UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

PREDIKCIA SITUAČNÉHO POVEDOMIA V KYBERNETICKEJ BEZPEČNOSTI POMOCOU METÓD STROJOVÉHO UČENIA

DIPLOMOVÁ PRÁCA

2024 **Bc. Jakub MOHLER**

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

PREDIKCIA SITUAČNÉHO POVEDOMIA V KYBERNETICKEJ BEZPEČNOSTI POMOCOU METÓD STROJOVÉHO UČENIA

DIPLOMOVÁ PRÁCA

Študijný program:	Informatika
Pracovisko (katedra/ústav):	ÚINF - Ústav informatiky
Vedúci práce ŠVK:	doc. JUDr. RNDr. Pavol Sokol, PhD.
Konzultant:	RNDr. Richard Staňa

Košice 2024

Bc. Jakub MOHLER

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Jakub Mohler
Študijný program:	informatika (jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	18. - informatika
Typ záverečnej práce:	Diplomová práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Predikcia situačného povedomia v kybernetickej bezpečnosti pomocou metód strojového učenia

Cieľ: 1) Vytvoriť dátovú sadu časových radov pre analýzu situačného povedomia v kybernetickej bezpečnosti.
2) Preskúmať existujúce metódy strojového učenia na predikciu situačného povedomia v kybernetickej bezpečnosti.
3) Navrhnúť model pre predikciu situačného povedomia v kybernetickej bezpečnosti a vyhodnotiť úspešnosť tohto modelu s existujúcimi štatistickými metódami.
4) Navrhnúť a implementovať systém na interaktívnu grafickú reprezentáciu jednokrokových a viackrokových predikcií.

Literatúra: 1) Husák, M., Jirsík, T., & Yang, S. J. (2020, August). SoK: contemporary issues and challenges to enable cyber situational awareness for network security. In Proceedings of the 15th International Conference on Availability, Reliability and Security (pp. 1-10).
2) Husák M, Komárková J, Bou-Harb E, Čeleda P. Survey of attack projection, prediction, and forecasting in cyber security. IEEE Communications Surveys & Tutorials. 2018 Sep 24;21(1):640-60.
3) Stana, R., Patrik, P., Gajdos, A., Pavol, S.: Network security situation awareness forecasting based on neural networks. 8th International conference on Time Series and Forecasting.

Kľúčové slová: Kybernetická bezpečnosť, situačné povedomie, honeypot, časové rady, strojové učenie

Vedúci:	doc. RNDr. JUDr. Pavol Sokol, PhD.
Konzultant:	RNDr. Richard Staňa
Oponent:	RNDr. Tomáš Bajtoš
Ústav:	ÚINF - Ústav informatiky
Riaditeľ ústavu:	doc. RNDr. Ondrej Krídlo, PhD.
Dátum zadania:	14.10.2022

Dátum schválenia: 30.04.2024

doc. RNDr. Ondrej Krídlo, PhD.
riaditeľ ústavu

Poďakovanie

Týmto sa chcem poďakovať vedúcemu svojej práce doc. RNDr. JUDr. Pavlovi Sokolovi, PhD., konzultantovi RNDr. Richardovi Staňovi a kolegovi Mgr. Patrikovi Pekarčíkovi za odborné vedenie, cenné rady a veľkú pomoc počas tvorby práce.

Abstrakt v štátnom jazyku

V tejto práci sa venujeme situačnému povedomiu, ktoré v oblasti kybernetickej bezpečnosti zohráva kľúčovú úlohu v identifikácii bezpečnostných hrozieb v dynamickom digitálnom prostredí. Cieľom predikcie situačného povedomia je lepšie predvídať a reagovať na kybernetické bezpečnostné hrozby. Pre sledovanie situačného povedomia môžeme využívať rôzne systémy, jedným z príkladov sú honeypoty a honeynety. Ich úlohou je napodobňovať reálny systém, a tým prilákať útočníkov a umožniť im vykonávanie útokov. Zaznamenávaním ich aktivít získame cenné informácie o aktuálne využívaných taktikách a technikách, čo nám následne umožňuje to reprezentovať ako situačné povedomie. Našim cieľom je pripraviť dataset časových radov obsahujúci dáta zozbierané z honeypotov a honeynetov čo zahŕňa analýzu vytvorených datasetov a výber relevantných parametrov, ktoré budeme skúmať. Kľúčovým prínosom našej práce je implementácia a vyhodnotenie predikčných modelov strojového učenia ako napríklad SVM, rozhodovacích stromov, LSTM a GRU. Naša práca zahŕňa aj implementáciu systému pre predikciu vývoja situačného povedomia vrátane grafického rozhrania, ktoré umožní interaktívne preskúmať získané dáta a identifikovať potenciálne bezpečnostné hrozby na základe predikcií, čím posilňuje obranné mechanizmy v kybernetickej bezpečnosti. Tento prístup zvyšuje situačné povedomie nie len o aktuálnych ale aj o budúcich hrozbách a tým otvára nové možnosti pre rozvoj pokročilých bezpečnostných nástrojov.

Kybernetická bezpečnosť, situačné povedomie, honeypot, časové rady, strojové učenie

Abstrakt v cudzom jazyku

In this paper, we discuss situational awareness, which in cybersecurity plays a key role in identifying threats in a dynamic digital environment. The goal of situational awareness prediction is to better anticipate and respond to cyber threats. We can use various systems to monitor situational awareness, one example is honeypots and honeynets. Their role is to mimic a real system and thus attract attackers and enable them to carry out attacks. By recording their activities, we gain valuable information about the tactics and techniques currently used, which then allows us to represent this as situational awareness. Our goal is to prepare a time series dataset containing data collected from honeypots and honeynets which involves analyzing the created datasets and selecting relevant parameters to investigate. A key contribution of our work is the implementation and evaluation of machine learning prediction models such as SVM, decision trees, LSTM and GRU. Our work also includes the implementation of a system for predicting the evolution of situational awareness, including a graphical interface to interactively explore the collected data and identify potential security threats based on the predictions, thus strengthening the defense mechanisms in cybersecurity. This approach increases situational awareness not only of current but also of future threats and thus opens up new opportunities for the development of advanced security tools.

Cybersecurity, situational awareness, honeypot, time series, machine learning

Obsah

Slovník termínov

Honeypot je bezpečnostný nástroj, ktorého hodnota spočíva v tom, že je skúmaný, napadnutý alebo kompromitovaný.

Honeynet je sieť honeypotov.

Threat hunting je metóda proaktívneho vyhľadávania hrozieb.

Zoznam ilustrácií

Úvod

Využitie nových technológií so sebou prináša nové typy bezpečnostných hrozieb a bezpečnostných incidentov. Ich počet neustále rastie. Z tohto dôvodu sa im musíme vedieť efektívne brániť. Kybernetická bezpečnosť sa stáva stále viac dôležitou, a preto organizácie investujú peniaze na ochranu svojej infraštruktúry.

S nárastom komplexity a objemu kybernetických hrozieb je nevyhnutné využívať pokročilé techniky a nástroje na identifikáciu a predvídanie potenciálnych bezpečnostných incidentov. V tomto kontexte sa metódy strojového učenia a neurónových sietí stávajú dôležitým nástrojom pre predikciu situačného povedomia v oblasti kybernetickej bezpečnosti. Tieto metódy dokážu analyzovať rozsiahle množstvo údajov, identifikovať vzory a vzťahy medzi dátami a následne generovať predpovede o možných hrozbách. Vďaka strojovému učeniu a neurónovým sieťam sa môžeme lepšie pripraviť na budúce útoky, zvýšiť efektivitu detekcie hrozieb a reagovať rýchlejšie a presnejšie na bezpečnostné incidenty.

Klasický prístup k tejto problematike bol postavený na základe reaktívnych činností, avšak s vývojom kybernetických hrozieb je stále jasnejšie, že reaktívny prístup k bezpečnosti už nie je postačujúci. Súčasným trendom je prechod od reaktívnych činností (riešenie, resp. koordinovanie bezpečnostného incidentu) k proaktívnym činnostiam (aktívne vyhľadávanie zraniteľností, tzv. threat hunting). Hlavným dôvodom tejto zmeny je často sa meniace správanie útočníkov a taktiež nové metódy a používané nástroje. Jedným zo spôsobov proaktívnych činností je spomínaný threat hunting.

Threat hunting znamená, že je potrebné aktívne skúmať prostredie, aby sa identifikovali potencionálne slabiny, ktoré by mohli byť zneužité útočníkmi. Ďalším spôsobom je zavedenie honeypotov a honeynetov do infraštruktúry. Sú to podvodné systémy, ktoré sú vytvorené s cieľom simulovať reálne prostredie, ktoré by mohlo byť cieľom útoku. Tieto dáta poskytujú jedinečný pohľad na taktiky útočníkov a umožňujú identifikovať nové hrozby. Teda hlavným cieľom použitia honeypotov a honeynetov je snažiť sa už vopred pripraviť na bezpečnostné hrozby, ktorým čelíme resp. môžeme čeliť v budúcnosti a v najlepšom prípade úplne zabrániť vzniku bezpečnostného incidentu [1].

Táto práca sa bližšie venuje problematike predikcie situačného povedomia, ktoré Mica Endsley definovala ako „vnímanie prvkov v prostredí v rámci času a priestoru, pochopenie ich významu a projekcia ich stavu v blízkej budúcnosti“ [27]. Toto situačné

povedomie pozostáva z vnímania, ktoré predstavuje zber relevantných údajov. Druhou časťou je porozumenie, ktoré znamená pochopenie situačného povedomia na základne zozbieraných údajov. Napokon tretiu časť tvorí predikcia situačného povedomia, čo predstavuje predpoveď toho, ako sa dané povedomie bude vyvíjať. Bližšie sa situačnému povedomiu budeme venovať v druhej kapitole tejto záverečnej práce.

Ako sme už vyššie uviedli, práca sa venuje problematike predikcie situačného povedomia. Ako základný zdroj bezpečnostných údajov, z ktorých toto povedomie vychádza, sú už spomínané podvodné systémy (honeypoty a honeynety). Tým sa bližšie venujeme v prvej kapitole a priblížime výhody použitia týchto systémov, typy honeypotov a koncept honeynetu. Pre analýzu situačného povedomia využívame metódy strojového učenia, ktoré porovnávame s metódami neurónových sietí. Bližšie sa situačným povedomím a možnosti jeho chápania zaoberáme v druhej kapitole tejto práce. Napokon súčasťou práce je návrh a implementácia riešenia, ktoré automaticky spracuje dáta z reálneho honeynetu, spracuje tieto dáta, pripraví časové rady a aplikuje rôzne modely. Návrhom a implementáciou riešenia sa venujeme v tretej kapitole.

1 Honeypoty a honeynety

Pôvodný termín honeypot, resp. honey trap pochádza zo špionážnej terminológie. Zahŕňa to použitie tajného agenta na vytvorenie situácie vhodnej na kompromitáciu cieľa [2]. Podobný význam má pojem honeypot aj v terminológii kybernetickej bezpečnosti.

Honeypot je možné definovať viacerými spôsobmi, či už od pohľadu, akým naň pozeráme alebo od účelu, aký má plniť. V roku 2002 Lance Spitzner definoval honeypot ako „bezpečnostný zdroj, ktorého hodnota spočíva v tom, že je skúmaný, napadnutý alebo kompromitovaný“ [3]. Podľa spoločnosti Kaspersky je honeypot „obetný počítačový systém, ktorý má ako návnada prilákať kybernetické útoky. Napodobňuje cieľ pre útočníkov a využíva ich pokusy o preniknutie na získanie informácií o útočníkoch a spôsoboch, akými operujú alebo odpútava ich pozornosť od iných cieľov“ [4].

Honeypot je počítačový systém, ktorý je umiestnený v sieti, aby naňho bolo útočené a tým získaval dáta o útočníkoch a ich postupoch. Navonok pôsobí ako akýkoľvek iný legitímny systém s operačným systémom, ktorý obsahuje priečinky, súbory, webové služby a iné, ale jeho motív, o ktorom útočníci nevedia je iný [5]. Honeypoty majú viaceré úlohy. Jednou z úloh môže byť chránenie iného zariadenia alebo systému pripojeného v sieti a to odpútaním pozornosti od nich. Taktiež môžu poskytovať skoré upozornenie o novom typu útoku a trendoch [6]. Nové typy útokov, ako napríklad malvér, môžeme následne využiť pri implementácii proaktívnych opatrení v antimalvérovom riešení pridaním jeho signatúr alebo digitálneho odtlačku do databázy. Z výskumného hľadiska nám poskytujú množstvo informácií a dát a možnosť hĺbkovej analýzy útočníkov a ich postupov počas a po zneužití honeypotu. Taktiež sa na základe týchto dát môžeme pokúsiť vytvoriť rôzne predikčné modely, pre vytváranie predikcií.

1.1 Výhody využitia honeypotov

Využitie honeypotov prichádza s množstvom výhod, ktoré si popíšeme v tejto časti [5].

- Honeypoty sú umiestnené v sieti iba na zaznamenávanie pokusov o útokoch a následnému ukladaniu týchto udalostí. Na základe týchto udalostí sa vieme dozvedieť o aktuálnych taktikách, technikách a nástrojoch používaných útočníkmi.

- Každá aktivita je vykonaná útočníkom. Keďže ide o návnadu, bežný používateľ sa nemá ako dostať k tomuto zariadeniu. Vďaka tomu sa vyhneme jednej z najzložitejších úloh, a to filtrovaniu, či ide o aktivitu útočníka alebo legitímnu aktivitu bežného používateľa.
- Nízke množstvo flase-positive záznamov. Táto vlastnosť je výsledkom predchádzajúceho bodu.
- Nie sú náročné na infraštruktúru. Ich jedinou úlohou je zaznamenávať sieťovú prevádzku, resp. prichádzajúce útoky.
- Taktiež nie sú náročné na inštaláciu a konfiguráciu.
- Poskytujú nám informácie o nových taktikách, technikách a nástrojoch používaných útočníkmi. Vďaka týmto informáciám sa môžeme lepšie pripraviť na nové trendy v útokoch.

Získané dáta môžu byť následne využité na hlbšiu dátovú analýzu, resp. v našom prípade trénovanie modelov strojového učenia a neurónových sietí. Tieto modely môžu byť navrhnuté tak aby analyzovali a predikovali potencionálne hrozby na základe vzorov, ktoré boli identifikované v týchto dátach.

1.2 Typy honeypotov

Honeypoty sa delia do viacerých skupín na základe rôznych sledovaných parametrov. Rôzne typy honeypotov môžu byť použité na identifikovanie rôznych typov hrozieb a útokov [4]. Jedným zo spôsobov klasifikácie honeypotov je na základe úrovni interakcie. Sú rozdelené do troch skupín, a to s nízkou, strednou a vysokou interakciou. Ďalším spôsobom klasifikácie honeypotov je rozdeliť ich podľa role akú spĺňajú. Podľa tohto delenia sú rozdelené na honeypoty na strane servera a klienta [7]. Posledným spôsobom delenia, ktorým sa budeme zaoberať je delenie podľa dát ktoré zbierajú, resp. typov útokov, ktoré sa na nich vykonávajú [4]. V nasledujúcich podkapitolách sa detailnejšie pozrieme na všetky spomínané rozdelenia typov honeypotov.

1.2.1 Rozdelenie podľa úrovne interakcie

Delenie honeypotov podľa úrovne interakcie s útočníkom je určené na základe možností, ktoré sú útočníkovi poskytnuté na danom honeypote. Je určené rozsahom možností akcií, ktoré sú útočníkovi povolené honeypotom vykonávať.

Honeypoty s nízkou úrovňou interakcie sú systémy, ktoré sa snažia napodobniť základne charakteristiky zariadenia s operačným systémom a bežiacimi službami. Toto je uskutočnené sprístupnením niekoľkých webových služieb na hosťovskom operačnom systéme. Nevýhodou je, že tento prístup neposkytuje útočníkovi celý operačný systém a teda nevieme získať kompletné informácie o postupoch útočníka. Príkladom pre tento typ honeypotov je Dionaea, ktorý zaznamenáva payloady útokov a malware [5, 7].

Honeypoty so strednou úrovňou interakcie poskytujú o niečo viac možností pre útočníka ako tie s nízkou úrovňou interakcie ale zároveň mu neposkytujú celý operačný systém so všetkými bežiacimi službami. Tieto honeypoty dokážu na isté aktivity generovať nejakú formu odpovede na rozdiel od honeypotov s nízkou úrovňou interakcie. Príkladom pre tento typ honeypotov je SSH honeypot Kippo, ktorý zaznamenáva útoky hrubou silou na heslá a celkovú interakciu so shell-om [7].

Honeypoty s vysokou úrovňou interakcie poskytujú úplný operačný systém so všetkými bežiacimi službami. Povolením prístupu útočníkovi k celému operačnému systému a všetkým jeho službám mu dovolí vykonať komplexné útoky použitím „všetkého čo má k dispozícii“. Výhodou tohto prístupu je získanie väčšieho množstva informácií o útočníkovi a jeho postupoch pri útokoch a nástrojoch, ktoré používa [5, 8].

V práci budeme využívať primárne honeypoty s nízkou a strednou úrovňou interakcie, ktoré poskytujú vhodnejšie údaje na učenie modelov strojového učenia a neurónových sietí a následnú predikciu.

1.2.2 Rozdelenie podľa role

Na honeypoty sa môžeme pozerať z pohľadu role akú majú. V tomto prípade sa to delí do dvoch kategórií, honeypoty na strane servera a klienta. Sú to dve rôzne stratégie nasadzovania honeypotov do infraštruktúry. Toto rozdelenie nám umožňuje využívať dve odlišné prístupy na detekciu a analýzu útokov.

Honeypoty na strane **servera (server-side honeypots)** sú nasadzované na serveroch alebo v sieťovej infraštruktúre. Ich hlavným cieľom je zachytávanie a analýza

útokov, ktoré sú zamerané na samotné servery alebo sieťovú infraštruktúru. Tieto útoky môžu zahŕňať pokusy o neoprávnený prístup, exploitovanie zraniteľností serverových aplikácií poprípade iné pokusy o útoky na serverovú a sieťovú infraštruktúru. Vďaka nim môžeme detegovať nové exploity, zbierať malvér a obohacovať výskum o množstvo dát o hrozbách, taktikách, technikách a nástrojoch, ktoré útočníci používajú.

Na druhej strane **klientske honeypoty (client-side honeypots)** sú zariadenia, ktoré napodobňujú správanie klientskych zariadení pri komunikácii so serverom. Hlavným cieľom klientskych honeypotov je zaznamenávať či dochádza ku útokom v sieti [7]. Tieto honeypoty sa sústreďujú na detekciu škodlivých aktivít, ktoré by mohli byť zamerané na koncové zariadenia. Môže sa jednať o rôzne formy útokov ako napríklad phishing.

V práci budeme pracovať s oboma stratégiami nasadenia honeypotov, teda na strane servera aj na strane klienta.

1.2.3 Rozdelenie podľa zbieraných dát a typov útokov

Definície rôznych typov honeypotov závisia od typu dát, ktoré zbierajú. Dáta sú založené na základe hrozieb, ktoré adresujú, tzn. niektoré môžu zbierať dáta o IP adresách útočníkov, iné zaznamenávať vykonané príkazy. Nižšie sú popísané niektoré z najčastejšie používaných typov honeypotov podľa zbieraných dát a ich potencionálny prínos.

Emailové (spamové) pasce sú zamerané na zachytávanie nežiadúcej pošty. Princíp emailových pascí je založený na vytvorení falošných emailových adries, ktoré sú ukryté tak, aby ich dokázali objaviť iba automatizované harvestery. Keďže adresy nie sú používané na bežné účely, tak akákoľvek komunikácia na tieto emailové adresy je považovaná za potencionálne škodlivú a berieme ju, že je iniciovaná útočníkom. Na základe zozbieraných správ, IP adries z ktorých boli odoslané a emailových adries útočníkov sa následne môžu zaviesť opatrenia na blokovanie emailov s podobným obsahom alebo s takýmito adresami odosielateľa [9]. Takýmto spôsobom sa dá napríklad obmedziť určitý počet phisingových kampaní.

Návnadová (decoy) databáza je databáza, ktorá môže útočníkom poskytovať možnosť vykonať útoky typu SQL injection alebo exploitácie SQL služieb [10]. Získané informácie môžu slúžiť na identifikáciu zraniteľností v systéme.

Malvérový honeypot napodobňuje aplikácie a API služby, ktoré môžu byť cieľom doručenia malvéru. Získané vzorky malvéru umožňujú dopĺňanie a vytváranie nových pravidiel pre anti-malvérové riešenia, ktoré sú založené na signatúrach a digitálnych odtlačkoch malvérov [10]. To pomôže organizáciám lepšie detegovať a reagovať na nové varianty malvéru.

Spider honeypot je typ honeypotu, ktorý napodobňuje webové stránky, ku ktorým sa dokážu dostať iba web crawlery. Detegovaním webových crawlerov je možné získať informácie o automatizovaných nástrojoch, ktoré skenujú internet. Získané dáta môžu byť následne využité na naučenie sa blokovať automatizovaných botov [4].

V práci sa budeme venovať najmä honeypotom, ktoré zbierajú informácie o IP adresách útočníkov.

1.3 Honeynet

Ako názov napovedá, honeynet je prepojenie viacerých honeypotov do siete. Honeynet rozširuje koncept honeypotu do vysoko kontrolovanej siete honeypotov. V tomto smere môžeme definovať virtuálny honeynet ako celkový honeynet bežiaci na jednom počítači vo virtuálnom prostredí [8].

Honeynet je špecializovaná sieťová architektúra nakonfigurovaná tak, aby dosiahla nasledujúce časti [7]:

- **Kontrola dát** – zaoberá sa obmedzením činností v honeynete.
- **Zachytávanie dát** – zahŕňa zachytávanie, monitorovanie a zaznamenávanie všetkých hrozieb a aktivít útočníkov v rámci honeynetu.
- **Zber dát** – zachytené údaje sa musia bezpečne posielať do centrálneho úložiska.

Takáto architektúra vytvára vysoko kontrolovateľnú sieť, v ktorej sa dajú kontrolovať, zaznamenávať a uchovávať všetky systémové a sieťové aktivity [8]. Virtuálny honeynet je efektívny spôsob ako analyzovať správanie útočníkov v simulovanom, ale kontrolovanom prostredí.

Jednou z najpoužívanejších služieb, ktorá ponúka množstvo honeypotov na jednom mieste je T-Pot [11].

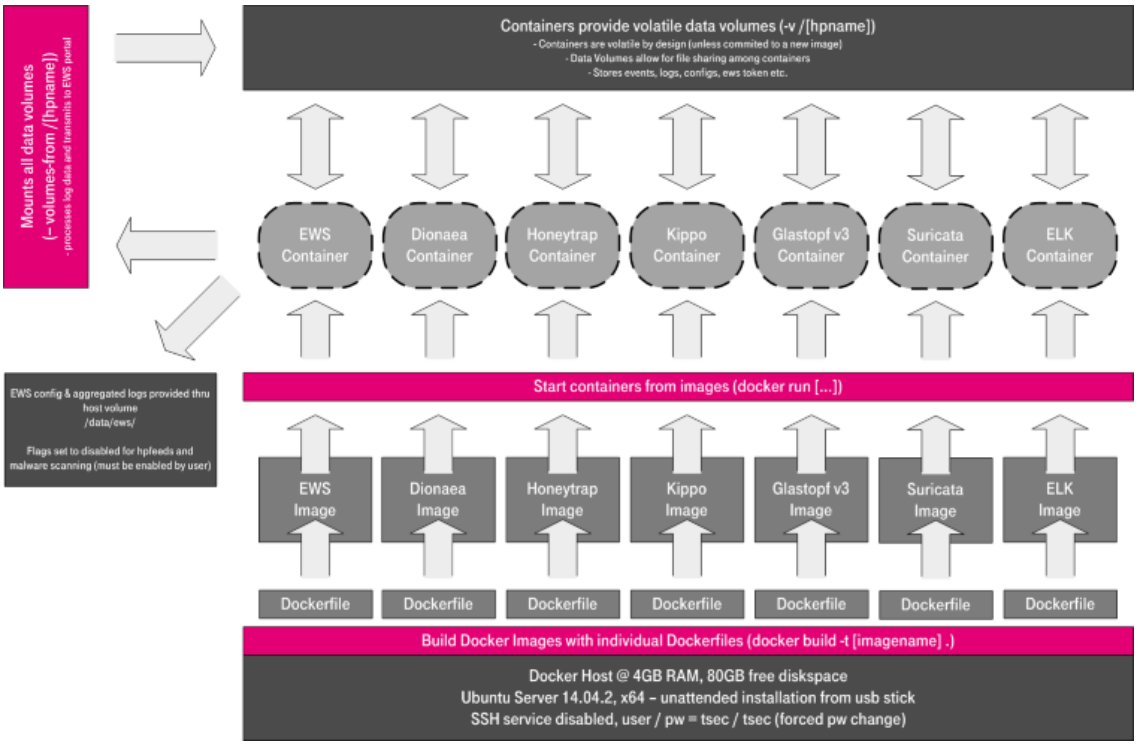
1.3.1 Honeynet T-Pot

T-Pot je multiplatformová služba, ktorá poskytuje celý honeynet s vizualizáciou dát pomocou Elastic Stack-u a množstvom zobrazovacích funkcií [11]. Podporuje viac ako 20 typov honeypotov.

Architektúra T-Potu, je založená na linuxovej distribúcii Debian 11. Pre beh takého množstva funkcií (honeypotov, atď.) súčasne využíva Docker [12] pre beh kontajnerov. T-Pot je založený na koncepte honeypotov a využíva kombináciu rôznych technológií na zhromažďovanie informácií o útokoch a na ich analýzu. Architektúra ako taká pozostáva zo 4 častí [11]:

- **Honeypoty** – sú použité rôzne honeypoty, ktoré zaznamenávajú a monitorujú všetky prichádzajúce útoky a interakcie s nimi.
- **Honeynet kontrolér –** je centrálny komponent, ktorý spravuje a koordinuje všetky honeypoty v rámci siete. Zabezpečuje konfiguráciu, monitorovanie a riadenie honeypotov.
- **Dátové úložisko** – všetky zozbierané dáta o útokoch, interakciách a iných bezpečnostných udalostiach sú ukladané do centrálneho databázového úložiska.
- **Analýza a vizualizácia** – poskytuje nástroje na analýzu a vizualizáciu zozbieraných údajov.

Na Obr. 1 je zobrazená časť tejto architektúry. Honeypoty sú uložené ako Dockerfile súbory, ktoré sú následne použité na vytvorenie ich obrazov. Použitím obrazov honeypotov sa štartujú kontajnery, ktoré následne operujú s hlavným kontrolerom.



Obr. 1 Časť architektúry TPOT-u [11]

T-Pot ponúka niekoľko služieb, ktoré sú rozdelené do 5 kategórií [11]:

- **Systémové služby** poskytované operačným systémom. Sú to základné služby ako napríklad SSH pre bezpečný vzdialený prístup alebo Cockpit pre webový vzdialený prístup a webový terminál.
- **Elastic Stack** zahŕňa tri základné služby: elasticsearch, logstash a kibana. **Elasticsearch** je databáza, do ktorej sa ukladajú všetky zaznamenané udalosti. **Logstash** je služba, ktorá zabezpečuje prenos dát z honeypotov do databázy. **Kibana** je zobrazovač dát pomocou ktorej sa zobrazujú všetky udalosti vo webovom prehliadači [13].
- T-Pot obsahuje ďalšie **nástroje** ako napríklad mapu útokov, na ktorej sa zobrazujú miesta (podľa geolokácie IP adresy) z ktorých sa v reálnom čase snažil niekto uskutočniť útok na niektorý z honeypotov. Ďalšími nástrojmi sú CyberChef, Spiderfoot, Elasticvue (webové rozhranie pre prácu s Elasticsearch klastrom) a NGINX poskytujúci bezpečné vzdialené pripojenie ku Kibane, CyberChef, Elasticvue, mape útokov a Spiderfoot.
- **Sieťový bezpečnostný monitoring** (Network Security Monitoring - NSM). T-Pot obsahuje viaceré nástroje na sledovanie a monitorovanie sieťovej prevádzky ako napríklad Suricata alebo Fatt.

- **Honeypoty**. Aktuálne je dostupných 22 docker obrazov rôznych honeypotov. Niektoré z nich si bližšie popíšeme v nasledujúcej podkapitole.

1.3.2 Honeypoty v T-Pote

Ako bolo spomínané v predchádzajúcej časti, T-Pot aktuálne obsahuje 22 použiteľných docker obrazov honeypotov. Každý z nich slúži na simulovanie inej služby, systému alebo zraniteľnosti.

ADBHoney je zo skupiny honeypotov s nízkou úrovňou interakcie. Je dizajnovaný pre Android Debug Bridge cez TCP/IP. Jeho cieľom je zachytávať malvér, ktorý je posúvaný obetiam s otvoreným portom 5555 [14].

Cisco ASA honeypot je taktiež zo skupiny honeypotov s nízkou úrovňou interakcie. Tento honeypot simuluje Cisco ASA komponenty, na ktorých sa dá detegovať zraniteľnosť, ktorá dovolí neautorizovaný vzdialený prístup pre vzdialené vykonanie kódu (CVE-2018-010) [15].

Conpot je honeypot na strane servera s nízkou úrovňou interakcie. Jeho úlohou je zhromažďovať informácie o motívoch a metódach útočníkov zameraných na priemyselné riadiace systémy (ICS) [16].

Cowrie je zo skupiny honeypotov so strednou až vysokou úrovňou interakcie zameraný na služby SSH a Telnetu. Jeho hlavnou úlohou je zaznamenávať útoky hrubou silou na tieto služby resp. protokoly. Taktiež dokáže zaznamenávať interakciu so shell-om vykonanú útočníkom. V režime so strednou interakciou (cez shell) emuluje UNIX-ový systém v jazyku Python. Vo vysoko interakčnom režime (cez proxy) funguje ako proxy server SSH a telnetu na pozorovanie správanie útočníka v inom systéme [17].

DDoSPot je skupina honeypotov ktoré sú zamerané na sledovanie a monitorovanie Distributed Denial of Service (DDoS) útokoch založených na UDP protokole. Poskytuje viacero honeypot služieb resp. serverov: DNS – snaží sa emulovať DNS službu, NTP – odpovedá na niektoré módy paketov Client, Control a monlist, Generic – emuluje odpoveď ľubovoľnej služby bez dodržania konkrétneho protokolu resp. algoritmu [18].

Úlohou honeypotu **Dionaea**, ktorý emuluje Windows prostredie a služby so zraniteľnosťami, je odchytávať malvér zneužívajúci tieto zraniteľnosti. V najlepšom prípade sa podarí získať až kópiu malvéru [19].

Honeypot **ElasticPot** simuluje zraniteľný Elasticsearch server zverejnený na internete [20].

Endlessh je honeypot, ktorý veľmi pomaly posiela nekonečný SSH baner. Týmto spôsobom udržiava klientov (útočníkov) v spojení s týmto serverom veľmi dlhú dobu a zabraňuje im tak útočiť na reálny server. Cieľom je mať reálny SSH server sprístupnený na inom porte a takto nechať útočníkov (hlavne script kiddies) "odstavených" na honeypote [21].

Heralding je jednoduchý, no za to veľmi užitočný honeypot, ktorý zbiera prihlasovacie údaje. Podporuje komunikáciu prostredníctvom niekoľkých protokolov (napr.: ftp, telnet, ssh, http/s, pop3, ...). Dáta sú v JSON formáte s atribútmi ako zdrojová a cieľová IP adresa a port, počet pokusov o prihlásenie a samotné prihlasovacie údaje s časovou pečiatkou. Je veľmi užitočný na obmedzenia vytvárania politík pre prihlasovacie údaje [22].

Honeytrap je zo skupiny honeypotov s nízkou interakciou, ktorý sa javí ako bežiaca TCP alebo UDP služba. Beží to ako daemon, ktorý štartuje služby servera na požadovaných portoch. Server emuluje známe služby posielaním sieťovej prevádzky do pripojeného hosta. Taktiež dokáže parsovať príkazy zasielané útočníkmi [23].

IPPHoney je honeypot simulujúci tlačiareň, ktorá podporuje Internet Printing protocol (IPP), ktorá je voľne dostupná z internetu [24].

Log4Pot je typ honeypotu, ktorý simuluje systém so známou Log4Shell zraniteľnosťou (CVE-2021-44228). Jeho úlohou je počúvať na rôznych portoch a zaznamenávať pokusy o exploitáciu Log4Shell zraniteľnosti. Následne detegovať a rekurzívne stiahnuť náklad (payload) z exploitu [25].

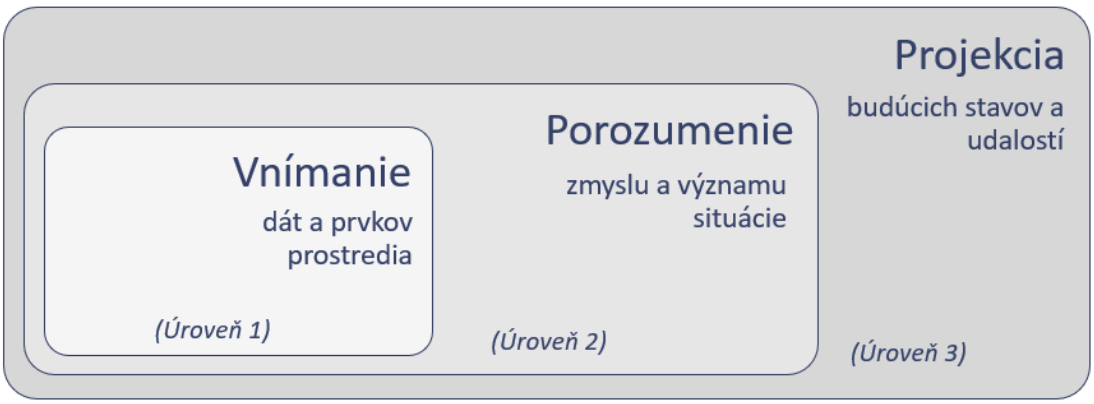
Honeypots je skupina 30 honeypotov s nízkou až vysokou úrovňou interakcie, ktoré dokážu sledovať a monitorovať sieťovú prevádzku, aktivity botov a zaznamenávať prihlasovacie údaje zadávané útočníkmi. Taktiež dokážu emulovať množstvo serverov s rôznymi službami na rôznych portoch. Tieto servery následné zaznamenávajú potrebné udalosti vykonané útočníkmi ako napríklad zdrojovú a IP adresu a port, prihlasovacie údaje alebo príkazy, ktoré sa útočník pokúsil vykonať [26].

2 Situačné povedomie a jeho predikcia

V predchádzajúcej časti tejto záverečnej práce sme sa venovali teoretickým základom honeypotov a honeynetov. Boli v nej vysvetlené definície a rôzne typy delení honeypotov, či už podľa úrovne interakcie, role ktorú zastávajú alebo typov zbieraných dát a útokov. Taktiež sme sa detailnejšie pozreli na architektúru platformy T-Pot a aké typy honeypotov obsahuje. V tejto kapitole si bližšie priblížime, čo je situačné povedomie a spôsoby jeho predikcie pomocou časových radov.

2.1 Definícia situačného povedomia

Mica Endsley definovala situačné povedomie ako „vnímanie prvkov v prostredí v rámci času a priestoru, pochopenie ich významu a projekcia ich stavu v blízkej budúcnosti“ [27]. Z tejto definície sa dajú extrahovať tri úrovne situačného povedomia zobrazené na Obr. 2.



Obr. 2 Úrovne situačného povedomia [27]

Z tohto pohľadu môžeme vnímanie (perception) uvažovať ako monitorovanie, resp. zachytávanie útokov prichádzajúcich na sieť honeypotov. Pochopenie (comprehension) korešponduje s porozumením situácie v kybernetickej bezpečnosti napr. formovaním kybernetických hrozieb. Najvyššou úrovňou je projekcia (projection), resp. predikcia, zahrňuje predikovanie zmien situačného povedomia v kybernetickej bezpečnosti [27].

Na predikciu situačného povedomia sa využívajú najmä metódy založené na spojitých modeloch. Príkladmi týchto metód môžu byť časové rady, strojové učenie alebo

sivé modely. Bežnými výsledkami týchto metód sú predpovedanie konkrétnej hodnoty, objemu alebo nájdenie vzorov v časových radoch na predikovanie útoku.

Použitie časových radov, sivých metód a taktiež aj metód strojového učenia je podrobnejšie rozoberaný v článku, z ktorého v tejto časti budeme vychádzať [28]. Pri časových radoch sledovali použitie modelov SMA, EWMA a ARIMA. Z metód strojového učenia boli porovnávané metódy neurónových sietí, konkrétne so spätným šírením chyby, rekurentných, RBF a Wavelet. Taktiež boli skúmané metódy SVM, teda metóda podporných vektorov, ale aj metódy dolovania údajov, rozhodovacích stromov a náhodných lesov.

2.2 Štatistické metódy

V tejto časti sa bližšie pozrieme na najčastejšie používané štatistické metódy na predikciu časových radov [32].

Autoregresia (Autoregression AR) predpovedá nasledujúcu hodnotu na základe lineárnej kombinácie predchádzajúcich pozorovaní. Matematický výpočet:

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

Kde X_t je hodnota v čase t, c je konštanta, φ sú koeficienty modelu a ε_t je náhodná chyba resp. šum.

Najvhodnejšie použitie tejto metódy je pre časové rady s jednou zložkou bez sezónnych komponentov. Implementácia tejto metódy v jazyku Python je nasledovná:

```
# príklad AR
from statsmodels.tsa.ar_model import AutoReg
# nafitujeme model
model = AutoReg(data, lags=1)
model_fit = model.fit()
# vytvoríme predikciu
pred = model_fit.predict(len(data), len(data))
print(pred)
```

Kĺzavý priemer (Moving Average MA) je metóda, ktorá modeluje predikciu v sekvencii ako lineárnu funkciu reziduálnych chýb zo stredného procesu v predchádzajúcich časových krokoch. Matematický výpočet:

$$X_t = \mu + \sum_{i=1}^{p} \theta_i \, \varepsilon_{t-i} + \varepsilon_t$$

Kde X_t je hodnota v čase t, μ je priemer časového radu, ε_t je náhodná chyba resp. šum v čase t, ε_{t-i} sú náhodné chyby resp. šum v predchádzajúcich časoch.

Táto metóda je taktiež vhodná pre predikciu jednorozmerných časových radov bez trendových a sezónnych zložiek.

Autoregresný integrovaný kĺzavý priemer (Autoregressive Integrated Moving Average ARIMA) predpovedá ďalší krok v sekvencii ako lineárnu funkciu diferencovaných pozorovaní a reziduálnych chýb v predchádzajúcich časových krokoch. Táto metóda integruje princípy predchádzajúcich dvoch metód ako aj krok diferencovania.

Je to jedna z najpoužívanejších metód predikcií časových radov. Je optimálna pre časové rady s jednou premennou, ktoré vykazujú trend ale nemajú sezónne výkyvy. Implementácia metódy ARIMA v pythone:

```
# príklad ARIMA
from statsmodels.tsa.arima.model import ARIMA
# nafitujeme model
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit()
# vytvoríme predikciu
pred = model_fit.predict(len(data), len(data), typ='levels')
print(pred)
```

2.3 Metódy neurónových sietí

Ďalším spôsobom ako sa môžu predikovať hodnoty časových radov sú pomocou neurónových sietí. V tomto prípade je možné použiť rôzne typy neurónových sietí ako aj ich kombinácie. Touto problematikou sa zaoberali v článku [33]. Použili celkovo 9 typov, aby zistili, ktoré typy neurónových sietí predikujú lepšie takéto dáta. Všetky siete

majú 1 výstupnú vrstvu s n neurónmi, pričom hodnota je určená počtom krokov predikcie. Teda ak by sme chceli jednokrokovú predikciu zvolíme n rovné 1.

Hustá sieť (DN) obsahovala 4 husté vrstvy po 1024, 512, 256, 128 neurónoch s aktivačnou funkciou relu a výstupnú vrstvu.

Dlhá krátkodobá pamäť (LSTM) obsahovala 3 LSTM vrstvy po 512 neurónoch so základnými parametrami a výstupnú vrstvu

Gated Recurrent Unit (GRU) obsahovala 3 GRU vrstvy po 256 neurónoch, 1 rekurentnú vrstvu s 128 neurónmi a výstupnú vrstvu.

1D konvolúcia (Conv1D) obsahovala 3 konvolučné vrstvy s filtrami 256 veľkosťami kernelov 7 a aktivačnou funkciou relu, 1 hustú vrstvu s 64 neurónmi a výstupnú vrstvu. Taktiež bola skúmaná 1D konvolúcia s inými veľkosťami filtrov a kernelov.

Enkóder-Dekóder Conv1D obsahovala 2 konvolučné vrstvy s filtrami 256 a 128, veľkosťami kernelov 7 a 5 a aktivačnou funkciou relu, 1 hustú vrstvu s 1 neurónom a aktivačnou funkciou sigmoid, 1 hustú vrstvu s 9216 neurónmi a lineárnou aktivačnou funkciou, 2 Conv1DTranspose vrstvy s 128 a 256 neurónmi, veľkosťami kernlov 5 a 7 a aktivačnou funkciou relu a výstupnú vrstvu.

Enkóder-Dekóder LSTM obsahovala 2 LSTM enkóder vrstvy, RepeatVector vrstvu, 2 LSTM dekóder vrstvy a výstupnú vrstvu. Táto neurónová sieť bola vytvorená na základe veľmi podobne enkóder-dekóder LSTM siete, ktorá mala po jednej LSTM vrstve. Sieť s dvoma LSTM vrstvami dosahovala lepšie výsledky.

Na základe tohto výskumu sme sa rozhodli pre použitie modelov neurónových sietí LSTM a GRU. V ďalších podkapitolách sa bližšie pozrieme na to, ako tieto siete vyzerajú, aké parametre sme pri trénovaní použili a aké výsledky sa nám podarilo takýmto spôsobom dosiahnuť.

2.3.1 LSTM

LSTM (Long Short-Term Memory), siete s dlhou krátkodobou pamäťou, neurónové siete sú špeciálnym typom rekurentných sietí navrhnuté tak, aby riešili problém dlhodobej závislosti. Zjednodušene povedané, ako názov napovedá, sú to neurónové siete, ktoré sú schopné si udržať informáciu po dlhšie časové obdobie. Vďaka

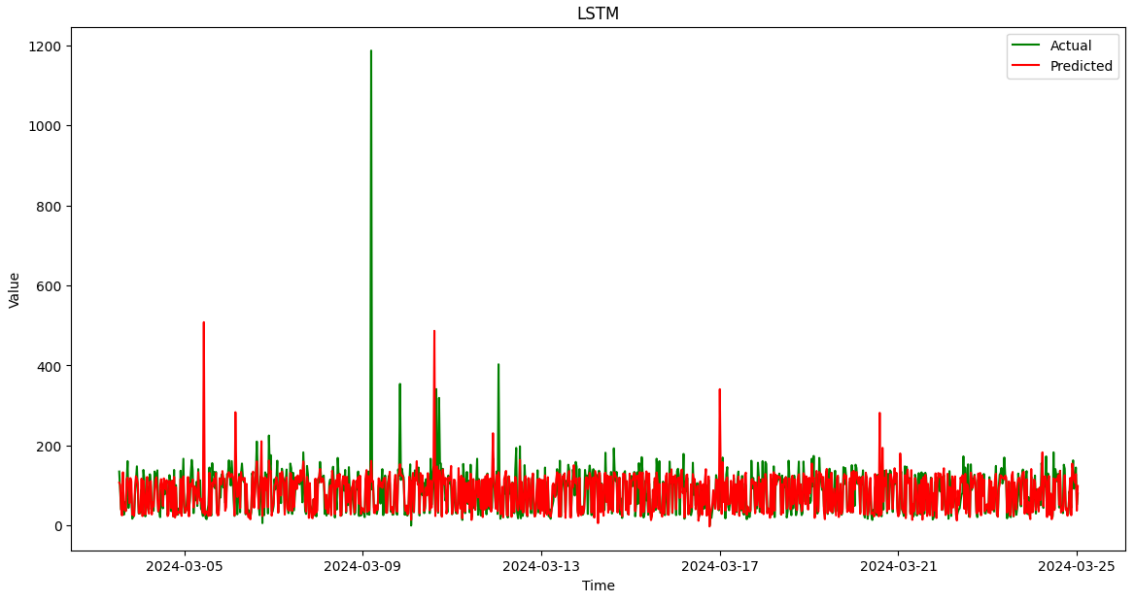
tomu dokážu zachytávať dlhodobé závislosti v dátach. Práve táto vlastnosť je kľúčová pri predikovaní použitím časových radov [34].

LSTM siete obsahujú takzvané bunky (cells), ktoré sa skladajú z niekoľkých základných komponentov:

- **Pamäťová bunka** – uchováva informácie naprieč rôznymi časovými krokmi.
- **Vstupná brána** – Rozhoduje, ktoré hodnoty zo vstupu sa aktualizujú v pamäťovej bunke. Používa funkciu sigmoid pre rozhodnutie a tanh pre škálovanie hodnôt do rozsahu -1 a 1.
- **Zabúdacia brána** – Rozhoduje, ktoré informácie z pamäťovej bunky sa zabudnú, taktiež používa sigmoidálnu funkciu na toto rozhodnutie.
- **Výstupná brána** – Rozhoduje, ktoré hodnoty z pamäťovej bunky sa použijú ako výstup v danom časovom kroku.

Náš model aktuálne pozostáva z dvoch vrstiev. Prvá vrstva je LSTM s počtom jednotiek 50 a aktivačnou funkciou relu. Druhá vrstva je výstupná, ktorá obsahuje jeden neurón. Model používa optimalizátor Adam a stratovú funkciu MSE. Model trénujeme v 20 epochách, veľkosť batchu je 32 a rozdelenie na validačnú množinu je 0,1. Na Obr. 3 môžeme vidieť zelenou farbou reálne hodnoty a červenou farbou predikované hodnoty.

Ako bolo spomenuté venujúcej sa verifikácii (2.1.1), model bol vyhodnocovaný na 7 časových radoch pomocou metrík MAE, MSE a MASE. Tento model dosahoval najlepšie výsledky predikovaním unikátnych IP adries. V tomto prípade bolo MAE = 19,368, MSE = 2658,389 a MASE = 0,323. V porovnaní s celkovým počtom útokov kde boli dosiahnuté nasledujúce hodnoty MAE = 1318,717, MSE = 11772639416 a MASE = 0,674 je vidieť že tento model bol úspešnejší. Je potrebné však brať do úvahy, že celkový počet útokov za časovú jednotku je niekoľko násobne väčší ako počet unikátnych IP adries, a preto nemôžeme priamo porovnávať skóre MAE a MSE.



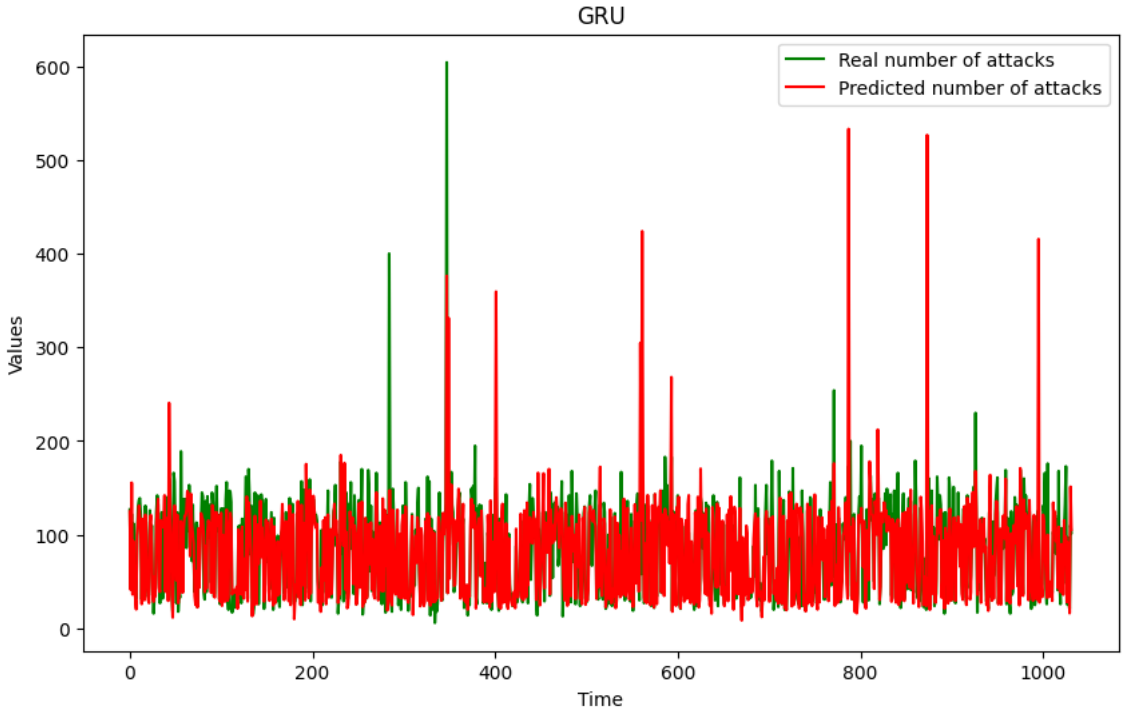
Obr. 3 Porovnanie predikcií unikátnych IP adries a reálnych hodnôt modelu LSTM

2.3.2 GRU

GRU (Gated Reccurent Unit), bránové rekurentné jednotky, sú neurónové siete, ktoré patria do kategórie rekurentných neurónových sietí. Zameriavajú sa na rovnaký problém, teda uchovanie si dlhodobejších závislostí. Na rozdiel od LSTM bunky, GRU bunka spája vstupnú a výstupnú bránu do jednej, takzvanej aktualizačnej brány. Výsledkom je, že takýto model je jednoduchší ako LSTM model a veľmi rýchlo narastá na popularite [34].

Model GRU neurónovej siete, ktorý sme otestovali pozostáva z 2 vrstiev. Prvá vrstva obsahuje 50 GRU jednotiek a počet časových krokov, ktoré budú použité na predikciu, v našom prípade 24. Nasleduje vrstva s jedným výstupným neurónom. Model používa optimalizátor Adam a stratovú funkciu MSE. Model sa trénuje pri 20 epochách s nastavením veľkosti batchu na 32.

Keďže metódy neurónových sietí LSTM a GRU sú stavbou veľmi podobné, túto podobnosť môžeme vidieť aj pri väčšine výsledkov. Pri predikovaní počtu unikátnych IP adries dosiahol tento model MAE = 21,003, MSE = 3914,425 a MASE = 0,249. Z týchto výsledkov môžeme usúdiť, že v tomto prípade bola presnosť týchto modelov veľmi podobná.



Obr. 4 Porovnanie predikcií unikátnych IP adries a reálnych hodnôt modelu GRU

2.4 Metódy strojového učenia

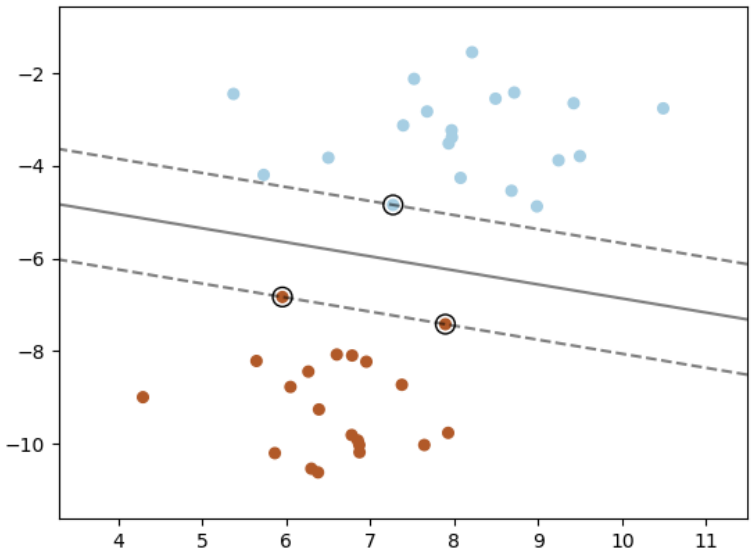
V nasledujúcej časti sa budeme venovať metódam strojového učenia. Strojové učenie zahŕňa množstvo rôznych metód a algoritmov. Medzi najpoužívanejšie patria metóda podporných vektorov (Support Vector Machine SVM) [35], rozhodovacie stromy [36] a pokročilé techniky napr. XGBoost [37].

Tieto metódy sú zvyčajne výpočtovo menej náročné v porovnaní s neurónovými sieťami. Neurónové siete vyžadujú rozsiahle výpočtové zdroje a rádovo dlhší čas na tréning, najmä pri veľkých dátových sadách. V tejto práci sa zameriame na použitie SVM, rozhodovacích stromov a metódy XGBoost.

2.4.1 SVM

SVM je súbor metód strojového učenia používaných najmä pri klasifikačných a regresných úlohách alebo na detekciu anomálií. V našom prípade budeme využívať regresiu. Hlavnou myšlienkou SVM je nájsť nadrovinu v priestore, ktorá jasne rozdelí body do rôznych kategórií. Pre nájdenie optimálnej nadroviny sa snažíme maximalizovať rozsah medzi najbližšími dátovymi bodmi rôznych tried, teda podporných vektorov. SVM

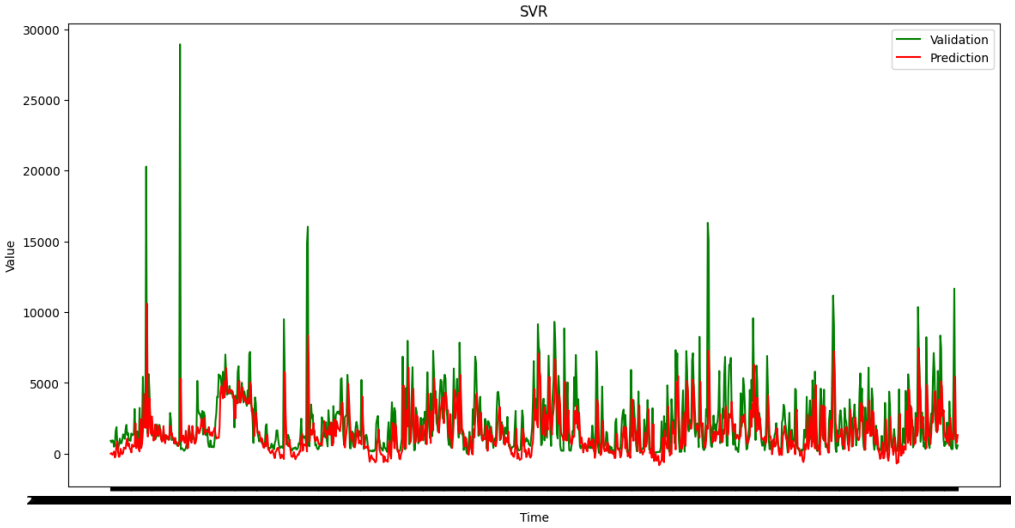
sú efektívne na rôznych typoch dát aj vďaka použitiu rôznych jadrových funkcií, ako napríklad lineárne, polynomiálne a RBF funkcie [35].



Obr. 5 Ukážka princípu SVM [38]

V záverečnej práci bol tento model využitý na predikciu potenciálnych bezpečnostných hrozieb na základe charakteristických vzorov z dát. Kľúčové parametre modelu sú typ jadra, penalizačný parameter C a gamma. Ako typ jadra sme vybrali rbf (radial basis function), parameter C bol nastavený na 1000 a gamma na 0.01. Tieto parametre boli vybrané na základe výsledkov predbežných testov, ktoré dokázali ich efektívnosť v našom konkrétnom prípade.

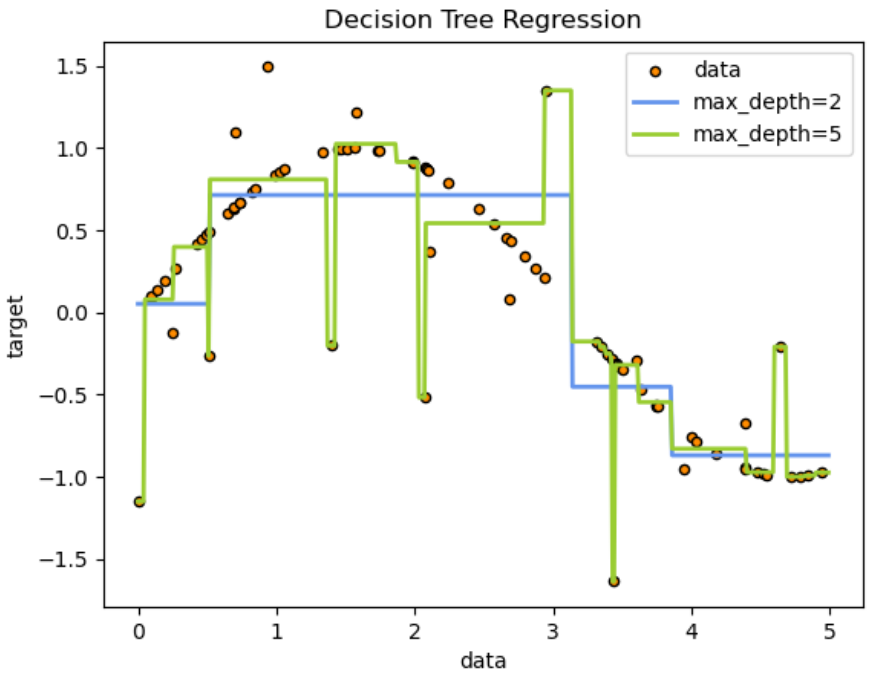
Na Obr. 6 sú zobrazené predikované hodnoty červenou farbou a reálne hodnoty zelenou farbou. Presnosť modelu môžeme vyhodnotiť taktiež na základe MAE, MSE a MASE. Tento model dosiahol pomerne dobré výsledky pri časovom rade počtu útokov na honeypot Dionaea. V tomto prípade bolo MAE = 252,106, MSE = 350619,662 a MASE 0,858. V prípade predikovania počtu unikátnych IP adries tento model dosiahol MAE = 22,477, MSE = 987,353 a MASE = 1,457. Z hodnoty MASE môžeme vyčítať, že v tomto prípade by bola naivná predikcia presnejšia.



Obr. 6 Porovnanie predikcií a reálnych hodnôt modelu SVM

2.4.2 Rozhodovacie stromy

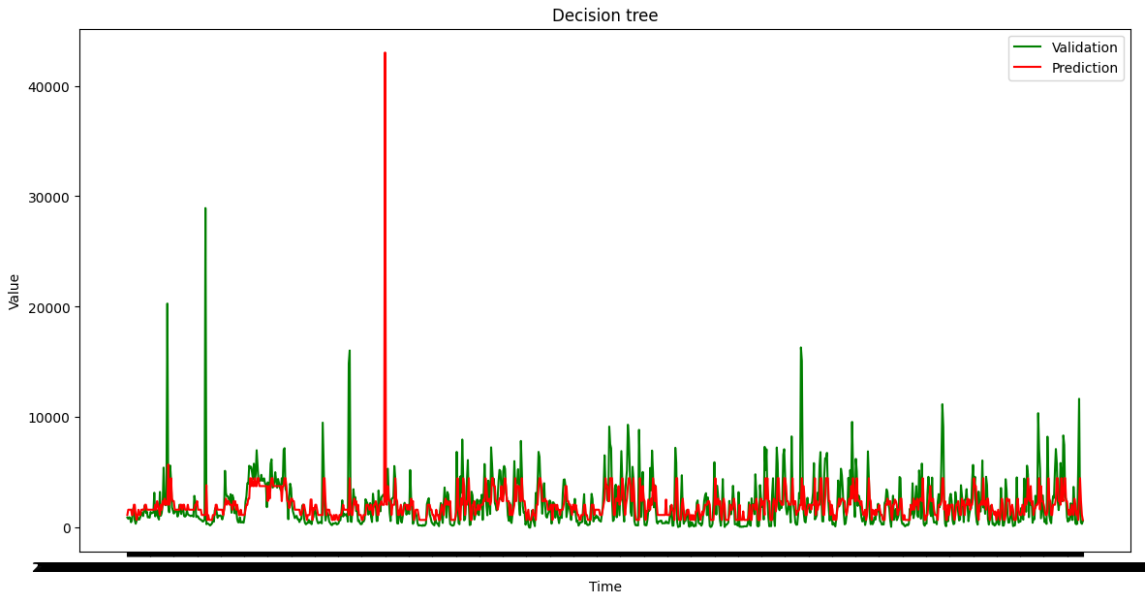
Rozhodovacie stromy sú taktiež metódou strojového učenia spadajúcej pod metódy učenia s učiteľom používanou na klasifikačné a regresné úlohy. Táto metóda pracuje na princípe rozdelenia dát na menšie homogénnejšie skupiny na základe určitých podmienok. To vytvára stromovú štruktúru rozhodnutí. Tvorenie stromu funguje na princípe výberu parametra, ktorý najlepšie v danom uzle dáta rozdelí.



Obr. 7 Ukážka použitia rozhodovacích stromov na regresnú úlohu [39]

Nevýhodou rozhodovacích stromov je vytváranie príliš zložitých stromov. Pre zabránenie tohto problému existujú rôzne mechanizmy. Medzi najdôležitejšie parametre modelu patrí napríklad hĺbka stromu, kritérium kvality rozdelenia alebo rýchlosť učenia. Pre rozhodovacie stromy boli nastavené parametre kritérium na hodnotu poisson a maximálna hĺbka stromu na 6.

Pri tomto nastavení parametrov sme dosiahli najlepšie výsledky pri časovom rade honeypotu Honeytrap. Tu dosiahol model rozhodovacích stromov hodnotu MAE = 52,208, MSE = 21254,005 a MASE = 0,9. Pri vyhodnocovaní predikcií ostatných časových radov dosahoval o niečo slabšie výsledky ako napríklad metóda SVM. Vykreslené predikované hodnoty červenou farbou a reálne hodnoty zelenou farbou môžeme vidieť na Obr. 6.



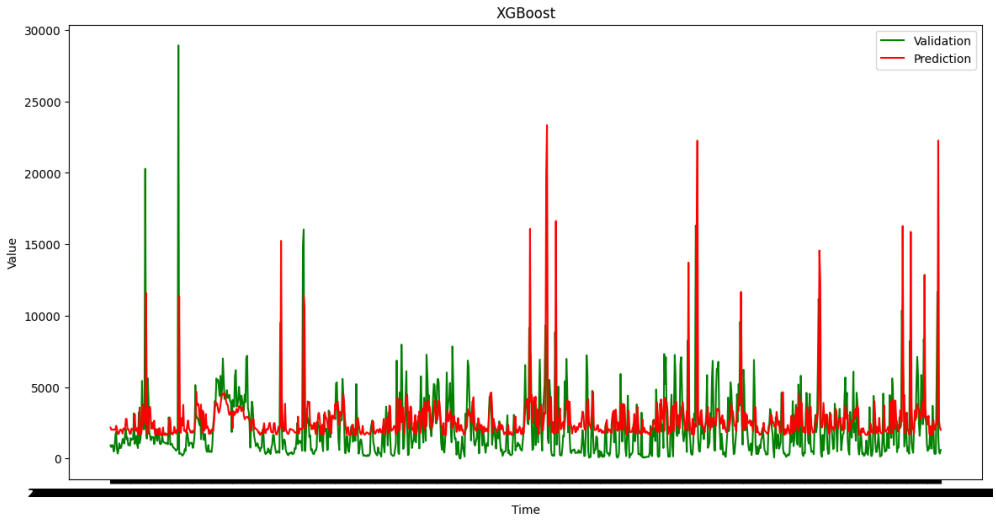
Obr. 8 Porovnanie predikcií a reálnych hodnôt modelu Rozhodovacieho stromu

2.4.3 XGBoost

Metóda **XGBoost** (skr. pre Extreme Gradient Boosting) je metóda strojového učenia založená na gradientovom boostingu. Táto metóda kombinuje viacerých slabších učiteľov, najmä rozhodovacie stromy, na vytvorenie silnejšej predikcie. Výhodou je taktiež nízka náročnosť na zdroje. Hlavným princípom je iteratívne pridávanie stromov do modelu tak, aby každý strom opravoval chyby predchádzajúcich stromov [40].

Model XGBoost sme trénovali s parametrami minimálnej váhy dieťaťa 5, podmnožinu pre každý strom na 0,7 maximálnu hĺbku na 5 a učiaci pomer na 0,05.

Tento model bol vo všeobecnosti menej úspešný ako SVM a rozhodovacie stromy. V jedinom prípade, kedy tento model dosiahol lepšie výsledky oproti rozhodovacím stromom, je pri predikovaní počtu útokov na port 23. MAE = 10,400, MSE = 490,707 a MASE = 1,048 a rozhodovacie stromy v tomto prípade mali MAE = 14,187, MSE = 8059,780 a MASE 1,429. Z tohto môžeme vidieť že XGBoost bola presnejšia ako rozhodovacie stromy, avšak na základe hodnoty MASE väčšej ako 1 môžeme usúdiť, že sú menej presné ako naivná predikcia.



Obr. 9 Porovnanie predikcií a reálnych hodnôt modelu XGBoost

2.5 Prístupy k verifikácii modelov

Veľmi dôležitou časťou pri predikovaní je overovanie správnosti a relevantnosti modelov, teda **verifikácia** modelov. Tieto metriky overujú rôznymi spôsobmi chybu medzi predikovanými hodnotami a reálnymi hodnotami. Na výber je viacero metrík, ktoré sa využívajú pri vyhodnocovaní presnosti predikcií modelov. Medzi najčastejšie používané metriky zaraďujeme [29]:

- **MAE** (Mean Absolute Error) je priemerná absolútna chyba.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|x_i - m_i|$$

Kde x_i je aktuálna hodnota a m_i je predikovaná hodnota.

- **MSE** (Mean Squared Error) priemerná štvorcová chyba.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - m_i)^2$$

Kde x_i je aktuálna hodnota a m_i je predikovaná hodnota.

- **MASE** (Mean Absolute Scaled Error) je priemerná absolútna škálovaná chyba. Ak je MASE < 1 znamená to, že je model lepší ako základná naivná predikcia.

$$MASE = \frac{MAE}{MAE_{naive}}$$

- **Diebold-Mariano Test** sa používa na porovnanie dvoch modelov na predikciu.

Diebold Mariano test je štatistická metóda používaná na porovnanie predikčnej presnosti dvoch modelov. Metóda, ktorý vyvinul Francis X. Diebold a Robert S. Mariano sa používa najmä pri analýze časových radov, kde zisťuje, či je rozdiel v predikčnej výkonnosti štatisticky významný. Test vypočíta testovaciu štatistiku na základe série rozdielov v stratovej funkcii (napr. kvadratická chyba). Výstupom Diebold Mariano testu sú dve hodnoty, DM štatistika a p-hodnota [30].

- **DM štatistika** udáva rozdiel v presnosti predpovede medzi dvoma porovnávanými predikčnými modelmi. Záporná hodnota naznačuje lepšiu presnosť prvého modelu, kladná lepšiu presnosť druhého modelu
- **P-hodnota** určuje pravdepodobnosť pozorovania vypočítaného rozdielu v prediktívnej presnosti medzi dvoma modelmi. Zjednodušene, určuje, či je rozdiel v presnosti modelov štatisticky významný.

V úvode máme dve hypotézy. **Nulová hypotéza** hovorí, že neexistuje rozdiel v prediktívnej presnosti medzi porovnávanými modelmi.

Alternatívna hypotéza je, že existuje rozdiel v prediktívnej presnosti medzi modelmi.

Nízka p-hodnota (zväčša < 0,5) ukazuje dôkaz proti nulovej hypotéze. To naznačuje, že porovnávaný rozdiel v presnosti je štatisticky významný a teda prvý model je lepší voči druhému. Vysoká p-hodnota (zväčša > 0,5) ukazuje slabý dôkaz proti nulovej hypotéze (teda nulovú hypotézu nezamietame), čo naznačuje, že akýkoľvek pozorovaný rozdiel v presnosti môže byť pravdepodobne dôsledkom náhody. Teda na základe dostupných údajov nie je možné vybrať lepší model [30].

Na implementáciu Diebold Mariano testu sme sa rozhodli použiť python knižnicu dieboldmariano [31]. Táto knižnica poskytuje implementovanú funkciu dm_test porovnania dvoch metód. Použitím tejto metódy sme vytvorili metódu pre automatizované porovnávanie viacerých metód spôsobom „každá s každou“.

```
# Vykonanie DM testu medzi každým paŕommodelov
results = {}
model_names = list(predictions.keys())
for i in range(len(model_names)):
    for j in range(i+1, len(model_names)):
        model1, model2 = model_names[i], model_names[j]
        dm_stat, p_value = dm_test(
            actual_values,
            predictions[model1],
            predictions[model2],
            one_sided=True
        )
        results[f'{model1} vs {model2}'] = (dm_stat, p_value)

# Výpis výsledkov
for comparison, (dm_stat, p_value) in results.items():
    print(f"{comparison}:\n \tDM Statistic = {dm_stat}, \tp-value
= {p_value}")
```

List predictions obsahuje všetky výsledky modelov, ktoré chceme porovnať vo forme názov modelu: predikcie.

Pre vyhodnotenie presnosti modelov sme vybrali 7 časových radov:

- Celkový počet útokov
- Počet unikátnych IP adries
- Útoky na port 22
- Útoky na port 23

- Útoky na honeypot Cowrie
- Útoky na honeypot Dionaea
- Útoky na honeypot Heralding

Na týchto časových radoch sme vyhodnotili všetky navrhnuté modely. V nasledujúcich podkapitolách sa bližšie pozrieme na štatistické metódy, metódy neurónových sietí a metódy strojového učenia.

2.6 Porovnanie presnosti modelov

Ako bolo spomínané v predchádzajúcich podkapitolách, všetky modely boli natrénované a následne vyhodnotené na 7 časových radoch. Pri samotných modeloch sme vyzdvihli niektoré zaujímavé výsledky. V tejto podkapitole sa bližšie pozrieme na porovnanie presnosti modelov na základe hodnôt MAE, MSE, MASE a na záver aj vykonaním Diebold Mariano testu.

Tabuľka Tab. 1 zobrazuje zaujímavé porovnania modelov a ich presnosti na rôznych časových radoch. Tabuľka pozostáva z označenia časového radu, metódy a následne hodnotami MAE, MSE a MASE. Štatistické metódy dosahovali podobné výsledky, pričom najlepšie z nich dosahovala metóda ARIMA. Preto v ako reprezentatívnu metódu zo štatistických metód budeme používať metódu ARIMA.

Časový rad	Metóda	MAE	MSE	MASE
Počet útokov	ARIMA	1573.427	5322889.578	0.868
	SVM	1266.728	4736500.358	0.981
	DT	1254.012	5815545.819	0.971
	XGBoost	1649.185	1649.185	1.277
	LSTM	1318.717	11772639.416	0.674
	GRU	1448.264	17705161.984	0.695
Unikátne IP adresy	ARIMA	44.086	3795.174	2.328
	SVM	22.477	987.353	1.457
	DT	18.236	3608.058	1.182
	XGBoost	29.379	1117.775	1.903
	LSTM	19.368	2658.389	0.323
	GRU	21.003	3914.425	0.249
Port 22	ARIMA	125.477	31583.553	0.875
	SVM	102.663	27448.281	1.060
	DT	108.428	34323.367	1.120
	XGBoost	113.612	27012.663	1.173
	LSTM	92.369	18309.606	0.586

	GRU	91.752	20163.605	0.567
Dionaea	ARIMA	377.194	388022.861	0.963
	SVM	252.106	350619.662	0.858
	DT	281.926	361734.366	0.959
	XGBoost	328.747	320901.283	1.118
	LSTM	268.181	274937.741	0.609
	GRU	291.269	267043.484	1.042
Honeytrap	ARIMA	99.088	26492.039	1.581
	SVM	64.397	21156.246	1.110
	DT	52.208	21254.005	0.900
	XGBoost	73.470	21512.813	1.266
	LSTM	67.237	30222.207	0.529
	GRU	70.933	38321.663	0.715

Tab. 1 Porovnanie presnosti skúmaných modelov

Pri počte unikátnych adries je z dosiahnutých výsledkov zobrazených v tabuľke Tab. 1 vidieť, že sa modely LSTM a GRU dopúšťali menších chýb ako ostatné testované modely. Taktiež na základe hodnoty MASE vidíme, že sú lepšie ako naivné predikcie. Podobné výsledky dosiahli aj pri časovom rade počtu útokov na port 22. V tomto prípade boli metódy strojového učenia horšie ako štatistické metódy. Zaujímavý výsledok sme dosiahli pri časovom rade útokov na honeypot Dionaea. Toto je jediný prípad, kedy model GRU dosiahol výrazne horšie hodnotu MASE, pričom v priemerných chybách sa až tak nelíšil. Podobné výsledky sme dosiahli aj pri opakovanom trénovaní a spúšťaní modelu. Celkovo dosiahli najlepšie výsledky modely neurónových sietí. Spomedzi nich dokázal lepšie predikovať v našom prípade model LSTM siete.

Ako ďalší spôsob porovnania predikčnej presnosti metód sme použili Diebold Mariano test. Pri vykonaní testu medzi testovanými modelmi strojového učenia môžeme vybrať niekoľko najlepších. Na základe výsledkov, ktoré sme dosiahli sme vykonali Diebold Mariano test medzi nasledujúcimi kombináciami testovaných metód.

SVM vs DecisionTree. DM Statistic = -0.6894835619880473, p-value = 0.24533738263700422. DM štatistika je negatívna, čo naznačuje, že model SVM má lepšiu prediktívnu presnosť, avšak rozdiel nie je štatisticky významný pretože p-hodnota nie je menšia ako 0,05. Teda nulovú hypotézu nezamietame.

SVM vs XGBoost. DM Statistic = -2.8014193287231124, p-value = 0.002591701499904126. DM štatistika je negatívna a v tomto prípade aj p-hodnota je výrazne menšia ako 0,05, teda model SVM má štatisticky významne lepšiu prediktívnu

presnosť. V tomto prípade nulovú hypotézu zamietame a teda do úvahy prichádza alternatívna hypotéza.

DecisionTree vs XGBoost. DM Statistic = -0.5328435668823154, p-value = 0.2971285728271357. DM štatistika je negatívna, ale p-hodnota je opäť väčšia ako 0,05. Teda ani v tomto prípade nemôžeme povedať, že jeden z modelov je štatisticky lepší ako druhý. V tomto prípade taktiež nezamietame nulovú hypotézu.

GRU vs LSTM. DM Statistic = -1.54851408, p-value = 0.06090273. DM štatistika opäť poukazuje, že prvé z porovnávaných metód je lepšia, avšak p-hodnota nie je dostatočne nízka na to, aby sme zamietli nulovú hypotézu. Teda rozdiel nie je štatisticky významný.

Na základe porovnaní modelov môžeme konštatovať, že modely neurónových sietí a metódy strojového učenia dosahujú lepšie výsledky ako štatistické metódy. Dokonca, LSTM a GRU neurónové siete dosahujú značne lepšie výsledky v porovnaní s metódami strojového učenia ako SVM, rozhodovacie stromy a XGBoost. Zároveň sú výpočtovo náročnejšie, avšak pre takéto vylepšenie presnosti predikcií to je možné tolerovať. Výpočtovo náročnejšie je najme trénovanie, ktoré sa nemusí vykonávať tak často ako samotné predikovanie.

3 Návrh a implementácia riešenia

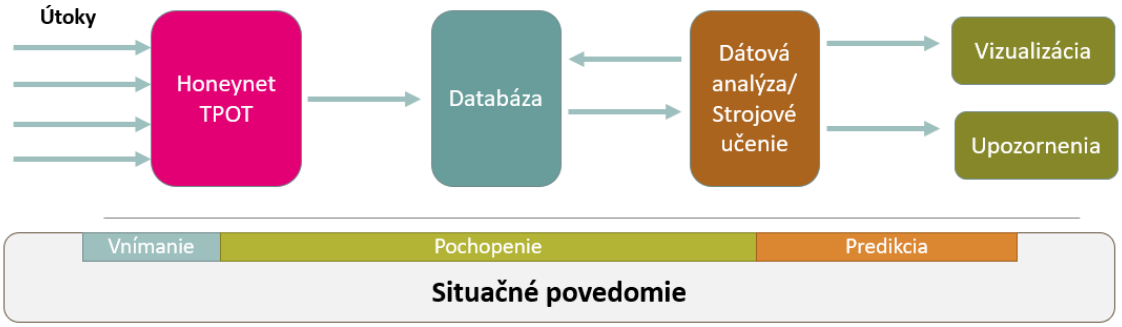
V tejto časti sa bližšie pozrieme na návrh riešenia problému predikcie situačného povedomia v kybernetickej bezpečnosti. Súčasťou návrhu sú honeypoty a honeynety a metódy strojového učenia a neurónových sietí pre dosiahnutie optimálnej prediktívnej schopnosti.

Hlavným cieľom systému na predikciu situačného povedomia v oblasti kybernetickej bezpečnosti je poskytnúť včasnú a dostatočne presnú informáciu o potencionálnych hrozbách, ktorá umožní organizáciám adekvátne a hlavne proaktívne reagovať na potencionálne bezpečnostné hrozby. Na základe toho je možné prijímať opatrenia na zabránenie zneužitia aktuálnych hrozieb a tým optimalizovať bezpečnostné stratégie a minimalizovať škody.

Náš návrh riešenia tohto problému spočíva v prilákaní útočníkov na nami nasadené podvodné systémy – honeypoty, ktoré budú simulovať zraniteľné systémy a aplikácie. Tieto systémy budú zaznamenávať všetky aktivity útočníkov, čo nám dá dostatočné množstvo dát, ktoré budú tvoriť základ pre dátovú analýzu a vypracovanie obrazu o aktuálnom situačnom povedomí v kybernetickej bezpečnosti. Následne budeme pomocou metód strojového učenia a neurónových sietí predikovať vývoj situačného povedomia, na základe čoho bude možné zaviesť bezpečnostné opatrenia. Návrh riešenia je rozdelený do piatich hlavných častí:

- **Zber dát** – definujeme a implementujeme mechanizmy na efektívny zber dát z honeypotov. To zahrňuje vytvorenie vhodného datasetu.
- **Dátová analýza, strojové učenie** – využijú sa metódy dátovej analýzy, strojového učenia a neurónových sietí na identifikáciu vzorov a zákonitostí v zozbieraných údajoch. Cieľom je extrahovať informácie o správaní útočníkov a vytvoriť modely pre predikciu.
- **Ukladanie dát** – v tejto časti sa navrhnú a implementujú úložiská, teda databázy pre získané dáta. Uchovávané dáta musia byť prístupné pre prípadné ďalšie analýzy a zobrazovanie výsledkov.
- **Vizualizácia** – implementujú sa nástroje na vizualizáciu výsledkov z predchádzajúceho kroku. Vizualizácie budú slúžiť pre lepšiu interpretáciu a pochopenie situačného povedomia v kybernetickej bezpečnosti

- **Upozornenia** – systém by mal na základe stanovených hraníc vedieť vygenerovať upozornenia na možné bezpečnostné hrozby.



Obr. 10 Schéma návrhu riešenia

Schéma návrhu nášho riešenia je zobrazená na obrázku Obr. 8. Vstupom pre naše riešenie sú útoky (Attacks), ktoré sú realizované útočníkmi na honeypoty a honeynety (Honeynets). Dáta z prichádzajúcich útokov sú dopytované a následne spracované. Následne je vykonaná dátová analýza a použitie metód strojového učenia (DA – Data Analysis/ML – Machine Learning), ktorá vyprodukuje predikčné modely, na základe ktorých sa budú vypočítavať požadované predikcie. V poslednom kroku sú dáta ukladané do databázy (Database). Následne sa dáta, vrátane predikcií vykresľujú (Visualisation) pre lepšiu interpretáciu a jednoduchšie pochopenie situačného povedomia v kybernetickej bezpečnosti. V paralelnom poslednom kroku sa generujú upozornenia (Alerts) na základe nami definovaných pravidiel a hraníc. V nasledujúcich kapitolách sa bližšie venujeme jednotlivým častiam navrhnutého riešenia.

3.1 Implementačné požiadavky

V tejto kapitole sa pozrieme na implementačné požiadavky, ktoré sú potrebné pre funkčnosť systému, resp. časti systému. Tieto požiadavky zabezpečujú, že systém bude schopný plniť svoju úlohu efektívne a spoľahlivo, pričom sa berie ohľad na aspekty ako výkon, bezpečnosť, škálovateľnosť a udržateľnosť. Na základe nami definovaného návrhu sme stanovili nasledujúce hardvérové a softvérové požiadavky, ktoré využívame.

Systém bol nasadený na serveri s procesorom Intel Xeon E5-2630 s virtualizačnou platformou Proxmox VE vo verzii 7.4-18. Tento server je host pre virtuálny stroj s operačným systémom Linux s verziou kernelu 5.15.0-116-generic a distribúciou Ubuntu 22.04.4 LTS. Virtuálny stroj disponuje 2 jadrami a 8GB operačnej pamäte RAM. Minimálna veľkosť úložiska je odporúčaná na 32GB pre dostatočné množstvo uložených dát. Pre správne fungovanie je potrebný prístup k Internetu. Softvérové špecifikácie pre správny chod všetkých systémov sú nasledovné:

- Python vo verzii 3.10.12,
- InfluxDB vo verzii 2.7.5,
- Influxdb client vo verzii 1.42,
- Grafana vo verzii 10.4.1,
- ElasticSearch vo verzii 8.13.0,
- Matplotlib vo verzii 3.8.4,
- Numpy vo verzii 1.26.4,
- Pandas vo verzii 2.2.1,
- Scikit learn vo verzii 1.4.2,
- Scipy vo verzii 1.13.0,
- Tensorflow vo verzii 2.16.1.

3.2 Zber dát

Základom realizácie riešenia je zber dát. V našom prípade sme sa rozhodli na zber dát použiť spomínané honeypoty a honeynety. Dôležitým problémom, ktorý bolo potrebné vyriešiť, je výber platformy, pomocou ktorej budeme sieť honeypotov riadiť. Rozhodli sme sa pre riešenie T-Pot, keďže ponúka viacero honeypotov prepojených do jedného honeynetu. Je to flexibilný a rozšíriteľný framework, ktorý umožňuje prispôsobenie a rozšírenie rôznych komponentov podľa našich potrieb a požiadaviek.

Architektúru platformy T-Pot sme podrobnejšie popísali v teoretickej časti tejto záverečnej práce (kapitoly 2.4 a 2.5). V tejto časti sa bližšie pozrieme na to, ako vyzerajú dáta z honeypotov, ako sú uložené v elasticsearch databáze a akým spôsobom sa na nich dopytovať.

Dáta sa z honeypotov presúvajú do centrálneho úložiska pomocou tzn. Logstashu. Logstash súbor je vytvorený raz denne, pričom v priebehu dňa sa doňho postupne pripisujú záznamy z honeypotov.

V tejto kapitole sa taktiež budeme venovať automatizácii procesu získavania dát. Najprv vysvetlíme, ako fungujú dopyty nad Elasticsearch databázou, následne spracovanie dát. Takýmto spôsobom je možné manuálne získavať dáta. Následne je potrebné proces zberu dát automatizovať pre zobrazovanie živých dát a následne ich predikcií.

3.2.1 Záznam v logstash súbore

V nasledujúcej časti si ukážeme, ako vyzerá jeden záznam. Na Obr. 11 je vidieť úvodné riadky záznamu obsahujúce atribúty ako napríklad "_index", ktorý označuje ktorému logstash súboru záznam prislúcha (resp. v našom prípade, v ktorý deň bol vygenerovaný), "_id" jednoznačné označenie záznamu a následne informácie o zdroji, teda útočníkovi resp. útoku.


```
{
    "_index": "logstash-2023.12.30",
    "_id": "t5cdvYwBqSCTsGqevBFb",
    "_version": 1,
    "_seq_no": 1402510,
    "_primary_term": 2,
    "found": true,
```


Obr. 11 Ukážka záznamu v json formáte

V časti "_source" zobrazenej na obrázku nájdeme podrobnejšie informácie o zdroji útoku. Jedným z najdôležitejších údajov je časová pečiatka, ktorá určuje presný čas útoku na honeypot. Následne sú v tejto časti informácie o zdrojovej a cieľovej IP adrese, porte a protokole, na základe čoho sú zistené aj približné geolokačné údaje o IP adrese. Geolokačné údaje sú základným zdrojom pre tvorenie mapy útokov spomínanej v predchádzajúcich častiach záverečnej práce. Geolokačné informácie sú na Obr. 12.

```
    "_source": {
        "t-pot_ip_int": "193.87.120.100",
        "src_ip": "161.230.230.44",
        "@timestamp": "2023-12-30T23:44:51.013Z",
        "src_port": 80,
        "t-pot_ip_ext": "193.87.120.100",
        "timestamp": "2023-12-30T23:44:51.013756+0000",
        "proto": "UDP",
        "dest_port": 53,
```


Obr. 12 Ukážka geolokáčneho obohatenia záznamu

Ďalej nasledujú spomínané geolokačné informácie IP adresy útočníka. Na základe IP adresy sa dajú dohľadať približná zemepisná šírka a dĺžka, a teda aj krajina resp. mesto v ktorom sa zdrojová IP adresa nachádzala (Obr. 13).


```
    "geoip": {
        "longitude": -8.6168,
        "region_name": "Porto",
        "country_code2": "PT",
        "postal_code": "4400-507",
        "continent_code": "EU",
        "timezone": "Europe/Lisbon",
        "location": {
            "lon": -8.6168,
            "lat": 41.1384
        },
        "region_code": "13",
        "as_org": "Vodafone Portugal - Communicacoes Pessoais S.A.",
        "country_name": "Portugal",
        "city_name": "Vila Nova de Gaia",
        "ip": "161.230.230.44",
        "asn": 12353,
        "country_code3": "PT",
        "latitude": 41.1384
```


Obr. 13 Ukážka informácie o útočníkovi

V poslednej časti záznamu na obrázkoch z honeypotu sa nachádzajú informácie o cieľovom zariadení, teda honeypote. Je tam jeho IP adresa spoločne s geolokačných informácií alebo názve organizácie (Obr. 14).

```
52          "dest_ip": "193.87.120.100",
53          "geoip_ext": {
54              "longitude": 21.1992,
55              "region_name": "Kosice",
56              "country_code2": "SK",
57              "postal_code": "040 11",
58              "continent_code": "EU",
59              "timezone": "Europe/Bratislava",
60              "location": {
61                  "lon": 21.1992,
62                  "lat": 48.6865
63              },
64              "region_code": "KI",
65              "as_org": "Zdruzenie pouzivatelov Slovenskej akademickej datovej siete",
66              "country_name": "Slovakia",
67              "city_name": "Košice",
68              "ip": "193.87.120.100",
69              "asn": 2607,
70              "country_code3": "SK",
71              "latitude": 48.6865
```


Obr. 14 Ukážka informácie o organizácii

Posledný dôležitý údaj v tomto súbore je názov honeypotu, ktorý danú udalosť zachytil a typ udalosti, o akú v danom prípade išlo (Obr. 15).


```
80          "t-pot_hostname": "whisperingicecream",
81          "in_iface": "eth0",
82          "@version": "1",
83          "event_type": "dns",
84          "path": "/data/suricata/log/eve.json"
```


Obr. 15 Názov honeypotu a typ udalosti

3.2.2 Dopyty nad Elasticsearch databázou

V podobe, akú sme si ukázali v predchádzajúcej podkapitole, sa dáta ukladajú do elasticsearch databázy zabudovanej v platforme T-Pot. Ku dátam sa vieme dostať viacerými spôsobmi. Jedným z nich je spomínaná Kibana. Kibana je prostredie pre zobrazovanie dát, v ktorom si vieme nakonfigurovať prehľadové panely. Výhodou tohto zobrazenia je ľahká čitateľnosť údajov v grafoch, možnosť filtrovania napríklad podľa dátumu (portu, honeypotu, atď.). Nevýhodou tohto zobrazenia by bola náročné získavanie dát na ďalšie spracovanie. Ďalšou možnosťou je ukladať si priamo záznamy z logstashu. Touto metódou je síce možné sa dostať ku surovým dátam, avšak logika za spracovaním a organizáciou týchto dát by bola veľmi náročná.

Z tohto dôvodu sme pre spracovanie dát vybrali tretiu možnosť, knižnicu Elasticsearch [41] v programovacom jazyku Python. Pomocou tejto knižnice je možné sa pripojiť priamo na elasticsearch databázu a následne nad ňou vytvárať potrebné dopyty. Pre pripojenie do elasticsearch databázy je potrebné zadať adresu, na ktorej databáza beží a prihlasovacie údaje. Možnosť overovania certifikátov je nastavená aby sa neoverovali z dôvodu, že server je samo podpísaný.

```
es = Elasticsearch(["https://xxx01.xxx.upjs.sk:64297/es"],
                   verify_certs=False,
                   basic_auth=(user_name, password))
es.ping()
```

Po vytvorení pripojenia do databázy môžeme začať vytvárať dopyty. Dopyt sa skladá z viacerých častí. Prvou časťou sú "aggs", ktoré obsahujú nami stanovené atribúty, podľa ktorých chceme dáta vyberať. Nasledujú polia, ktoré chceme zahrnúť, v závere ďalšie atribúty v časti "querry" ako napríklad z ktorých honeypotov sa majú dáta vybrať a časové rozmedzie, z ktorého chceme dáta vybrať. Ako ukážku dopytu si zoberieme celkový počet útokov a počet unikátnych IP adries za určité časové obdobie zachytené všetkými honeypotmi. V prvej časti nájdeme určený časový interval v akom chceme mať dáta agregované a následne kardinalitu zdrojových IP adries za daný interval. Za tým nasledujú všetky polia, podľa ktorých chceme agregovať.

```
resp = es.search(
    body={
  "aggs": {
    "3": {
      "date_histogram": {
        "field": "@timestamp",
        "fixed_interval": "30m",
        "time_zone": "Europe/Bratislava",
        "min_doc_count": 1
      },
      "aggs": {
        "2": {
          "cardinality": {
            "field": "src_ip.keyword"
```

```
          }
        }
      }
    }
  }
```

Atribút "must" obsahuje všetky podmienky, ktoré chceme aplikovať na dopyt. Prvá podmienka hovorí o tom, že chceme prehľadávať všetky záznamy (to zabezpečíme pomocou *), druhá podmienka určuje z akého honeypotu dáta požadujeme. V poslednej časti označenej atribútom "range" určujeme časové obdobie, za ktoré chceme dopytovať dáta. Nižšie uvádzame ukážku dopytu na honeynet.

```
  "query": {
    "bool": {
      "must": [
        {
          "query_string": {
            "query": "*",
            "analyze_wildcard": True,
            "time_zone": "Europe/Bratislava"
          }
        },
        {
          "query_string": {
                "query": "type:\"Adbhoney\" OR type:\"Ciscoasa\" OR
type:\"CitrixHoneypot\" OR type:\"ConPot\" OR type:\"Cowrie\" OR
type:\"Ddospot\" OR type:\"Dicompot\" OR type:\"Dionaea\" OR
type:\"ElasticPot\" OR type:\"Endlessh\" OR type:\"Glutton\" OR
type:\"Hellpot\" OR type:\"Heralding\" OR type:\"Honeypy\" OR
type:\"Honeysap\" OR type:\"Honeytrap\" OR type:\"Honeypots\" OR type:
\"Log4pot\" OR type:\"Ipphoney\" OR type:\"Mailoney\" OR
type:\"Medpot\" OR type:\"Rdpy\" OR type:\"Redishoneypot\" OR
type:\"Tanner\"",
            "analyze_wildcard": True,
            "time_zone": "Europe/Bratislava"
          }
        }
      ],
      "filter": [
        {
```

```
          "range": {
            "@timestamp": {
              "format": "strict_date_optional_time",
              "gte": "2023-12-06T23:00:00.000Z",
              "lte": "2023-12-20T00:30:00.000Z"
            }
          }
        }
      ],
      "should": [],
      "must_not": []
    }
  }
```

Časť "filter" ostáva v tomto prípade prázdna, avšak ak by sme chceli filtrovať napríklad všetky záznamy, ktoré smerovali na port 22, v takom prípade by to bolo obsiahnuté v tejto časti. Taktiež atribút "must_not" ostáva prázdny. Tento atribút slúži na pridanie podmienok, ktoré atribúty nesmú obsahovať.

Odpoveď na dopyt je vo formáte JSON, ktorý obsahuje základné údaje o dopyte a následne časť "aggregations". Tento segment obsahuje takzvané buckety, ktoré už obsahujú nami požadované dáta. Atribút "key_as_string" je časová pečiatka v intervale, v ktorom sme si určili pri dopyte, teda v našom prípade 30 minútový. Následne počet dokumentov, ktorý určuje v našom prípade počet útokov za daný časový interval. Hodnota atribútu "value" určuje v tomto prípade počet unikátnych IP adries, z ktorých boli tieto útoky vykonané.

```
{'took': 921,
 'timed_out': False,
 '_shards': {'total': 128, 'successful': 128, 'skipped': 0, 'failed':
0},
 'hits': {'total': {'value': 10000, 'relation': 'gte'},
  'max_score': None,
  'hits': []},
 'aggregations': {'3': {'buckets': [{'key_as_string': '2023-12-
07T18:00:00.000+01:00',
```

```
     'key': 1701968400000,
     'doc_count': 348,
     '2': {'value': 11}},
    {'key_as_string': '2023-12-07T18:30:00.000+01:00',
     'key': 1701970200000,
     'doc_count': 1107,
     '2': {'value': 35}},
...
```


Spracovaním týchto dát získaných z dopytu sa dostaneme ku časovému radu vo forme časová pečiatka, celkový počet útokov, unikátne IP adresy z ktorých boli útoky vykonané. Ukážka tohto spracovania je zobrazená na Obr. 16.

	Timestamp	Attack_counts	Unique_ips
0	2023-12-07T18:00:00.000+01:00	348	11
1	2023-12-07T18:30:00.000+01:00	1107	35
2	2023-12-07T19:00:00.000+01:00	1905	32
3	2023-12-07T19:30:00.000+01:00	859	28
4	2023-12-07T20:00:00.000+01:00	1088	37
...	...	...	...
586	2023-12-19T23:00:00.000+01:00	4360	111
587	2023-12-19T23:30:00.000+01:00	1718	69
588	2023-12-20T00:00:00.000+01:00	509	51
589	2023-12-20T00:30:00.000+01:00	665	36
590	2023-12-20T01:00:00.000+01:00	630	137

591 rows × 3 columns

Obr. 16 Ukážka spracovaných dát do formy časového radu

Podobným spôsobom sú dopytované a spracovávané ďalšie vybrané dáta. Ide o počet unikátnych IP adries, útoky použitím portu 22 (ssh), 23 (Telnet), 445 (SMB) a 5900 (VNC) a útoky na rôzne typy honeypotov (adbhoney, ciscoasa, citrixhoneypot, conpot, cowrie, ddospot, dicompot, dionaea, elasticpot, heralding, honeytrap, ipphoney,

mailoney, redishoneypot, tanner). Spomedzi týchto 21 časových radov sme vybrali užší rozsah na základe pravidelnosti a množstva dát. V niektorých prípadoch obsahoval časový rad nedostatočné množstvo počtu útokov čo by pri trénovaní modelov nedosahovalo požadovanú úroveň výsledkov.

3.2.3 Automatizácia dopytov

Dopyty, ktoré sme si v predchádzajúcej podkapitole ukazovali, sú manuálne. Teda pre získanie dát je nutné ho manuálne spustiť. To je však neefektívne z pohľadu fungovania systému pre zobrazovanie živých dát a ich predikcií.

Podobne ako skript na manuálne sťahovanie dát z Elasticsearch databázy, aj pri skripte na automatizované získavanie dát je potrebné sa do databázy pripojiť. Vytvorenie konektora na databázu a následné overenie spojenia je podrobnejšie popísané v kapitole 3.1.2 Dopyty nad Elasticsearch databázou.

Následne sme zadefinovali potrebné funkcie pre vytvorenie DataFrame pre rôzny časové rady. Ukážka funkcie vytvorenia DataFrame pre časové rady rozdelené podľa typu honeypotu.

```
def create_hps_df(resp):
    dates = []
    counts = []
    for key in resp["aggregations"]["3"]["buckets"]:
        dates.append(key["key_as_string"])
        counts.append(key["doc_count"])
    data_count_unique = {
        "Timestamp": dates,
        "Attack_counts": counts,
    }
    df = pd.DataFrame(data_count_unique)
    return df
```

Keďže počet útokov v časovom okne 30 minút môže byť nulový, je potrebné DataFrame doplniť nulami. Kvôli tomuto kroku je potrebné časové pečiatky prekonvertovať potomocu funkcie pd_to_datetime(), nastaviť indexovanie a následne doplniť nulami. Keďže nulové hodnoty sú po dopyte reprezentované hodnotami NaN, je potrebné použiť funkciu fillna(), ktoré tieto hodnoty zmení na 0.

```
def fill_with_zeros(df):
    df_pom = df.copy()
    df_pom['Timestamp'] = pd.to_datetime(df_pom['Timestamp'])
    df_pom = df_pom.set_index('Timestamp')
    df_filled = df_pom.resample('30T').sum().fillna(0)
    df_filled.reset_index(inplace=True)
    return df_filled
```

Ďalším krokom je zadefinovať si premenné. V tomto prípade sa jedná o zoznamy portov, podľa ktorých budeme sťahovať dáta pre vytvorenie časových radov podľa portov cez ktoré sa útočník snažil vykonať akcie. Taktiež vytvoríme zoznam všetkých honeypotov, ktoré sledujeme, pre vytvorenie časových radov podľa honeypotov, na ktoré boli útoky vykonávané.

V tejto fáze máme pripravené všetky potrebné metódy a premenné. Nasleduje vytvorenie dopytov, ktoré je bližšie popísane v kapitole 3.2.2 pod názvom „Dopyty nad Elasticsearch databázou“.

V poslednej fáze nad odpoveďou na dopyt vytvoríme DataFrame, vyplníme ho nulovými hodnotami a uložíme vo formáte csv.

```
df = create_ips_uniq_df(resp)
df_filled = fill_with_zeros(df)
df_all = df_filled[['Timestamp', 'Attack_counts']]
df_uniq = df_filled[['Timestamp', 'Unique_ips']]
df_uniq.rename(columns={'Unique_ips': 'Attack_counts'},
inplace=True)
save_to_csv(df_all, 'countIPS')
```

```
save_to_csv(df_uniq, 'uniqIPS')
print("df all uniq done.")
```

Takýmto spôsobom vytvoríme súbory pre všetky sledované porty, honeypoty a celkový počet útokov a unikátnych IP adries. Tento spôsob získavania dát je navrhnutý s dôrazom na jednoduché rozšírenie sledovaných parametrov, teda napr. pre rozšírenie sledovaných portov stačí pridať ďalšie porty do premenných. Takýto skript je pripravený na automatické spúšťanie.

Pre automatizáciu celého zberu dát sme sa rozhodli využiť vstavaný nástroj v operačnom systéme Linux, a to cron [42]. Cron je unixový nástroj, ktorý sa používa na plánovanie automatického spúšťania príkazov, skriptov a úloh. To je možné vykonať nastavením cron pravidla v rámci operačného systému Linux. Syntax programu Cron je veľmi jednoduchá. V prvých bodoch nastavíme interval, v akom sa má úloha opakovať, a následne zadáme príkaz, ktorý sa má vykonať (Obr. 17).

```
# ┌──────────── minuty (0 - 59)
# │ ┌──────────── hodiny (0 - 23)
# │ │ ┌──────────── den v měsíci (1 - 31)
# │ │ │ ┌──────────── měsíc (1 - 12)
# │ │ │ │ ┌──────────── den v týdnu (0 - 6) (od neděle do soboty)
# │ │ │ │ │
# │ │ │ │ │
# │ │ │ │ │
# * * * * * <příkaz k provedení>
```

Obr. 17 Ukážka syntaxe Cron úlohy

Keďže vytvárame časové rady s 30 minútovým oknom, tak pre nás bude vhodné nastaviť úlohu, aby sa spúšťala každých 30 minút. Cron úloha teda vyzerá nasledovne: */30 * * * * /usr/bin/python3 /home/jakub/diplomka/download_script_live.py

Týmto spôsobom máme zabezpečené pravidelné sťahovanie nových dát.

3.3 Ukladanie dát

Jedným z problémov, s ktorými sa stretávame, je ukladanie dát získaných z honeypotov. Pokusy o útoky prebiehajú neustále čo v prípade honeypotov a honeynetov generuje veľké množstvo dát. Dôležitou súčasťou je vybrať vhodný spôsob, akým tieto dáta budeme ukladať.

V rámci riešenia predikcie situačného povedomia potrebujeme dáta vo forme časových radov. Keďže honeypoty generujú dáta vo formáte JSON, bolo potrebné sa s týmto problémom vysporiadať. V rámci riešenia tohto problému sme zistili možnosť efektívneho tvorenia dopytov nad databázou elasticsearch. Dáta z dopytu sú síce vo formáte JSON (viď podkapitola 3.2.2), avšak netreba ich zložito spracovávať, pretože dopyt môže byť vytvorený tak, že už sú vo forme časových radov. Teda tieto dáta môžeme vložiť do štruktúry DataFrame z knižnice Pandas a následne uložiť do databázy vhodnej na prácu s časovými radmi. Teda tieto dáta môžeme vložiť do štruktúry DataFrame z knižnice Pandas. Následne bude vykonaná dátová analýza a použijú sa metódy strojového učenia.

3.3.1 Porovnanie databáz

Pre efektívne ukladanie a narábanie s dátami bolo potrebné vybrať vhodnú databázu. Keďže dáta sú vo forme časových radov, hlavná funkcionalita, ktorú musí databáza poskytovať je efektívna a jednoduchá práca s časovými radmi. Databáza musí taktiež spĺňať ďalšie podmienky ktoré sú bližšie špecifikované v nasledujúcom odseku a tabuľke Tab. 2.

Je na výber viacero možností ako napríklad Elastic [43], Influxdb [44] alebo Promotheus [45] a relačné databázy PostgreSQL [46] a MySQL [47]. V Tab. 2 sú najdôležitejšie parametre, ktorými sme sa riadili pri výbere databázy.

Názov	**Elastic**	**Infulxdb**	**Prometheus**	**PostgreSQL**	**MySQL**
Primárny databázový model	Vyhľadávací engine	Databáza časových radov	Databáza časových radov	Relačný	Relačný
Implementačný jazyk	Java	Go	Go	C	C, C++

Podporovaný operačný systém	Windows, Linux, OS X, Java VM, ...	Linux, OS X	Windows, Linux	Windows, Linux, OS X, ...	Windows, Linux, OS X, ...
Rok vydania	2010	2013	2015	1989	1995
Python	Áno	Áno	Áno	Áno	Áno
Otvorený kód	Áno	Áno	Áno	Áno	Áno
Integrácia do Grafany	Áno	Áno	Áno	Áno	Áno

Tab. 2 Porovnanie databáz

Doteraz zisteným problémom použitia riešenia, ktoré ponúka Elastic je náročnosť na hardvér a nedostupnosť voľne dostupnej verzie modelov strojového učenia, ktoré Elastic ponúka. Tento problém bude podrobnejšie rozoberaný v ďalšej časti. Z vyššie uvedených databáz sú dve relačného typu (MySQL a PostgreSQL) a ďalšie dve (Influxdb, Prometheus) sú databázy časových radov. Pre náš návrh je práve toto najdôležitejší parameter, keďže budeme chcieť ukladať dáta výlučne vo forme časových radov. Tieto dve novšie databázy zamerané na dáta vo forme časových radov sú implementované v jazyku Go, staršie MySQL a PostgreSQL sú v jazykoch C a C++ a Elastic v jazyku Java. Všetky databázy podporujú operačný systém Linux a okrem Influxdb aj Windows. Všetky databázy podporujú programovací jazyk Python, sú s otvoreným kódom a je podporovaná integrácia do prostredia Grafany.

Z vyššie spomenutých možností vyšla ako najvhodnejší kandidát na túto úlohu databáza InfluxDB. Na rozdiel od databázy časových radov Prometheus ponúka možnosť zapisovať aj hodnoty v textovom formáte, čo môže byť využité pri zložitejších návrhoch s potrebou dodatočného označenia záznamu.

3.3.2 Návrh databázy

Databáza InfluxDB má vlastné pomenovania pre rôzne elementy. Prvým je organizácia (**organization**), ktorá predstavuje pracovný priestor pre skupinu používateľov. Všetky ďalšie elementy musia patriť pod organizáciu. Všetky dáta sa v InfluxDB ukladajú do vedier (**bucket**). Bucket je v podstate databáza. Následne v buckete sú ďalšie elementy, ktoré usporadúvajú a indexujú dáta. Keďže sa jedná o databázu primárne určenú pre časové rady, všetky dáta musia mať stĺpec časovej

pečiatky (**timestamp**), ktoré sa ukladajú ako epochtime v nanosekundách. Ďalším dôležitým elementom je tzv. meranie (**measurement**). Measurement je stĺpec, ktorí člení dáta podľa meraní. Tieto názvy sú reťazce. Measurement funguje ako kontajner pre tagy, polia a časové pečiatky. Pole (**field**) obsahuje kľúč v premennej _field a hodnotu kľúča _value. Kľúč reprezentuje názov poľa, hodnota kľúča reprezentuje pridelenú hodnotu. Dôležitou skutočnosťou je, že polia nie sú indexované. Posledným elementom je označenie (**tag**). Tag taktiež poskytuje pár kľúč-hodnota avšak na rozdiel od poľa. Hodnota v tomto prípade musí byť reťazec [48].

Pre účely tejto práce sme navrhli dve databázy. Jedna databáza je určená pre nami vytvorený dataset časových radov útokov. Druhá databáza slúži pre zobrazovanie živej prevádzky.

Prvá databáza sa skladá z nasledujúcich elementov:

- **bucket**: test_hp
- **_measurement**: network_activity
- **_field**: cowrie, dionaea, heralding, honeytrap, no_attacks, port_22, port_23, port_445, port_5900, unique_ips

- **bucket**: test_hp
- **_measurement**: predictions
- **_field**: cowrie_svm, cowrie_dt, cowrie_xgb, ...

_measurement predikcií obsahuje v elemente _field všetky sledované časové rady v kombinácii so všetkými testovanými metódami.

Schéma druhej databázy, ktorá slúži na zobrazovanie živej prevádzky vyzerá nasledovne:

- **bucket**: test_live
- **_measurement**: honeypot_data
- **_field**: no_attacks, unique_ips, port_22, port_23, port_445, port_5900, cowrie, dionaea, heralding, honeytrap, ...

- **bucket**: test_live
- **_measurement**: predictions
- **_field**: no_attacks_svm, no_attacks_dt, no_attacks_xgb, ...

Podobne ako predchádzajúca databáza, v elemente _field sú taktiež kombinácie časových radov s testovanými predikciami.

Navrhnuté databázy sú v režime push, teda dáta do databázy posielame pomocou skriptov, ktoré sú detailne popísané v nasledujúcich podkapitolách. Takýto návrh umožňuje jednoducho pridávať alebo odoberať sledované časové rady, resp. dáta z ďalších predikčných metód.

3.3.3 Vkladanie dát do databázy

V úvode kapitoly 3.3.1 sme porovnávali rôzne možnosti databáz, ktoré prichádzali do úvahy pre naše použitie. Po analýze sme vybrali databázu InfluxDB. Na základe tohto výberu je prispôsobené aj vkladanie dát do databázy. Pri vytvorení databázového servera sme dostali základné informácie potrebné pre nasledujúce pripojenie sa, jedná sa o token.

Pre pripojenie do databázy používame knižnicu influxdb_client [49]. Konektor na databázu potrebuje tri parametre: token, názov organizácie a url adresu. Následne si vytvoríme klienta na zapisovanie a API pre zapisovanie a vytváranie dopytov.

```
write_client = influxdb_client.InfluxDBClient(url=url,
token=INFLUX_TOKEN, org=org)
write_api = write_client.write_api(write_options=SYNCHRONOUS)
query_api = write_client.query_api()
```

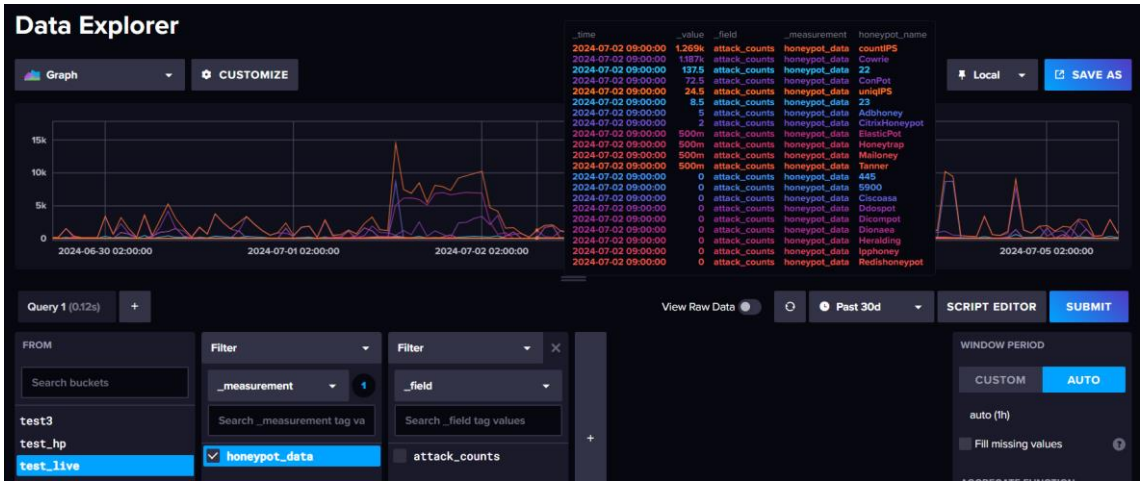
Vkladanie dát do databázy zahŕňa špecifikovanie takzvaného bodu (point), kde sa majú dáta zapísať a následne parametre tagu, poľa a časovej pečiatky.

```
point = Point('honeypot_data')\
  .tag('honeypot_name', name)\
  .field('attack_counts', row['Attack_counts'])\
  .time(datetime.strptime(row['Timestamp'], '%Y-%m-%d
%H:%M:%S%z'), WritePrecision.NS)
points.append(point)
```

Posledným krokom pre zapísanie dát do databázy je potrebné pomocou API pre zápis do databázy tieto dáta zapísať.

```
write_api.write(bucket=bucket, org=org, record=points)
```

Po pripojení sa cez webové rozhranie si môžeme skontrolovať úspešnosť zápisu dát do databázy. Na obrázku Obr. 18 je ukážka webového rozhrania InfluxDB. Po zvolení si databázy, do ktorej sme dáta zapísali a nastavení filtrov môžeme vidieť ukážku dát. Po priložení kurzoru vidíme v našom prípade konkrétne počty útokov na rôzne typy honeypotov resp. portov.



Obr. 18 Ukážka webového rozhrania InfluxDB

3.3.4 Automatizácia vkladania do databázy

Pre kompletnú automatizáciu celého procesu je potrebné po získaní dát ich vedieť automatizovane aj vložiť do našej databázy. Na základe kapitoly 3.3.3 kde sme opisovali vytvorený skript pre vkladanie dát do databázy manuálne, môžeme vytvoriť automatizovaný skript podobne ako pri získavaní dát.

Keďže vkladanie dát je priamo nadviazané na získavanie dát, môžeme vynechať spúšťanie cez úlohy nástroja cron, kde by sme museli riešiť synchronizáciu s úlohou získavania dát. Preto sme v závere skriptu pre získavanie dát doplnili nasledujúci kód.

```
exec(open('/home/Jakub/diplomka/db_live_push_script.py').read())
```

Tento kód, ktorý je umiestnení na konci skriptu získavania dát spustí skript pre vkladanie dát do databázy. Vďaka tomuto riešeniu sme sa vyhli synchronizácii procesu

spúšťania vkladania dát do databázy až po kompletnom skončení získavania. Skript vkladania dát do databázy je bližšie popísaný v predchádzajúcej podkapitole. Tým pádom tak ako sťahovanie dát, rovnako aj vkladanie dát do databázy sa vykonáva každých 30 minút.

3.4 Predikcia situačného povedomia

Po uložení dát vo forme časových radov nastane fáza dátovej analýzy. V tejto fáze sa budeme zaoberať použitím metód strojového učenia a neurónových sietí na predikciu situačného povedomia v kybernetickej bezpečnosti, ktoré budeme porovnávať so štatistickými metódami.

Ako bolo spomínané v predchádzajúcej časti, použitie riešenia, ktoré ponúka Elastic je nevýhodné kvôli nedostupnosti voľne dostupnej verzie modelov strojového učenia. Na základe tohto problému sme sa rozhodli navrhnúť vlastné modely na predikciu. Následne bolo potrebné tieto modely natrénovať a vyhodnotiť ich úspešnosť na dostupných dátach a v porovnaní so štatistickými metódami.

V tejto práci sme sa rozhodli použiť metódy SVM (Support Vector Machine), rozhodovacích stromov (Decision Tree), XGBoost a z neurónových sietí LSTM (Long Short-Term Memory) a GRU (Gated Reccurent Unit). Metódy neurónových sieti sme implementovali pomocou knižnice Tensorflow, ostatné pomocou knižnice sklearn.´

3.4.1 Príprava dát

Dôležitým prvkom pri využívaní modelov strojového učenia je príprava dát. Prvý krok prípravy dát, ktorý zahŕňa získanie dát z honeypotov a honeynetov a následnú konverziu do formy časových radov, resp. súborov vo formáte csv sme si popísali v kapitole 3.2. V tomto kroku je potrebné dáta vhodne upraviť pre vstup do metód strojového učenia.

Jedným so spôsobov úpravy dát je použitie tzv. lagu, čo predstavuje posun dát o niekoľko časových jednotiek. Toto posunutie zabezpečí vytvorenie nových atribútov a paternov na základe predchádzajúcich dát. Ako príklad ukážky môžeme zobrať celkový počet zaznamenaných útokov. Tento časový rad sa skladá z časovej pečiatky a celkového počtu útokov (Obr. 19).

	timestamp	no_attacks
0	2023-12-08 01:00:00+01:00	4895
1	2023-12-08 01:30:00+01:00	5135
2	2023-12-08 02:00:00+01:00	3337
3	2023-12-08 02:30:00+01:00	820
4	2023-12-08 03:00:00+01:00	1444
...	...	...
5179	2024-03-24 22:30:00+01:00	11662
5180	2024-03-24 23:00:00+01:00	1524
5181	2024-03-24 23:30:00+01:00	445
5182	2024-03-25 00:00:00+01:00	338
5183	2024-03-25 00:30:00+01:00	591

5184 rows × 2 columns

Obr. 19 Časový rad celkového počtu útokov

Následne sme vytvorili dáta s posunmi. Dáta boli posunuté postupne o 48 hodín, čím vzniklo 48 nových časových radov s posunmi (Obr. 20).

	timestamp	no_attacks	lag_1	lag_2	lag_3	lag_4	lag_5
48	2023-12-09 01:00:00+01:00	341	1199.0	3896.0	1200.0	733.0	3779.0
49	2023-12-09 01:30:00+01:00	381	341.0	1199.0	3896.0	1200.0	733.0
50	2023-12-09 02:00:00+01:00	1663	381.0	341.0	1199.0	3896.0	1200.0
51	2023-12-09 02:30:00+01:00	458	1663.0	381.0	341.0	1199.0	3896.0
52	2023-12-09 03:00:00+01:00	1782	458.0	1663.0	381.0	341.0	1199.0
...	...	...	...	...	...	...	...
5179	2024-03-24 22:30:00+01:00	11662	5331.0	346.0	303.0	382.0	2192.0
5180	2024-03-24 23:00:00+01:00	1524	11662.0	5331.0	346.0	303.0	382.0
5181	2024-03-24 23:30:00+01:00	445	1524.0	11662.0	5331.0	346.0	303.0
5182	2024-03-25 00:00:00+01:00	338	445.0	1524.0	11662.0	5331.0	346.0
5183	2024-03-25 00:30:00+01:00	591	338.0	445.0	1524.0	11662.0	5331.0

5136 rows × 50 columns

Obr. 20 Časový rad celkového počtu útokov s posunmi

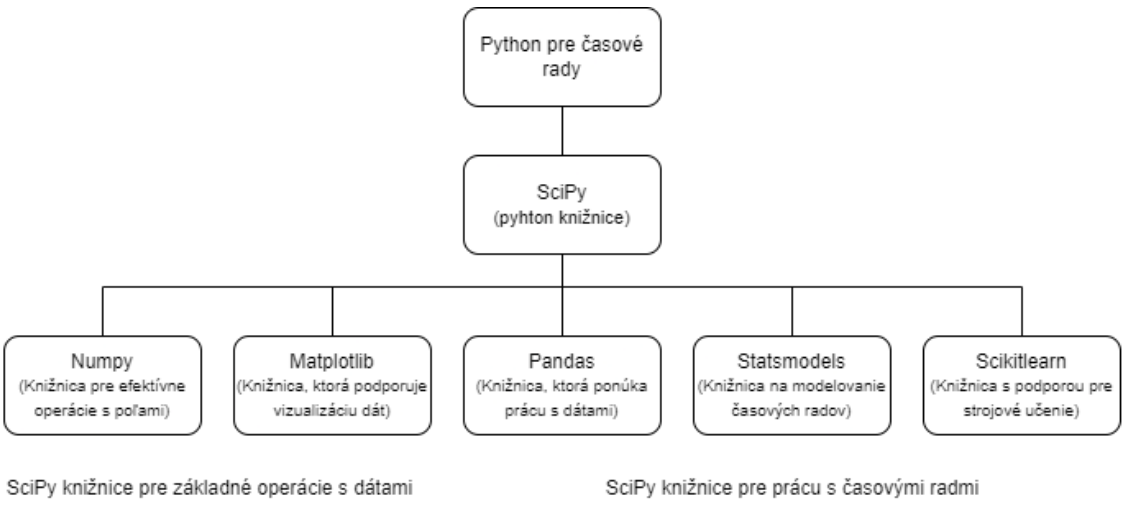
Ďalším používaným spôsobom je rozdelenie dát na tréningovú a testovaciu množinu. Týmto spôsobom dokážeme overiť, ako dobre modely generalizujú na nových

dátach. Tréningová množina je poskytnutá modelu počas učiacej fázy. Na týchto dátach sa model učí. Naopak testovacia množina sa používa na ohodnotenie modelu. Použitím kontroly na testovacích dátach môžeme objektívne posúdiť, či je model schopný generalizovať a následne dostatočne presne predikovať. V našom prípade sme množinu rozdelili na tréningovú a testovaciu v pomere 80:20.

Posledná úprava dát, ktorej sme sa venovali je štandardizovanie. Na túto úpravu sme použili StandardScaler z knižnice sklearn. Štandardizácia je metóda prípravy údajov, ktorá zahŕňa úpravu vstupných dát tak, že sa najprv vycentrujú, teda od každého bodu sa odpočíta priemer a následne sa vydelia štandardnou odchýlkou. Týmto spôsobom vznikajú dáta s priemerom 0 a štandardnou odchýlkou 1.

3.4.2 Výber vhodných knižníc

Pri výbere a návrhu modelov bude potrebné sa rozhodnúť, aké knižnice budú použité. Z doterajšej analýzy tohto problému sme sa dopracovali ku knižnici SciPy. Je to súbor Python knižníc, ktoré dokážu efektívne pracovať s dátami vo forme časových radov a taktiež ponúkajú implementované metódy strojového učenia [36].



Obr. 21 Knižnica SciPy [50]

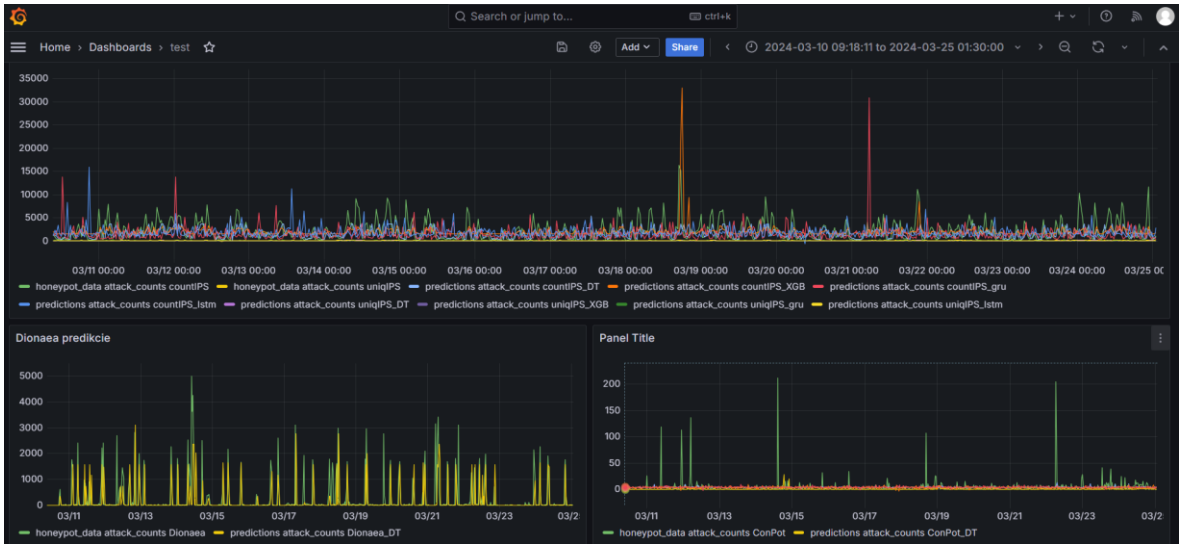
Ďalšiu knižnicu, ktorú sme sa rozhodli použiť, je knižnica Tensorflow. Je to knižnica s otvoreným zdrojovým kódom vyvíjaná spoločnosťou Google Brain Team. Je navrhnutá tak, aby uľahčovala implementáciu rôznych modelov strojového učenia,

predovšetkým neurónových sietí. Poskytuje prostredie pre modelovanie a trénovanie neurónových sietí na rôznych platformách (CPU, GPU, TPU) [51].

3.5 Vizualizácia

Vizualizácia dát hrá kľúčovú úlohu v oblasti situačného povedomia kybernetickej bezpečnosti. Hlavnou výhodou vizualizácie dát je, že umožňuje prehľadné a intuitívne zobrazenie komplexných súborov informácií, čo pomáha k lepšiemu a rýchlejšiemu odhaleniu hrozieb a anomálií. Taktiež pomáha odhaliť vzorce, trendy a spojitosti, ktoré by mohli byť iba pohľadom na dáta ťažko rozoznateľné.

Pre zobrazovanie dát sme sa rozhodli použiť nástroj Grafana [52]. Je to multiplatformový nástroj s otvoreným kódom na analýzu dát a ich vizualizáciu. Obrovskou výhodou tohto nástroja je detailná dokumentácia a priama implementácia spojenia s databázou InfulxDB. Dokáže robiť dopyty, vizualizovať dáta v prehľadovom panely a vytvárať upozornenia, ktoré budeme spomínať v ďalšej časti. Príklad toho, ako vyzerá prehľadový panel, je zobrazený na obrázku Obr. 22. Výhodou systému Grafana je, že prehľadové panely sú plne variabilné. To znamená že ich vieme upraviť, aby zobrazovali dáta, ktoré potrebujeme.



Obr. 22 Príklad vizualizácie časových radov pomocou nástroja Grafana [40]

3.5.1 Inštalácia a konfigurácia nástroja Grafana

Ako bolo spomenuté v kapitole o implementačných požiadavkách (3.1), jednou z nich je bežiaca služba Grafana. Proces inštalácie je podrobne spísaný v dokumentácii [53], a preto sa mu nebudeme do detailov venovať. Pre správne fungovanie Grafany je potrebné mať splnené minimálne požiadavky a následne prejsť inštalačným procesom. Ďalším krokom pre úspešné spustenie je úprava konfiguračných súborov. Tento proces je taktiež detailne zdokumentovaný [54]. Najdôležitejšie parametre, ktoré musia byť správne nakonfigurované sú prihlasovacie meno, heslo a port na ktorom má byť služba dostupná. Po úspešnej konfigurácii môžeme službu spustiť. Spustenie a overenie služby je možné pomocou nasledujúcich príkazov:

```
sudo systemctl start grafana-server
sudo systemctl status grafana-server
```

Po úspešnom naštartovaní by mal byť status služby stanovený ako „active“ (viď. Obr. 23).

```
jakub@research-jm:~/diplomka$ sudo systemctl status grafana-server
[sudo] password for jakub:
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-07-23 10:48:22 UTC; 1 week 0 days ago
       Docs: http://docs.grafana.org
   Main PID: 755 (grafana)
      Tasks: 27 (limit: 9388)
     Memory: 224.9M
        CPU: 30min 33.245s
     CGroup: /system.slice/grafana-server.service
             ├─755 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run
             └─893 /var/lib/grafana/plugins/marcusolsson-csv-datasource/gpx_csv-datasource_linux_amd64
```

Obr. 23 Ukážka overenia statusu služby Grafana

Následne bude služba dostupná vo webovom rozhraní na porte, ktorý sme použili pri konfigurácii (default 3000). Ďalším krokom je vytvorenie spojenia s našou databázou.

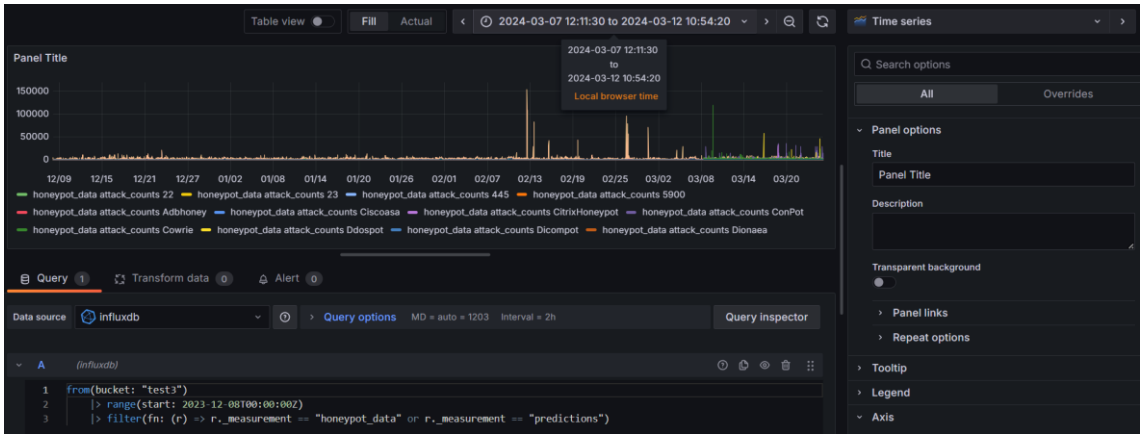
Pre vytvorenie spojenie medzi grafanou a databázou je opäť možné postupovať podľa dokumentácie [55]. Po výbere názvu spojenia, je potrebné si vybrať dopytovací jazyk. InfluxDB ponúka v nástroji Grafana na výber z troch dopytovacích jazykov: SQL, Flux a InfluxQL, pričom my sme sa rozhodli pre druhú možnosť, teda Flux. V časti HTTP je potrebné zadať URL adresu databázového servera InfluxDB. Ďalším krokom je výber autentifikácie, kde je možné si vybrať z viacerých možností, v našom prípade sme zvolili

základnú autentifikáciu pomocou prihlasovacieho mena a hesla. Posledný krok pri vytváraní spojenia je ID organizácie, ktoré bolo vytvorené pri konfigurovaní databázového servera. V tejto fáze je spojenie s databázovým serverom úspešne vytvorené.

3.5.2 Vytvorenie prehľadového panelu

Po inštalácii, konfigurácii a následnom spojení služby Grafana s databázovým serverom je možné cez webové rozhranie vytvárať prehľadové panely. Prehľadové panely slúžia na vizualizáciu dát. Súčasťou jedného prehľadového panelu môže byť viacero panelov s rôznymi metrikami (Obr. 21). Ako prvé musíme vytvoriť panel s vizualizáciou. Grafana ponúka na výber rôzne typy vizualizácií, ktoré vieme zvoliť v pravom hornom rohu obrázku Obr. 23 (v našom prípade je zvolené Time series, teda časový rad). Môže sa jednať o formát časového radu, heat mapy, tabuľky alebo iných. V pravom rolovacom paneli je možné nastaviť ďalšie parametre panelu ako napríklad názov, legendu, typ čiar, škálovanie a mnoho ďalších. Väčšina z týchto úprav je vizuálnych.

V dolnej časti obrázku Obr. 24 je konfigurácia panela z pohľadu dát, ktoré bude zobrazovať. Keďže ako databázu využívame InfluxDB a máme takto vytvorené aj spojenie, tak ako zdroj dát (Data source) sme určili influxdb (názov z časti konfigurácie spojenia). Následne je potrebné vytvoriť dopyt, ktorý je v jazyku, ktorý sme si zvolili pri konfigurácii, v našom prípade Flux.



Obr. 24 Ukážka konfigurácie panela s vizualizáciou

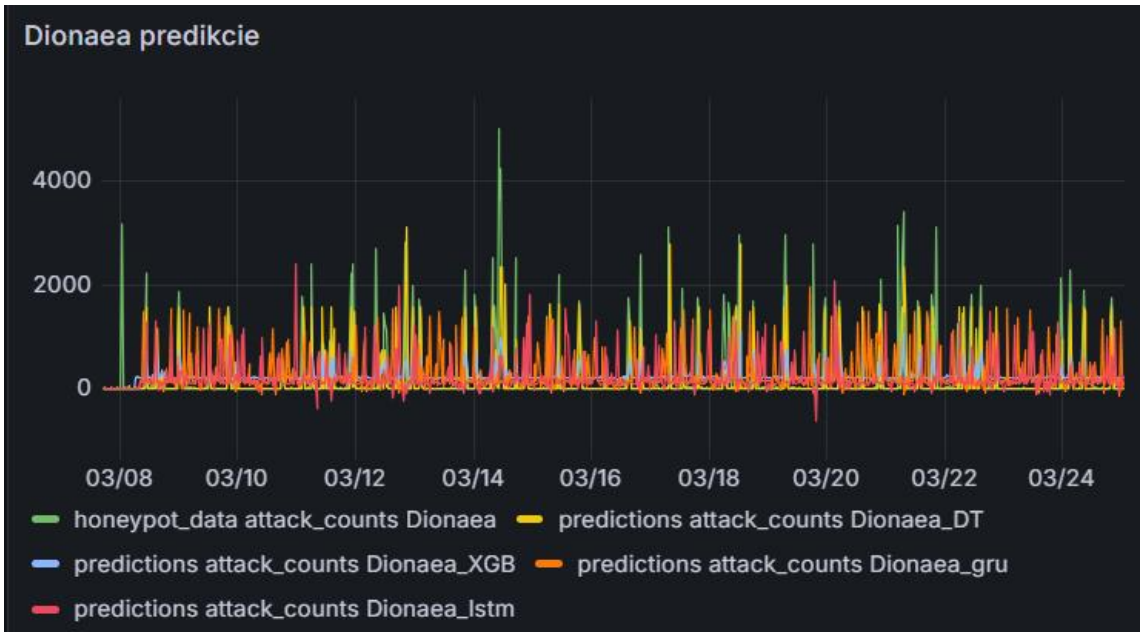
```
from(bucket: "test3")
```

```
          |> range(start: 2023-12-08T00:00:00Z)
           |> filter(fn: (r) => r._measurement == "honeypot_data" or
r._measurement == "predictions")
```

Dopyt, ktorý je zobrazený na obrázku nám zobrazí dáta z bucketu test3, ktoré sú v časovom horizonte, ktoré začína 8.12.2023 o 00:00 a spĺňajú podmienku, že _measurement je honeypot_data alebo predictions. To znamená, že z databázy test3 zoberieme všetky dáta, ktoré sú čisté dáta z honeypotov alebo predikcie.

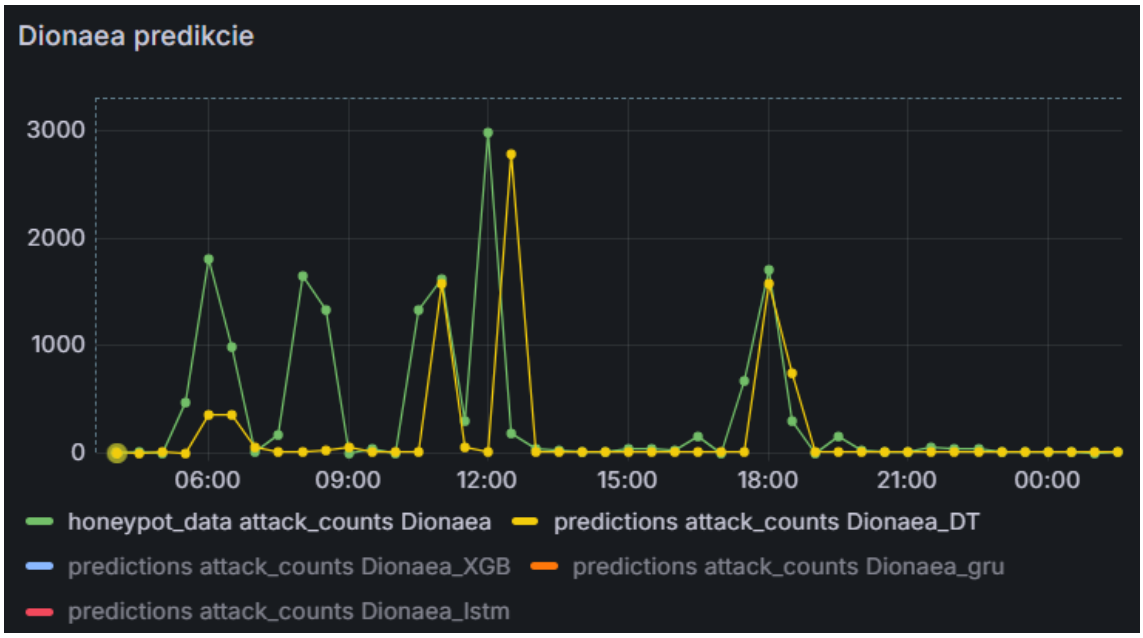
Zložitejší dopyt obsahuje napríklad vizualizácia počtu útokov zameraného na honeypot Dionaea spolu s predikciami všetkých nami skúšaných modelov. Vizualizácia tohto dopytu vyzerá nasledovne.

```
      from(bucket: "test3")
        |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
         |> filter(fn: (r) => (r._measurement == "honeypot_data" or
r._measurement == "predictions") and
        (r.honeypot_name == "Dionaea") or
          r.honeypot_name == "Dionaea_DT" or r.honeypot_name ==
"Dionaea_SVM" or r.honeypot_name == "Dionaea_XGB" or r.honeypot_name ==
"Dionaea_lstm" or r.honeypot_name == "Dionaea_gru")
```



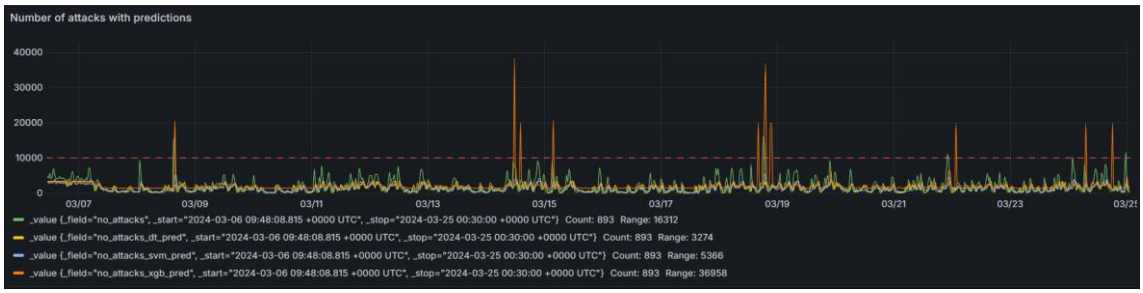
Obr. 25 Ukážka vizualizácie počtu útokov na honeypot Dionaea

Ako môžeme vidieť na obrázku Obr. 25, zobrazenie všetkých predikcií je vcelku neprehľadné. V tomto nástroji sa dá jednoducho, kliknutím na jednotlivé názvy v legende panelu, vybrať iba požadované. Podobným intuitívnym spôsobom je možné sa zamerať iba na podozrivé resp. záujmové obdobie. Stačí v paneli označiť oblasť, ktorú si chceme priblížiť. Takýto pohľad potom zachytáva dáta omnoho podrobnejšie, ako môžeme vidieť na obrázku Obr. 26.



Obr. 26 Ukážka filtrovania v nástroji Grafana

Ďalšou zaujímavou funkciou je nastavenie zobrazenia prahu v paneli vizualizácie. Takýto prah môže pomôcť analytikovi vizuálne zistiť potenciálny problém. Prah sa nastavuje pri vytváraní resp. úprave panelu v pravej spodnej časti pod záložkou Thresholds. V tejto časti môžeme zvoliť či sa má jednať o konkrétne číslo, alebo percentuálnu hodnotu. Následne sa prah zobrazí v paneli nasledovne (Obr. 27).



Obr. 27 Ukážka nastavenia prahu vo vizualizácii

V ďalšej podkapitole sa bližšie pozrieme, ako sa dajú v nástroji Grafana generovať upozornenia, ktoré nás dokážu informovať rôznymi kanálmi o potenciálnych hrozbách. V našom prípade to môže byť veľmi užitočné, pretože dáta, ktoré zobrazujeme obsahujú predikcie daných sledovaných aktivít. Teda takýmto spôsobom by sme boli schopní dostávať upozornenia vopred.

3.6 Generovanie upozornení

Cieľom tejto práce je predikovať situačné povedomie v kybernetickej bezpečnosti na základe ktorého je možné zavádzať bezpečnostné opatrenia vopred. Preto je dôležité zaviesť spôsob, akým by nás systém upozornil na vzniknuté anomálie, resp. blížiace sa hrozby.

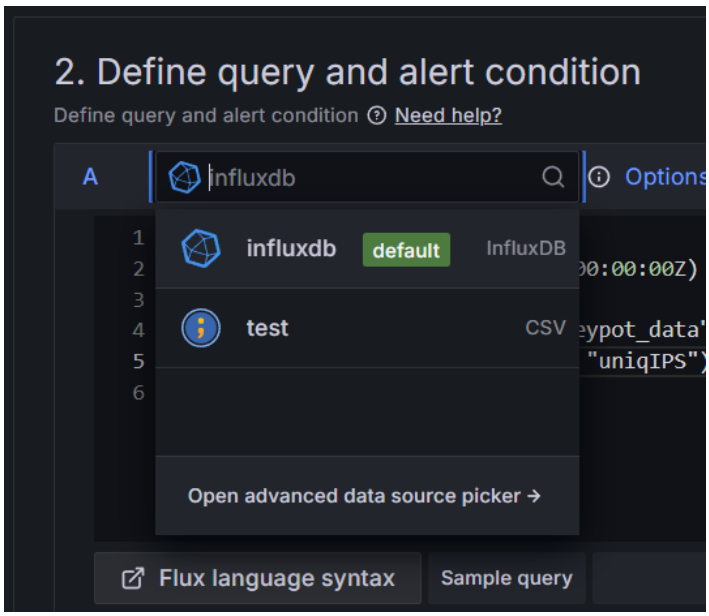
Ako je spomínané v predchádzajúcej časti, na vizualizáciu sme sa rozhodli použiť nástroj Grafana [39]. Jednou z funkcionalít tohto nástroja je vytváranie upozornení. Grafana dokáže vytvárať upozornenia na základe nami definovaných pravidiel. Pravidlá na vytváranie upozornení môžu byť vytvorené na základe dát z rôznych zdrojov čo nám umožňuje kombinovať dáta na vytváranie upozornení. Následne nás môže o anomálii informovať prostredníctvom vstavaného agenta na upozornenia, odosielaním upozornení

do iného agenta na manažovanie upozornení alebo prostredníctvom emailu, SMS či do služby Slack. Vytvorenie upozornení prechádza troma krokmi:

- Vytvorenie kontaktného bodu
- Vytvorenie pravidla pre generovanie upozornenia
- Vytvorenie notifikačnej politiky

V časti Alerting, Contact points je potrebné ako prvé vytvoriť kontaktný bod. Kontaktný bod potrebuje názov, spôsob integrácie napr. email a emailovú adresu, na ktorú budú upozornenia následne zasielané.

Vytváranie pravidiel pre generovanie upozornení v prostredí Grafana je nasledujúce. Po zadaní názvu pravidla, je potrebné definovať dopyt. Dopyt určuje dáta, nad ktorými sa budú vyhodnocovať podmienky. Ak máme k dispozícii väčšie množstvo zdrojov dát, je potrebné si vybrať zdroj dát, z ktorého chceme čerpať dáta (Obr. 28).



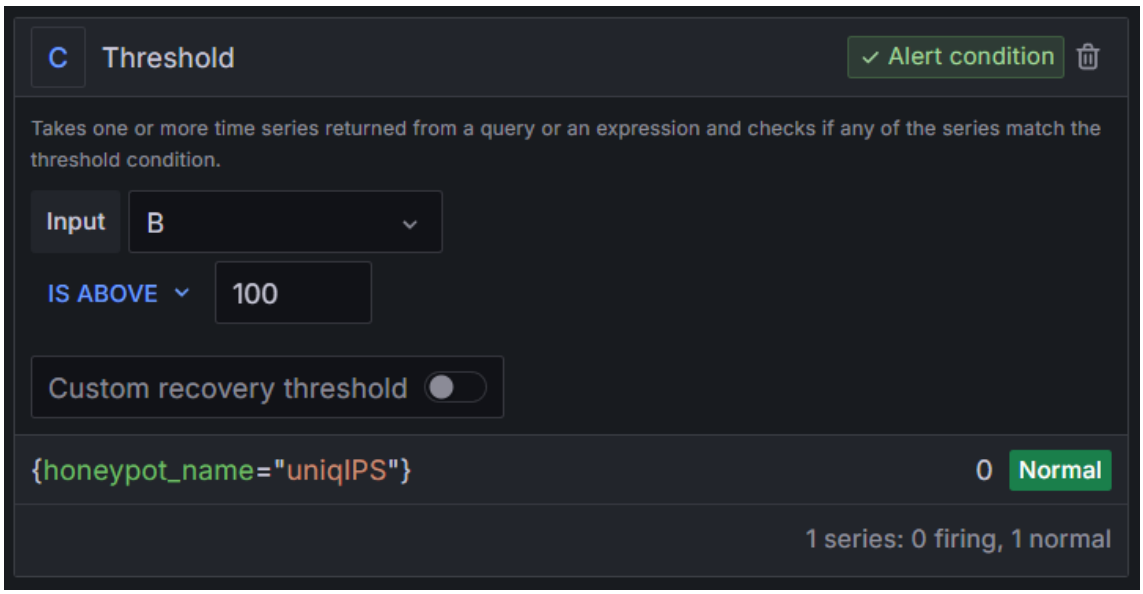
Obr. 28 Ukážka výberu zdroju dát

Následne vytvoríme dopyt pre dáta, ktoré chceme pravidlom sledovať. Tento dopyt zoberie dáta z bucketu test_live, teda z databázy, kde prichádzajú živé dáta. Následne vyfiltruje dáta, od dátumu 4.1.2024, ktoré sú v _measurement honeypot_data a jedná sa o unikátne IP adresy.

```
from(bucket: "test_live")
    |> range(start: 2024-04-01T00:00:00Z)
    |> filter(fn: (r) =>
```

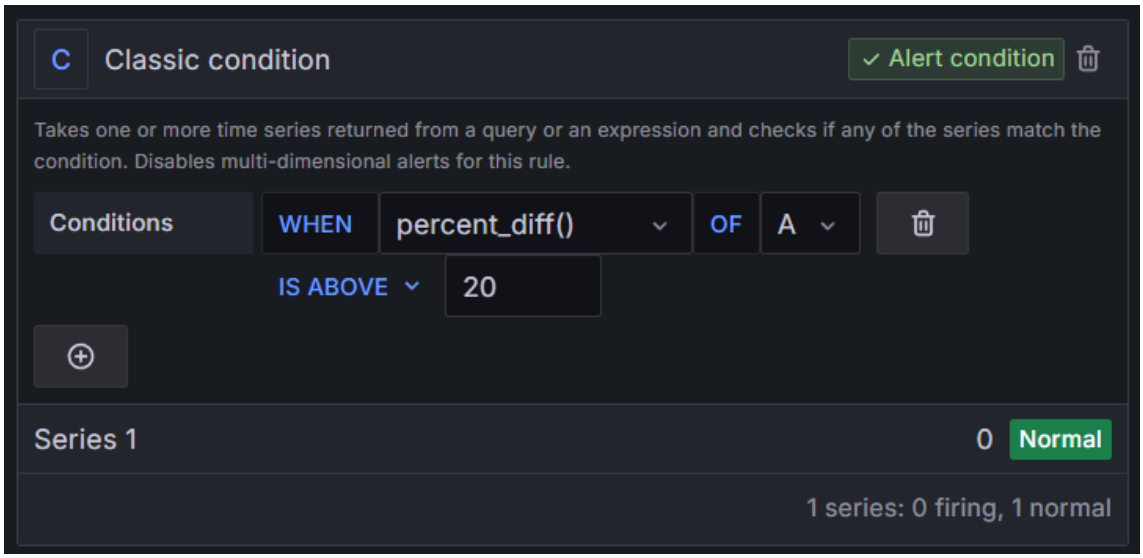
```
                (r._measurement == "honeypot_data")  and (r._field ==
"attack_counts")
                and (r.honeypot_name == "uniqIPS")
            )
```

Ako ďalší krok musíme nastaviť pravidlo, na základe ktorého sa rozhodne, či sa upozornenie pošle. Pre túto podmienku si môžeme vybrať z viacerých možností, jednou z nich je prah. V tomto prípade nakonfigurujeme upozornenie tak, že ak prekročí nami daný prah, tak sa odošle upozornenie. Napríklad ako vidíme na obrázku Obr. 29, ak počet unikátnych IP adries presiahne 100 v danom časovom okne, vytvorí sa upozornenie.



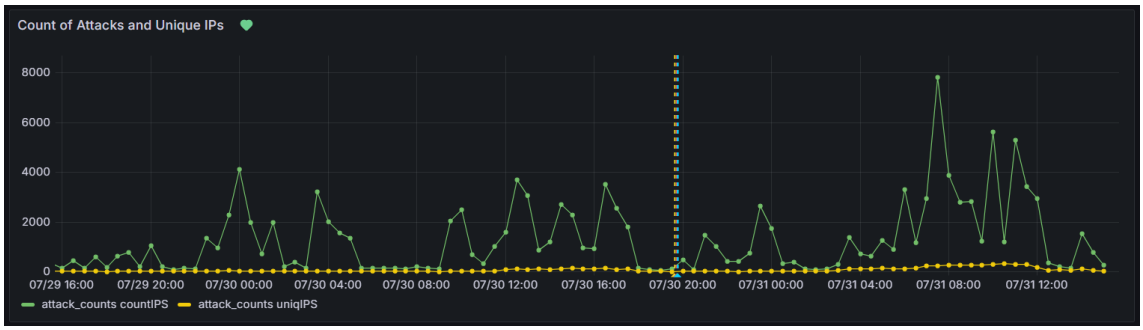
Obr. 29 Príklad vytvorenia prahu pre upozornenie

Ďalšou možnosťou je vytvorenie zložitejšej podmienky, kde vieme zadefinovať rôzne parametre. Ako príklad môžeme uviesť podmienku z obrázku Obr. 30. Táto podmienka vytvorí upozornenie, ak vznikne percentuálny rozdiel oproti predchádzajúcej hodnote o viac ako 20%. Takáto podmienka dokáže zachytiť prudký nárast napríklad v počte zachytených unikátnych adries.



Obr. 30 Príklad vytvorenia podmienky pre upozornenie

Následne je potrebné špecifikovať priečinok, do ktorého sa dané pravidlo zaradí (pre lepšiu prehľadnosť) a evaluačnú skupinu. Evaluačná skupina zabezpečuje, že pravidlá v rovnakej skupine sa vyhodnocujú konkurentne v tom istom časovom intervale. Ďalším krokom je konfigurácia označení, ktoré slúžia ako informácia v upozornení, o aký typ sa jedná, a zároveň na ich základe vieme vytvárať notifikačné politiky. Ako posledný parameter, ktorý musíme zvoliť je zobrazovací panel a z neho konkrétny panel v ktorom sa upozornenie bude zobrazovať. Vytvorenie upozornenia v paneli vyzerá nasledovne.



Obr. 31 Zobrazenie upozornenia v paneli

Záver

V tejto práci sme sa zaoberali teoretickou stránkou honeypotov a honeynetov, ako aj konkrétnym návrhom riešenia problému predikcie situačného povedomia v kybernetickej bezpečnosti. Venovali sme sa definícii honeypotu a rozdeleniu honeypotov do kategórií podľa role, akú napĺňajú. Podľa interakcie, akú s nimi môže útočník mať a podľa zbieraných dát a typov útokov, ktoré sa na nich dajú vykonávať. Taktiež sme sa zaoberali architektúrou platformy T-Pot a typmi honeypotov, ktoré obsahuje.

V ďalšej časti práce sme skúmali oblasť strojového učenia, kde sme sa v úvode venovali najmä základnému pochopeniu daného pojmu a metódam, ktoré sa zvyknú používať. Následne sme sa zaoberali štatistickými metódami používanými pri predikcii časových radov. V ďalšej časti sme sa pozreli na neurónové siete a ich použitie v oblasti kybernetickej bezpečnosti pri predikcii na časových radoch. Z tejto analýzy sme sa bližšie zaoberali LSTM a GRU rekurentným neurónovým sieťam. Metódami strojového učenia sme sa zaoberali v závere, kde sme sa bližšie pozreli na metódy SVM, rozhodovacích stromov a XGBoost. V závere tejto kapitoly sme vyhodnotili a porovnali presnosť skúmaných modelov pomocou metrík MAE, MSE a MASE. Metriky MAE a MSE sledujú chybu v predikcie, MASE sleduje výkonnosť modelu oproti naivnej predikcii. Taktiež sme na porovnanie výkonnosti modelov medzi sebou použili Diebold Mariano test. Z výsledkov tejto časti sme došli k záveru, že modely neurónových sietí LSTM a GRU dokážu presnejšie predikovať na tomto type dát.

V časti návrhu riešenia sme predstavili náš návrh, ktorý pozostával z piatich častí: zber dát, ukladanie dát, dátová analýza a strojové učenie, vizualizácia a generovanie upozornení. V prvej podkapitole sme popísali požiadavky pre správne fungovanie daného systému. Následne sme sa podrobne venovali každej časti návrhu, pri ktorých sme rozoberali úvodné problémy a následné riešenia. Taktiež sme sa venovali automatizácii celého procesu od zberu dát až po ich predikovanie a zobrazovanie. V časti predikcie situačného povedomia sme sa venovali od výberu vhodných knižníc, cez prípravu dát, až po verifikáciu a porovnanie modelov.

Z výsledkov práce môžeme konštatovať, že na predikciu situačného povedomia v oblasti kybernetickej bezpečnosti je vhodné použiť modely neurónových sietí LSTM a GRU aj napriek ich náročnejšej výpočtovej zložitosti. Tieto modely sú presnejšie ako štatistické metódy AR, MA a ARIMA a taktiež aj ako modely strojového učenia SVM,

rozhodovacích stromov a XGBoost. Samozrejme tieto výsledky závisia od povahy datasetu a vytvorených časových radov.

Prvým cieľom práce bolo vytvoriť dátovú sadu časových radov pre analýzu situačného povedomia v kybernetickej bezpečnosti. Tomuto cieľu sa venujeme v tretej kapitole v častiach zberu a ukladania dát. Druhým cieľom práce bolo preskúmať existujúce metódy strojového učenia na predikciu situačného povedomia v kybernetickej bezpečnosti. Tento cieľ sa nám podarilo naplniť v prvej a druhej kapitole. Navrhnutiu modelu na predikciu situačného povedomia v kybernetickej bezpečnosti a následnému vyhodnoteniu s existujúcimi štatistickými metódami sme sa venovali v druhej kapitole. Posledný cieľ, a síce návrh a implementácia systému na interaktívnu grafickú reprezentáciu jednokrokových a viackrokových predikcií, je bližšie rozobratý v tretej kapitole.

Počas tvorby tejto záverečnej práce bolo najväčším problémom prepojenie jednotlivých nástrojov a systémov, zabezpečenie ich správneho fungovania a automatizácie procesu od zberu dát až po ich zobrazovanie. Preto túto časť hodnotíme aj ako najväčší prínos tejto práce, teda vytvorenie nástroja, resp. systému pre interaktívne grafické reprezentovanie situačného povedomia.

Na základe dosiahnutých výsledkov v práci vidíme potenciál pre ďalšie rozšírenie a ďalší výskum v danej oblasti najmä v skúmaných modeloch strojového učenia a neurónových sietí. Ako zaujímavú tému vidíme kombináciu rôznych modelov na vytvorenie nových, prepracovanejších modelov a ich predikcií. Taktiež je možné sa zaoberať pridaním ďalších typov honeypotov a následne skúmaniu ďalších vytvorených časových radov. V neposlednom rade vidíme priestor na rozšírenie grafického rozhrania a podmienok pre vytváranie upozornení.

Zoznam použitej literatúry

[1] Husák, M., Jirsík, T., & Yang, S. J. SoK: contemporary issues and challenges to enable cyber situational awareness for network security. In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*, August 2020, pp. 1-10.

[2] Honeypots in espionage fiction. [online]. Dostupné z: https://en.wikipedia.org/wiki/Honeypots_in_espionage_fiction

[3] Spitzner, L. *Honeypots: Tracking Hackers*. USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] What is a honeypot? [online]. Dostupné z: https://usa.kaspersky.com/resource-center/threats/what-is-a-honeypot

[5] Mairh, A., Barik, D., Verma, K., & Jena, D. Honeypot in network security: a survey. In: *Proceedings of the 2011 International Conference on Communication, Computing & Security (ICCCS '11)*. New York, NY, USA: Association for Computing Machinery, 2011, pp. 600-605. https://doi.org/10.1145/1947940.1948065

[6] Mokube, I., & Adams, M. Honeypots: concepts, approaches, and challenges. In: *Proceedings of the 45th annual southeast regional conference (ACM-SE 45)*. New York, NY, USA: Association for Computing Machinery, 2007, pp. 321-326. https://doi.org/10.1145/1233341.1233399

[7] Sokol, P., Pekarčík, P., & Bajtoš, B. Data Collection and Data Analysis in Honeypots and Honeynets. [online]. Dostupné z: https://web.archive.org/web/20180421062733id_/http://spi.unob.cz/papers/2015/2015-19.pdf

[8] Abbasi, F., & Harris, R. Experiences with a Generation III virtual Honeynet. *In: Proceedings of the 2009 ATNAC*; 2009, pp. 1-6. 10.1109/ATNAC.2009.5464785.

[9] What Is an Email Honeypot and How To Avoid It. [online]. Dostupné z: https://selzy.com/en/blog/what-is-an-email-honeypot-and-how-to-avoid-it/

[10] What is a honeypot? How they are used in cybersecurity. [online]. Dostupné z: https://www.malwarebytes.com/blog/news/2021/05/what-is-a-honeypot-how-they-are-used-in-cybersecurity

[11] TPOT. [online]. Dostupné z: https://github.com/telekom-security/tpotce

[12] Docker. [online]. Dostupné z: https://www.docker.com/

[13] Elastic Stack. [online]. Dostupné z: https://www.elastic.co/elastic-stack

[14] ADBHoney. [online]. Dostupné z: https://github.com/huuck/ADBHoney

[15] Cisco ASA honeypot. [online]. Dostupné z: https://github.com/Cymmetria/ciscoasa_honeypot

[16] Conpot. [online]. Dostupné z: https://github.com/mushorg/conpot?tab=readme-ov-file

[17] Cowrie. [online]. Dostupné z: https://github.com/cowrie/cowrie

[18] DDoSPot. [online]. Dostupné z: https://github.com/aelth/ddospot

[19] Dionaea. [online]. Dostupné z: https://dionaea.readthedocs.io/en/latest/introduction.html

[20] ElasticPot. [online]. Dostupné z: https://gitlab.com/bontchev/elasticpot

[21] Endlessh. [online]. Dostupné z: https://github.com/skeeto/endlessh

[22] Heralding. [online]. Dostupné z: https://github.com/johnnykv/heralding

[23] Honeytrap. [online]. Dostupné z: https://github.com/armedpot/honeytrap/

[24] IPPHoney. [online]. Dostupné z: https://gitlab.com/bontchev/ipphoney

[25] Log4Pot. [online]. Dostupné z: https://github.com/thomaspatzke/Log4Pot

[26] Honeypots. [online]. Dostupné z: https://github.com/qeeqbox/honeypots

[27] Endsley, M.R. Situation awareness global assessment technique (SAGAT). In: *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*; 1988, pp. 789-795 vol.3.

[28] Husák, M., Komárková, J., Bou-Harb, E., & Čeleda, P. Survey of Attack Projection, Prediction, and Forecasting in Cyber Security. *IEEE Communications Surveys & Tutorials*. 2019, 21. issn: 1553-877X. https://doi.org/10.1109/COMST.2018.2871866

[29] Chuhg, A. MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared – Which Metric is Better? (2020). [online]. Dostupné z: https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e

[30] Diebold mariano test. [online]. Dostupné z: https://real-statistics.com/time-series-analysis/forecasting-accuracy/diebold-mariano-test/

[31] Diebold Mariano. [online]. Dostupné z: https://pypi.org/project/dieboldmariano/

[32] Endsley, M.R. Situation awareness global assessment technique (SAGAT). In: *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*; 1988, pp. 789-795 vol.3.

[33] Sokol, P., et al. Network security situation awareness forecasting based on statistical approach and neural networks. *Logic Journal of the IGPL*. 2023, 31.2: 352-374.

[34] Understanding LSTM. [online]. Dostupné z: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[35] SVM. [online]. Dostupné z: https://scikit-learn.org/stable/modules/svm.html

[36] Decision Tree. [online]. Dostupné z: https://scikit-learn.org/stable/modules/tree.html

[37] XGBoost. [online]. Dostupné z: https://xgboost.readthedocs.io/en/stable/

[38] Ilustrácia SVM. [online]. Dostupné z: https://scikit-learn.org/stable/_images/sphx_glr_plot_separating_hyperplane_001.png

[39] Ilustrácia Decision Tree. [online]. Dostupné z: https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

[40] What is XGBoost Algorithm?. [online]. Dostupné z: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

[41] Python Elasticsearch Client. [online]. Dostupné z: https://elasticsearch-py.readthedocs.io/en/v8.12.0/

[42] Cron. [online]. Dostupné z: https://www.man7.org/linux/man-pages/man8/cron.8.html

[43] Elastic Stack. [online]. Dostupné z: https://www.elastic.co/elastic-stack

[44] InfluxDB. [online]. Dostupné z: https://www.influxdata.com/

[45] Prometeus. [online]. Dostupné z: https://prometheus.io/

[46] PostgreSQL. [online]. Dostupné z: https://www.postgresql.org/

[47] MySQL. [online]. Dostupné z: https://www.mysql.com/

[48] InfluxDB data elements. [online]. Dostupné z: https://docs.influxdata.com/influxdb/cloud/reference/key-concepts/data-elements/

[49] InfluxDB client. [online]. Dostupné z: https://github.com/influxdata/influxdb-client-python

[50] SciPy. [online]. Dostupné z: https://scipy.org/

[51] Tensorflow. [online]. Dostupné z: https://www.tensorflow.org/

[52] Grafana. [online]. Dostupné z: https://grafana.com/

[53] Grafana inštalácia. [online]. Dostupné z: https://grafana.com/docs/grafana/latest/setup-grafana/installation/

[54] Konfigurácia grafany. [online]. Dostupné z: https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/

[55] Get started grafana influxDB. [online]. Dostupné z: https://grafana.com/docs/grafana

Prílohy

Príloha 1: Úprava dát pomocou posunutia (lag), rozdelenie na tréningovú a testovaciu množinu, škálovanie dát.

```
lag_features = ['no_attacks']
lag_hours = 48

for lag in range(1, lag_hours + 1):
    df_attacks[f'lag_{lag}'] = df_attacks['no_attacks'].shift(lag)

df_lagged = df_attacks.dropna()
X = df_lagged[[f'lag_{lag}' for lag in range(1, lag_hours + 1)]].values
y = df_lagged['no_attacks'].values
split_point = int(len(X) * 0.8)
X_train, X_test = X[:split_point], X[split_point:]
y_train, y_test = y[:split_point], y[split_point:]
timestamps_test = df_lagged['timestamp'][split_point:].values
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

Príloha 2: Vytvorenie, trénovanie predikcia modelu SVM.

```
svm_model = SVR(kernel='rbf', C=1000, gamma=0.01, epsilon=0.01)
svm_model.fit(X_train_scaled, y_train)
y_pred_svm = svm_model.predict(X_test_scaled)
y_pred_svm
num_nans_to_prepend = len(df_attacks) - len(y_pred_svm)
predicted_series = np.concatenate((np.full(num_nans_to_prepend, np.nan),
y_pred_svm))


df_predicted = pd.DataFrame({
    'timestamp': df_attacks['timestamp'],
    'predicted_no_attacks': predicted_series
})
```

Príloha 3: Vytvorenie, trénovanie predikcia modelu DecisionTree.

```
dt_regressor = DecisionTreeRegressor(criterion='poisson', max_depth=6)
dt_regressor.fit(X_train, y_train)
y_pred_dt = dt_regressor.predict(X_test)
y_pred_dt
```

Príloha 4: Vytvorenie, trénovanie predikcia modelu XGBoost.

```
xgb_regressor =
xgb.XGBRegressor(objective='reg:squarederror',min_child_weight=5, subsample =
0.7, learning_rate=0.005, max_depth=5)
xgb_regressor.fit(X_train_scaled, y_train)
y_pred_xgb = xgb_regressor.predict(X_test_scaled)
y_pred_xgb
```

Príloha 5: Vytvorenie, trénovanie predikcia modelu LSTM.

```
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(seq_length, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.1)
predictions = model.predict(X_test)
predictions
predictions = scaler.inverse_transform(predictions)
y_test = scaler.inverse_transform(y_test)
```

Príloha 6: Vytvorenie, trénovanie predikcia modelu GRU.

```
model = Sequential([
   GRU(50, input_shape=(n_steps, 1)),
   Dense(1)
])
model.compile(optimizer=Adam(), loss='mean_squared_error')
model.fit(X_train, y_train, epochs=20, batch_size=32, verbose=1,
validation_split=0.2)
predicted = model.predict(X_test)
predicted = scaler.inverse_transform(predicted)
y_test_inverse = scaler.inverse_transform(y_test)
```

Príloha 7: Metriky porovnávania modelov MAE, MSE, MASE.

```
mae = mean_absolute_error(y_test, y_pred_svm)
mse = mean_squared_error(y_test, y_pred_svm)
naive_forecast = np.roll(y_test[1:], 1)
mase = mean_absolute_scaled_error(y_test[1:], y_pred_dt[1:], naive_forecast)
```