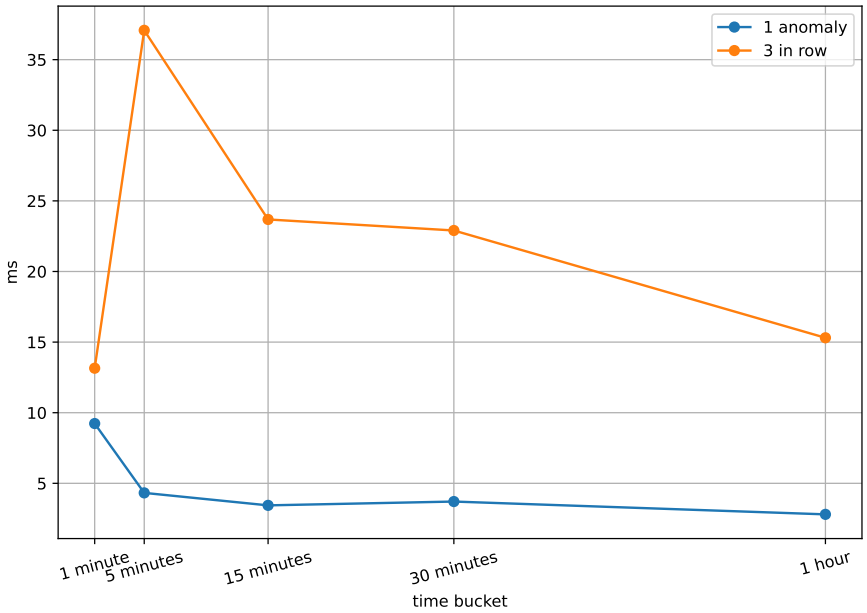


Obr. 7.2: Hodnoty merania z tabuľky 7.4 pre detekciu anomálií pod prahovou hodnotou.

detekovaní anomálií podľa priemeru je spôsobený väčším intervalom dát. Tento interval by sa dal zmenšiť na 1 deň s tým, že by sa stále zachoval priemer pre 5 dní rodín.

	VZDIALENOSŤ OD PRIEMERU	
	1.hodnota	**3 za sebou**
1 min	9,233	13,151
5 min	4,323	37,078
15 min	3,439	23,685
30 min	3,709	22,904
1 hod	2,802	15,305

Tabuľka 7.5: Výsledky merania dĺžky vykonávania detekcie anomálie na základe vzdialenosti od priemeru v ms.



Obr. 7.3: Hodnoty merania pre detekciu anomálií na základe vzdialenosti od priemeru.

7.3 Užívateľské testovanie

Systém bol otestovaný firemným analytikom, ktorý mal za úlohu skúsiť vytvorené scenáre zobrazené nižšie, a odpovedať na otázky.

1. *Zisti najvyšší počet agregovaných hitov rodiny Andromeda za posledných 5 dní.*
    - Ako rýchlo si sa vedel zorientovať a získať požadované informácie?
    - Sú informácie v grafe agregované vhodným spôsobom? Pozri sa na časové razítka intervalov a či sú informácie rozdelené do vhodného počtu intervalov (max. počet intervalov je 30) alebo by mal byť počet intervalov menší/väčší?
    - V akom časovom intervale ťa zaujímajú dáta najčastejšie? (napr. posledná hodina, deň, týždeň)?
    - Prezri si celú stránku. Využil by si informáciu o distribúcii v dňoch týždni? Myslíš si, že je táto informácia reprezentovaná vhodným spôsobom?

2. *Vytvor notifikačný kanál pre upozorňovanie na anomálie a prihlás sa na upozornenia do DMs.*

    - Rozumieš funkciám parametrov, ktoré sa nastavujú pri tvorení notifikačného kanálu?
    - Podarilo sa ti vytvoriť nový notifikačný kanál?

3. *Stiahni graf zobrazujúci vývoj statických a behaviorálnych pravidiel pre rodinu Sytro vo formáte jpeg.*

    - Podarilo sa ti načítať požadovaný graf a stiahnuť ho vo formáte jpeg? Skús stiahnuť aj zvyšné formáty. Sú formáty png a csv vytvorené vhodným spôsobom?
    - Je ponuka formátov pre export dát dostatočná? Ak nie, aký formát chýba?

4. *Získaj informáciu o 5 najviac rozšírených klasifikačných pravidlách v systémoch cuckoo a gvma.*

    - Za aký dlhý čas sa ti podarilo nájsť požadovanú informáciu?
    - Načítavali sa požadované dáta primeraný čas?

5. *Zisti priemernú hodnotu za minútu pre klasifikačné pravidlo kraton_known_sequences za posledný deň a 12 hodín.*

    - Podarilo sa ti nájsť a zobraziť požadované informácie?
    - Aký relatívny čas si napísal do textového poľa pre filtrovanie času?
    - Ako rýchlo sa ti podarilo pochopiť princíp filtrovania podľa relatívneho času? Príde ti filtrovanie podľa relatívneho času ako vhodná voľba alebo by si preferoval filtrovanie podľa absolútneho času?
    - Je význam všetkých hodnôt v štatistickej tabuľke zrozumiteľný?

6. *Pozri si prehľad informácií za posledný týždeň.*

    - Ako rýchlo sa ti podarilo dané informácie nájsť?
    - Sú zobrazované informácie dostatočné? Ak nie, aké informácie chýbajú?

Pri prvom scenári mal užívateľ mierny problém sa zorientovať na stránke a nájsť konkrétnu stránku rodiny Andromeda, na ktorú sa musí užívateľ dostať dvojitým kliknutým v tabuľke so všetkými rodinami. Pri ďalších scenároch už užívateľ nemal problém nájsť požadovanú informáciu. Užívateľ zhodnotil, že počet intervalov v grafe je dostatočný, ale bolo by vhodnejšie, keby by boli v grafe zobrazované rovnaké časy. Užívateľ odpovedal, že by bol využil distribúciu šírenia v dňoch v týždni, ak by tam bola abnormalita, kedy by kampane daného malwaru v istý deň, prevyšovali ostatné dni. Parametre pri tvorbe notifikačných kanálov neboli užívateľovi zjavné. Scenár 2 sa nepodarilo užívateľovi splniť celý kvôli chybe, ktorá neumožňovala načítať vytvorené objekty. Táto chyba bola odstránená. 3. scenár sa podaril užívateľovi splniť bez problémov a ponuka formátov pre export je podľa neho dostatočná. Užívateľ dokázal veľmi rýchlo pochopiť filtrovanie pri pravidlách, avšak nahlásil problémy s dlhším filtrovaním dát, ktoré trvalo 1,5 minúty. Užívateľ nemal problém s filtrovaním podľa času a podľa neho mu pomohol aj príklad pod filtrovacím poľom. Filtrovanie podľa

relatívneho času hodnotí pozitívne, ale ocenil by aj filtrovanie podľa absolútneho času. Štatistické dáta sa nemenia spolu s filtrovaním časom. Užívateľ by pri celkovom prehľade dát ocenil aj informáciu o tom, či je aktuálny trend klesajúci alebo rastúci.

Orientácia na stránke a spôsob získavania informácií boli pre užívateľa intuitívne, ale význam parametrov, napríklad v štatistickej tabuľke alebo pri tvorbe notifikačných kanálov, už tak intuitívny nebol, preto by bolo vhodné vytvoriť tooltip vysvetľujúci význam jednotlivých parametrov a pri tabuľke pridať zobrazenie významu prvkov, napr. cez event handler: *onHover*. Na základe užívateľovej odpovede by bolo vhodné, aby implicitne graf zobrazoval údaje za posledný mesiac, pretože tieto užívateľa zaujímajú načastejšie. Taktiež by bolo vhodné upraviť časové údaje v grafe na pravidelné namiesto presného rozdelenia celého časového úseku na 30 dielov. Užívateľ sa stretol s problémami pri tvorbe notifikačného kanálu, tieto problémy boli opravené. Na základe spätnej väzby od užívateľa, by bolo vhodné zlúčiť tlačidlá `subscribe/unsubscribe` do jedného. Pri takejto verzii sa ponúkajú dve riešenia. Prvé riešenie by zobrazovalo zoznam e-mailov, ktoré sú prihlásené na odber, a dalo by sa jednoducho z ich odberu odhlásiť kliknutím na tlačidlo `X`, pričom prihlasovanie na odber pri tejto verzii by bolo zachované cez textové pole. Druhé ponúkané riešenie by bolo v podobe tlačidla, ktoré by reflektovalo aktuálny stav prihláseného užívateľa na stránke, ale pre takúto funkcionalitu by musel byť systém rozšírený o prihlásenie, napr. pomocou OAuth 2.0. V takomto prípade by nebolo nutné ani zadávať e-mailovú adresu a užívateľ by si jednoducho menil stav odberu kliknutím na jedno tlačidlo. Na stránke chýba informácia v prípade nefungujúcej backendovej služby, ktorú je nutné doplniť. Rýchlosť načítavania dát bola primeraná, až na načítavanie dát pri filtrovaní podľa systému, ktorú treba zlepšiť (viď 7.2.2). Celkovo bol užívateľ s ponúkanou funkcionalitou spokojný.

Kapitola 8

Záver

Cieľom tejto práce bolo pre malwarových analytikov vytvoriť systém, vizualizujúci charakteristiku šírenia malwaru a jeho detekcie. Okrem úlohy plniacu vizuálnu stránku, bolo cieľom práce zlepšiť rýchlosť stránky oproti predchádzajúcemu systému s využitím time-series databázy a vytvoriť platformu, poskytujúcu nový spôsob pre detekcie anomálií.

Prvým krokom pre riešenie tejto práce bolo zoznámiť sa s jazykom YARA a jeho využitím vo firme Avast, keďže pravidlá jazyka YARA slúžia ako primárny zdroj dát pre systém. Riešenie celkového systému bolo rozdelené na menšie úlohy, medzi ktoré patrilo okrem iného, zoznámenie sa so spôsobom posielania správ medzi jednotlivými systémami vo firme, preskúmanie time-series databáz alebo zoznámenie sa s možnosťami detekcie anomálií. Úspešne bol vytvorený systém plniaci viaceré funkcie pre prácu s dátami, tj. prijatie a spracovanie dát, ukladanie dát, vizualizácia požadovaných informácií a detekcia zmien. Základ celého systému spočívalo vo vytvorení time-series databázy a konzumenta správ, ktorý spracováva a ukladá aktuálne informácie o šírení malwaru do databázy. Tieto dáta sú ďalej pomocou API transformované na vhodnú formu reprezentujúcu požadovanú informáciu. Spracované dáta od API sú vizualizované vo forme grafov a tabuliek na webovej stránke. Na stránke nájde užívateľ napr. informácie o aktuálne najviac rozšírených malwarových rodinách, agregovanom počte hitov pre objekty, grafy zobrazujúce šírenie v čase a vzťah medzi behaviorálnymi a statickými pravidlami. Užívateľ sa dozvie taktiež informácie o rozložení šírenia v dňoch týždni alebo v systémoch. Na stránke sa okrem toho nachádza správa notifikačných kanálov slúžiacich pre detekciu anomálií. V rámci práce si užívateľ môže vybrať z 3 druhov detekcie anomálií a v rámci teoretickej časti boli preskúmané ďalšie metódy detekcie anomálnych hodnôt a zhodnotené možnosti využitia týchto metód. Bola overená rýchlosť detekcie anomálií, ale aj rýchlosť API a rýchlosť načítávania webového rozhrania. Rýchlosť detekcie anomálií sa dá označiť za dobrú. Rýchlosť API a webového rozhrania by sa dala pre niektoré prípady zlepšiť. Systém podstúpil aj užívateľské testovanie a bol zhodnotený firemným analytikom, ktorý mal za úlohu vyskúšať pripravené testovacie scenáre a odpovedať na otázky skúmajúce jeho reakciu. Pre jednoduchú správu a prenositeľnosť boli moduly nasadené cez kontajnery služby Docker.

Pre budúcu prácu by som videla možnosť zlepšenia modulu detekcie anomálií využitím, napr. jazyka R, spolu s preskúmaním zložitejších metód pre detekciu anomálií, medzi ktorými by mohli byť zaujímavé, okrem metód štatistických aj metódy založené na strojovom učení, napr. metódy založené na vzdialenosti alebo metódy založené na clusteroch. Detekcia anomálií by mohla byť rozšírená aj o iné prípady detekcie, ako sledovanie vzťahov medzi jednotlivými typmi pravidiel alebo sledovanie zmien naprieč viacerými systémami. Vhodné by bolo aj hlbšie preskúmanie charakteristiky dát a na základe veľkej variability medzi jed-

notlivými typmi objektov by bolo možné vytvoriť algoritmus založený na charakteristike štatistických parametrov pre výber správnej metódy pre detekciu anomálneho správania, pretože pravdepodobne neexistuje jedno uniformné riešenie, ktoré by vhodne detekovalo anomálie pre všetky datasety. Druhým možným typom skúmaného algoritmu by bola nápoveda pre nastavenie požadovaných parametrov pri detekcii pre optimálne výsledky pre jednotlivé datasety, na základe ich štatistických vlastností. Medzi budúcu dodatočnú funkcionalitu webovej stránky by bolo vhodné pridať filtrovanie podľa absolútneho času a ďalších grafov pre nasledujúce prípady: zobrazenie šírenia malwaru/pravidiel na základe systému, šírenie podľa typu malwaru, frekvencia šírenia v dňoch mesiaca alebo hodinách dňa, napr. v podobe heatmapy.

Literatúra

[1] *Emulator* [online]. Kaspersky [cit. 2022-03-14]. Dostupné z: https://www.kaspersky.com/enterprise-security/wiki-section/products/emulator.

[2] *Odpoveď na otázku na stránke stackoverflow: Alternative for a many to many relation between a hypertable and a 'normal' table* [online], 19. júna 2020 [cit. 2022-4-20]. Dostupné z: https://stackoverflow.com/questions/62451952/alternative-for-a-many-to-many-relation-between-a-hypertable-and-a-normal-tabl.

[3] *Oficiálna dokumentácia Cuckoo sandbox* [online]. 2020 [cit. 2021-12-18]. Dostupné z: https://cuckoo.readthedocs.io/en/latest/introduction/what/.

[4] *Oficiálna dokumentácia TimeScaleDB - Hypertables and chunks* [online]. TimeScaleDB, 2021 [cit. 2021-12-17]. Dostupné z: https://docs.timescale.com/timescaledb/latest/overview/core-concepts/hypertables-and-chunks/.

[5] *Oficiálna dokumentácia YARA* [online]. VirusTotal, 2021 [cit. 2021-12-5]. Dostupné z: https://yara.readthedocs.io/en/latest/index.html.

[6] *Oficiálna dokumentácia platformy Grafana* [online]. Grafana Labs, 2022 [cit. 2022-04-17]. Dostupné z: https://grafana.com/docs/.

[7] Ahmed, M., Naser Mahmood, A. a Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*. 2016, zv. 60, s. 19–31, [cit. 2021-12-16]. DOI: https://doi.org/10.1016/j.jnca.2015.11.016. ISSN 1084-8045. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1084804515002891.

[8] Blázquez García, A., Conde, A., Mori, U. a Lozano, J. A. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery. Apríl 2021, zv. 54, č. 3, [cit. 2022-04-19]. DOI: 10.1145/3444690. ISSN 0360-0300. Dostupné z: https://doi.org/10.1145/3444690.

[9] Braei, M. a Wagner, S. Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. [online]. arXiv. Apríl 2020, [cit. 2022-4-17]. DOI: 10.48550/ARXIV.2004.00433. Dostupné z: https://arxiv.org/abs/2004.00433.

[10] Chandola, V., Banerjee, A. a Kumar, V. Anomaly Detection: A Survey. *ACM Computing Surveys* [online]. New York, NY, USA: Association for Computing Machinery. Júl 2009, zv. 41, č. 3, s. 58, [cit. 2021-12-16]. DOI: 10.1145/1541880.1541882. ISSN 0360-0300. Dostupné z: https://doi.org/10.1145/1541880.1541882.

[11] Freedman, M. *13 tips to improve PostgreSQL Insert performance* [online]. TimeScale, jún 2020 [cit. 2021-12-21]. Dostupné z: https://blog.timescale.com/blog/13-tips-to-improve-postgresql-insert-performance/.

[12] Goldberg, I., Wagner, D., Thomas, R. a Brewer, E. *A Secure Environment for Untrusted Helper Applications* [online]. University of California, Berkeley, júl 1996 [cit. 2021-12-05]. Dostupné z: https://www.usenix.org/legacy/publications/library/proceedings/sec96/full_papers/goldberg/goldberg.pdf.

[13] Hahn, K. *Malware family naming hell is our own fault* [online]. gdatasoftware.com, júl 2021 [cit. 2021-12-03]. Dostupné z: https://www.gdatasoftware.com/blog/malware-family-naming-hell.

[14] Hanousek, J. a Charamza, P. *Moderní metody zpracování dat - matematická statistika pro každého*. 1. vyd. Praha: Grada a.s., 1992 [cit. 2022-04-19]. 216 s. ISBN 80-85623-31-5.

[15] Hayes, A. *Multiple Linear Regression (MLR)* [online]. Investopedia, 2. januára 2022 [cit. 2022-1-11]. Dostupné z: https://www.investopedia.com/terms/m/mlr.asp.

[16] Ijaz, M., Durad, M. H. a Ismail, M. Static and Dynamic Malware Analysis Using Machine Learning. In: *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)* [online]. 2019, s. 687–691 [cit. 2021-7-12]. DOI: 10.1109/IBCAST.2019.8667136. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8667136.

[17] Jamalpur, S., Navya, Y. S., Raja, P., Tagore, G. a Rao, G. R. K. Dynamic Malware Analysis Using Cuckoo Sandbox. In: *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)* [online]. 2018, s. 1056–1060 [cit. 2021-7-12]. DOI: 10.1109/ICICCT.2018.8473346. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8473346.

[18] Koret, J. a Bachaalany, E. *The Antivirus Hacker's Handbook*. 1. vyd. John Wiley & Sons, Inc., 2015. ISBN 978-1-119-02875-8.

[19] Kristína, M. *Nové trendy v strojovom učení – štatistický prístup*. 1. vyd. Košice: Technická univerzita v Košiciach, september 2016 [cit. 2022-01-11]. 96 s. ISBN 978-80-553-2602-3.

[20] Kulkarni, A. a Booz, R. *What the heck is time-series data (and why do I need a time-series database)?* [online]. TimeScaleDB, december 2020 [cit. 2021-12-10]. Dostupné z: https://blog.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/.

[21] Mareš, P. a Rabušic, L. *Normalita rozložení a z skóry; zobecňování výběrových výsledků na základní soubor* [online]. 2003 [cit. 2022-4-20]. Dostupné z: https://is.muni.cz/el/1423/podzim2004/SOC708/um/59381/SOC108_708PRIKLADY_DEMO03_normalnirozlozeni.pdf.

[22] Moser, A., Kruegel, C. a Kirda, E. Limits of Static Analysis for Malware Detection. In: *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)* [online]. 2007, s. 421–430 [cit. 2022-01-11]. DOI:

10.1109/ACSAC.2007.21. Dostupné z:
https://ieeexplore.ieee.org/document/4413008.

[23] Neubauer, J., Sedlačík, M. a Kříž, O. *Základy statistiky: Aplikace v technických a ekonomických oborech*. 2. vyd. Praha: Grada Publishing a.s., 2016 [cit. 2022-04-20]. 280 s. ISBN 978-80-247-5786-5.

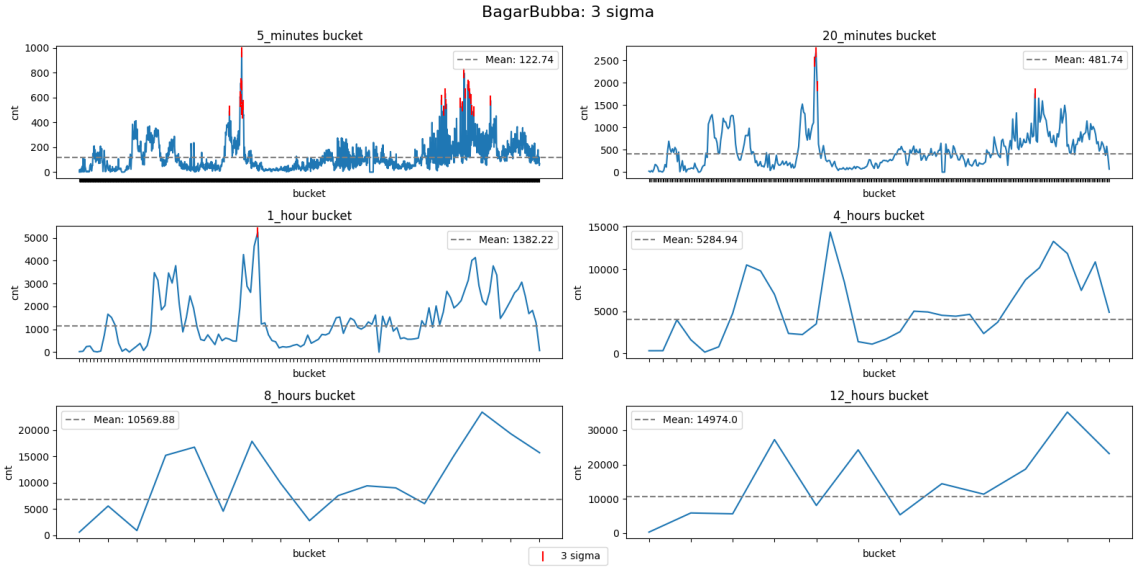
[24] Rafferty, M., Brogan, P., Hastings, J., Laverty, D., Liu, X. A. et al. Local Anomaly Detection by Application of Regression Analysis on PMU Data. In: *2018 IEEE Power Energy Society General Meeting (PESGM)* [online]. Portland, OR, USA: IEEE, 2018, s. 1–5 [cit. 2022-01-10]. DOI: 10.1109/PESGM.2018.8586320. Dostupné z: https://ieeexplore.ieee.org/document/8586320.

[25] Samariya, D. a Thakkar, A. A Comprehensive Survey of Anomaly Detection Algorithms. *Annals of Data Science*. November 2021, [cit. 2022-4-19]. DOI: 10.1007/s40745-021-00362-9. Dostupné z:
https://link.springer.com/content/pdf/10.1007/s40745-021-00362-9.pdf.

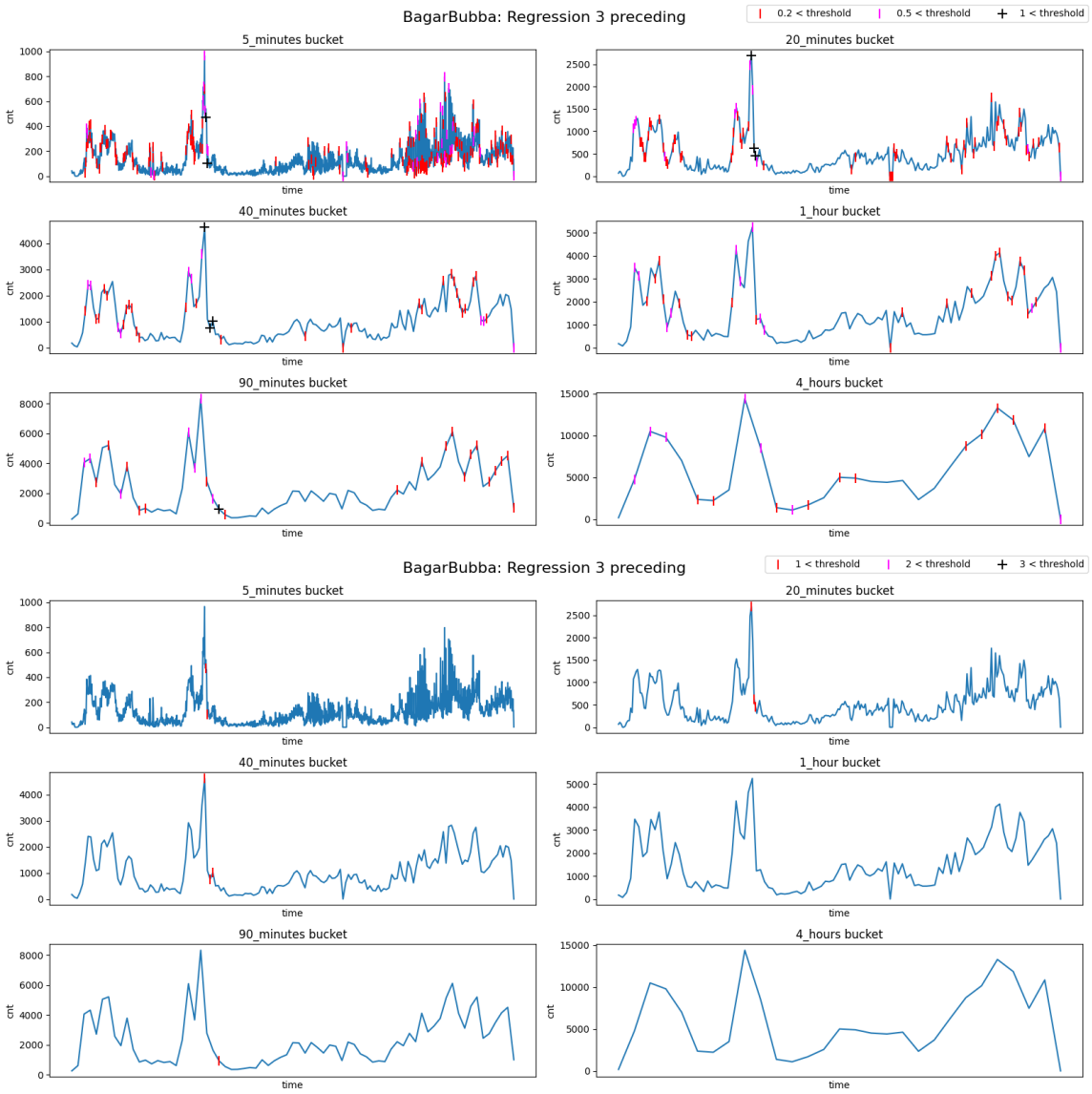
[26] Ször, P. *The Art of Computer Virus Research and Defense*. 1. vyd. Addison-Wesley Professional, 2005. ISBN 0-321-30454-3.

Príloha A

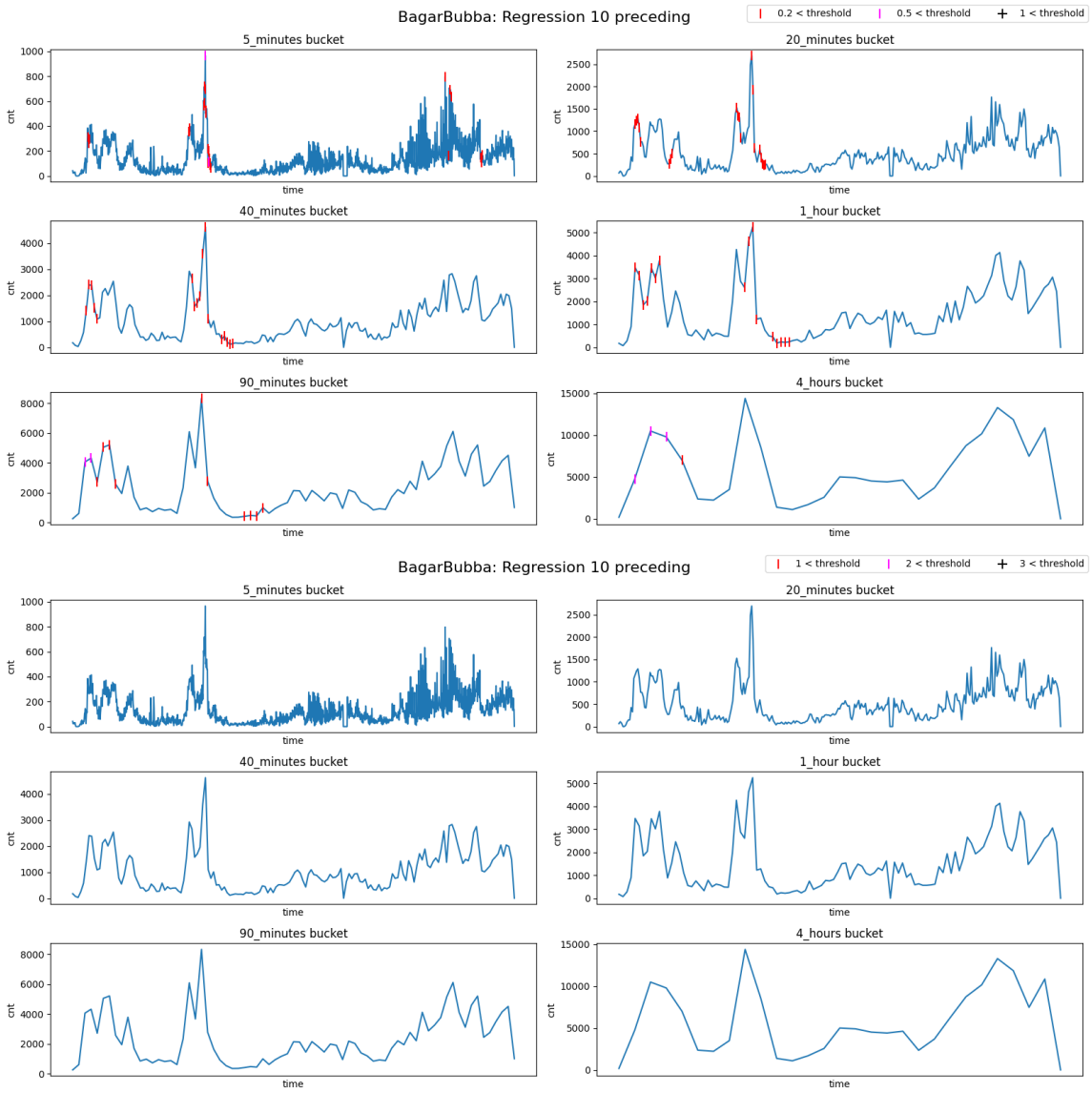
Výsledky experimentov



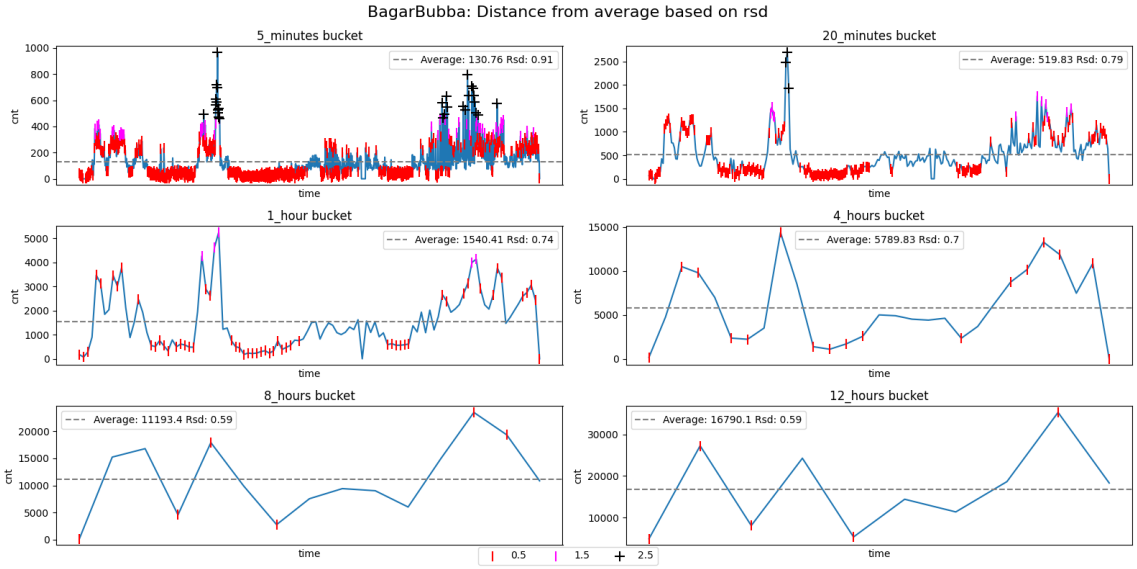
Obr. A.1: Metóda 3 sigma pre BagarBubba.



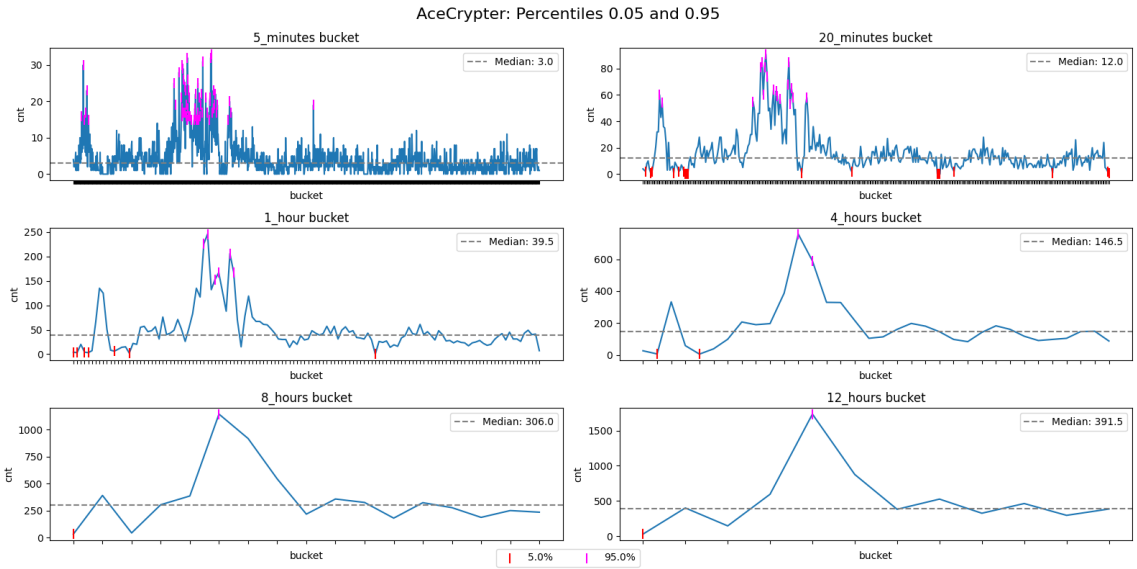
Obr. A.2: Metóda s využitím smernice regresnej priamky pre 3 časové okná pre rodinu BagarBubba.



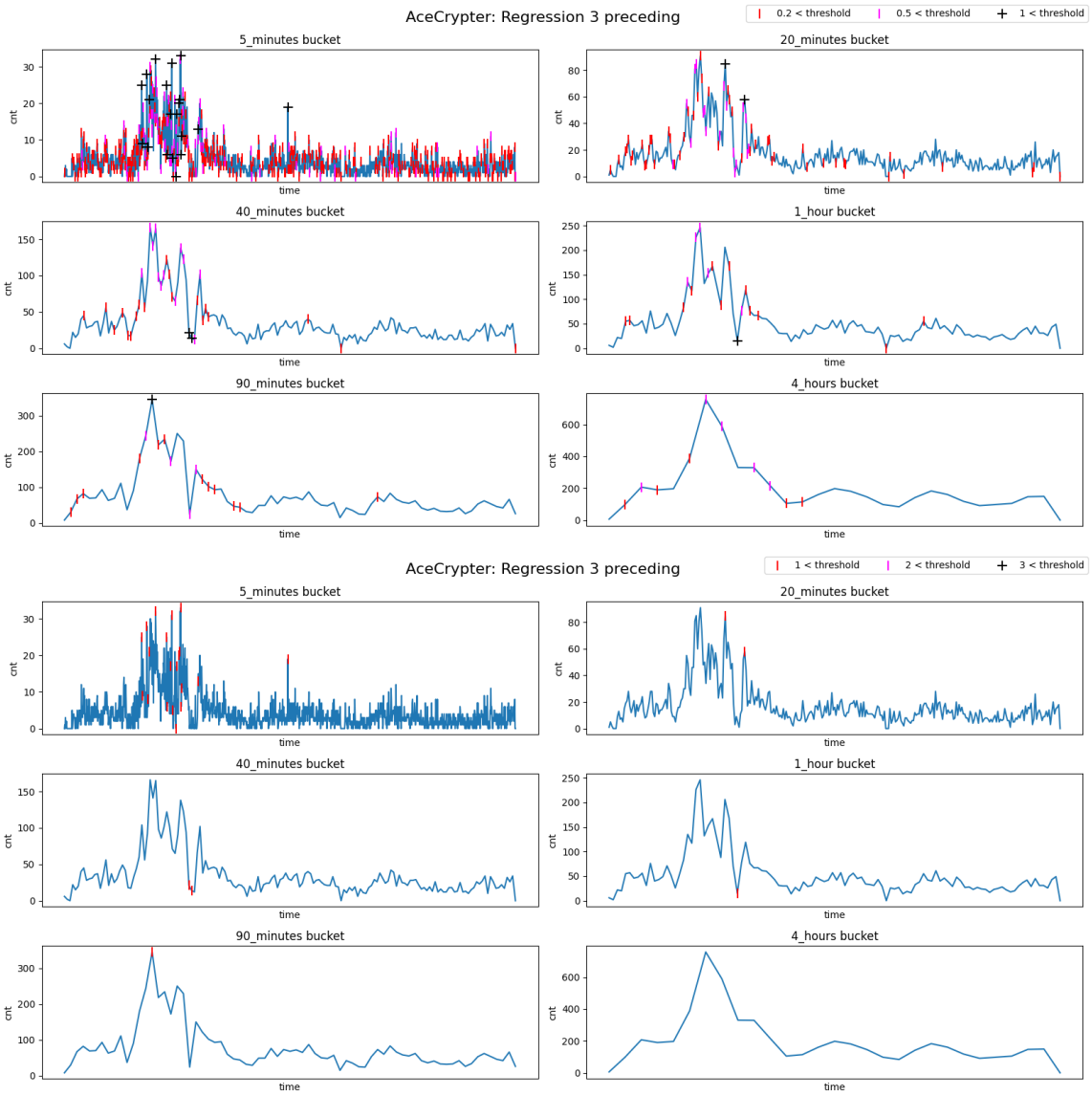
Obr. A.3: Metóda s využitím smernice regresnej priamky pre 10 časových okien pre rodinu BagarBubba.



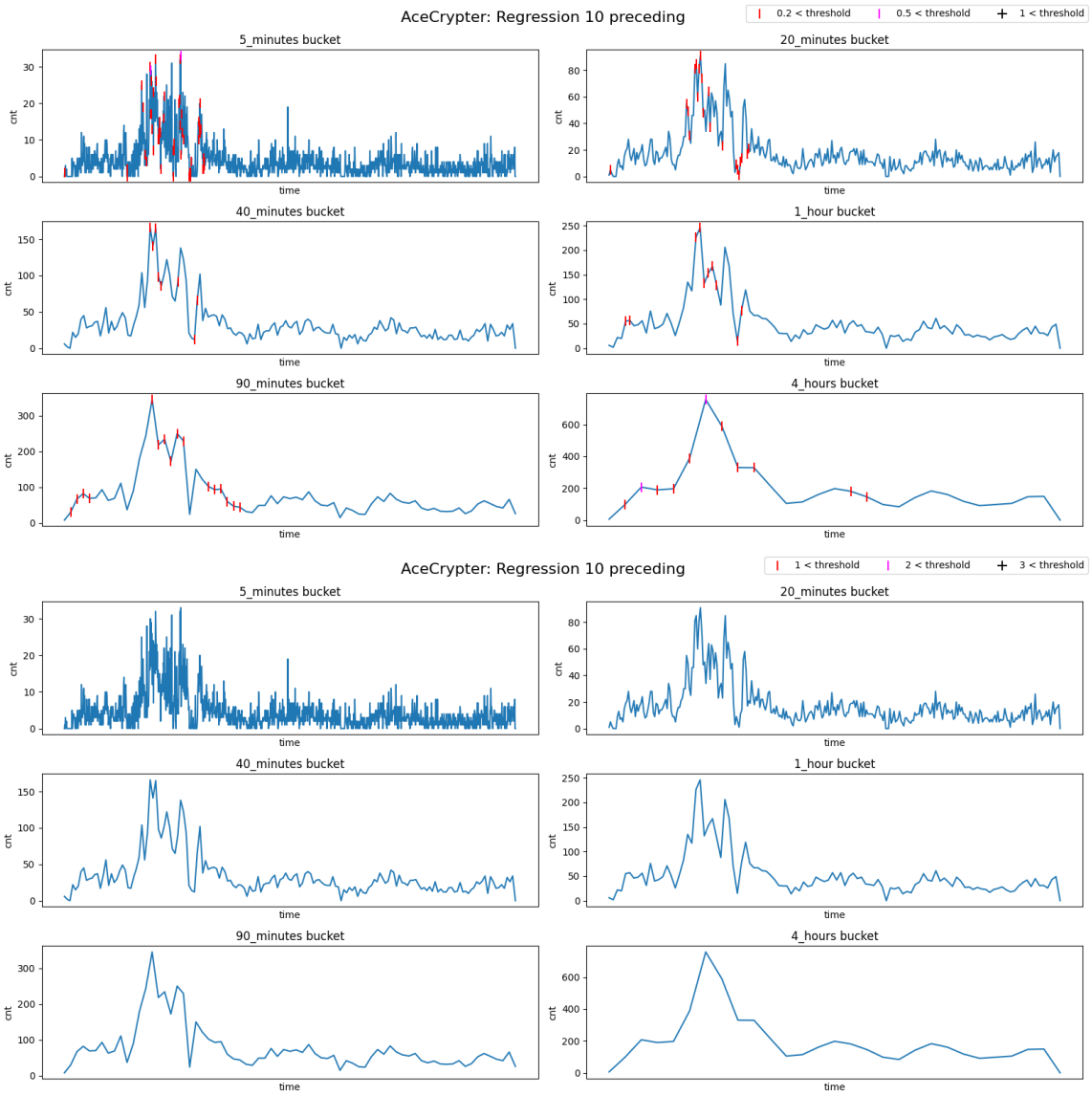
Obr. A.4: Metóda na základe vzdialenosti od priemeru pre rodinu BagarBubba pre hraničné hodnoty v 3 intervaloch: $0,5 < 1,5 < 2,5+$.



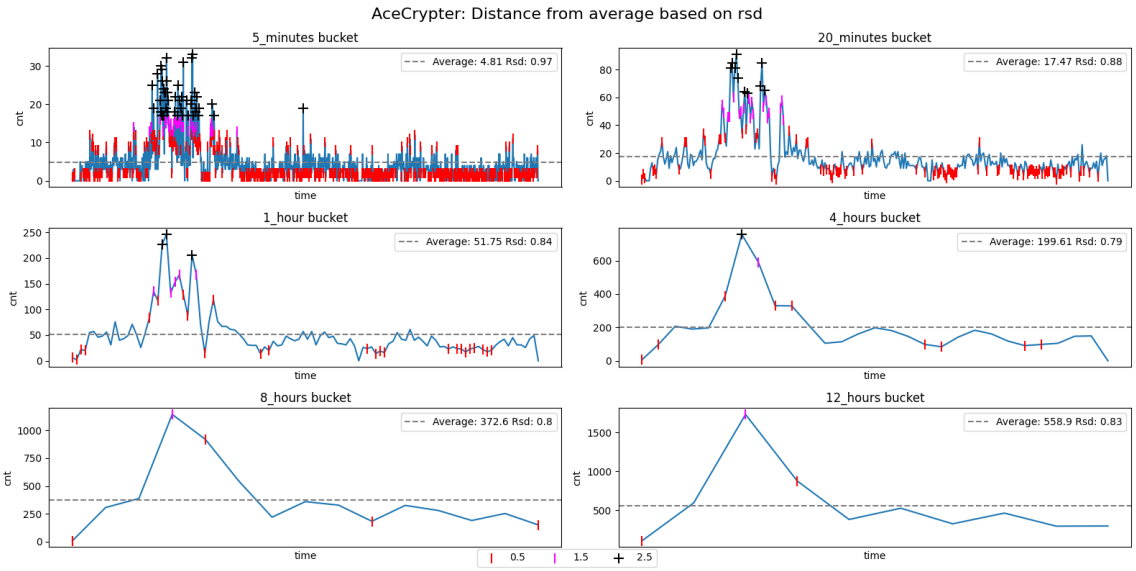
Obr. A.5: Metóda 5% a 95% percentilov pre rodinu AceCrypter.



Obr. A.6: Metóda s využitím smernice regresnej priamky pre 3 časové okná pre rodinu AceCrypter.



Obr. A.7: Metóda s využitím smernice regresnej priamky pre 10 časových okien pre rodinu AceCrypter.



Obr. A.8: Metóda na základe vzdialenosti od priemeru pre rodinu AceCrypter pre hraničné hodnoty v 3 intervaloch:$0, 5 < 1, 5 < 2, 5+$.

Príloha B

Dostupné API endpointy

- `GET: /api/strains/count` - agregovaný počet pre rodiny
- `GET: /api/rules/count` - agregovaný počet pre pravidlá
- `GET: /api/strains/min` - minimálny agregovaný počet pre rodiny
- `GET: /api/rules/min` - minimálny agregovaný počet pre pravidlá
- `GET: /api/systems/strains` - distribúcia v systémoch pre rodiny
- `GET: /api/stats/` - štatistické vlastnosti objektu
- `GET: /api/rules/distribution/` - distribúcia podľa typu pravidla
- `GET: /api/strains/malware` - distribúcia podľa typu malwaru
- `GET: /api/rules/category/` - distribúcia podľa kategórie malwaru
- `GET: /api/strains/weekdays/` - distribúcia rodiny v dňoch v týždni
- `GET: /api/systems/` - systémy uložené v databáze
- `GET: /api/{strain}/ruletypes` - šírenie typov pravidiel v čase pre rodinu
- `GET: /api/strains/` - šírenie rodín v čase
- `GET: /api/rules/` - šírenie pravidiel v čase
- `GET: /api/channels/` - uložené notifikačné kanále
- `GET: /api/channel/{id}/` - konkrétny notifikačný kanál
- `GET: /api/channel/{id}/last/` - posledná anomália notifikačného kanálu
- `GET: /api/objects/` - všetky registrované objekty v rámci detekcie
- `GET: /api/objects/{name}/` - konkrétny objekt na základe mena
- `GET: /api/objects/{id}/` - konkrétny objekt na základe id
- `GET: /api/anomalies/distance/` - vzdialenosť od priemeru

- `GET: /api/timebuckets/gapfill/` - početnosť v časových oknách s možnosťou určenia prahových hodnôt (vypĺňa prázne intervaly)
- `GET: /api/timebuckets/` - početnosť objektu v časových oknách (bez prázdnych intervalov)
- `GET: /api/channels/{id}/subscriptions/` - odberatelia pre daný kanál
- `GET: /api/channels/{id}/obj/` - sledovaný objekt pre kanál
- `GET: /api/object_types/` - typy objektov
- `GET: /api/anomaly_types/` - typy anomálií
- `PATCH: /api/channel/{id}/last/` - aktualizovanie poslednej anomálie konkrétneho notifikačného kanálu
- `POST: /api/channels/` - pridanie notifikačného kanálu
- `POST: /api/objects/` - pridanie nového objektu
- `POST: /api/channels/{id}/subscribe/` - pridanie nového odberateľa pre notifikačný kanál
- `POST: /api/channels/{id}/unsubscribe/` - odhlásenie odberateľa pre notifikačný kanál
- `POST: /api/channels/slackchannel/subscribe` - prihlásenie upozornení do dedikovaného slack kanálu

Príloha C

Konfiguračný súbor

```
{
    rabbitmq: {
        username = "user"
        password = "psw"
        host = "localhost"
        port = 5672
        virtual_host = "/"
    }

    database: {
        username = "username"
        password = "psw"
        host = "localhost"
        port = 8088
        database = "consumer_db"
        driver = "asyncpg"
    }

    test_database: {
        username = "test"
        password = "test"
        host = "172.20.0.1"
        port = 8087
        database = "test_db"
        driver = "asyncpg"
    }

    logLevel: {level = DEBUG }

    slack: {bot_token = "xoxb-327140xxxxxx" }
}
```

Výpis C.1: Ukážka štruktúry konfiguračného súboru

Príloha D

Obsah přiloženého pamäťového média

- `docs/` - technická správa vo formáte pdf a zdrojové kódy technickej správy (LaTeX)
- `src/` - zdrojové kódy programu
  - `src/local/` - zdrojové kódy pre lokálne spustenie. Manuál sa nachádza v prílohe E.
  - `src/production/` - zdrojové kódy pre spustenie na firemnom servery. Neobsahujú konfiguračný súbor.
- `experiments/` - grafické výstupy pre metódy detekcie anomálií

Príloha E

Manuál pre lokálne spustenie

Súbory v priečinku `src/local/` obsahujú lokálnu upravenú verziu. V tejto verzii je vytvorená trieda `FakeRabbitMQ` nachádzajúca sa v súbore `fake_rabbitmq/rabbitmq.py`, ktorá simuluje tvorbu dát v náhodnom časovom intervale. Trieda `Consumer` sa z tohto dôvodu odlišuje od triedy v priečinku `/production/`, kde sa využívajú dáta z firemného serveru RabbitMQ. Medzi druhý rozdiel medzi lokálnou a produkčnou verziou patrí funkcionalita detekcie anomálií. Pretože SlackBot je viazaný na workspace vytvorený na platforme Slack, v lokálnej verzii by táto aplikácia prinášala prínos iba v prípade inštalácie SlackBota na platforme Slack a zmene tokenu v konfiguračnom súbore, a preto je posielanie upozornení nahradené výpisom na stdout a detekované anomálie sú zapisované do dedikovaného súboru `anomalyDetection.log`.

Súbory v priečinku `src/production/` obsahujú produkčnú verziu, ktorá funguje iba s internými údajmi firmy Avast. Pre túto verziu chýba konfiguračný súbor.

V zložke `src/local/` sa nachádza:

- `docker-compose.yml`
- `images/` s dopredu vytvorenými docker images pre jednoduché spustenie.

```
Načítanie priložených docker images:

docker load -i images/image.tar

Spustenie všetkých aplikácií:

docker-compose up

Zastavenie všetkých aplikácií:

docker-compose down
```