
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ**
**DEPARTMENT OF COMPUTER SYSTEMS**


# VYHLEDÁVÁNÍ PODOBNOSTÍ V SÍŤOVÝCH BEZPEČNOSTNÍCH HLÁŠENÍCH

**SIMILARITY SEARCH IN NETWORK SECURITY ALERTS**

**DIPLOMOVÁ PRÁCE**
**MASTER'S THESIS**


**AUTOR PRÁCE** **Bc. IMRICH ŠTOFFA**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Ing. MARTIN ŽÁDNÍK, Ph.D.**
**SUPERVISOR**



**BRNO 2020**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

# Zadání diplomové práce


Student: **Štoffa Imrich, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Vyhledávání podobností v síťových bezpečnostních hlášeních**
**Similarity Search in Network Security Alerts**
Kategorie: Počítačové sítě
Zadání:

1. Nastudujte problematiku síťových bezpečnostních hlášení. Zaměřte se na systém Warden.
2. Analyzujte vhodné nástroje pro zpracování velkého množství bezpečnostních hlášení.
3. Navrhněte metodu pro vyhledávání skupin entit (např. IP adres), kdy skupinu tvoří takové entity, které se v datech několikrát vyskytují ve shodný časový interval.
4. Navržené řešení implementujte.
5. Funkčnost implementace ověřte na datech ze systému Warden.
6. Zhodnoťte dosažené výsledky a navrhněte možná rozšíření.

Literatura:

- Dle pokynů vedoucího práce.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Žádník Martin, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 25. října 2019

# Abstrakt

Systémy pre monitorovanie sietí zachytávajú veľké množstvo anomálií a podozrivých aktivít IP adries. Z týchto informácií, bezpečnostných udalostí, je len malé množstvo natoľko dôležitých, že si vyžadujú pozornosť administrátora. Majorita udalostí teda ostáva nespracovaná. Výsledky experimentov naznačujú, že analýzou týchto dát môžu byť odhalené koordinované skupiny IP adries a informácie o špecifických koreláciách udalostí. Metóda pre získavanie týchto znalosti zlepší prehľad administrátorov o dianí na sieti a je odpoveďou na narastajúce požiadavky na extrakciu užitočných informácií pri monitorovaní sietí. Nová metóda pre vyhľadávanie skupín IP, ktoré vykazujú vysokú mieru podobností komunikácie, bola implementovaná. Zakladá sa na korelácií sekvencií, ktoré reprezentujú vzory príchodu udalostí v čase. Metóda je testovaná na reálnych dátach z platformy pre zdieľanie, ktorá akumuluje 2.2 milióna udalostí denne. Výsledky ukázali, že metóda zachytila ozajstné koordinované skupiny IP adries.



# Abstract

Network monitoring systems generate a high number of alerts reporting on anomalies and suspicious activity of IP addresses. From a huge number of alerts, only a small fraction is high priority and relevant from human evaluation. The rest is likely to be neglected. Assume that by analyzing large sums of these low priority alerts we can discover valuable information, namely, coordinated IP addresses and type of alerts likely to be correlated. This knowledge improves situational awareness in the field of network monitoring and reflects the requirement of security analysts. They need to have at their disposal proper tools for retrieving contextual information about events on the network, to make informed decisions. To validate the assumption new method is introduced to discover groups of coordinated IP addresses that exhibit temporal correlation in the arrival pattern of their events. The method is evaluated on real-world data from a sharing platform that accumulates 2.2 million alerts per day. The results show, that method indeed detected truly correlated groups of IP addresses.



# Kľúčové slová

bezpečnostné udalosti, bezpečnostné hlásenia, korelácia, IP adresy, klastrovanie, zhlukovanie, monitorovanie siete, strojové učenie, rozpoznávanie vzorov, kolektívne anomálie, detekcia botnetov.

# Keywords

Alerts, correlation, IP address, sequence clustering, situational awareness, machine learning, pattern recognition, collective anomaly, botnet detection.


# Citácia

ŠTOFFA, Imrich. *Vyhledávání podobností v síťových bezpečnostních hlášeních.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Žádník, Ph.D.

# Vyhledávání podobností v síťových bezpečnostních hlášeních

## Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Martina Žádníka. Další informace mi poskytli Honza Wrona a Tomáš Čejka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Imrich Štoffa
5. júna 2020


## Poďakovanie

Ďakujem môjmu vedúcemu práce Martinovi Žádníkovi za odborné vedenie a trpezlivosť pri hodinách strávených vysvetľovaním vzoriek dát, grafov a implementačných prekážok. Ďakujem mojim rodičom za ich dôveru vo mňa a nekonečnú podporu. A nakoniec ďakujem mojím priateľom za to, že pri mne stáli keď som to potreboval.

# Obsah

# Kapitola 1

# Úvod

Svet ako ho dnes poznáme stojí z veľkej časti na technológiách. Za týmto štádiom však stojí neskutočne dlhá cesta vývoja. Každodenný život generácií ľudí, kde každý jednotlivec disponoval schopnosťou učiť sa objavovať a tvoriť, v kombinácií s neúprosným tlakom prírody pôsobiacim na ľudstvo viedol k optimalizácií najčastejších činností. Toto v súhrne spôsobilo, že ľudské kolektívne vedomie nadobudlo isté know-how. Najskôr ako prežiť, neskôr ako žiť a dnes snáď aj ako žiť a nechať žiť. Implementácie riešení z tohoto beztvarého know-how vidíme všade okolo seba. Zvlášť to, čo dnes ešte stále používame zložilo skúšku času. Pre lepšiu predstavu nech čitateľ uváži jazyk. Schopnosť človeka rozprávať jazykmi je mu tak vlastná, že mozog má na jeho používanie celé centrum. Je nesporné, že schopnosť koordinovať viacero jednotlivcov dáva skupine väčšiu šancu na úspech. Schopnosť verbálneho dorozumievania očividne obstála v skúške času a ľudstvo na nej ďalej stavalo. Snáď schopnosť človeka abstrahovať, premýšľať nad nehmotnými vecami, a plánovať zaviedla do komunikácie koncept pravdy a lži. Zrejme je lož vo konkurenčnom boji nevídanou zbraňou. Technicky povedané umožňuje zručnému jednotlivcovi dostať kolektív do stavu, ktorý vyhovuje práve jemu a je v záujme každého jednotlivca aby posúdil, či je bezpečné konať na základe danej informácie. Jednotlivec by teda mal hodnotiť zdroje informácií zvlášť, ak na základe týchto informácií plánuje robiť rozhodnutia.

Abstraktne povedané, internet je tiež jedným z médií pre komunikáciu, a teda podedil problémy rovnakej povahy. V tomto prípade vplyv komunikácie nadobúda nevídaných rozmerov. Význam internetu pre jednotlivcov ako aj pre ekonomiku celých štátov zrejme nie je treba ďalej vysvetľovať. Rýchly prenos informácií sa stal samozrejmosťou a umožnil neskutočne urýchliť vývojové procesy v spoločnosti. Instant messaging, sociálne siete, online žurnalistika, online nakupovanie, vyhľadávače obsahu, cielená inzercia, zoznamky a hromady ďalších aplikácií môže využívať a využíva človek 21. storočia. Filozofické duše sa môžu pýtať, či sú ľudia pripravení a hodní novo-nadobudnutých schopností a aké to bude mať ďalšie dopady pre spoločnosť.

Internet, je celosvetová sieť počítačov. Je lepšie na internet nahliadať ako na sieť sietí, kde zodpovednosť nad jednotlivými čiastkovými sieťami preberajú skupiny administrátorov. Aby vôbec bolo možné vidieť, čo sa na sieti deje, musí infraštruktúra podporovať monitorovanie. Monitorovanie môže prebiehať na rôznych úrovniach, od zberu štatistík o komunikácií, po sledovanie záznamov o aktivite (logy), až po zachytávanie a analýzu netflow dát, či paketov. Existujú systémy umožňujúce detektovať, či dokonca obmedzovať útoky na základe monitorovania a včasnej intervencie. Štandardne sa celý systém označuje skratkou SIEM.

Cieľom tejto práce je extrahovať z veľkej dátovej sady skupiny IP adries, ktoré vykonávajú škodlivú činnosť v podobných časových intervaloch a poskytnúť tak administrátorom ďalší kúsok informácie, ktorá prispeje administrátorom k tomu aby udržali sieť spoľahlivú. V práci bol preskúmaný návrh metódy klastrovania bezpečnostných udalostí, metóda bola implementovaná a jej činnosť overená vykonaním experimentov na dátach z reálnej siete.

V kapitole 2 je v krátkosti popísaný monitoring siete, vysvetľuje čo sú anomálie v dátach ako sa nich stanú bezpečnostné udalosti a akú majú udalosti vzťah voči škodlivým elementom na počítačových sieťach. Sú popísané príklady detektorov a malware. Snahou bolo uviesť čitateľa do kontextu bezpečnosti na počítačových sieťach a porozprávať o hrozbách číhajúcich na internete. V kapitole 3 je popísané aký je stav výskumu na tému analýza bezpečnostných udalostí a aké možnosti poskytuje strojové učenie pre rozpoznávanie vzorov v bezpečnostných udalostiach. Kapitola 4 rozvádza a popisuje problémy a riešenia potrebné pre implementáciu prototypu metódy SECT. Kapitola 5 obsahuje popis zaujímavých alebo dôležitých pseudo-algoritmov ktoré metód používa. V kapitole 6 sú popísané experimenty a nálezy v dátach.

# Kapitola 2

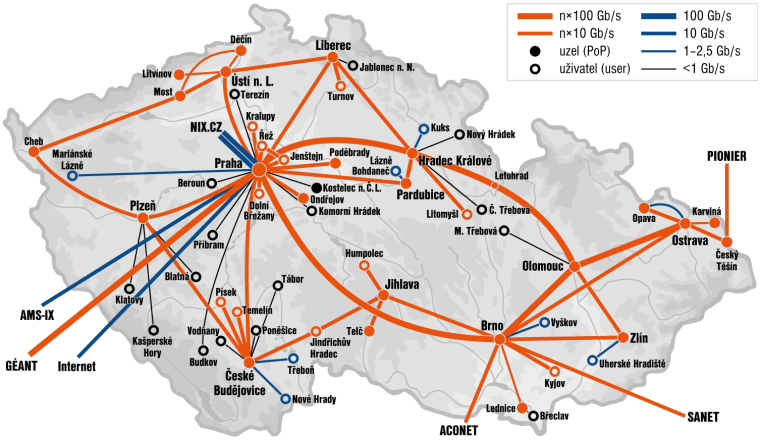
# Monitorovanie sietí a bezpečnostné hlásenia

Rozsiahle siete je potrebné sledovať a riadiť, preto sa monitoruje premávka na hraničných bodoch. Na týchto bodoch dochádza k zachytávaniu a exportu netflow dát. Dáta sa ďalej spracúvajú pomocou rôznych detektorov pre extrakciu užitočných informácií, medzi nimi aj bezpečnostných udalostí[1]. Podrobnejší zoznam generátorov/detektorov bezpečnostných udalostí nájdete v sekcii 2.3. Netflow dáta sa zvyknú krátkodobo skladovať, pre účely ďalšej analýzy. Dlhodobému skladovaniu bráni vysoký prietok dát. Napríklad v Česku na sieti CESNET prietok v dosahuje hodnoty rádovo $100k$ flow/$s$ v súhrne[24].

Internet ako sieť sietí je komplexne prepojený systém. Dá sa predpokladať, že anomálie detekované v jednej autonómnej sieti ovplyvnia aj ďalšie. Je teda prirodzené usúdiť, že administrátori by si mohli ceniť systém, ktorý by im umožnil tieto informácie zdieľať.

Pre účely správy a monitorovania siete je potrebný zber a analýza dát o dianí a prevádzke. Medzi jednotlivé zdroje dát patrí zachytávanie a analýza netflow, paketov, analýza logov zo zariadení s rôznymi úlohami, blacklisty, reputačné databázy, proaktívne skeny a ďalšie.

[1] pomenovanie pre záznam generovaný niektorým z detektorov, popisuje incident




Obr. 2.1: Topológia siete CESNET2[3]

## 2.1 Hrozby

Sieť a zariadenia do nej pripojené sú vystavené hrozbám. Pripojenie umožňuje v rámci protokolov ovplyvňovať činnosť zariadení v siete pomocou komunikácie. Mechanizmy protokolov a programové vybavenie uzlov na sieti umožňujú široké spektrum operácií a efektov, pričom niektoré môžu by nežiadúce. Pripojené entity môžu niektoré mechanizmy zneužiť.

Sieť alebo uzol môžu byť úmyselne preťažované útočníkom a pre administrátorov je dôležité o tomto vedieť. Informáciu o dianí administrátori získajú prostredníctvom telemetrie siete a odvodených informácií. Z týchto zozbieraných a analyzovaných dát je potrebné vyvodiť závery a (ne)vykonať akcie. Pri spracovaní sa v dátach vyskytujú informácie rôznych podobách a je úlohou administrátorov aby vyvodili závery. Je ťažké ručnou analýzou rozlíšiť aké dáta sú normálne a čo už je spôsobené činnosťou hackerov, škodlivým softvérom alebo napríklad výpadkom na sieti. Pre pohľad na dáta z vyššej perspektívy je potrebné použiť vhodné nástroje pre vyhľadávanie vzorov a detekciu anomálií.

### Anomálie

Za anomáliu možno považovať odchýlku od normálu. Mnoho vedeckých zameraní sa spolieha, že skúmané procesy fungujú na základe istých pravidiel. Tieto procesy sú produktom zmien stavov systému, čo sa navonok prejaví ako zmeny istej pozorovateľnej vlastnosti. Z týchto dát sa pri skúmaní formuluje hypotéza, teda predpoklad o chovaní systému. Hypotézu je možné overiť experimentom na inej alebo väčšej sade dát. Overená hypotéza sa stane predpokladom, no ten nemusí platiť večne, môže byť porušený, ak sa chovanie skrytého procesu príliš odchýli od normálu. Systém sa dostane do nepozorovaných stavov. Variácie chovania procesu sa musia preniesť v istej forme do variácie v pozorovaniach. Úlohou detekcie anomálií je rozpoznať ich a ideálne poskytnúť dostatok informácií pre odvodenie variácie v chovaní, ktoré anomáliu spôsobili. Základným problémom však je, že neexistuje jediný jednoznačný proces, ktorý by toto umožňoval[28].

V prípade aplikácie detekčného algoritmu s binárnym výsledkom, môžu nastať štyri prípady: True positive (TP) a true negative (TN) sú prípady, kedy sa výsledok detekcie zhoduje s realitou. False positive (FP) indikuje nesprávnu detekciu anomálie, rovnako ako false negative (FN).

U väčšiny reálnych systémov nie je perfektná detekcia možná. Úlohou analytika je teda vyvinúť mechanizmy tak aby FP a FN boli minimalizované, prípadne vyladiť ich pomer tak aby odpovedal bola minimalizovaná celková cena chýb. Klasifikácia dát na anomálne alebo normálne môže byť popísaná aj spojitou hodnotou, prípadne aj hodnotu relatívnej miery abnormálnosti dvojice vzoriek dát.

V kontexte počítačovej bezpečnosti musia byť detekčné algoritmy prispôsobené tomu, že chovanie siete je silne závislé na kolektívnej vôli ľudí. Komunikácia na sieti (súhrnne) nie je ani deterministická, ani náhodná. Zber vzoriek dát počas dostatočne dlhého obdobia lepšie zaručí, že vzory sa budú nejakú dobu spoľahlivo opakovať, to znamená budú existovať vzory, ktoré majú istú mieru predikovateľnosti. Pre príklad: Keď útočník, respektíve proces generujúci hypotetický škodlivý sledovaný vzor zistí, že jeho postup pre hádanie hesiel ďalej nefunguje a rozhodne sa že zacieli inú zraniteľnosť, skokovo zmení chovanie v snahe reagovať na bezpečnostný mechanizmus, ktorý jeho pôvodný zámer zmaril. Takéto rozhodnutie sa bude ťažko predikovať. Denné variácie v toku dát na sieti a informácia, že množstvo neidentifikovaných skenov[2] s tokom dobre koreluje (čo je predpoklad) predsta-

[2]Sken je pokus o naviazanie komunikácie s cieľom zistiť dostupnosť zariadenia, či služby

vuje ľahšie predikovateľnú anomáliu. Neočakáva sa však, že bude dochádzať k abruptným zmenám chovania na globálnej mierke v sledovanom systéme.

V taxonómií anomálií podľa prehľadového článku[16] sa rozlišujú nasledujúce typy:

- *Bodová anomália*: Ak sa jediná vzorka vymyká očakávanému rozloženiu v dátovej sade, považuje sa za bodovú anomáliu. Realistickým príkladom je ak bežná spotreba paliva pre osobu $X$ je 5 litrov za deň, ale v jeden náhodný deň zrazu stúpne na 50 litrov, potom ide o bodovú anomáliu.

- *Kontextuálna anomália*: Jedna vzorka z údajov, ktorá je neobvyklá voči nejakému konkrétnemu kontextu sa nazýva kontextová alebo podmienečná anomália. Napríklad výdavky z kreditnej karty počas Vianoc sú nižšie ako normálne, avšak v danom časovom kontexte sú všetky vzorky v danom období vyššie, môže ísť o kontextovú anomáliu.

- *Kolektívna anomália*: Keď je množina podobných vzoriek v dátach neobvyklá vzhľadom na zbytok sady. Napríklad keď sú udalostí 20 IP adries v dátovej sade kde sa vyskytujú tisícky, perfektne zarovnané v čase na rovnakých intervaloch.

Odhad hraníc anomality býva predmetom skúmania v analýzach. Central limit theoreme hovorí, že jav, ktorý závisí od veľkého množstva faktorov, sa začne javiť ako náhodný a je možné ho popisovať daným rozložením. Znalosť charakteristík rozloženia umožní nastavenie rozhodovacej hranice. To síce funguje, ale nezohľadňuje prípadný drift rozloženia (postupná zmena očakávanej charakteristiky). Lepší spôsob nastavenia hraníc popisuje práca [37], zakladá sa na teórií Extreme value theoreme. Výskumníci v práci odvodili štatisticky spoľahlivý detektor anomálií s automatickým nastavením rozhodovacích hraníc, ktorý je možné použiť aj na prúdoch dát[38].

### Zdroje

Zdrojom anomálií často býva činnosť škodlivého softvéru. Existuje veľa typov takéhoto softvéru a dokáže sa prejavovať mnohými spôsobmi. Jednotlivé komponenty, či čiastkové operácie takéhoto softvéru zanechávajú vzory v dátach a v momente kedy je významnosť vzoru dostatočná alebo metóda dostatočné presná dochádza k možnosti detekcie ich prítomnosti. Presné mapovanie vzorov na programy je pravdepodobne výnimkou, avšak vzory je možné kategorizovať a prideliť im aspoň čiastočný význam. Každý 'malware' má svoje vzory chovania a časť z nich môžu zachytiť detektory.

#### Malware

Malware, teda 'malicious software' označuje softvér, ktorý bol napísaný s úmyslom poškodiť užívateľa. Súhrnne sa tým označujú vírusy, červy, trojské kone všetkých účelov. V priebehu rokov boli tieto programy písané s rôznymi úmyslami, od snahy žartovať, cez pokusy osvojiť si programovanie a pochopiť zložité systémy až po pomsty predošlým zamestnávateľom. S rozšírením internetu okolo roku 2003 začalo vznikať veľké množstvo malware s cieľom obohacovania sa. Počítače infikované backdoormi umožnili útočníkom zneužívať výpočtové zdroje a informácie napadnutých užívateľov pre šírenie spamu, nezákonného obsahu, či vykonávať DDoS útoky.

Vplyv malware je dnes podstatne viac limitovaný vďaka technikám separácie služieb na sieti pomocou firewallu, presmerovaniu komunikácie známych botnetov na špeciálne ser-

very (sinkholing) a nekonečnej snahe bezpečnostných výskumníkov a vývojárov operačných systémov[8].

Množstvo programov, ktoré sa používajú na počítačoch rastie rovnako ako počet pripojení, a s tým aj množstvo neodhalených bezpečnostných dier a príležitostí. Niektoré exploity[3] môžu byť neuveriteľne komplexné (Heartbleed, EternalBlue), iné naopak jednoduché, avšak nebezpečné čisto kvôli množstvu zariadení, ktoré môžu potencionálne ovplyvniť (Mirai IoT botnet). Napríklad Mirai botnet sa šíri cez telnet jednoduchým hádaním prednastavených hesiel zo zoznamu, a v roku 2016 spôsobil DDoS s dovtedy nevídaným dátovým tokom[11].

Boj o kybernetický priestor nikdy nekončí a zlepšovanie obranných systémov zároveň tlačí na tvorcov malware aby prichádzali novými sofistikovanejšími exploitami.

Malware je možné klasifikovať z rôznych hľadísk, kategórie nie sú vzájomne výlučné.

- *Transportný mechanizmus* - Vírus, Červ
- *Mechanizmus prieniku do systému* - Trojský kôň, Rootkit, Backdoor
- *Zámer* - Šírenie spamu, Nevyžiadaná reklama, Krádež informácií, Podvrhnutie služby, Výpalníctvo

Najčastejším typom malware zvykli byť vírusy a červy. Svoje mená prebrali na základe spôsobu ich šírenia. Vírus je škodlivý kód, ktorý je cielený na zraniteľný program a istým mechanizmom, napríklad pomocou 'stack overflow' vykoná kód, ktorý sa v pôvodnom programe nevyskytoval, z miesta, z ktorého by k tomuto nemalo vôbec dôjsť. Vírus teda zabezpečí spustenie svojho kódu prostredníctvom iného programu. Červ zasa aktívne vyžíva možnosti počítača komunikovať v sieti s ostatnými k tomu, aby sa mohol replikovať. Často krát sa jednalo o e-maily so súborom balíka MS Office, ktorý obsahoval skript (VBA makro), ktoré okrem škodlivej aktivity na lokálnom počítači kódovalo tiež rozoslanie rovnakého emailu ďalším kontaktom. Nedávny incident vo Fakultnej nemocnici Brno bol pravdepodobne spôsobený práve skriptom v prílohe e-mailu. Súdiac podľa informácie, že nemocnica potrebovala meniť disky v napadnutých v počítačoch šlo najskôr o ransomware[6].

Červ bol prvým z malware, vznikol v roku 1988 a červy dodnes pracujú na rovnakom základnom princípe. Skenujú sieť a využívajú zraniteľné počítače na to, aby sa ďalej kopírovali. SQL Slammer v roku 2003 využil zraniteľnosť vtedajšej verzie MS SQL servera, kde stačilo aby server prijal správne malformovanú požiadavku, a to spôsobilo rýchle rozosielanie rovnakých požiadaviek od siete na náhodné IP adresy. V priebehu niekoľkých minút slammer vyradil služby na tisícoch ovplyvnených serverov a zastavil alebo spomalil k hranici nepoužiteľnosti veľké časti internetu.

Aby malware dosiahol svoje ciele, musí by schopný dostať sa do systému a zotrvať v ňom dostatočne dlho bez intervencie. V prípade *trojského koňa* pomáha prieniku do systému utajenie. Užívateľ nainštaluje program do svojho počítača a aby sa vyhol poplatkom za licenciu, prepíše pôvodný exe súbor modifikovaným exe s crackom z internetu. Týmto sa do systému dostáva škodlivý kód po vzore starogréckeho trojského koňa. Takýto softvér potom utajuje používateľovi svoju sekundárnu funkciu. Tvorcovia malwaru dokonca zachádzajú až tak ďaleko, že danú sekundárnu činnosť legalizujú na základe podmienok používania, ktoré musí užívateľ odsúhlasiť pri inštalácii.

Technikou, ktorú malware používa pre upevnenie svojej pozície v systéme je *rootkit*. Rootkit zabezpečuje utajenie procesov malware pred užívateľom, odolnosť proti ukončeniu týchto procesov a umožní zisk administrátorských práv. *Backdoor* je metóda obchádzania

[3]programy písane špeciálne pre využitie niektorej zraniteľnosti

normálnych autentifikačných procedúr. Kompromitovaný systém obyčajne vytvorí tajnú službu, cez ktorú je možné dostať prístup do systému, obyčajne je ňou terminál. Pôvodnou myšlienkou snáď bolo zjednodušenie technickej podpory sieťových zariadení a tak výrobcovia zariadení zavádzali do svojich systémov tajné prístupové kanály. Býva možné zistiť prístupové údaje z binárnych súborov firmvéru. Takto sú dodnes zraniteľné napríklad IP kamery, obzvlášť, keď cez svoje webové rozhranie vyzradia verziu svojho firmvéru.

### Botnety

Botnet je spojením mnohých hrozieb do jedného balíčka, typicky sa skladá z jedného servera (napr. internet relay chat) a viacerých botov. Tvorcovi a používateľovi botnetu sa hovorí botherder. Siete stoviek až tisícok botov sú dnes považované za malé botnety. Botnet po napadnutí hostiteľa musí vedieť, ako kontaktovať stroje v roli C&C. C&C potrebuje uchovávať zoznam členov botnetu. C&C odosiela príkazy botom. Napadnuté stroje môžu generovať anomálnu sieťovú premávku ako skeny, bruteforce útoky[4] a vôbec akýkoľvek útočný nástroj, ktorý útočník zadá. Botnety vynikajú tým, že umožňujú koordinovať veľké množstvo botov a anomálne zaostriť premávku na cieľ.

Botnet sa líši od iného malware v tom, že musí byť schopný vykonávať akcie na klientoch bez toho, aby sa botherder musel priamo prihlasovať na napadnuté stroje. Veľa botov musí byť schopných vykonávať úlohy so spoločným cieľom a to s minimálnou intervenciou od útočníka.

Aby malware systém spĺňal definíciu botnetu musia byť tieto podmienky splnené:

- Je naprogramovaný aby sa šíril
- Usiluje sa o zotrvanie na hostiteľskom stroji
- Vykonáva príkazy od botherdera priamo, či nepriamo.
- Implementácie botnetov majú tendenciu sa zlepšovať, tak aby boli odolnejšie voči technikám vyvinutým na jeho potlačenie.

### Životný cyklus botnetu

Počas svojho fungovania botnety vykonávajú istú sadu operácií. Tá predstavuje životný cyklus a charakterizuje botnety ako také. Cyklus bota začína napadnutím hostiteľa. Toto sa deje zneužitím niektorej bezpečnostnej slabiny alebo preľstením užívateľa k tomu, aby kompromitoval svoj stroj. Sled operácií je nasledovný:

- *Exploit* - Napadnutím hostiteľa vzniká nový bot
- *Registrácia* - Bot kontaktuje botherdera (resp. jeho C&C), a registruje sa do botnetu
- *Zabezpečenie pozície* - Zahŕňa získanie proti antivírusového modulu a zabezpečenie, že bot ostane pripojený.
- *Cyklus počúvanie príkazov* - Príjem príkazov z C&C alebo od ostatných botov.

  Stiahnutie balíčka s úlohou.

  Vykonanie úlohy.

  Nahlásenie výsledkov (a návrat na príjem príkazov).

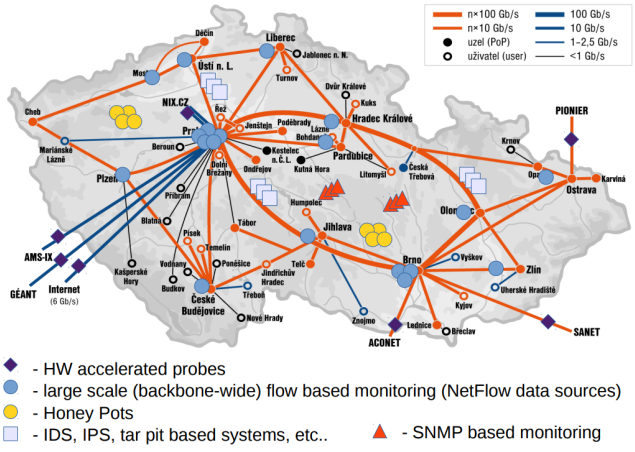
[4] Útok hrubou silou je opakovanie pokusov o získanie prístupu hádaním hesiel.

- *Opustenie* - Ukončí cyklus, zahladí stopy a opustí klienta.

Útočník môže zadať botnetu v podstate akúkoľvek úlohu (v rámci kapacitných možností) ale obyčajne sa botnety používajú na útoky typu: DDoS, DoS, AdWare, SpyWare, RansomWare. Z pohľadu útočníka botnet predstavuje, distribuované úložisko a výpočtovú kapacitu za minimálnu cenu, takmer bez zodpovednosti, pretože užívatelia koncových počítačov často netušia, že ich stroj je botom. Útočník, ak porušuje zákony, činí tak prostredníctvom zariadení, ktoré napadol, teda v cudzom mene. Tieto zdroje môže a využíva pre akékoľvek vlastné ciele, zvyčajne je to finančný zisk alebo zisk reputácie v komunite hackerov. Vystopovanie útočníka nebýva jednoduché a poškodení si často nedokážu odôvodniť/dovoliť zložitý proces vyšetrovania, kvôli časovej alebo finančnej náročnosti. Hrozba botnetu spočíva v neoprávnenom použití informácií, výpočtovej a sieťovej kapacity napadnutých strojov. Okolo roku 2007 sila potenciálnych DDoS útokov spustených cez botnety bola tak veľká, že dokázala ohrozovať aj infraštruktúru celých štátov[36, 31].

## 2.2 Detektory

Aby administrátori mohli sledovať telemetriu a udalosti na sieti musia surové dáta prejsť cez nástroje pre extrakciu vzorov a anomálií. Na tieto nástroje sa bude v texte odkazovať ako na detektory. Obecne sa celej hardvérovej a softvérovej výbave pre monitoring a manažment udalostí hovorí 'Security Information and Event Management' systém (SIEM)[39]. Interne v systéme SIEM pracujú detektory z rôznymi zdrojmi dát. Napríklad analýza flow záznamov je jedným zo zdrojom bezpečnostných udalostí v SIEM. V nasledujúcich sekciách je krátky popis niekoľkých typov detektorov používaných pri monitorovaní siete CESNET2. Na obrázku 2.2 je zobrazené geografické rozloženie detektorov na sieti CESNET2, zber flow dát prebieha primárne na hraničných linkách pomocou akcelerovaných sond:




Obr. 2.2: Umiestnenia niektorých detektorov na sieti CESNET2[3]

### Honeypot

Honeypot je špeciálny server, ktorého úlohou je vystaviť sa útokom a získavať o nich informácie. Požívajú sa najmä na včasné detekovanie šírenia malware a analýzu jeho chovania. Malware sa časom mení, a jeho chovanie je treba znova a znova analyzovať (menia sa rôzne

verzie, pribúdajú moduly, exploity). Získané informácie je možné použiť pre aktualizáciu antivírusových systémov. Honeypot systémy detekujú činnosť pri neoprávnených prístupoch do systému a automaticky zbierajú informácie o chovaní. Je to rýchlejšie ako zbierať dáta z funkčných napadnutých systémov.

## Tarpit

Tarpit je druh honeypot systému, Je to program určený pre spomalenie alebo znemožnenie škodlivých aktivít na sieti, ktoré spoliehajú na rýchlu odozvu a umožňuje vyčerpávať sieťové zdroje útočníka. Tarpit server si pridelí časť alebo celý nevyužitý rozsah IP adries na chránenej sieti. V prípade prichádzajúceho pripojenia je útočník požiadaný aby komunikáciu fragmentoval.

LaBrea tarpit sleduje ARP komunikáciu, tá slúži pre zistenie L2 adries na základe L3 adries. V prípade viacnásobnej požiadavky ARP pre získanie L2 jedného konkrétneho stroja (ktorý neexistuje) v krátkom intervale za sebou. LaBrea systém podvrhne svoju L2 adresu útočníkovi a presmeruje pripojenie na seba. Na smerovači takto vytvorí virtuálny stroj s danou IP, čo spôsobí, že komunikácia bude smerovať na LaBrea.

LaBrea sa javí ako normálny stroj, ale na požiadavky odpovedá tak, že komunikáciu fragmentuje a spomaľuje. V podstate odpovedá len na SYN pakety vytvorením pripojenia (SYN/ACK), pričom vyžaduje fragmentáciu prichádzajúcich paketov a odpovedá až na konci intervalov vyhradených pre výmenu správ. Červom šíriacim sa po sieti značne spomaľuje postup[9].

### NEMEA

Network Measurements Analysis je modulárny systém pre prúdovú analýzu flow dát. Pozostáva z viacero nezávislých modulov spojených cez komunikačné rozhranie message passing interface. V moduloch sa k flow záznamu pripájajú extra informácie ako štatistiky, detekované bezpečnostné udalosti, pred spracované dáta etc.. Podozrivé chovanie sa zaznamenáva (loguje), z týchto záznamov tiež môžu byť generované bezpečnostné udalosti[20]. Moduly v systéme NEMEA sú napríklad:

- *HostStats* - Modul počíta štatistiky komunikácie pre každého klienta (IP), a tieto záznamy sú následne vyhodnocované aplikáciou sád pravidiel, teda jednoduchými detektormi anomálií rôznych typov. Sledujú sa počty paketov, bajtov, tokov, TCP príznakov komunikácie a odhadnuté počty cieľových adries. Modul vyhodnocuje pravidlá pre detekciu skenov IP a portov, SSH lámania hesiel (bruteforce), DDoS amplifikované cez DNS a ďalšie.

- *blacklist filter* - Modul sleduje a ukladá komunikáciu s adresami z blacklistov, pre účely analýzy prípadne použitia ako dôkazov pri vyšetrovaní incidentov.

- *ddos detector* - Detektor DDoS útokov pracuje na základe analýzy krátkej histórie komunikácie. Modul sleduje koľko paketov tečie do každého monitorovaného cieľa osobitne a v prípade, že zmena množstva komunikácie tečúcej k jednému z cieľových klientov prekročí detekčnú hranicu, nahlási sa DDoS útok.

### fail2ban

Fail2ban je program, ktorý sleduje záznamy s pokusu o autentifikáciu heslom. IP adresy, ktoré vykazujú škodlivú činnosť, teda podľa pravidiel prekročia počet pokusov o prehlásenie, budú po nastavenú dobu filtrované. Obecne sa fail2ban používa pre aktualizáciu pravidiel firewallu, tak aby na nejakú dobu zakázal komunikáciu s danými IP. Požíva sa na znemožnenie bruteforce hádania hesiel. Nezaručuje ale, že k neoprávnenému prihláseniu dôjde. V momente detekcie môže byť vykonaná ľubovoľná akcia napríklad generovanie bezpečnostnej udalosti[5].

## 2.3 Bezpečnostné udalosti

V kontexte bezpečnosti a monitorovania počítačových sietí je bezpečnostná udalosť informáciou, že na sieti došlo k činnosti, ktorá je považovaná za škodlivú. Zdrojom týchto udalostí sú detektory. Bezpečnostné udalosti sú vlastne odhalené a klasifikované anomálie z dát získaných monitorovaním koncových uzlov a siete.

V SIEM systémoch sa bezpečnostné udalosti akumulujú do databáz a analyzujú. Aby bola hodnota informácie čo najvyššia je potrebné zvoliť správny formát dát. Vyžaduje sa vyváženie medzi flexibilitou a výkonom.

### Formát IDEA

IDEA je skratkou pre intrusion detection extensible alert. Aj keď existuje viacero formátov pre výmenu bezpečnostných udalosti medzi detektormi. IDEA formát je pokusom definovať základné požiadavky pre reprezentáciu bezpečnostných udalostí. Je založený na skúsenostiach s predošlými formátmi, pričom sa snaží zachovať pozitívne rysy týchto formátov. Snahy o vyvinutie formátov bezpečnostných udalostí nie sú ničím novým, avšak pri návrhu IDEA boli uvážené ako požiadavky bezpečnostných tímov tak aj skúsenosti s existujúcimi formátmi.

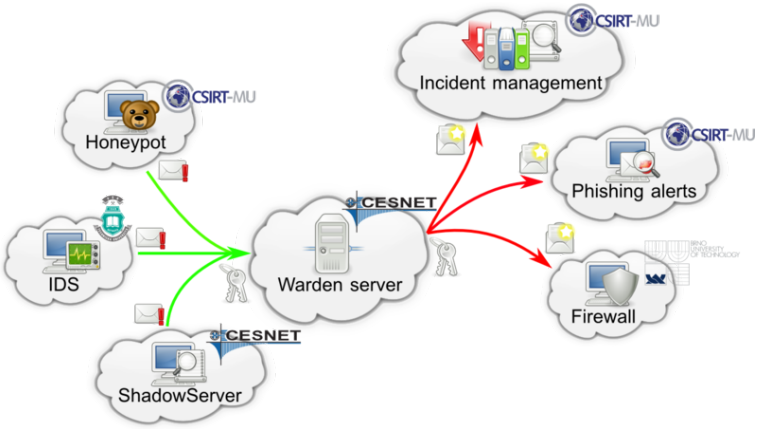
IDEA používa JSON formát a špecifikuje základnú schému štruktúru do hĺbky dvoch úrovní kľúčov. Maximálne jedna úroveň zanorenia dát umožní efektívne spracovať a analyzovať dáta v relačných DB. Formát umožňuje seba referenciu na kľúče v prvej úrovni. Každý záznam má rozumne jedinečný identifikátor. Detektory môžu pridávať nove kľúče, ak nekolidujú so základnou schémou.[21]

Medzi dôležité kľúče patrí časová značka, jej formát je v UTC. Pole EventTime označuje kedy udalosť nastala, DetectTime označuje kedy došlo k detekcii. Source označuje informácie o zdroji a obsahuje kľúč IP4 alebo IP6, ktoré vymenúvajú adresy ktorých sa udalosť týka. Rovnaké je to pre Target. Kľuč Node obsahuje dôležité informácie o detektore samotnom. Pre príklad je vzorová správa priložená v A.

### Systémy Warden

Systém Warden slúži na efektívne zdieľanie informácií o bezpečnostných udalostiach. V súčasnej dobe je vyvíjaný a prevádzkovaný hlavne pre potreby akademickej siete CESNET2, ktorá je prevádzkovaná združením CESNET, pre jeho členov a partnerov.

Mnohé CERT/CSIRT tímy sa v súčasnosti intenzívne venujú prevádzke a vývoju užitočných nástrojov a systémov v oblasti monitorovania sieťovej prevádzky, služieb a odhaľovaniu anomálií v komunikácií. Tieto nástroje a detektory bývajú založené na báze netflow, IDS,




Obr. 2.3: Architektúra Warden[14]

honeypot systémov, analýze logov etc.. Ich výstupy potom tímy používajú pre svoju potrebu a pre zabezpečenie svojej siete. Tieto dáta a informácie by ale mohli byť využité aj ďalšími bezpečnostnými tímami pôsobiacimi v iných sieťach.

Bezpečnostné tímy tak stoja pred nepríjemnou dilemou. Dáta môžu rozoslať správcom alebo tímom sietí, ktorých sa dotýkajú. Môžu ich zahodiť, alebo v lepšom prípade zaslať do veľkých databáz typu Shadowserver etc.. Prvá cesta, rozoslanie relevantným správcom, je náročná a môže zaťažiť dotknuté tímy ďalšou réžiou, ktorú nepôjde efektívne zvládať. Druhá cesta, zahodenie dát, znamená to neefektívne plytvanie užitočnými dátami ba dokonca aj ignorovanie rastúcich bezpečnostných hrozieb. Tretia cesta, zberné miesto, predstavuje núdzové riešenie prinášajúce početné negatíva, ktoré musia tímy starostlivo zvážiť. Aké dáta je vhodné a možné zasielať do podobných miest? Existuje záruka, že bude s dátami vhodne zaobchádzať? Nebude zbytočne dochádzať k odkladu a skresľovaniu?

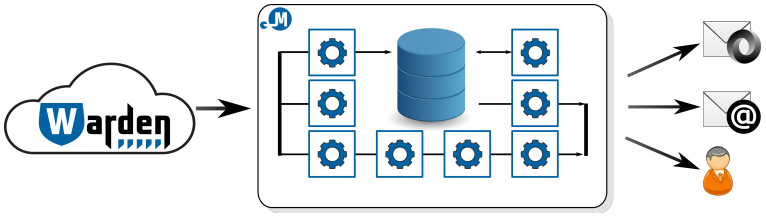
Túto otázku rieši systém Warden. Umožňuje bezpečnostným tímom efektívne zdieľať a využívať informácie o detekovaných anomáliách z prevádzky sietí a služieb. Správcovia doň môžu na dobrovoľnej báze zasielať dáta o detekovaných hrozbách a zároveň si môžu sťahovať údaje, ktoré ich zaujímajú a ktoré potrebujú.

Warden systém predstavuje pre bezpečnostných výskumníkov bohatý zdroj informácií. Hodnotné informácie sa skrývajú nie v jednotlivých bezpečnostných udalostiach ale v tom, že ich je veľké množstvo naprieč dlhým časovým intervalom, denne pribudnú odhadom 2.2 milióna udalostí. Táto sada prestavuje akýsi odtlačok chovania siete a komunikácie na nej. Analýzu týchto dát smeruje k lepšiemu pochopeniu, predikovaniu a potlačeniu nežiadúceho chovania na sieti. Pre zdieľanie a uchovávanie bezpečnostných udalostí v systéme Warden sa používa práve formát IDEA[14].

Systém sa skladá z hlavného Warden servera a dvoch typov klientov. Server zaobstaráva všetku základnú činnosť. Vysielajúci klient odosiela informácie poskytnuté zapojenou organizáciou na server. Prijímajúci klient získava informácie požadované zapojenou organizáciou zo serveru, je možno ho nastaviť aby prijímal len vyžadované typy udalostí. Komunikácia je zabezpečená.

### Možné prípady užitia systému Warden

- *Obrana vlastnej siete pred vonkajšími útočníkmi* - Vďaka dátam zo systému Warden možno posilniť obranu vlastnej siete niekoľkými spôsobmi. Jedným z nich je využitie




Obr. 2.4: Architektúra systému Mentat[12]

dát v nástrojoch tretích strán. Napríklad vo fail2ban nástroji môžu byť použité dáta o detekovaných útokoch na iných zapojených sieťach a útočníci sú preventívne zablokovaní ešte predtým, než dôjde k útoku na vlastnú sieť. Systém Warden by v tomto prípade fungoval ako akýsi Český dynamický blacklist.

- *Odhaľovanie napadnutých strojov z vlastnej siete* - Dáta zo systému Warden možno tiež použiť na vyhľadávanie napadnutých a zle nakonfigurovaných strojov vo vlastnej sieti. Dobrým príkladom je využitie dát z honeypot systémov patriacich do ostatných zapojených organizácií. Z týchto je možné hľadať IP adresy strojov z vlastnej siete a podniknúť patričné opatrenia pre nápravu. Vo všeobecnosti je možné hľadať akékoľvek udalosti, kde sú IP adresy z rozsahu vlastnej siete v pozícii útočníka.

- *Výskumné a experimentálne účely* - Dáta zo systému Warden je možné využiť mimo bežných prevádzkových aplikácií, tiež k experimentálnym a výskumným účelom. Veľké množstva dát sú ideálne pre hľadanie a overovanie rôznych korelácií medzi udalosťami. Dájú sa napríklad overovať súvislosti medzi útokmi na jednotlivé inštitúcie a získať tak podstatne väčší obraz o aktivite útočníkov v čase. Ďalšou možnosťou je vytvorenie reputačnej databázy IP adries (alebo rozsahov, či autonómnych systémov).

### Mentat

Mentat je distribuovaný modulárny SIEM (Security Information and Event Management System) navrhnutý pre monitoring sietí. Jeho architektúra umožňuje prijímať, ukladať, analyzovať, spracovávať a reagovať na veľké množstvo bezpečnostných udalostí pochádzajúcich z rôznych detektorov a aj tretích strán.

Systém Mentat je platforma umožňujúca jednotný zber udalostí z rôznych zdrojov a jednotne ich spracovávať. Vznikol pre potreby CESNET/CSIRT tímu. Bol navrhnutý ako modulárny distribuovaný systém s dôrazom na bezpečnosť, rozšíriteľnosť a škálovateľnosť. Jadro Mentat-u sa skladá jednoduchých modulov typu daemon, kde každý rieši jednu jednoduchú úlohu. Výsledkom je rozšíriteľný paralelizovateľný systém. Celý systém používa dátový model IDEA a dáta získava zo systému Warden[12].

### NERD

NERD je softvérová služba, ktorá získava, ukladá a agreguje rôzne dáta o škodlivých sieťových entitách, primárne IP adresy, a poskytuje ich v jednoducho pochopiteľnom formáte pre svojich používateľov. Cieľom NERD-u je pozberať čo najviac informácií o škodlivých IP a voľne ich poskytnúť na jednom mieste komunite bezpečnostných výskumníkov. NERD prijíma informácie z Wardenu a ďalších zdrojov. Záznamy o entitách obohacuje o metadáta a ďalšie informácie.

V záznamoch je možné nájsť:

- reverse DNS záznam
- číslo autonómneho systému ASN
- whois informácie
- geolokácia
- blacklisty, v ktorých sa IP nachádza vrátane histórie
- passive DNS
- dáta z DShield
- značky odvodené z ďalších zdrojov

Naviac, informácie o IP sa sumarizujú do reputačného skóre, ktoré reprezentuje akúsi mieru hrozby[13, 17].

### GreyNoise

GreyNoise je systém, ktorý zbiera, analyzuje a označuje hromadné internetové skenovacie a útočné aktivity. Tisícky ľudí a organizácií denne zbierajú informácie o čo najväčšom počte pripojených strojov. Zámery však môžu byť rôzne, výskumníci prehľadávajú internet, aby mohli zisťovať a nahlasovať zraniteľnosti, a sledovať botnety. Útočníci prehľadávajú internet, aby našli zraniteľné zariadenia, ktoré môžu ohroziť a zneužiť na vlastné účely. Mnoho ďalších skupín prehľadáva internet z neznámych alebo tajných dôvodov. Existujú spoločnosti, ktoré pôsobia ako 'vyhľadávacie nástroje' a venujú sa hromadnému skenovaniu internetu, ako napríklad Shodan, Censys, Sonar projektu Rapid7 a mnoho ďalších. Je náročné rozpoznať, ktoré skeny sú normálne a kto je ich pôvodcom. Čistá magnitúda dát, ktorú detektory skenov generujú dokáže v podstate zahltiť výskumníkov informáciami. Aby sa rozlíšilo, ktoré skeny sú normálne, a ktoré cielené je potrebné dlhodobo monitorovať sieť na globálnej škále a dáta analyzovať a agregovať. Na základe výsledkov je potom možné zistiť, či je daný sken sledovaný všade alebo sa jedná o cielenú operáciu. GreyNoise poskytuje dotazovací nástroj, ktorým sa dajú získavať informácie s rôznym detailom. Navyše sleduje globálne trendy, respektíve na aké konkrétne porty a služby sa momentálne cieli. Toto uľahčuje život výskumníkov, administrátorov a sieťových analytikov[7, 29, 30].

# Kapitola 3

# Analýza z bezpečnostných udalostí na sieti

Sady veľkých množstiev bezpečnostných udalostí predstavujú potencionálny zdroj informácií užitočných pre rôzne aplikácie. Problémom ostáva, ako a čo z daných dát zisťovať. V tejto práci je snahou využiť potenciál sady udalostí a extrahovať užitočné informácie o skupinách IP. V praxi sa dnes skúmajú anomálie a korelácie v dátach s cieľom zlepšiť prehľad o situácií a detekovať alebo dokonca predchádzať útokom.

Využitie metód strojového učenia pre detekciu anomálií v dianí na sieti je intenzívne skúmané a prináša výsledky pri nasadení na obranu a monitorovanie siete. Mohiuddin Ahmed a ďalší v prehľadovom článku [16] komplexne analyzovali situáciu vo výskume. Hoci sa detekcia anomálií zdá ako priamočiara záležitosť a existuje široká škála nástrojov, riešenie je ďalej predmetom výskumu. Heterogenita dát a prostredia nasadenia komplikuje riešenie. Primárnymi problémami je, že neexistujú univerzálne metódy pre detekciu anomálií na sieťach, chýbajú verejné dátové sady s označením a komunikácia na sieti sa v čase vyvíja a postupne mení čo je ešte normálne, a čo nie. Rozpoznanie anomálneho javu závisí na spôsobe spracovania dát a je potrebné vedieť, vzhľadom k čomu je jav anomálny. Navyše zo sietí prichádzajú nekončiace toky udalostí, ktoré je treba spracovať ideálne v jednom priechode.

Medzi metódami používanými k detekcií anomálii je napríklad klastrová analýza. Prevažne sa používa na označenie komunikácie ako benígnej alebo malígnej. Adaptácia k-means (x-means) bola úspešne použitá pre detekciu DDoS. V prístupe pomocou informačnej teórie bola využitá viacdimenzionálna korelácia v kombinácii earth mover distance na detekciu DDoS[18, 40].

Co-clustering upravuje spôsob práce s príznakmi oproti klasickému klastrovaniu, agreguje dáta ako podľa riadkov tak aj podľa stĺpcov, čo má za následok zachovanie väčšieho množstva informácie. Jedná sa teda zároveň aj o formu redukcie dimenzionality[32].

Benjamin J. Radford a kolektív v práci [33] popisujú spracovanie flow dát a detekciu útokov pomocou *Long short-term memory* neurónových sietí. Netflow dáta boli agregované po hodinách do tokenov a kĺzavým oknom spracované v LSTM. LSTM sieť sa ukázala byť efektívna na odhalenie SQL injection útoku. Práca však použila umelo vytvorený dátaset a nie je známe, či výsledky budú aplikovateľné na reálnu dátovú sadu.

Bayesovské klastrovanie umožňuje klatrovaciu analýzu dát s neistotou [41], to využili autori v článku [42] a kombináciou s rozhodovacími stromami vytvorili dobrý hybridný

klasifikátor. Jednotlivé sieťové anomálie sa klastrovaním roztriedili do typov a až po tom klasifikovali špecifickejším modelom[16, Sekcia 2.3].

Ďalšou témou v spracovaní udalostí je ich korelácia. Výskum prebieha už pomerne dlhú dobu existuje hromada článkov na danú tému, v snahe dosiahnuť rôzne ciele. Pojem korelácia má v danom kontexte nie presne štatistický význam. Výskumníci v sfére počítačovej bezpečnosti zvyčajne myslia agregáciu, zoskupovanie, filtráciu alebo zlučovanie udalostí do scenárov. Existujú tiež prehľadové články o aktuálnom *state of the art*.

Husák a kolektív v článku [25] vymenúva metódy a prístupy k predikcii útokov, ich zámeru a ďalších krokov. Porovnali diskrétne aj spojité modely. (attack graphs, bayesovské siete, markové modely). Zámerom bolo odhadnúť ďalšie kroky útokov, predikovať kedy a aký ciel má útok zasiahnuť a predpovedať celkovú situáciu.

Znalosť synchronizačnej postupnosti umožnila v článku [15] detekovať botnetom napadnuté stroje. Vykonali tak pomocou korelácie sekvencií bezpečnostných udalostí kategorizovaných do jednotlivých krokov životného cyklu botnetu. Zistili, že niektoré typy botnetov vykonávajú po infekcií špecifickú synchronizáciu s C&C, a tú je možné detekovať, resp. korelovať s postupnosťou v udalostiach.

Starší prehľadový článok [35] z 2006 preskúmal techniky korelácie udalostí a navrhol dvojakú klasifikáciu článkov - podľa toho ako agregujú udalosti a aké techniky používajú. Tvrdia, že bezpečnostné systémy sa líšia v typoch detekcií, a že nízko úrovňové udalosti sa musia spracovávať na vyššej úrovni v systémoch pre správu. Navrhuje rámec pre koreláciu udalostí zloženú zo 6 častí: Normalizácia, Agregácia, Korelácia, Redukcia falošných detekcií, Analýza stratégie útoku a Prioritizácia. Tieto časti sú extrakciou najlepších postupov pri spracovaní udalostí a ich vyhodnocovaní. Informácie o korelácii sú dôležité pre túto prácu. Autori delia existujúce prístupy na zamerania na scenáre, pravidlá, štatistiku a časové rady.

Prieskumy sa zhodujú na potrebe transformácie low-level udalostí do formy, kde sú filtrované a agregované do meta-informácií, čo má poskytovať náhľad na udalostí na vysokej úrovni. Avšak publikované články sa prevažne venujú vzťahom medzi udalosťami v kontexte útočných scenárov s jedným alebo viacerými štádiami. Články sa tiež zhodli na tom, že chýbajú verejne dostupné dátové sady pre trénovanie a vyhodnocovanie nových metód.

V preskúmaných článkoch sa výskumníci snažili vypracovať techniky korelácie pre zníženie počtu udalostí a to tým ze vytvárali vzťahy medzi nimi v podobe scenárov útokov a organizovali udalostí do nich. Naproti tomu, metóda SECT prezentovaná v tejto práci má za cieľ odvodiť koreláciou nové informácie z udalostí pre zlepšenie prehľadu o dianí na sieti. Metóda SECT odvodzuje nové informácie vyhľadávaním skupín entít, ktoré generujú bezpečnostné udalosti v podobnom vzore, to ich činí podozrivými. Skúmanie tohoto prístupu sa v článkoch nepodarilo objaviť.

Metóda SECT vychádza z a nadväzuje na predošlú prácu [23], ktorá síce implementovala vyhľadávanie podobností medzi entitami, avšak výkonovo nestačila na analýzu intervalov dlhších ako pár hodín. Metóda v spomenutej práci fungovala na základe porovnávania zoznamov časových značiek jednotlivých entít. Tiež počítala s istou toleranciou nezhôd. Jednotlivé IP adresy, korelované entity, zoskupovala po pároch a ďalej sa snažila budovať väčšie skupiny podľa nasledujúceho vzoru: Skupina $\{A, B, C\}$ je korelovaná, ak všetky dvojice vnútri skupiny sú korelované, skupina $\{A, B, C, D\}$ ak všetky trojice sú korelované, atď. Výkonnostne však metóda nestačila a nepriniesla použiteľné výsledky, kvôli nemožnosti analyzovať významne dlhé časové úseky. Metódu možno v krátkosti popísať výrazom sequence clustering.

## 3.1 Metódy strojového učenia

Potreba analyzovať a vyhľadávať vzory vo veľkých objemoch dáte nie je novinkou. Vznikli celé prúdy rôznych prístupov a techník, ako dané ciele dosiahnuť a vytvoriť pri tom tie najlepšie metódy. Strojové učenie používa techniky kombinácie dát s predom známymi informáciami na to, aby sa odhadla skrytá funkcia. Pri spracovaní dát v strojovým učením je dôležité vybrať vhodný typ metódy. Výber závisí na nasledovných vlastnostiach dát a problému:

- Veľkosť, kvalita a povaha dát
- Výpočtové prostriedky dostupné pre úlohu
- Čas na riešenie úlohy
- Cieľ analýzy

Základné delenie algoritmov pre strojové učenie je nasledovné:

### Supervised learning

Algoritmy pre trénovanie s učiteľom vytvárajú prediktory na základe dátovej sady s označenými dátami. Dátová sada pre každú vzorku definuje jej správne označenie. Napríklad MNIST, je sada ručne písaných číslic s priradenými numerickými hodnotami. Algoritmus sa učí mapovaciu funkciu medzi vstupnými dátami a ich označeniami. Vhodnosť zvoleného algoritmu pre učenie sa dá posúdiť validáciou. Validácia sa robí tak, že sa model otestuje na dátach, ktoré sa pri učení nevyskytli a spočíta sa úspešnosť.

- *Klasifikácia* - Ak sa údaje používajú na odhad kategórie vzorky, ide klasifikáciu s učiteľom. Ak existujú iba dve triedy, ide o binárnu klasifikáciu, ak existuje viac kategórií označuje sa to ako klasifikácia viacerých tried (multiclass classification).
- *Regresia* - Je ako klasifikácia, ale predikuje sa spojitá hodnota.
- *Predikcia* - Je odhad budúcej hodnoty predikovanej premennej na základe minulých a súčasných údajov respektíve modelu. Najčastejšie sa používa na analýzu trendov. Bežným príkladom je odhad tržieb na budúci rok na základe tržieb z aktuálneho a predchádzajúceho roku.

### Semi-supervised learning

Výzvou pri použití algoritmov pre trénovanie s učiteľom je skutočnosť, že označovanie údajov môže byť drahé, časovo náročné alebo nemožné. Ak sú označené dáta dostupne v obmedzene miere, je možné ich použiť v spojitosti s neoznačenými vzorkami na zlepšenie učenia s učiteľom.

### Unsupervised learning

Učenie bez učiteľa je keď ma stroj k dispozícii len neoznačené údaje. Úlohou je odhalenie skrytých vzorov alebo rozložení, z ktorých analyzované údaje vychádzajú. (štruktúra klastrov, topológia, rozhodovací strom).

- *Klastering* - Zoskupenie údajov analýzou vzťahov medzi vzorkami tak, aby si vzorky v jednej skupine boli vzájomne podobné a líšili sa od ostatných skupín. Toto sa často používa na rozdelenie celého súboru údajov do skupín, ktoré sa ďalej samostatne analyzujú. Podobnosť sa určuje pomocou funkcie vzdialenosti na vzorkách analyzovanej sady.

- *Redukcia dimenzionality* - Zníženie počtu uvažovaných premenných. V mnohých aplikáciách majú surové údaje veľké množstvo vlastnosti pričom niektoré sú pre úlohu nadbytočné alebo irelevantné. Redukcia počtu dimenzií pomáha nájsť skutočné skryté vzťahy.

### Reinforced learning

Posilňované učenie analyzuje a optimalizuje správanie agenta na základe spätnej väzby vo virtuálnom prostredí. Algoritmus skúša rôzne sekvencie akcií, aby zistil, ktoré prinášajú najväčšiu odmenu. Robí na základe nejakého definovaného cieľa. Chybová funkcia a oneskorené ocenenie odlišujú posilňované učenie od ostatných techník[10].

Cieľom metódy vyvinutej v tejto práci je zistiť, aké podobnostné vzory sa vyskytujú v komunikácií medzi entitami. Jednotlivé udalosti už sú označené, snahou teda bude identifikovať, v akých sekvenciách entity generujú udalosti. Udalosti sa musia predspracovať do sekvencií tak, aby mohol byť aplikovaný unsupervised learning, respektíve klastrovanie.

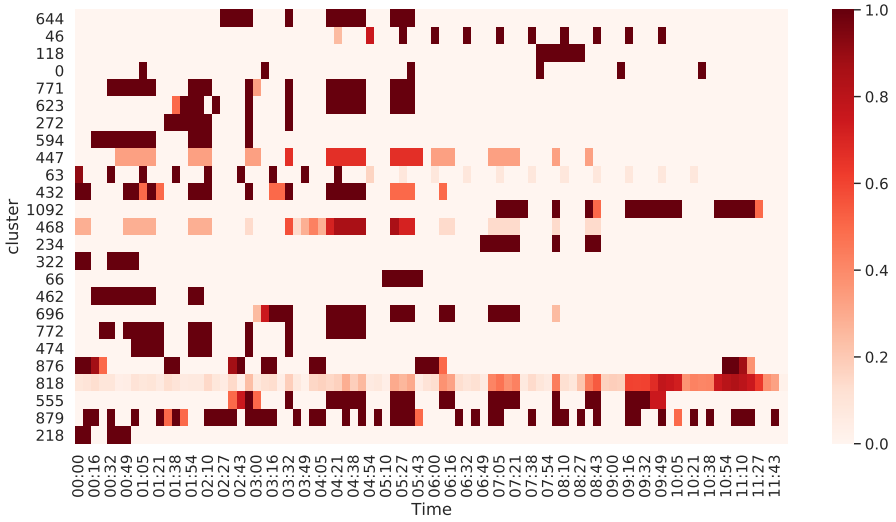
## 3.2 Klastrovacie algoritmy

Existuje veľké množstvo algoritmov pre použitie na rôznych typoch a veľkostiach dátových sád. Popis bežne známych možno nájsť napríklad na stránke scikit-learn[1]. Vo vzťahu ku klastrovaniu binárnych vektorov s väčšou dĺžkou vzniká problém pri skúmaní dát, je potrebné redukovať počet dimenzií, čo potencionálne spôsobu stratu detailov. To je možné riešiť vhodnou vizualizáciou. Koncept globulárnych klastrov sa na binárnych vektoroch skresľuje, globulárne klastre budú mať málo nezhôd na malom počte bitov. V prípade analýzy centroidu klastra sa globularita ukáže ako nízky počet miest, kde je hodnota nižšia ako je pre daný klaster zvyklé. Tento efekt možno vidieť na obrázku 3.1, kde v riadkoch sú časové sekvencie reprezentujúce centroidy klastrov. V stĺpcoch sú jednotlivé časové intervaly. Tmavé regióny označujú hodnoty blízko 1. V riadkoch, klaster 447, ma nízku hustotu a navyše je globulárny, 818 je už skôr konvexný. Klaster 555, 696 a 876 je dobrý príklad toho, čo sa pri klastrovaní binárnych sekvencii bude považovať za dobrý výsledok.

V dátach sa očakáva veľmi veľa skupín, preto ručný odhad počtu klastrov nepripadá do úvahy. Klastre s variabilnou hustotou sa v dátach nutne vyskytnú, pretože sa budú vyskytovať husto aj riedko zaplnené regióny v dátach. Voľba metriky toto tiež významne ovplyvní.

### Aglomeratívne klastrovanie

V skutočnosti sa jedná o triedu algoritmov. Ich základnou myšlienkou je, že sa klastre vytvárajú postupným zlučovaním. Na začiatku sú všetky body vo vlastnom klastri, pre každý klaster sa na základe nejakého kritéria vyberie ďalší, s ktorým sa bude zlučovať. Pokračuje sa v spájaní, kým nie sú všetky klastre spojené do jedného. Výsledkom je hierarchia, binárny strom, ktorý zachytáva vetvenie klastrov a v najhlbšej vrstve v listoch obsahuje jednotlivé vektory dátovej sady. Základným spôsobom výberu kandidátov na zlúčenie je




Obr. 3.1: Príklad centroidov klastrov binárnych vektorov

jednoduché spojovanie, *single linkage*. To spôsobí výber najbližšieho suseda pre zlúčenie na základe vzdialenosti a spojenia v strome, preto uzly môžu byť ohodnotené vzdialenosťou, pri ktorej došlo k spojeniu dvoch susedných klastrov. Komplexnejšie spojovacie funkcie používajú strednú vzdialenosť medzi klastrami (po dvojiciach členov) alebo vzdialenosť centroidov klastrov. Complete linkage používa maximum vzdialeností dvojíc v klastrov. Až je spočítaná hierarchia, je možné vytvoriť rez stromom v nejakej výške tak, aby bola splnená požiadavka na počet klastrov. Tiež sa dá zvoliť maximálna vzdialenosť, nad ktorou sa už zlučovanie nevykoná a zo stromu sa získajú klastre. Výhodou je, že klastre môžu rásť a nasledovať skryté rozloženie miesto toho, aby boli nútené vytvárať guľovité zhluky. Výber rezu strmou v praxi býva spravidla zložitejší. Jednou z metód je prieskumom vývoja počtu klastrov pri narastajúcej výške rezu a hľadaním miesta zlomu, výsledky však nemusia byť jednoznačné. Navyše, aglomeratívne klastrovanie neberie do úvahy outlierov v dátach[4]. Príklad výsledných klastrov je na obrázku 3.2a, rovnaké farby symbolizujú priradenie bodov spoločného klastra. Poznámka v zátvorke udáva nastavenie metódy.

- *Predpoklad*: Globulárne klastre v dátach nie sú nutným predpokladom (závisí na linkage), šum sa však neberie do úvahy a klastre ním budú narušené.
- *Stabilita*: Algoritmus dáva naprieč behmi na rovnakých dátach rozumne podobné výsledky pri podobných nastaveniach.
- *Parametre*: Voľba optimálnej maximálnej vzdialenosti alebo počtu klastrov môže byť komplikovaná.
- *Výkon*: Pri výbere dobre implementácie je dobrý.

### DBSCAN

Algoritmus založený na odhade hustoty v dátach. Predpokladá sa, že klastre ležia v zhustených regiónoch, nevyžaduje priradenie všetkých bodov do klastrov, a teda nejedná sa o

algoritmus ktorých by dáta iba delil. Skôr ide o extrakciu zhustených zhlukov a ich oddelenie od riedkych považovaných za šum. Prakticky je DBSCAN podobný ako aglomeratívne klastrovanie. DBSCAN najprv transformuje dáta v priestore podľa hustoty, body v zhustených regiónoch sa nechajú a body v riedkych regiónoch sa posunú ešte ďalej. Následne sa aplikuje klastrovanie so single linkage, výsledkom čoho je spojovací strom (dendogram). Dendogram sa rozdelí na základe parametra vzdialenosti epsilon (`eps`) a získajú sa klastre. Osamotené klastre a danej úrovni rezu sa považujú za šum a ďalej sa neuvažujú. Algoritmus je možné optimalizovať použitím lokálnych priestorových dotazov, čo posúva výkonnosť implementácií do veľkostí, ktoré sa inak ako pomocou k-means nedajú spracovať.

`Eps` sa môže vyberať ťažko, čo bude vyžadovať analýzu rozloženia vzdialeností pre správaný odhad. Algoritmus môže byť citlivý na nastavené parametre. Transformácia podľa hustoty závisí na parametri `min_samples`. Kombinácia parametrov definuje istú minimálnu hustotu a vo výsledku DBSCAN nájde klastre len hustotou vyššou a rovnakou. Algoritmus dobre nepracuje na dátach so zhlukmi s rôznou hustotou[4].

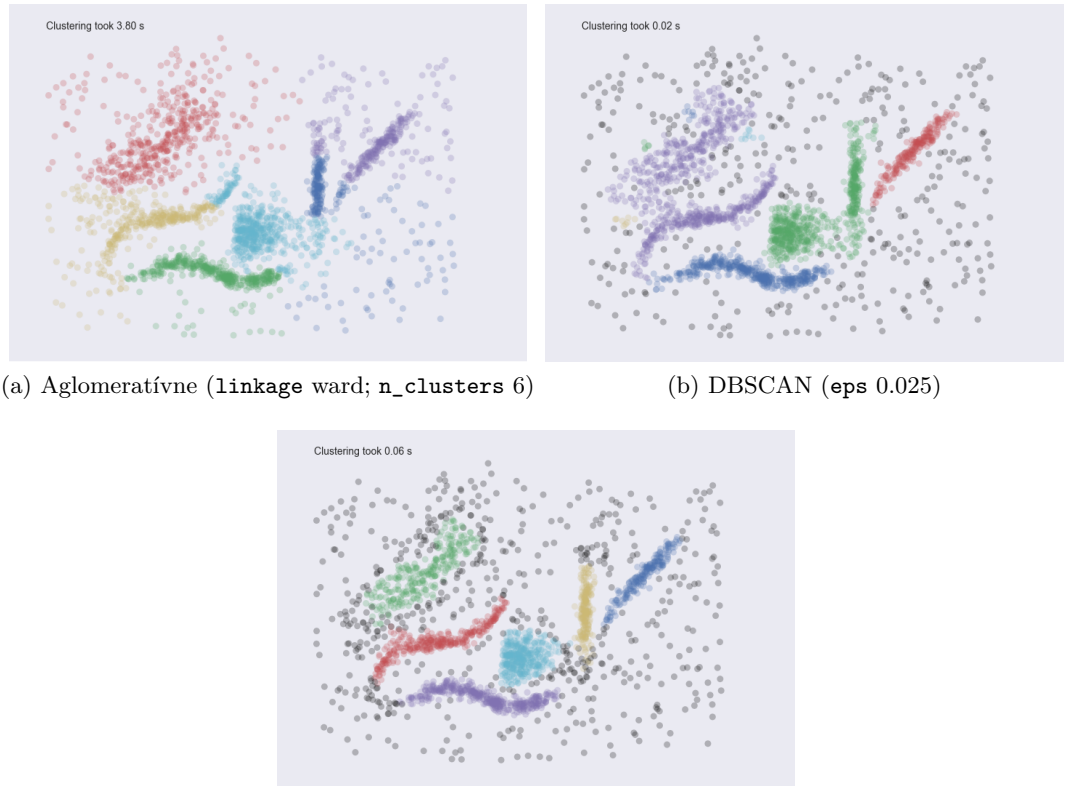
Na obrázku 3.2b je vidieť farebné rozdelenie klastrov pri `eps` = 0.025. Sivou je označený šum. Klastre sú je pomerné dobré, hoci niektoré sú zbytočne zlúčené a našlo zopár veľmi malých klastrov, ktoré mohli byť súčasťou veľkého suseda.

- *Predpoklad*: Klastre nemusia byť globulárne, šum sa odstráni, variácie hustoty klastrov spôsobí nie optimálne rozdelenia.
- *Stabilita*: Stabilný naprieč behmi, nestabilný voči variácii epsilonu.
- *Parametre*: Epsilon je maximálna vzdialenosť okolia klastra, môže byť ťažké ju optimálne odhadnúť.
- *Výkon*: Veľmi dobrý, veľkosť sady môže byť na úrovni k-means.

### HDBSCAN

Algoritmus HDBSCAN je pomerne nový, pochádza od rovnakých autorov ako DBSCAN. Ich cieľom bola metóda ktorá spracuje dáta, kde sa vyskytujú variácie v hustote klastrov. Metóda pracuje rovnako ako DBSCAN až po krok, kde má dôjsť k rezu dendogramu. Miesto rezu podľa konštantne zadanej hodnoty epsilon sa sofistikovanejšie spracuje dendogram, určia sa rezy a zlúčia uzly tak, aby zo vzniknutých klastrov odpadali malé množstva bodov. Cieľom je dendogram zmenšiť a zabezpečiť, aby len malý počet klastrov strácal body. Výsledný strom sa spracuje rezmi v rôznych výškach s cieľom vybrať stabilné klastre s variabilnou hustotou. Tento prístup zároveň eliminuje potrebu konštantne nastavovať parameter epsilon a nahradzuje ho intuitívnejší `min_cluster_size`, ktorý riadi koľko bodov sa odštepuje a znamená približne aký najmenší klaster bude analýza uvažovať[4, 19, 27].

- *Predpoklad*: Ako u dbscan ale variabilná hustota klastrov.
- *Parametre*: `min_cluster_size` je rozumný parameter pre prieskum dát, `min_samples` je zdedený a v HDBSCAN sa nastavuje ťažko. Používa sa prednastavená hodnota.
- *Stabilita*: Stabilný naprieč behmi aj voči nastaveniu parametrov, aj pri použití podvzorkovania.
- *Výkon*: Zatiaľ na úrovni implementácie aglomeratívneho klastrovania fastcluster.




(a) Aglomeratívne (`linkage` ward; `n_clusters` 6)

(b) DBSCAN (`eps` 0.025)

(c) HDBSCAN (`min_cluste_size` 15)

Obr. 3.2: Porovnanie klastrovacích metód[4]

## 3.3 Metriky na priestore binárnych vektorov

Podobnostné a vzdialenostné funkcie zohrávajú kritickú rolu v metódach pre rozpoznávanie vzorov ako klasifikácia a zhlukovanie. Výsledok a rýchlosť výpočtu často závisí od výberu tej správnej metódy, preto bolo za poslednú dekádu vyvinuté nemalé úsilie na identifikáciu tých najvýznamnejších variant funkcií. Týchto metrík je však veľa a rozdiely medzi nimi môžu byť marginálne, prípadne relevantné len pre veľmi špecifické aplikácie.

Prehľadový článok [22] analyzuje podobnosti jednotlivých metrík a experimentálne identifikoval základné skupiny. Medzi významné metriky patria napríklad: Jaccardova, Hammingova a kosinová podobnosť. Každá má svoje špecifické vlastnosti pole aplikácií. Každá sa hodí pre analýzu problémov preložiteľných do vzdialeností binárnych vektorov. Medzi problémy patria napríklad spracovanie neštruktúrovaných dát, klasifikácia textov atď..

**Jaccard index** Jaccard index je klasická miera podobnosti sád obecne, s množstvom praktických aplikácií v získavaní znalostí, dáta mining-u a strojovom učení. Je definovaný ako pomer zjednotenia voči prieniku sád $A$, $B$. Z indexu vyplýva definícia vzdialenosti ako:

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

kde

$$J(\emptyset,\emptyset) =_{def} 1$$

. Jaccardova vzdialenosť spĺňa všetky axiómy metriky[26].

**Hammingová vzdialenosť** Je to najmenší počet pozícií, na ktorých sa reťazce rovnakej dĺžky líšia. Pri aplikácií na binárne vektory je to počet bitov v ktorých sa vektory nezhodujú. V prípade, že vektory majú pevnú dĺžku $n$, je hammingova vzdialenosť metrikou na danom vektorovom priestore.

$$d_H(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} |a_i - b_i|$$

Primárne využite má v teórií kódovania. Pri klastrovaní s hammingovou metrikou je u niektorých algoritmov možné zadať aká je vzdialenosť susedov. Hodnota priamo udáva, aký je maximálny počet rozdielov medzi vektormi v jednom klastri (aglomeratívne, complete linkage), respektíve udáva hustotu klastra ako počet rozdielov na člena (DBSCAN, single linkage).

**Kosínová podobnosť** Táto miera sa používa ako podobnosť medzi dvoma nenulovými vektormi na priestore so skalárnym súčinom. Je ekvivalentná výpočtu kosínusu uhla dvoch vektorov, a preto je rovnomenná. Výsledok nadobúda hodnoty od -1 do 1 udáva mieru rovnobežnosti a zanedbáva faktor magnitúdy v dátach. Používa sa hlavne na priestore pozitívnych vektorov, sem patria aj binárne vektory, kde výstup funkcie je ohraničený. Ohraničenie platí pre ľubovoľný počet dimenzií, a používa sa na vysokodimenzionálnych priestoroch. Pre príklad, v dolovaní informácií z textu sa každému slovu priradí vlastná dimenzia. Početnosť výskytu slova v analyzovanom texte je tak priamo hodnotou v danej dimenzii vektora. Kosínová vzdialenosť takýchto vektorov udáva hodnotou, ako sa tematicky zhodujú porovnávané dokumenty.

Funkciu možno odvodiť z euklidovského skalárneho súčinu.

$$\cos(\theta) = d_c(A, B) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Rečou matematiky je to pomer skalárneho súčinu voči súčinu veľkostí. V prípade, že $A$, $B$ reprezentujú frekvencie uhol (výsledok) nepresiahne 90 stupňov. V oblasti dáta mining-u sa tiež používa ako miera súdržnosti vnútri klastrov. Použitie v praxi tiež dobre mapuje na použitie v klastrovani binárnych časových sérií.

V biológii je známa podobná metrika pod názvom koeficient Otsuka-Ochiai,

$$K = \frac{|A \cap B|}{\sqrt{|A| \times |B|}}$$

kde $A$ a $B$ sú binárne vektory, v tomto prípade $K$ odpovedá kosínovej podobnosti.

Existujú aj ďalšie techniky výpočtu vzdialeností, ktoré umožňujú identifikovať zhodu posunutých vektorov alebo optimálne transformovať jeden vektor na druhý. Bolo by užitočné pri klastrovaní časových binárnych vektorov, aby sa povolili krátke posuny. Prekážkou je časová náročnosť výpočtu, pretože okrem výpočtu základnej metriky je potrebné vykonať druh optimalizácie. Medzi tieto metódy patrí napríklad: Dynamic time warping, Levenshtein distance a Earth mover distance. DTW sa snaží nájsť také vzájomné mapovanie

vektorov, aby bola ich vzdialenosť čo najnižšia. Niekoľko parametrov umožňuje optimalizovať túto metódu. Lehvensteinova vzdialenosť pracuje podobne. Miesto hľadania zarovnania vektorov sa hľadá postupnosť operácii insert, delete a substitute symbolov taká, že cena transformácie vektora je minimálna. Eearth mover distance na pravdepodobnostných rozloženiach minimalizuje cenu transformácie jedného rozloženia na druhé, pričom cenu udáva súhrnné množstvo presunutých jednotiek vynásobené vzdialenosťou presunu. Experimenty s Dynamic time warping-om však odhalili, že časová náročnosť výpočtov tohoto typu neumožní použitie na bezpečnostných udalostiach, spracovateľné sú iba stovky vektorov naraz a komplexita stúpa s dĺžkou.

Znalosť sémantiky jednotlivých vzdialeností umožňuje lepšie pochopiť výsledky klastrovania binárnych vektorov.

# Kapitola 4

# Návrh metódy pre vyhľadávanie skupín IP v bezpečnostných udalostiach podľa časových vzorov

V tejto práci bola navrhnutá metóda pre koreláciu bezpečnostných udalostí v čase. Pracovne bola pomenovaná ako *Security Event Correlation in Time* (SECT). Cieľom tejto metódy je odhaliť skupiny IP adries, ktoré generujú *bezpečnostné udalosti* v zhodných časových intervaloch. Časové intervaly sú reprezentované binárnymi vektormi s konštantnou dĺžkou.

Na počítačových sieťach je potrebné monitorovať komunikáciu a bezpečnostné incidenty. K tomuto správcovia sietí používajú rôzne technológie a riešenia. Čo sa týka detailnosti, monitoring prebieha na rôznych úrovniach v sieti. Existujú servery sledujúce pripojenia na nepriradené časti adresného priestoru, známe ako honeypot systémy. Ich cieľom je analýza aktivít škodlivého softvéru na internete. Sledujú sa toky komunikácií na významných bodoch v sieti, na hraničných linkách atď. Tieto sa koncentrujú v dátových centrálach pre účely analýzy. O triedu detailnejšie informácie poskytuje cielený záchyt paketov komunikácie, ktorý sa tiež môže analyzovať pre účely hlásenia bezpečnostných udalostí. Variant je viacero a predošlé príklady sú uvedené pre lepšie pochopenie problematiky. Z týchto dát sa generujú bezpečnostné udalosti, a to pomocou sledovania štatistických ukazateľov vypočítaných pre jednotlivé pripojenia, sledovaním anomálií, či rozpoznaním vzorom. Výsledné udalosti popisujú napríklad horizontálne a vertikálne skeny na sieti, zachytené infekcie malwarom a ďalšie. Hlavnou pointou je, že bezpečnostné udalosti nesú rozmanité množstvo informácií sú generované riešeniami pre monitorovanie aktivity a ich zdrojové dáta budú často od podstaty odlišné.

Vyššie spomenuté riešenia pre monitorovanie budú v tejto práci ďalej spadať pod výraz *detektory*. Zistenia z detektorov sa predávajú správcom vo forme bezpečnostnej udalosti, tieto predstavujú záznamy v špecifickom formáte, napríklad IDEA[21]. Analýzou bezp. udalostí sa dáva do súvislosti široké spektrum javov čo môže skrývať zaujímavé zistenia vo forme kolektívnych anomálií. Výsledkom hypotetickej metódy pre analýzu bezp. udalosti, založenej na korelácií vzorov v čase by bolo zistenie v tvare: entity $\{x, y, z\}$, $\{u, v, w\}$, ... sa vyskytujú v zhodných časových intervaloch. Istý problém predstavuje, validácia a interpretácia takýchto výsledkov. Návrh metódy predpokladá spracovanie bezp. udalostí vo formáte IDEA popísaný v sekcii 4.2. Metóda však dokáže spracovať všetky typy udalostí, ktoré obsahujú zdrojovú entitu a časovú značku vzniku. Ďalšie informácie obsiahnuté v bezp. udalosti napomôžu analýze výsledkov.

### Výskumné otázky

Akademická sieť CESNET monitoruje svoju sieť CESNET2 vlastnou verziou SIEM systému. Bezpečnostné udalosti sa uchovávajú a posielajú do systému Warden pre zdieľanie ďalším organizáciám. Zároveň s tým sa systém Mentat umožňuje výskumníkom vykonávať dotazy nad akumulovanou sadou. Samostatné bezp. udalosti predstavujú informáciu využiteľný len krátky čas. Dianie na sieti sa zmení, niektoré stroje zmenia svoje dynamické IP adresy a znalosti nie sú úplne aktuálne. Ich uchovávanie sa rýchlo stáva zbytočným. Ak však dôjde k zdieľaniu, údržbe a cielenému obohacovaniu sád bezp. udalostí o ďalšie informácie, vzniká zdroj cenných informácií pre výskum v bezpečnosti počítačových sietí. Vo veľkých sadách udalostí narastá význam časovej dimenzie, čo umožňuje hlbšiu analýzu a možnosť získavať odpovede na konkrétnejšie otázky o častom dianí na skúmaných sieťach. Otázka, na ktorú sa snaží odpovedať metóda SECT znie: **Existuje významná koordinovaná činnosť na monitorovanej sieti označená ako škodlivá?** Význam tejto otázky vo vzťahu k bezpečnostným udalostiam sa ďalej skúma v pod otázkach:

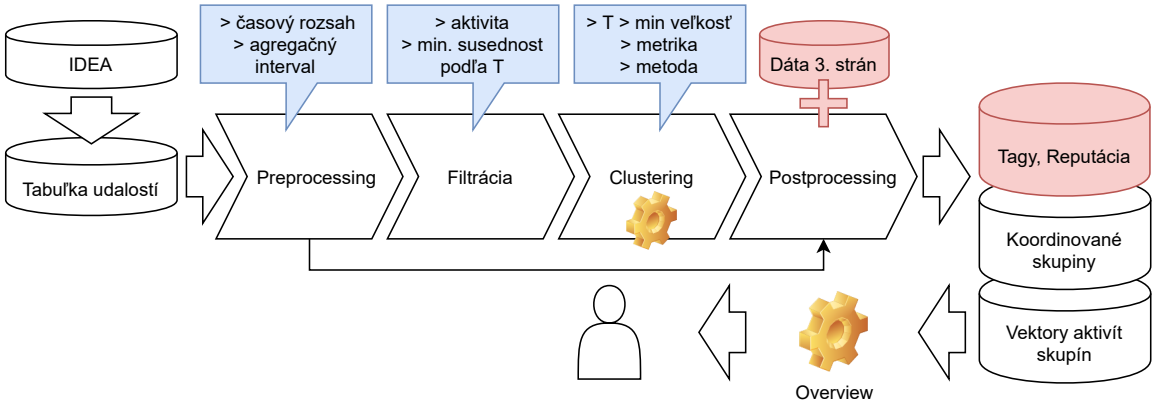
- Čo bude použité za kritérium koordinácie?
- Ako sa zistia koordinujúce skupiny?
- Aké skupiny entít (IP) koordinujú?
- Existujú v ďalších dostupných dátach informácie, ktoré by potvrdili platnosť odhalených skupín?
- Existuje informácia, ktorá by vyvrátila škodlivosť činnosti skupiny?

Prvé dve otázky sú predmetom návrhu metódy. Zvyšné otázky budú skúmané v sekcii 4.5. V jadre SECT metódy sa používa klastrovací algoritmus pre spájanie IP adries do skupín podľa vzorov chovania. SECT rieši problém hľadania koordinovaných skupín IP v postupných krokoch zobrazených na obrázku 4.1. Modré prvky tam označujú premenne ovplyvňujúce krok, červené komponenty sú voliteľné. Disky označujú dátové vstupy a výstupy.

### Metóda SECT v krokoch

- Predspracovanie udalostí a časových vektorov IP
- Filtrácia dát
- Klastrovanie
- Postprocessing klastrov na skupiny
- Overenie skupín

Metóda použije dátový model na obrázku 4.2. Tabuľky sa postupne budujú počas analýzy, preto sú niektoré polia označené farebne. Zelené polia sú značky klastrov, modré polia sú vyplnené na základe časových značiek udalostí, červené sú príznaky vypočítané z vektoru aktivít. Primárne sa budú analyzovať výsledky v tabuľke skupina a príznaky.




Obr. 4.1: Metóda SECT v krokoch

## 4.1 Kritérium koordinácie medzi entitami

Metóda SECT rozhoduje kritérium koordinácie nasledovne: Pre jednotlivé entity sa spracujú udalosti do formy vektorov, dĺžka vektora závisí od skúmaného časového rozsahu a veľkosti intervalu, do ktorého sa agreguje informácia o aktivite. Vektory budú obsahovať sériu {`True, False`} hodnôt. Nad týmito vektormi sa ďalej zistí relácia korelácie pomocou analýzy vzájomných vzdialenosti. Ak entity spĺňajú kritérium koordinácie potom vektory ich aktivít budú v relácii korelácie. Presnejší zápis definícií bude nasledovný:

**Definícia 4.1** (Vektor aktivity)**.** Nech vektor $x \in$ kóduje aktivitu entity v sledovanom časovom rozsahu nasledovne. Vektor obsahuje číslo `1` alebo `True` práve keď, má daná entita zaznamenanú aspoň jednu bezpečnostnú udalosť v odpovedajúcom podintervale sledovaného rozsahu.

Nech existuje množina vektorov aktivít (ďalej len vektory)

$$\Phi \subseteq \{(\{0,1\}^n)\}^*$$

reprezentujúca aktivitu skúmaných entít. Premenná $n \in \mathbb{N}$ udáva dĺžku vektoru a teda počet sledovaných podintervalov.

S využitím predošlých definícií sa bude predpokladať, že platí predpoklad o koordinácii.

**Definícia 4.2** (Korelácia je dôsledkom koordinácie)**.** Ak existuje koordinácia medzi entitami, potom platí relácia korelácie medzi odpovedajúcimi vektormi aktivít.

$$\texttt{koordinácia}(e_x, e_y) \implies \texttt{korelácia}(x, y)$$

Vo vzťahu k tomuto sa definuje relácia korelácie.

V súvislosti s analýzou vektorov aktivít sa výrazom korelácia myslí práve relácia korelácie tak ako je ďalej definovaná.

**Teorém 4.1** (Relácia korelácie)**.** *Nech medzi dvoma vektormi $x, y$ platí relácia korelácie $\simeq$ práve keď ich* vzdialenosť *je menšia ako rozhodujúca hranica* T*.*

$$\delta(x, y) < T$$

Korelácia môže nastať ešte z ďalších dôvodov. Do tej doby nech čitateľ vezme na vedomie, že relácia korelácie sa rozhoduje na základe vektorov odvodených zo vstupných dát, v nich existuje šum v a náhodné korelácie.

V dobe návrhu mi nebolo známe akú metriku je vhodné zvoliť, ani akú hodnotu $T$ je optimálne použiť. Rozhodol som sa preto dočasne pracovať s *Jaccardovou* metrikou, ktorá nadobúda v hodnoty v rozsahu $\langle 0, 1\rangle$. Intuitívne táto metrika dáva do pomeru počet zhodných `True` intervalov, voči počtu intervalov kde sa aspoň v jednej z dvojíc vyskytlo `True`. To, že hodnotu výsledku ovplyvňuje len aktívna časť dvojice je výhodou, pretože indikuje percentuálny pomer zhody dvoch sád. Nech by bola povolená tolerancia povedzme 5%. To znamená, že ak majú dva vektory spoločnú aktivitu v 20 intervaloch, jednému vektoru môže aktivita v jednom intervale chýbať. Metrika a hodnota $T$ sú akýmisi stupňami voľnosti v konfigurácii SECT metódy. Optimálne hodnoty sa zistia z experimentov a teda uvedené konštanty nech sú iba prvotnými odhadmi. Podrobnejší popis metrík a klastrovacích metód je v sekcii 3.2.

Nech pre koordinované skupiny platí, že v rámci skupiny sú všetky vektory vzájomne korelované. Vyvstáva problém vyhľadávania skupín a koreláciu je treba rozšíriť na skupiny.

**Teorém 4.2** (Korelácia skupiny)**.** *Nech $\simeq$ je relácia z Teorému 4.1 . Potom pre skupiny $G$ entít $e$ platí:*

$$e \in G \iff \exists x \in G\ (e \simeq x)$$

*Teda, prvok $e$ patrí do $G$ ak koreluje s jedným z členov skupiny.*

Zavedením korelácie nad skupinou vzniká možnosť, že sa v jednej skupine vyskytnú vektory $x, y$ s vlastnosťou $\delta(x, y) \geq T$. Definícia korelácie skupín 4.1 vedie na použite klastrovancích algoritmov a vzniká parameter metódy SECT:

- `distance_threshold` Parameter udáva hodnotu vzdialenosti okolia klastra v algoritmoch. Odpovedá popisovanému $T$ ako je zadané v definícií 4.1.

$T$ zároveň odpovedá vzdialenosti okolia klastrov v prípade použitia metódy, ktorá pracuje so single linkage. Istá miera nezhody je v rámci skupiny povolená už z definície korelácie a je žiadúca v metóde SECT. Nevýhodou je možnosť vzniku korelovaných skupín kde v rámci skupiny je maximálne $T$ príliš vysoké. Vznik takýchto skupín bude minimalizovaný výberom a nastavením klastrovacieho algoritmu. Spôsob nastavenia parametrov bude riešiť implementačná časť. Z praktického hľadiska je žiadúce aby metóda prebehla v dostatočne krátkom čase a získavala relevantné výsledky. Je potrebné brať na vedomie, že výpočet matice vzájomných vzdialeností má zložitosť $O(n^2)$. Dáta bude potrebné filtrovať ešte pred klastrovaním, pretože sa budú spracovávať veľké objemy dát a snahou bude analyzovať dlhšie a dlhšie časové rozsahy. Deň, týždeň či mesiac, zväčšujúci sa rozsah má za následok predlžovanie vektorov aktivít a vyšší počet analyzovaných entít.

Riešenie zostavenia korelovaných skupín je popísané v kapitole 4.4. Výsledkom SECT je množina skupín koordinovaných entít, preto sa musí zabezpečiť platnosť implikácie opačnej k formule v definícii 4.2:

**Definícia 4.3** (Koordinované skupiny)**.** Nech po minimalizácií šumu a náhodných korelácií v dátach pre výsledné skupiny platí:

$$\texttt{korelácia}(x, y) \iff \texttt{koordinácia}(e_x, e_y)$$

Ako ku náhodným koreláciám môže dochádzať je popísané v sekcií 4.3. Viac informácií vlastnostiach dát a o šume je v nasledujúcej sekcií.

Prechod od korelovaných ku koordinovaným skupinám popisuje sekcia 4.5.

Náhodné korelácie potlačia minimalizáciou šance ich výskytu vo výsledkoch a to filtráciou dát pred vstupom do analýzy. V prípade dostupnosti odhadu pravdepodobnosti náhodnej korelácie je možné stanoviť limity pre minimálnu veľkosť skupiny a minimálnu aktivitu na základe maximálnej akceptovateľnej pravdepodobnosti.

## 4.2 Predspracovanie udalostí do vektorov aktivít

Dátová sada pozostáva z jednodenných súborov so záznamami v textovej podobe. Súbory obsahujú bezpečnostné udalosti vo formáte IDEA, môžu byť zároveň komprimované do formátu gzip. Formát udalostí odpovedá zoznamu riadkov štruktúrovaného textu. Každý riadok zaznamenáva jednu bezpečnostnú udalosť. Jeden deň bezpečnostných dát analyzovaných v tejto práci obsahuje približne $2.2M$ udalostí a v nich figuruje cca $400k$ IP adries. V nekomprimovanej podobe má textový súbor veľkosť 1.5 GB.

Základná štruktúra udalosti je statická, ibaže IDEA formát je vlastne JSON s definovanou štruktúrou. Prístup ku položkám vo formáte JSON je pomalší ako v prípade jednoduchej tabuľkovej štruktúry kvôli nutnosti použitia parseru. Prístup k položkám sa dá optimalizovať použitím preprocessingu resp. predspracovania dát. Hlavným cieľom predspracovania je previesť surové dáta do tvaru vhodného pre spracovanie metódami strojového učenia. Predpokladá sa, že jeden súbor bude analyzovaný s rôznymi parametrami, predspracovany IDEA súbor sa preto oplatí uložiť pre budúce behy. Jednotlivé bezp. udalosti sa teda prevedú do zjednodušenej tabuľky. V nej sú zachytené len informácie relevantné pre analýzu. Príkladom môže byť tabuľka 4.1, kde v hlavičke sú kľúče IDEA, z ktorých sa extrahujú dáta do stĺpca, pod kľúčom sú požadované typy a hodnoty obsahujú kompletný alebo skrátený zápis.

Každý záznam obsahuje čas, v ktorom k bezpečnostnej udalosti došlo. Túto funkciu spĺňajú EventTime alebo DetectTime IDEA polia, pričom sa preferuje EventTime kvôli vyššej presnosti. IP slúži k identifikácii entity v detekcii. Polia Node.Name a Node.Type tieto budú prispievať ku charakterizácií skupiny na konci analýzy. Takto redukované IDEA správy sú pripravené na prevod do vektorov.

Bezpečnostné udalosti sa forme zjednodušenej tabuľky sa uložia pre použitie v ďalších behoch analýz. Ideálne by bolo dáta ukladať až po prevode do vektorov, to ale nie je praktické a to z viacerých dôvodov. Hlavným dôvodom je snaha umožniť analýzu s iným časovým rozlíšením. Ak by dáta boli uložené až po prevode do vektorov, možné rozlíšenia analýzy by boli už iba násobkami základného rozlíšenia, narástla priestorová náročnosť pre uloženie a zamedzilo by sa pripojeniu polí pre charakterizáciu. Navyše, zjednodušená

| Source.IP<br>string | EventTime<br>timestamp | Node.Name<br>string | Node.Type<br>string |
|---|---|---|---|
| 51.91.154.72 | 1583424775 | [{'Name': 'cz.cesnet.au1.wa... | ['Recon.Scanning'] |
| 94.182.241.190 | 1583405108 | [{'Name': 'cz.cesnet.au1.wa... | ['Recon.Scanning'] |
| 140.14.178.9 | 1583399243 | [{'AggrWin': '00:15:00', ... | ['Other', 'Test'] |
| 122.51.83.60 | 1583392554 | [{'Name': 'cz.cesnet.collec... | ['Attempt.Login', 'Test'] |
| 83.209.6.220 | 1583365837 | [{'Name': 'cz.cesnet.au1.wa... | ['Recon.Scanning'] |

Tabuľka 4.1: Polia popisujúce udalosti pri predspracovaní

tabuľka udalostí je aj tak, čo sa pamäťovej náročnosti týka na rovnakej úrovni ako riedka matica, čo je ideálny formát pre použitie v tejto práci. Nevýhodou tohoto prístupu je nutnosť pred každou analýzou prevádzať udalosti na vektory.

### Prevod udalostí do vektorov

Analýza skupín potrebuje pracovať s udalosťami v takej forme aby reprezentovali aktivitu entít v čase pomocou vektorov. V kontexte predspracovania je toto podstatne výraznejšia zmena formy dát. Postup prevodu je nasledovný, zvolí sa parametre prevodu. Tými sú časové rozlíšenie a časový rozsah.

- `aggregation` rozlišovací interval, šírka dimenzie vektora aktivity
- `dfrom` filter časových značiek, udáva počiatok analýzy
- `dto` udáva koniec analýzy

Zoznam vyjadruje označenia pre jednotlivé parametre Počet intervalov vektorov. akt. je daný časovým rozsahom a agregačným oknom. Z tabuľky sa vyberú len udalosti, ktorých časové značky ležia vnútri skúmaného intervalu. Pre každú IP sa vytvorí vektor aktivity. Komponenty vektorov sa položia rovné $False$. Vektor aktivity $v_x$ entity $x$ obsahuje $True$ len na indexoch, kde entita $x$ vygenerovala aspoň jednu udalosť s časovou značkou v odpovedajúcom intervale. Vektory aktivít sú vektormi booleovských hodnôt, preto je možné pri korelácii použiť tiež binárne metriky. Všetky vektory sa spoja do matice $M(r, s)$, kde $r$ je počet IP, $s$ je počet intervalov.

Nad vektormi aktivity v $M$, sa spočíta vektor príznakov (*feature*). Hodnoty sa prevedú do tabuľky $M\phi(r, :)$, priradenie riadkov k entitám musí odpovedať tomu v $M$. Použili sa nasledujúce príznaky:

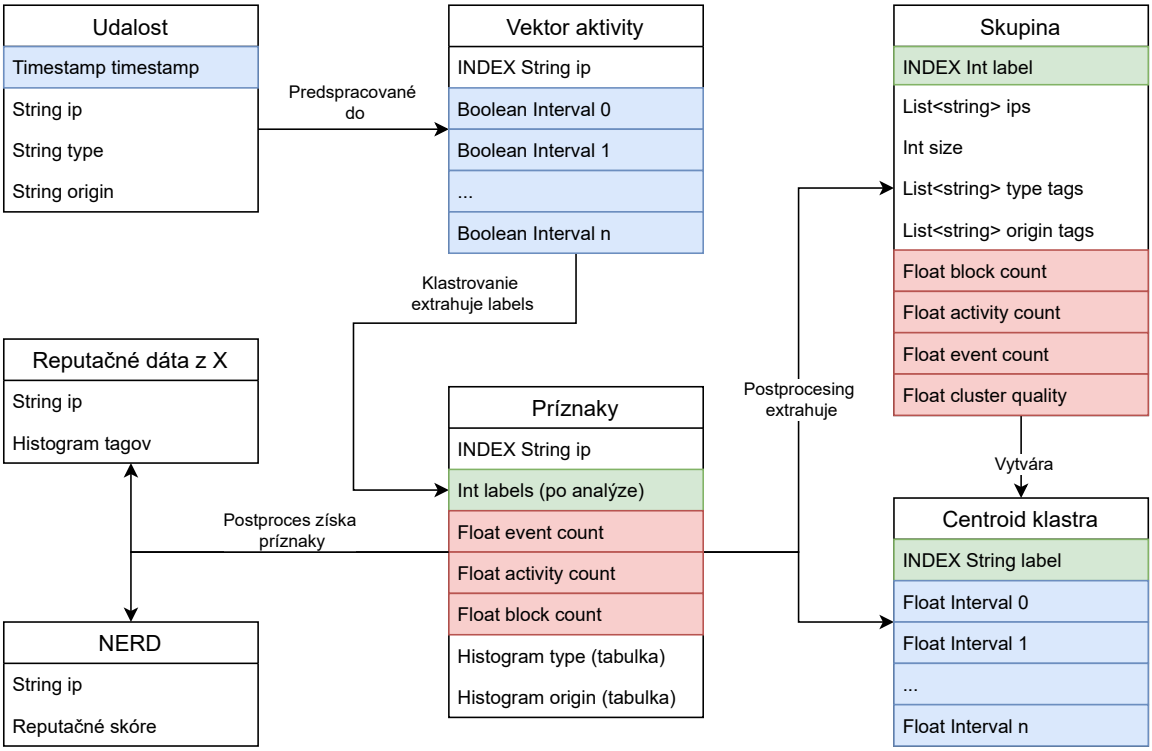
- *event count* Celkový počet udalostí pre danú IP.
- *activity count* Aktivita vektora, teda počet intervalov, v ktorých bola IP aktívna.
- *block count* Počet blokov intervalov, kde bola IP aktívna. Susedné intervaly patria do spoločného bloku práve keď sú oba aktívne.

Tieto príznaky umožňujú vyberať entity s rôznymi vlastnosťami, to má význam pri overení výsledkov. V implementácií sa však zistilo, že filtrovanie podľa počtu udalostí a aktivity napomáha zrýchleniu behu algoritmu pomocou zahadzovania nezaujímavých entít. Príznak *activity count* je dôležitý hlavne pre účely vylúčenia náhodných korelácií z výsledkov.

Model na obrázku 4.2 zobrazuje rozdelenie informácií v analýze do tabuliek. Farby odlišujú polia od bežných. Červené polia sú príznaky spočítané z časových vektorov, modré sú časové hodnoty a zeleným sú indexy klastrov, teda skupín. Šípky naznačujú vzťahy pri vytváraní modelu.

### Zistenia po predspracovaní dát

Po transformácií udalostí a predspracovaní dáta významne menia formu. Pri návrhu metódy som potreboval vedieť očakávanú veľkosť sady dát, aby som zvážil ako efektívne vyhľadať skupiny. Z vektorov aktivít boli zistené trendy v príznakoch, ktoré som zhrnul nasledujúcich pozorovaní.




Obr. 4.2: Dátový model analýzy

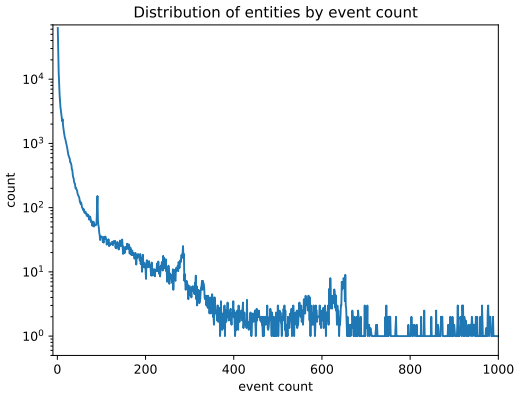
1. Vyskytne sa omnoho viac IP s malým počtom eventov ako s veľkým
2. So zväčšujúcim sa časovým rozsahom analýzy rastie počet jedinečných IP.
3. So zväčšujúcim sa agregačným oknom sa zvyšuje generalizačná schopnosť detektora.
4. So zmenšujúcim sa agregačným oknom sa zvyšuje citlivosť na `šum` v dátach.

Predbežné zistenia z dát pomohli lepšie sa rozhodovať vo veciach návrhu korelácie skupín. Informácia o rozložení početnosti entít podľa *activity count* na obrázku 4.3b poskytuje hodnoty pre odhad pravdepodobnosti náhodnej korelácie pre n-tice. Ten je dôležitý pri nastavení prahu pre filtrovanie dátovej sady pred krokom korelácie skupín, pretože vyšší počet jedinečných IP zvyšuje časovú náročnosť analýzy skupín.

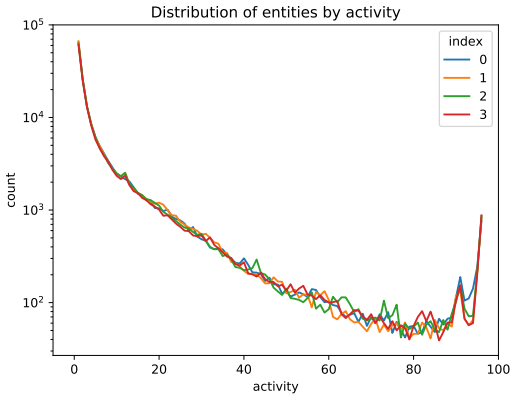
Graf na obrázku 4.3a ukazuje aké sú počty IP s rôznymi množstvami vygenerovaných udalostí a potvrdzujú dohad 1). Z tabuľky 4.3 sa tiež zistilo, že počet entít bude neúnosne stúpať pri analýze intervalov dlhších, ako štvrť dňa a preto je potrebné zaviesť filtrovanie ešte pred samotným výpočtom korelácie skupín. Rozloženie početnosti z grafov ukazuje, že filtrácia by mala mať malý vplyv na nájdené validné skupiny. Je to pretože väčšina entít má zanedbateľný počet udalostí a aj hodnôt aktivity, to zobrazuje tabuľka 4.2. Prečo dáva zmysel filtrovať aj výsledné skupiny so špecifickou aktivitou je podrobnejšie vysvetlené v sekcii 4.3.

## 4.3 Náhodné korelácie vektorov

Za začiatku kapitoly v sekcii 4.1 bolo definované, čo sa skrýva pod pojmom koordinácia. Pri rozhodnutí, či na základe korelácie dochádza aj ku koordinácii je potrebné vylúčiť možnosť




(a) Početnosť entít podľa počtu udalostí




(b) Početnosť entít podľa aktivity pri intervale $agg = 900s$

Obr. 4.3: Predbežná analýza aktivity a početnosti udalostí

náhodnej korelácie. Pre IP adresy generujúce bezpečnostné udalosti v podobných časoch je pravdepodobnosť, že ich činnosť je koordinovaná. To inak znamená pre entity, ktorých časové vektory aktivít sa zhodujú existuje pravdepodobnosť, že dané entity koordinujú alebo došlo k náhodnej zhode. Aby bola istota rozhodnutia o korelácií dostatočne vysoká, musí sa vylúčiť, že korelácia nastala na základe náhodnej zhody. Táto sekcia rieši ako odhadnúť pravdepodobnosť náhodnej zhody.

Začiatok tejto sekcie vznikal s intuitívnym predpokladom o 'nenáhodnosti korelácie' skupín. Budú existovať IP adresy, ktorých aktivita je príliš nízka, a nemá význam takéto skupiny sledovať, pretože medzi nimi vznikne veľké množstvo náhodných korelácií. Podobne sa vyskytnú sa IP, ktoré budú aktívne takmer vždy, čo spôsobí splývanie viacerých potencionálne separátnych skupín.

Po bližšom štúdiu som formuloval pohľad na problém: Nech sledujeme vektory aktivít o dĺžke $n$. Vektory aktivít sú prvkami množiny $U = \{0, 1\}^n$. Veľkosť množiny $|U_a|$ je vyjadrená kombinačným číslom. Nech, pre príklad, za koreláciu považujeme jedine presné zhody vektorov (pri T=0). Priestor, z ktorého sú vektory $v_a$ s aktivitou $a$ má počet možných

| | event count | activity | blocks |
|---|---|---|---|
| count | 167278.000000 | 167278.000000 | 167278.000000 |
| mean | 14.808573 | 7.887304 | 3.538757 |
| std | 152.461020 | 14.503242 | 4.728274 |
| min | 1.000000 | 1.000000 | 0.000000 |
| 25% | 1.000000 | 1.000000 | 1.000000 |
| 50% | 2.000000 | 2.000000 | 2.000000 |
| 75% | 8.000000 | 7.000000 | 4.000000 |
| max | 56825.000000 | 96.000000 | 47.000000 |

Tabuľka 4.2: Štatistický popis dátovej sady

prvkov daný:

$$|U_a| = \binom{n}{k} = \frac{n!}{a! \cdot (n-a)!}$$

Priestor ja najväčší, keď $a \to n/2$. Šanca, že ku korelácií dôjde náhodne je daná počtom entít s vektorom aktivity $v_a$ spadajúceho do v priestoru $|U_a|$.

**Predpoklad 4.1** (Uniformné rozloženie udalostí)**.** *Nech nulovou hypotézou je, že ku koreláciám dochádza náhodne a vektory aktivít sú v priestore rozložené uniformne. Pravdepodobnosť každej z možností je:*

$$p_a = \frac{1}{|U_a|}$$

*potom, pravdepodobnosť, že vo vzorke detekovaných vektorov aktivity $V_a$ dôjde k zhode n-tice je daná Bernoulliho vetou. Ako počet nezávislých opakovaní sa použije veľkosť množiny $|V_a|$.*

**Teorém 4.3** (Binomické rozloženie)**.** *Nech pravdepodobnosť náhodnej udalosti $X$ je v každom pokuse (bernoulli trials) $P(X) = p$, potom pravdepodobnosť $P(k, x)$ toho, že v sérii $k$ opakovaní náhodného pokusu nastane náhodná udalosť $X$ práve $x$-krát $(0 \leq x \leq k)$ je rovná:*

$$Bi\ P[X = x](k, x) = \binom{k}{x} p^x \cdot (1-p)^{k-x}$$

.

Pre kompletnosť vysvetlenia uvážme nasledujúci príklad: Nech SECT beží nad 8 hodinovým časovým intervalom rozdeleným po 15 minútach. IP adresy nech sú rozdelené rovnomerne naprieč celým spektrom aktivity, to udáva počet opakovaní nech sa zvolí $k = 8000$. Vektor aktivity pre každú z IP je dlhý $n = 8 \cdot 60/15 = 32$. Veľkosti priestorov možností pre jednotlivé aktivity udáva:

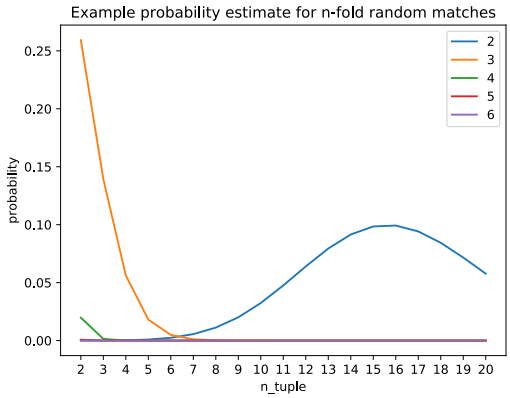
$$|U_2| = \binom{32}{2} = 496$$
$$|U_3| = \binom{32}{3} = 4960$$
$$|U_4| = \binom{32}{4} = 35960$$
$$|U_a| = \binom{32}{a} = \dots$$

Nech $k$ použité vo výpočte považujeme za počet opakovaní. Hodnoty pravdepodobností z rovnice teoréme 4.3 pre 2-20 ice boli vynesené do tabuľky. Graf je na obrázku 4.4. Zo

| Dní kumulatívne | $\sum$ udalosti | $\sum$ jedinečné IP |
|---|---|---|
| 2020-03-05 | 2569953 | 169292 |
| 2020-03-06 | 4999811 | 261206 |
| 2020-03-07 | 7480056 | 336523 |
| 2020-03-08 | 9908594 | 402066 |

Tabuľka 4.3: Kumulatívna veľkosť sady

zdrojových dát predošlého grafu vyplýva tabuľka, ktorá zobrazuje aká je pravdepodobnosť výskytu n-tice, kde $x < 21$, pri 8000 opakovaniach. Každé opakovanie reprezentuje výskyt jednej IP v kontexte analýzy SECT.




Obr. 4.4: PDF náhodných zhôd n-tíc podľa Bernoulliho vety, kde $k = 8000$, $x \in \langle 2 : 20 \rangle$, $p_a = 1/|U_a|$

| Activity | $\sum$ Random match prob. |
|---|---|
| 2 | 0.861060 |
| 3 | 0.479247 |
| 4 | 0.021363 |
| 5 | 0.000768 |
| 6 | 0.000039 |

Tabuľka 4.4: Pravdepodobnosť výskytu zhodnej n-tice vektorov, kde $1 < x < 21$, podľa grafu na obrázku 4.4

Zrejme platí, že kým je priestor, v ktorom má ku koreláciám dochádzať podstatne väčší ako počet opakovaní, je pravdepodobnosť náhodnej korelácie zanedbateľne malá. K tomuto je treba pridať zistenie, že $|U_a|$ rýchlo klesá s rastúcou aktivitou, na základe grafu na obrázku 4.3b.

Tabuľka 4.4 zobrazuje pravdepodobnosti vygenerovania násobnej zhody. Je však možné použiť odhad strednej hodnoty $x$ s maximálnou pravdepodobnosťou a rozptyl okolo tejto hodnoty[2].

$$E(x) = xp$$
$$D(x) = xp(1-p)$$

V tomto zjednodušenom prípade sa namiesto binomického použije normálne rozloženie. Normálne rozloženie s daným parametrami umožní odfiltrovanie pravdepodobne náhodných skupín. Aby sa však vyhlo presným počtom s nepresnými odhadmi je možno zvoliť horné a dolné ohraničenie aktivity a dolné ohraničenie veľkosti skupiny. To poslúži ako vstup do filtra udalostí ešte pred získaním skupín alebo pri verifikácií výsledkov.

Použitie binomickej vety pre odhad náhodnosti výsledkov však nezohľadňuje možnosť korelácie okrem zhodných aj podobných vektorov. V tejto práci som skúsil nasledujúce prístupy ako tento problém vyriešiť:

- *Náhodnosť skupiny 'per partes' zo zhôd* - Náhodnosť presnej zhody je daná Binomickým rozložením, pre veľké skupiny sa však výpočet rozpadne. Je možné odhadnúť ich zjednodušeným vzorcom a vyhnúť sa dlhému reťazcu exponentov vo výpočte. Pri vysokej hodnote faktoriálu pre výpočet $|U_a|$ možno prehlásiť $1/|U_a| \approx 0$. Použitie by vyžadovalo pravdepodobnosť skupiny zistiť z pravdepodobností podskupín so zhodnými vektormi. Pravdepodobnosti sa sčítajú. Výsledná pravdepodobnosť najskôr nebude odpovedať odhadu aký by poskytla binomická veta v kontexte, ktorý by zohľadňoval odchýlky vektorov podľa $T$.

  *TLDR: Kombinácia pravdepodobností pomocou váženého priemeru nedá spoľahlivé výsledky.*

- *Filtrácia podľa modelu* - Na základe pravdepodobnostného modelu bola zistená charakteristika náhodnosti zhôd v dátach. Z grafov na obrázkoch 4.5a a 4.5b sa usúdi nastavenie trojice filtračných parametrov. Zvolí sa maximálna akceptovateľná pravdepodobnosť náhodnej zhody $P$ a odčítajú sa parametre. Preferuje sa nastavenie minimálnej a maximálnej aktivity, pretože podľa aktivity je možné vektory efektívne filtrovať ešte pred klastorvaním.

  Výsledné skupiny sa prefiltrujú podľa príznaku veľkosti. Zvyšné výsledky sú pravé z istotou $1 - P$. Krok od náhodnosti presnej zhody ku náhodnosti korelácii sa rieši aproximáciou. Funguje to, pretože pri nastavení odchýlky vzdialenosti $T$=*0.05* v korelácii pomocou jaccardovej metriky nie je možné korelovať vektory menšie ako 20, bez toho aby sa zhodovali presne. To znamená, že pre malé aktivity sa klastrovanie správa ako vyhľadávanie presných zhôd. Vektory s aktivitou v hornej hranici možných hodnôt budú korelované pri väčšom absolútnom počte rozdielov, to však kompenzuje rádovo nižšia početnosť vektorov s danou aktivitou v dátach. Viďte obrázok 4.3b, kde na pravej strane je početnosť entít radikálne menšia ako na ľavej

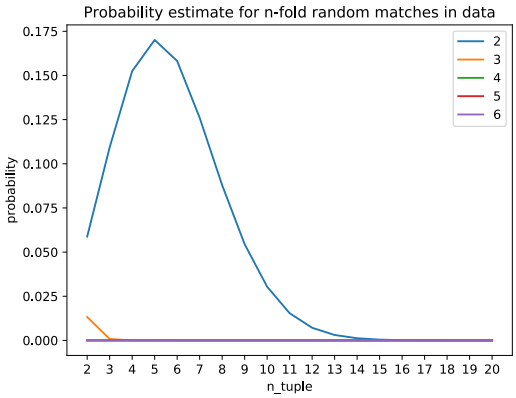
  *TLDR: Odľahčí výpočet ale zahadzuje viac výsledkov ako je nútné.*

Pre účely overenia skupín sa použije varianta filtrácie podľa modelu. Na základe toho sa vytvárajú nasledujúce filtračné parametre, ako parametre metódy SECT. Pomer aktívnych intervalov sa vzťahuje voči dĺžke vektora aktivity.

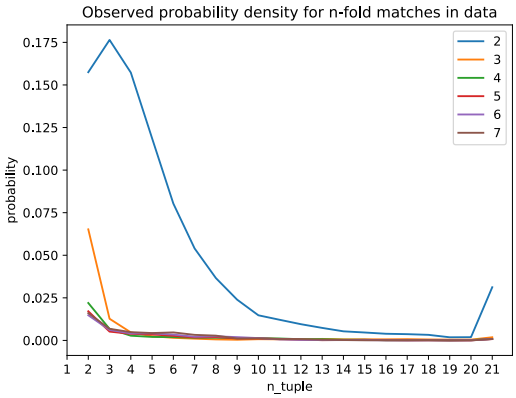
- `min_activity` Udáva minimálny pomer aktívnych intervalov, aby bola entita uvažovaná v analýze.
- `max_activity` Udáva maximálny pomer aktívnych intervalov.
- `min_cluster_size` Minimálna veľkosť skupiny uvažovaná v analýze. Primárne je snaha udržať hodnotu čo najnižšie a filtrovať až výsledky ak je to nutné.

### Predbežné zistenia z dátovej sady

Po príprave predspracovania pripravené som vykonal predbežnú analýzu sady. Zvolil som časový interval dlhý jeden deň, s *agregačným intervalom* `aggregation` = 900, teda 15 minút a $T = 0$, dáta sa nefiltrovali. Bola vykonaná agregácia vektorov aktivít podľa zhody. Z dát




(a) Očakávané PDF *náhodnej* zhôd n-tic v dátach




(b) PDF objavených zhodných n-tic v dátach

Obr. 4.5: Porovnanie odhadu a pozorovania korelácií v dátach

bolo zistené rozloženie veľkostí skupín naprieč spektrom aktivity. To je zachytené na obrázku 4.5b, kde percentuálne hodnoty na ose y udávajú aký je pomer počtu zhôd rôznych veľkostí oproti celkovému počtu vektorov s danou aktivitou. Na základe počtu vektorov s jednotlivými aktivitami som podobným spôsobom aký demonštruje príklad z predošlej sekcie odhadol na základe binomickej vety očakávané rozloženie korelácií. Toto odhadované rozloženie zobrazuje graf na obrázku 4.5a. Po porovnaní nájdeného a odhadovaného rozloženia som usúdil nasledujúce zistenia.

- Dochádza ku korelácii nepravdepodobne veľkých skupín.
- Malé skupiny sú rádovo pravdepodobnejšie aj pri vyšších hodnotách aktivity.

Pozorované odchýlky ovplyvňujú nasledujúce faktory:

- Existujú platné veľké koordinované skupiny v dátach.
- Vyplnenie vektorového priestoru nie je rovnomerné, pretože *periodický* export udalostí detektorov zanáša do dát šum.

U malých skupín je vyššia pravdepodobnosť náhodnej korelácia, preto sa malých koordinovaných skupín s nízkou aktivitou vyskytuje veľa, navyše, šum tento efekt ešte znásobí. Predpoklad 4.1, o rozložení vektorov aktivít ostáva zachovaný napriek pozorovaniu. Považuje sa, že rozloženie vektorov odpovedá dianiu na sieti ako takému. Možným vysvetlením narušenia ostáva periodický export a skreslenie časových značiek. Problém s veľkým počtom náhodných korelácií môže čiastočne riešiť výber menšieho rozlišovacieho intervalu. Problémom však ostáva spomenutý šum. Akým spôsobom sa potvrdí korelácia skupiny bude navrhnuté v sekcii 4.5.

Ako dodatok uvádzam tabuľku so súhrnnými hodnotami pravdepodobností výskytu korelovanej skupiny (veľkosť n-tice do 20) vo vzorke. Podľa nej je už pri aktivite 4 veľmi nepravdepodobné, že by sa vôbec nejaká skupina vyskytla.

Pre účely nastavenia detektora sa však ďalej prihliada na možnosť náhodnej zhody tak ako je popisná v sekcií 4.3. Bernoulliho veta v kombinácií s informáciami z dát poslúži pre informovaný odhad o hodnotách minimálnej veľkosti skupiny a tiež minimálnej a maximálnej aktivity, ktorá bude pri korelácií uvažovaná. Toto poslúži pre minimalizáciu náhodných

| Activity | $\sum$ Random match prob. | $\sum$ Observed match prob. |
|---|---|---|
| 2 | 0.975159 | 0.904322 |
| 3 | 0.014088 | 0.097425 |
| 4 | 0.000029 | 0.046202 |
| 5 | 8.66e-08 | 0.041516 |
| 6 | 3.76e-10 | 0.042895 |

Tabuľka 4.5: Odhad a pozorovanie zhôd n-tíc v dátach

korelácií. Hodnoty sa budú voliť agresívnejšie, pretože prechod od analýzy podľa presných zhôd vektorov ku klastrovaniu ($T > 0$) situáciu s náhodnými koreláciami zhorší.

## 4.4 Klastrovanie časových vektorov

Metóda SECT vyhľadáva skupiny s podobným vzorom generovania bezpečnostných udalostí v čase. V jadre tejto metódy sa riešim problém zostavenia daných skupín. Ako definované, koreláciu dvojice je možno rozhodnúť jednoducho, na základe prahu $T$ a vzdialenosti susedným bodom. Takto možno identifikovať blízkych susedov každého vektoru. Definícia korelácie skupín entít formálne vyjadruje ako má vyzerať výsledok o krok ďalej.

Pre zostavenie skupín z matice vzájomných vzdialeností slúžia klastrovacie algoritmy. V prípade $T$=0 nemá zmysel uvažovať klasické klastrovacie algoritmy. Skupiny sa získajú vytvorením zoznamov entít, ktoré majú rovnaké vektory aktivít. Tento prístup bude spôsobovať, že sa nezistia všetci členovia koordinovanej skupiny ak dôjde k odchýlkam časových značiek na hraniciach intervalov. Význam zavedenia $T$>0 má riešiť práve tento jav. Vplyv nepresností sa bude zvyšovať so zmenšujúcou sa veľkosťou intervalu.

Variácia agregačného intervalu pri predspracovaní bude meniť rozloženie vzdialeností medzi entitami, čo bude hrať významnú rolu pri klastrovaní. Preto je potrebné nejakým spôsobom riadiť výsledný počet klastrov. Kritériom filtrácie nech je veľkosť klastrov a miera ich kvality, pričom žiadúce sú čo najväčšie a čo najkvalitnejšie klastre.

Klastrovacích algoritmov je veľa, avšak, nasadenie na dátach vstupujúcich do SECT vytvára požiadavky na základe kt. je možné zúžiť výber. Počet klastrov udávajú dáta a nedá sa pevne určiť. Klastrovacie metódy nesmú vyžadovať zadanie počtu klastrov. Ladenie tohoto parametru neprichádza do úvahy, pretože skupín môžu byť stovky, a to by nebolo únosné. Algoritmy by mali byť schopné oddeliť šum resp. outlierov.

Jedným zo spôsobov ako vypočítať, ktoré entity patria k sebe je Aglomeratívne klastrovanie. To na základe vzdialeností medzi vektormi, vyberie najbližších susedov a vytvorí skupiny. V každom ďalšom priechode zlučuje skupiny a body na základe vzdialeností do nadskupín. Spôsob výpočtu vzdialenosti bodu a skupiny definuje *linkage* spojovacia funkcia. Klastre sa získajú vhodným rezom spojovacieho stromu, a to buď na základe požadovaného počtu klastrov alebo na základe maximálnej zlučovacej vzdialenosti.

Keďže je možné riadiť počet skupín pomocou maximálnej medziklastrovej vzdialenosti, bude aglomeratívne klastrovanie jedným z uvážených algoritmov. Aglomeratívne klastrovanie však nedokáže izolovať šum. Šumom v klastri bude entita, ktorá síce spĺňa požiadavky na vzdialenosť, avšak danej skupine predstavuje outliera. Problém šumu lepšie zvládajú algoritmy založené na hustote klastrov, tzv. density based clustering, medzi ktoré patria napríklad HDBSCAN, DBSCAN a OPTICS. Metódy sú bližšie popísané v kapitole 3.2.

Intuitívne mal parameter $T$ v aglomeratívnom klastrovaní význam, že udával minimálnu vzdialenosť medzi dvojicou skupín, a zároveň aj maximálnu vzdialenosť dvojice entít v skupine. HDBSCAN vykazuje iné chovanie, intuitívne bude $T$ rôzne pre rôzne skupiny. Hodnota $T$ bude závisieť na hustote dát v okolí miesta, kde sa daná skupina nachádza.

Density based algoritmy zavádzajú vlastné parametre pre nastavenie. Z použitých sem patrí napríklad minimálna veľkosť klastra. Tento parameter sa popisuje ako *eps*, a riadi koľko blízkych susedov musí mať bod aby sa klaster ďalej rozširoval z daného bodu.

Vektory aktivít, ktoré nevyhovujú požiadavkám na minimálnu aktivitu podľa odhadu náhodnosti korelácii sa zahodia. Z výslednej sady sa spočíta matica vzájomných vzdialeností. Ďalšou možnou optimalizáciou je, že sa zahodia všetky vektory, ktoré nemajú korelovaných susedov.

Prečistená matica vstupuje do vybraného klastrovacieho algoritmu. Výsledné hodnoty *labels* sa použijú pre priradenie entít do skupín. Tabuľka vektorov aktivít sa rozšíri o príslušnosť do skupiny, ako je naznačené v diagrame na obrázku 4.2. Cez dvojicu hodnôt IP a *labels* sa v ďalších krokoch vygenerujú potrebné tabuľky. Metrika je jednou z voľných premenných a bude ovplyvňovať, aké klastre sa identifikujú mimo presných zhôd. Krok klastrovania zavádza do metódy ďalšie parametre:

- `method` Zvolená metóda klastrovania, jedna z 'match', 'agglomerative', 'dbscan', 'hdbscan'.
- `metrika` Zvolená funkcia vzdialenosti. Primárne jaccardovská metrika.

Priradenie parametrov pre jednotlivé klastrovacie metódy používa ešte nasledujúce parametre SECT `distance_threshold` a `min_cluster_size`. Presné priradenie popisuje implementácia.

### Filtrácia vektorov na základe príznakov a susednosti

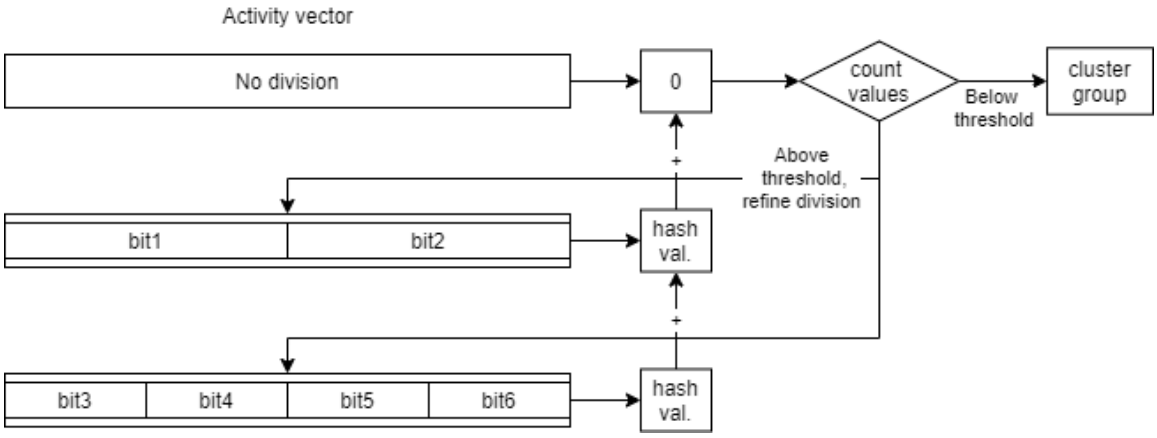
Klastrovacie algoritmy pracujú s maticou vzájomných vzdialeností. Ako zobrazuje tabuľka 4.3, z hodnotami z predbežnej analýzy, jeden deň bezpečnostných udalostí obsahuje cca 160000 entít, čo stanovuje požiadavky na operačnú pamäť okolo 100GB. To vynútilo zavedenie filtrovania entít podľa dôležitosti pre spracovanie s cieľom znížiť pamäťové nároky klastrovania.

### Rozdelenie klastrovania na nezávislé behy

V postupnom spracovaní dát je pred klastrovaním predradené filtrovanie podľa aktivity. Môže sa stať, že dátová tabuľka bude stále príliš dlhá a matica vzdialeností sa nevmestí do operačnej pamäte. Riešením je rozdeliť sadu na slabo závislé podsady. To sa docieli rozdelením dát *hash* funkciou podľa súhrnného obsahu vektorov aktivít.

Hash funkcia pridelí rovnaké číslo vektorom na základe súhrnnej aktivity v častiach vektorov. Vo vektore sa rekurzívne sumarizujú hodnoty. V každej ďalšej hĺbke zanorenia výpočtu sa zdvojnásobuje delenie a zistí prítomnosť aktivity získaných častiach. Výsledkom bude n-bitová hodnota, kde $n$ je počet súhrnných častí delenia a *True* označuje, že vektor bol aktívny v danej súhrnnej časti. Delia sa len skupiny, ktoré sú väčšie ako limit. Takto je možné rekurzívne deliť sadu vektorov, kým sa nedosiahne rozdelenie na samostatne spracovateľné celky. Tento proces je ilustrovaný diagramom na obrázku 4.6. Vektory s rovnakou hash sa spracujú v samostatných behoch klastrovania, čím sa dosiahne zníženie pamäťo-

vej náročnosti. Medzi behmi budú potencionálne existovať korelujúce vektory, no pokiaľ sa obmedzí počet delení, vplyv na veľké skupiny bude minimálny.




Obr. 4.6: Diagram rozdelenia sady pomocou sumarizácie aktivity

## 4.5 Overenie korelovaných skupín

Po tom, čo prebehne klastrovanie sú nájdené skupiny zaznamenané do tabuľky. Obohatia sa o dostupné informácie a výsledky sa zostavia do tabuliek podľa modelu na obrázku 4.2. Podľa definície 4.3 je treba minimalizovať vplyv náhodnej korelácie a šumu.

### Potlačenie náhodných korelácií

V súlade s poznatkami zo sekcie 4.3 o náhodných zhodách je možné odhadnúť pravdepodobnosť, že sa skupina vyskytla na základe náhody. Na potlačenie náhodných korelácií sa použije vhodne parametrizovaná binomická veta. Teória bola popísaná len pre skupiny s presnými zhodami, preto sa zvolil spôsob potlačenia náhodnosti pomocou filtrácie podľa modelu 4.3. Získaný odhad pravdepodobností sa použije pre výber intervalu aktivity (min a max) minimálnej veľkosti. Zo vstupných dát sa odčítajú nevyhovujúce vektory.

### Potvrdenie konzistencie klastrov

Podľa toho, ako sa nastaví klastrovanie môžu byť skupiny viac či menej konzistentné vzhľadom na aktivitu. Pri použití klastrovacieho algoritmu je potrebné sledovať nejakú mieru prekrytia vektorov. Je treba zohľadniť dva faktory a to: či sú získané klastre kvalitné, a či majú vektory v skupine dostatočne veľký prekryv.

Overenie kvality klastrov sa vykoná výpočtom vhodnej metriky nad dátami, ktoré vyhoveli filtrácii vstúpili do klastrovania. Z hľadiska klastrovacieho algoritmu je treba overiť kvalitu nájdených skupín. Metrík pre overenie kvality klastrovania je viacero. Použitie zužuje výber len na metriky, ktoré pracujú bez znalosti ozajstných priradení (ground truth), ideálne by tiež mali byť vhodne pre density based algoritmy a mali by zohľadňovať šum v dátach. Ako metrika bola vybraná shilouette, kvôli predošlým skúsenostiam s ňou[1, Sekcia 2.3.10],[34].

Pri vizuálnej inšpekcií klastrov som preferoval *skupiny s dobrým prekrytím vektorov*, tzn. jednotlivé vektory v skupine sú až na niektoré intervaly zhodné. Vizuálna inšpekcia sa

ale nedá efektívne vykonávať nad stovkami skupín, preto sa nad vektormi spočíta stredná odchýlka od priemernej aktivity v skupine.

$$inhomogenity = \sum_{x \in G} \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \overline{x})^2}$$

. V texte sa na túto metriku bude odkazovať ako na nehomogénnosť skupiny. Nehomogénnosť sa môže vyhodnocovať na rôznych typoch informácií prevedených do histogramov. U skupín s presnou zhodou a vysokou aktivitou intuitívne nebude pochýb o platnosti korelácie. Na vyššie uvedené metriky: kvalita klastrovania a nehomogennosť akvitivy, možno tiež nazerať ako na príznaky pre relatívne porovnanie analýz s rôznym nastavením voči sebe. Žiadúca je vysoká kvalita klastrov a nízka nehomegenita typov.

### Potlačenie vplyvu šumu na výsledky

Z nečasových informácií v dátach a z dát z 3. strán sa získajú histogramy tagov. Histogram bude numericky popisovať kombináciu tagov, ktorá bola priradená danej IP reputačnou databázou. Z toho sa získajú príznaky 'nehomogénnosť tagov' a 'odchýlka od priemeru' pre jednotlivé skupiny. Tieto slúžia ako indikátor toho, ako dobre sa zhodujú typové informácie vnútri klastra a ako veľmi sa klaster líši od priemerného rozloženia tagov v celej dátovej sade. Jej vysoká hodnota môže naznačovať, že sa jedná o zhodu na základe šumu. V tomto štádiu metóda naráža na problém detekcie anomálii. Je problematické aký vzťah vzhľadom na histogramy tagov by mali skupiny mať.

Ja žiadúce mať nízku nehomogénnosť tagov, a vysokú odchýlku od priemeru tagového histogramu v dátovej sade.

### Vyhodnotenie objavených skupín

V tomto štádiu analýzy je potrené skupiny popísať príznakmi. Korelácie samotné neposkytujú informácie o obsahu, preto sa k skupinám agregujú vyššie uvedené typové a tagové informácie. Zostavia sa tabuľky popisujúce skupiny z rôznych hľadísk podľa modelu na obrázku 4.2.

- Vlastnosti skupiny (zoznam entít, veľkosť skupiny, zoznam typov udalostí, zoznam zdrojov udalostí)
- Vzor aktivity skupiny
- Súhrnné príznaky zo vzoru aktivity (priemerná aktivita, priemerný počet blokov v skupine, priemerný počet udalostí, priemerná doba medzi začiatkami blokov)
- Metriky pre overenie klastrovania (shilouette, nekonzistencia)
- Metriky pre overenie skupín na základe zloženia príznakov (nekonzistencia, podobnosť voči celej sade)

  Idea typ

  Idea zdroj

  GreyNoise tagy

  Nerd tagy

- Ďalšie ad-hoc príznaky

Pre agregáciu informácií k skupinám sa použije priemer pre číselné hodnoty a zjednotenie pre zoznamy. Príznaky pre overenie skupín sa zostavia na základe normalizovaných histogramov zložení.

### Štatistické vyhodnotenie behu

Pri štatistickom vyhodnotení sa použije podsada príznakov a pre každý beh sa z nich vypočíta kovariančná matica. Získané absolútne hodnoty nadobudnú bližší význam až pri porovnávaní voči sebe. Stanoví sa základná hodnota kovariancie v dátovej sade nastavením analýzy na vyhľadávanie presných zhôd. Zistí sa rozdiel kovariancií. Ideálny výsledok potom bude: Analýza XY ma väčšie klastre, ktoré sú vnútorne homogénnejšie, silhouette metrika naznačuje, že prekrytie aktivít je trošku horšie ale skupiny sú homogénnejšie vzhľadom na Greynoise a Nerd tagy. Porovnajú sa ešte histogramy veľkosti skupín, a typové zloženie skupín.

### Ručné preskúmanie výsledkov

Ručné vyhodnotenie výsledkov sa urobí vizuálnou analýzou. Na základe ad-hoc kritérií sa identifikuje niekoľko zaujímavých a nezaujímavých skupín, nálezy sa diskutujú.

Vzor komunikácie skupiny bude charakterizovať vektor so súhrnnou aktivitou. Získa sa tak, že sa po stĺpcoch sčítajú vektory aktivít členských entít a výsledok sa normalizuje počtom entít v skupine. Pre účely ručnej inšpekcie skupín je pripravený algoritmus pre značkovanie skupín podľa dát z 3. strán. Napríklad skupina sa označí tagom Attempt. Login, ak sa v nej v dostatočnej miere vyskytuje daný typ udalostí. To umožní organizovať skupiny podľa primárnych typov. Tabuľka skupín a príznakov umožní poloautomaticky hlbšie skúmať koordinované skupiny.

### Dlhodobo aktívne skupiny

Jednou zo zaujímavých informácií o skupinách je miera ich perzistencie. Ak sa vo výsledkoch nájde 80 skupín koľko z nich sa objaví na ďalší deň? Bude vykonaný experiment, kde sa analyzuje viacero jednodenných výsledkov, vývoj skupín a ich typové zloženia a príznaky. Bude sa skúmať rozdielnosť opakujúcich sa skupín oproti zbytku skupín.

# Kapitola 5

# Implementácia metódy SECT pomocou zhlukovania vektorov aktivít

Riešenie bolo implementované použitím kombinácie technológií. Ako implementačný jazyk bol zvolený python verzie 3.6. Pre manipuláciu s dátami sa používa knižnica pandas v kombinácii s numpy. Tvorba grafov je riešená pomocou matplotlib a seaborn. Pre spracovanie dát sa použili knižnice scikit-learn, HDBSCAN. Pri prvotnom prieskume a niektorých vizualizáciách sa použila moderná redukcia dimenzionality umap, moderný nástupca t-SNE. Ad-hoc dotazovanie a finálna analýza výsledkov prebiehala v prostredí jupyter na vzdialenom, výkonnejšom stroji.

S dátami sa pracovalo prevažne pomocou základných pandas objektov DataFrame a Series. Často sa používali metódy loc/iloc umožňujúce, meniť či vyberať podmnožinu riadkov alebo stĺpcov tabuľky na základe indexu alebo kľúčov.

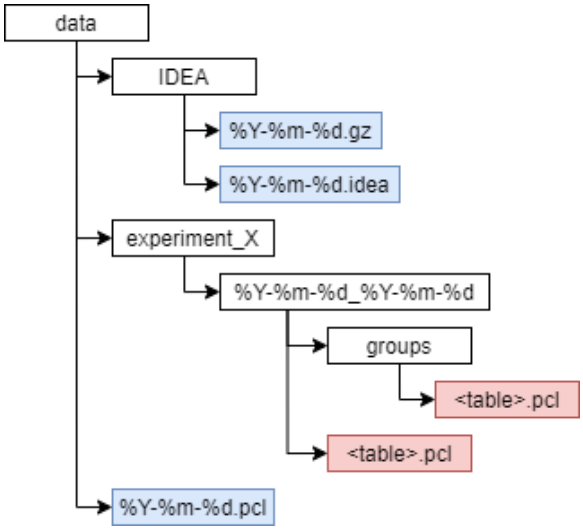
```
df.loc[rowIndex,colNames]
```

Implementácia je rozdelená na dve časti. Jednou sú hlavne skripty na spracovanie dát, klastrovanie a podporné metódy. Druhou častou sú jupyter notebooky.

Keďže klastrovanie je náročne na pamäť, bola práca presunutá na vzdialený stroj. Na vzdialenom stroji bolo vytvorené potrebné prostredie a spustený Jupyter-lab server. Spúšťanie analýz sa robilo cez Jupyter notebooky, pretože to bol lepší spôsob ako vzdialená konzola. To si však vyžadovalo ukladať výsledky po analýze na disk a v notebooku ich načítať.

Vstupné dáta v podobe IDEA a predspracovaných suborov a výsledky sú na disku organizované to hierarchickej štruktúry. Vstupné dáta a priečinky s výsledkami, sú pomenované podľa rozsahov dátumov `dFrom_dTo` vrátane posledného dňa. Meno experimentu obyčajne udáva meno klastrovacej metódy použitej v analýze. Štruktúra priečinkov je na obrázku 5.1.

Nasledujúce sekcie obsahujú časti implementácie zjednodušené do pseudokódu.




Obr. 5.1: Štruktúra uloženia výsledkov a vstupných dát

## 5.1 Predspracovanie udalostí

Dáta v surovej podobe IDEA boli predspracované do efektívnejšej formy. Prvotné dáta sú vo forme jednodenných blokov v textových súborov s radkami vo formáte JSON. Predspracovanie sa deje v dvoch krokoch. Najprv IDEA do tabuľky udalostí, potom agregácia do vektorov aktivít.

### Selekcia polí a konverzia do tabuľky udalostí

IDEA súbor sa prechádza riadok po riadku, čo odpovedá udalostiam. Riadok sa interpretuje ako objekt JSON a extrahujú sa požadované polia. Ako rýchly spôsob konverzie sa ukázal byť prepis polí IDEA do CSV formátu v pamäti a následný import do DataFrame objektu.

```
import datetime
import json
import sys
if sys.version_info[0] < 3:
    from StringIO import StringIO
else:
    from io import StringIO

#IDEA pseudokod predspracovania
def preprocess(file_path):

    # priprava hlavicky CSV
    csv_str = "ip,timestamp,origin,type\n

    with open(file_path, 'r') as data:
        line = data.readline()
        while line:
```

```
            x = json.loads(line)
            try:
                origin = str(x.get('Node', ['None']))
                evt_type = str(x.get('Category', ['None']))
                ip = extract_ip(x) # vyber IPv4 alebo IPv6

                # vybera prmarne EventTime inak DetectTime
                # nacita cas v roznych formatoch a ulozi ako timestamp
                timestamp = extract_time(x)

                # budovanie CSV retazca
                csv_str += f"{ip},{timestamp},{origin},{evt_type}\n"
            except ValueError as err:
                print(err, end=', ')
                print('while processing line {}'.format(line_num))

            line = data.readline()

    # je to rychlejsie ako postupne budovanie DataFrame
    # DataFrame rozpozna typy stirng, int, string, string
    res = pd.read_csv(StringIO(csv_str))
    return res
```

Výsledok sa uloží na disk v komprimovanom formáte pre ďalšie použitie. Meno súboru je dátum vo formáte '%Y-%m-%d.pcl'.

### Prevod udalostí na vektory aktivít

Transform metóda konvertuje tabuľku udalostí na tabuľku vektorov aktivít pre jednotlivé entity, zároveň s tým sa vytvorí tabuľka príznakov (vynechané). Vektory konštantnej dĺžky sú nutné pre spracovanie v klastrovacom algoritme. Parametre, ktoré riadia dĺžku vektorov a ich časové rozlíšenie sú agregačné okno `aggregation` a časový rozsah analýzy `dfrom`, `dto`.

```
def transform(self, df, tfrom, tto):

    dfc = df.copy() # nemen vstupn dta

    # konvertuj cas do slotu a pridaj novy stpec do dfc
    dfc['slot'] = np.floor((df.timestamp - tfrom) / self.aggregation)
    dfc = dfc.loc[(dfc['slot'] >= tfrom) & (dfc['slot'] < tto),:]
    # vytvor zoznam slotov pre kazdu ip a zisti pocet udalsoti
    feature = dfc['slot'].groupby(dfc['ip']).agg([list, 'count'])

    # uloz dlzku vektora
    self.vect_len = np.int((tto-tfrom)/self.aggregation + 1)

    # generuj one hot zakodovane priznaky a agreguj do histogramov
    # tabulka typov per ipf
```

```
    type = pd.get_dummies(dfc['type'])
    type = type.groupby(dfc['ip']).agg(sum)
    # tabulka zdrojov per ip
    origin = pd.get_dummies(dfc['origin'])
    origin = origin.groupby(dfc['ip']).agg(sum)

    # ziskaj vektory aktivit
    feature['series'] = feature['list'].apply(get_bin_series,
    args=[self.vect_len])

    # spocitaj priznaky do tabulky feature
    feature['activity'] = feature['series'].apply(sum)
    feature['blocks'] = feature['series'].apply(count_blocks)

    # prevod vektorov do tabulky vektorov aktivit,
    # ma rovnaky index ako feature!
    act_vect = pd.DataFrame(data=np.stack(feature['series'],
        dtype=np.bool), index=feature.index)

    del feature['series']
    del feature['list']

    # vrat jednotlive tabulky
    return act_vect, feature, type, origin
```

DataFrame.apply metóda aplikuje funkciu po riadkoch na tabuľku. Vektor aktivity sa získava zo zoznamu slotov. Na to slúži funkcia `get_bin_series(...)`.

```
#pseudokod apply
def apply(self, func):
    col = pd.Series(index=self.index)
    for val in self.index:
        col[val] = func(self.loc[val,:])
    return col

#zostav vekt aktivity inkrementaciou na indexoch
def get_bin_series(slot_list, length):
    # priprav vektor
    vector = np.zeros(length)
    # inkrementuj na indexoch
    np.add.at(vector, np.array(slot_list).astype(np.int), 1)
    # vrati numpy bool vektor
    return vector > 0
```

## 5.2 Jadro výpočtu

Metódy potrebné pre transformáciu tabuľky udalostí do vektorov a následnú klasifikáciu sú zabalené v spoločnom objekte. Pri konštrukcií objektu sa nastavia parametre analýzy, ktoré

sa propagujú do jednotlivých metód. Objekt neslúži pre predávanie tabuliek s výsledkami medzi krokmi.

```
import pandas as pd
import numpy as np
...

class TemporalClusterer:
    def __init__(self,
        min_activity=0.05, max_activity=0.8, # filter podla priznakov
        aggregation=900, # pocet sekund v jednom intervale vektorov
        min_cluster_size=2, # minimalna velkost klastrov
        dist_threshold=0.05, # max vzidalenost suseda pri klastrovani
        metric='jaccard', method='hdbscan', # nastavenie metriky a metody
        prune_distmat=False, # filtovanie matice podla blizkych susedov
        batch_limit=40000): # limit pre delenie vypoctu
        ...
        #return self instance

    # rozdelenie behu na podproblemy
    def series_divide(self, division, series, limit=10000):
        ...
        return division

    # transformacia udalosti na vektory
    # staci raz ak sa aggregation nemen
    def transform(self, df, tfrom, tto):
        ...
        return act_vect, feature, type, origin

    # predikcia skupin, spustenie klastrovania
    def fit_predict(self, act_vect, feature):
        ...
        return labels

    # agregacia informacii do vyslednych tabuliek
    def postprocess(self, labels, act_vect, feature, query_nerd, third_party_path):
        ...
        return clusters, series, stats, dfnerd
```

### Filter a klastrovanie

Na vektory sa aplikuje filtrácia podla príznakov (tabuľka feature). Sada sa rozdelí, ak je príliš veľká (parameter `batch_limit`[1]) a spočíta sa matica vzdialeností. Ak je nastavený parameter `prune`, matica sa prefiltruje podľa parametrov `min_cluster_size` a `dist_threshold`.

[1] Určí sa podľa množstva dostupnej pamäte, 40000 vektorov zaberie 11.92 GB

Výsledná matica vstupuje do vybraného klastrovcieho algoritmu. Neklastrované vektory sú označené ako šum ($labels = -1$).

```
def fit_predict(self, act_vect, feature):
    # init
    labels = pd.Series(data=-1, index=act_vect.index)
    division = pd.Series(data=0, index=act_vect.index)
    labels_ofs = 0

    # filtrovanie podla priznakov
    act_vect = act_vect.loc[
        ((feature.activity >= self.min_activity) &
        (feature.activity < self.max_activity)), :]

    # delenie sady vektorov ak presahuju limit
    if len(act_vect) > self.limit:
        division = self.series_divide(division, act_vect, self.limit)

    # zisti velkost skupin
    z = division.value_counts()

    # pre kazdy skupinu
    for group in z.index:
        # vyber podskupinu
        vect = act_vect.loc[division == group]
        if len(vect) > 1:
            # ziskaj maticu vzdialenosti
            pairwise = pd.DataFrame(
                squareform(pdist(vect, self.metric)),
                index=vect.index, columns=vect.index)

            # filtrovanie - ip bez susedov sa zahodia
            # o blizkosti rozhoduje dist threshold
            # obycajne je chceny aspon jeden sused
            if self.dist_threshold is not None and self.prune:
                matches = pairwise.apply(
                    lambda x: np.sum(x <= self.dist_threshold)) \
                    .where(lambda x: x >= self.min_cluster_size) \
                    .dropna()
                # vdaka matici v dataframe je jednoduchy filter
                pairwise = pairwise.loc[matches.index, matches.index]

            # spusti klastrovanie
            clustering = Clustering(metric='precomputed', ...)
            labels_part = clustering.fit_predict(pairwise)

            labels_part = pd.Series(labels_part + labels_ofs, index=pairwise.index)
            # oznac podsadu
```

```
            labels_ofs = labels_part.max() + 1
            labels.loc[labels_part.index] = labels_part
    return labels
```

Ak je sada vektorov väčšia ako limit, vykoná sa delenie problému podľa hash funkcie. Proces delenia lepšie zobrazuje diagram na obrázku 4.6. Vektory sa postupne označujú do skupín podľa toho, v ktorých regiónoch sú aktívne. Delenie sa rekurzívne opakuje nutných počet krát. Snahou je minimalizovať počet skupín. V prípade, že sa skupina nerozdelí dostatočne, zo sady sa vyberú entity s vyšším počtom eventov (zoradí a vezme sa top n riadkov).

```
def series_divide(self, division, series, limit):
    division.loc[:] = 0

    # postup od najvacsieho kroku nasobku 2
    for x in range(np.ceil(np.log2(len(series.columns))).astype(np.int), 2, -1):

        # spocitaj skupiny, zoradenie podla velkost
        z = division.value_counts()
        stopper = True
        # postupne vyberaj bity do kt. sa sumarne hodnoty ulozia
        # a vytvorili tak jedinecne cislo skupiny
        idx = 0

        for w in z.index:
            # prejdi len skupiny vacsie ako maximum
            if z[w] > limit:
                # zjemni delenie
                work_set = series.loc[division == w]
                ws_div = division.loc[division == w]

                # sumarizuj po blokoch a zakoduj do cisla
                for y in range(0, len(series.columns), 2 ** x):
                    # sumarizuj cast aktivity a bit po bite uloz
                    ws_div += ((work_set.iloc[:, y:y + 2 ** x]\
                        .apply(max, axis=1)) * (2 ** idx))
                    idx += 1
                # uloz podrozdelenie
                division.loc[ws_div.index] = ws_div
                stopper = False
        if stopper:
            break
    return division
```

### Mapovanie parametrov klastrovacích algoritmov

V kapitole s návrhom sú postupne popísané parametre metódy SECT. Časť z nich sa priamo používa pre riadenie klastrovacích metód. Priradenie parametrov SECT na parametre metódy klastrovania popisujú nasledujúce riadky.

Hdbscan je najviac automatická metóda, je ideálne použiť ju s predfiltrovaním matice vzdialeností, pretože takto sa dobre kontroluje hustota klastrov. Hodí sa pre použitie s jaccardovou metrikou.

```
# Clustering = HDBSCAN
labels_part = HDBSCAN(
    # riadi velkost klastrov
    min_cluster_size=self.min_cluster_size,
    metric='precomputed',
    ).fit_predict(pairwise)
```

Dbscan je menej automatický typ density based zhlukovania ako hdbscan. Skôr by sa nemalo používať predfiltrovanie matice, algoritmus to ošetruje sám. Je potrebné zadať `distance_threshold`, inak sa použije 0.05

```
# Clustering = DBSCAN
eps = self.dist_threshold
if self.dist_threshold is None:
    eps = 0.05
labels_part = DBSCAN(
    # max vzdialenost k susedom
    eps=eps,
    # min mnozstvo susedov aby sa bod stal zdrojom sirenia
    min_samples=self.min_cluster_size,
    metric='precomputed'
).fit_predict(pairwise)
```

Aglomeratívny klastering dáva najviac konzervatívne výsledky. Kvalita klastrov sa vždy dodrží podľa `distance_threshold` hodnoty. Výsledok obsahuje všetky skupiny aj osamotené body, preto je použitý filter klastrov menších ako bolo parametrom zadané.

```
# Clustering = AgglomerativeClustering
AC = AgglomerativeClustering(
    # metrika
    affinity='precomputed',
    # spojovanie max vzdialenostou dvojice v zhlukoch
    linkage='complete',
    # hranica nad kt. uz k spojeniu nedojde
    distance_threshold=self.dist_threshold,
    # nezadava sa, bola definovana vzdialenost
    n_clusters=None
)
labels_part = AC.fit_predict(pairwise)
labels_part = pd.Series(labels_part + labels_ofs, index=pairwise.index)
# AC nepodporuje outlierov, skupiny mensie ako min sa oznacia ako sum.
labels_part = labels_part.loc[
```

```
labels_part.map(labels_part.value_counts()) >= self.min_cluster_size]
```

Získané výsledky po transformácií v postprocesingu uložia do priečinkovej štruktúry podľa diagramu na obrázku 5.1.

# Kapitola 6

# Vyhodnotenie

Dátová sada obsahuje bezpečnostné udalosti získané z platformy pre zdieľanie Warden. Z dostupných dát bol vybraný týždeň, rozsah 2020-04-16 až 2020-04-22 vrátane. Široké spektrum detekčných mechanizmov nasadených naprieč niekoľkými organizáciami prispieva do Warden-u. Udalosti sú popísané moderným formátom IDEA[21]. Príklad správy je v prílohe A. Entity (IP) v priloženej dátovej sade sú anonymizované. Sada pozostáva z jednodenných záchytov správ a pomocných súborov s reputačnými informáciami o IP z 3. strán (GreyNoise, Nerd). Približne 12 miliónov udalostí bolo zozbieraných z 34 IDS, honeypot systémov a ďalších detektorov. Väčšina detektorov je založená na analýze flow. Pomocné súbory obsahujú približne 2.7 milióna IP, sada sa čiastočne prekrýva s entitami z udalostí. Obsahuje IP označené tagmi v zoznamoch a širokom formáte, každý zdroj je vo vlastnej tabuľke.

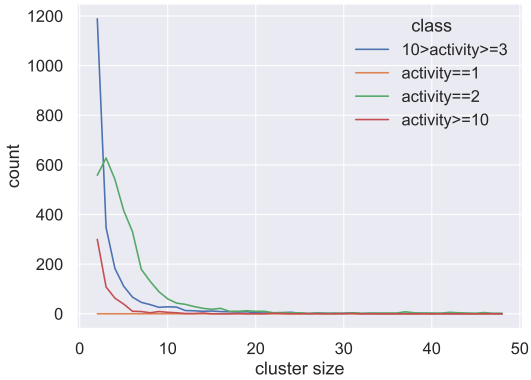
Nastavenie SECT metódy sa medzi experimentami v nasledujúcich sekciách môže líšiť, avšak, ak nie je spomenuté inak používa sa nastavenie na metódu hdbscan s jaccardovou metrikou, agregačný interval 900 (15 minút), prah aktivity $[10, 86]$ a vzdialenosť $T = 0.1$ so zapnutým filtrom vstupu (parameter pruning). Teda presný zoznam nastavení je: `method`=hdbscan, `metric`=jaccard, `aggregation=900`, `min_activity=10/96`, `max_activity=86/96`, `distance_threshold` $= 0.1$, `min_cluster_size` $= 2$ a `pruning`=`True`.

## 6.1 Vlastnosti skupín pri presnej zhode

Bola vykonaná analýza sady, s nastavením korelačnej metódy na presné zhody (match). Filtrácia podľa aktivity bola nastavená na čo najpermisívnejšiu, teda v intervale $[1, 95]$, aby sa ukázala charakteristika celej sady. Analyzované dáta sú z dňa 2020-04-17. Zámerom bolo preskúmať výskyt skupín a pravdepodobnosti náhodných zhôd s nimi spojené podľa teórie 4.3. Výsledky experimentu na grafoch 6.1 zobrazujú histogramy. Počty sú rozdelené podľa aktivity v klastroch, pozrite legendu. Na obrázku 6.1a je vidieť, ako pravdepodobnosť podľa modelu na základe binomickej vety závisí na aktivite. V koreláciách skupín s aktivitou 1 sa vyskytla anomália, binomický model predikuje najvyššiu pravdepodobnosť výskytu skupín o veľkostiach cca $580 - 610$ s pravdepodobnosťou okolo 30%, pričom sa vyskytlo niekoľko skupín s veľkosťou menej ako 200 a viac ako 800 (ďaleko od odhadu najpravdepodobnejšej hodnoty). V dátach je veľké množstvo skupín s presnými zhodami, veľká časť z nich je pri nízkych hodnotách aktivity, to ukazuje obrázok 6.1b, kde farebne odlíšené krivky popisujú počty detekovaných skupín s aktivitou v rôznych rozsahoch (class).

Z celkového počtu 141000 IP nemalo žiadnu zhodu 51000 (cca 35%), zbytok sa vyskytol v 6100 skupinách s rôznymi veľkosťami. Bližšie štatistiky sú v tabuľke 6.1, ktorá popisuje


(a) Histogram pravdepodobnosti náhodných korelácií




(b) Histogram veľkosti klastrov

Obr. 6.1: Dekompozícia parametrov nájdených zhodných skupín v sade

| | size | events | activity | blocks | probability |
|---|---|---|---|---|---|
| deviation | 78.83429 | 11.397109 | 5.694532 | 2.569528 | 6.766685e-02 |
| minimum | 2.00000 | 1.004894 | 1.000000 | 0.000000 | 0.000000e+00 |
| quantlie 25% | 2.00000 | 2.000000 | 2.000000 | 2.000000 | 2.834084e-07 |
| quantile 50% | 3.00000 | 2.111111 | 2.000000 | 2.000000 | 4.460950e-02 |
| quantile 75% | 6.00000 | 4.500000 | 4.000000 | 2.000000 | 1.036322e-01 |
| maximum | 973.00000 | 304.000000 | 88.000000 | 31.000000 | 1.864003e-01 |

Tabuľka 6.1: Štatistický popis zhodných skupín

sadu príznakov skupín. Size je veľkosť skupiny, events je priemerný počet udalostí v skupine, activity je počet aktívnych bitov vo vektore, blocks je počet súvislých blokov True vo vektore a probability je odhad pravdepodobnosti.

## 6.2 Porovnanie klastrovacích metód

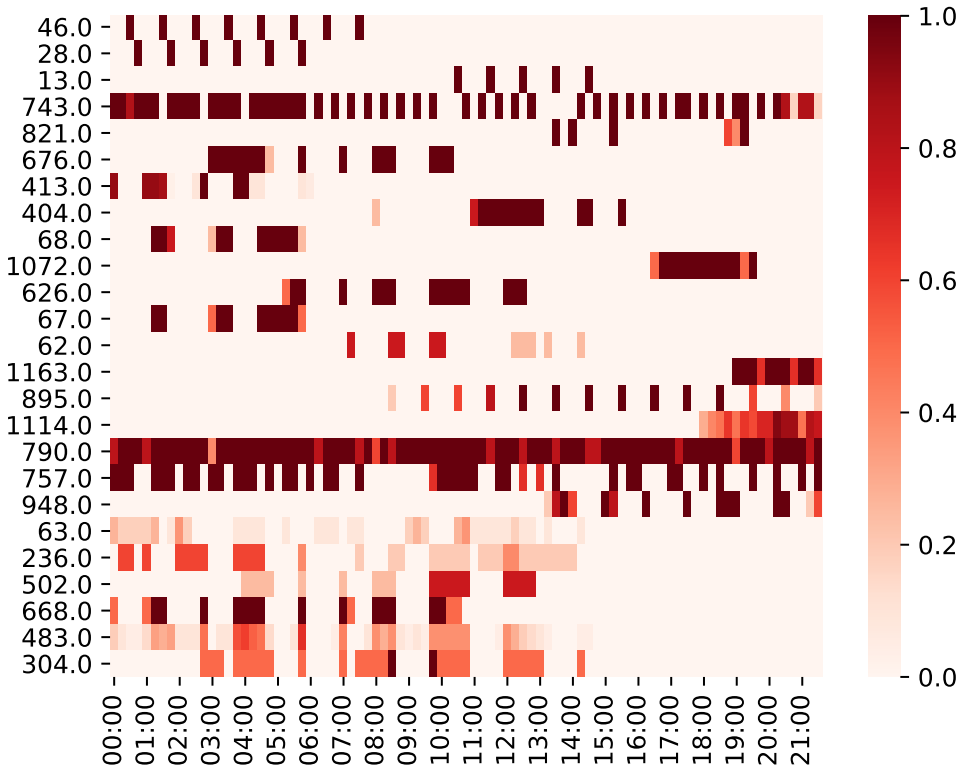
SECT metóda umožňuje výber klastrovacieho algoritmu. Aby sa zistilo, ako vyplýva jeho výber na počet a kvalitu nájdených skupín, bola vykonaná analýza jedného dňa a porovnali sa štatistiky výsledkov. Výsledky môžu byť ťažko čitateľné, preto bola sada analyzovaná tiež vyhľadaním presných zhôd. Výsledky presných zhôd udajú štatistickú tabuľku, ku ktorej je možné vzťahovať ostatné výsledky. Nastavenie je všeobecne spoločné, filtrácia podľa aktivity bola nastavená aby potlačila náhodné korelácie ale zároveň poskytla viac výsledkov do štatistík, je teda v intervale $[5, 90]$. Filtrácia susedov podľa $T$ je zapnutá len pre metódu hdbscan. Analyzované dáta sú z dňa 2020-04-17. Zámerom bolo získať podklady pre informovaný vyber metódy v budúcom použití.

| | count | mean | std |
|---|---|---|---|
| match size | 1398.0 | 3.908441 | 5.679225 |
| match silhouette | | 1.000000 | 0.000000 |
| agglomerative size | 1835.0 | 3.870845 | 7.061129 |
| agglomerative silhouette | | 0.775906 | 0.291004 |
| dbscan size | 1175.0 | 29.021277 | 291.836169 |
| dbscan silhouette | | 0.868645 | 0.275660 |
| hdbscan size | 1203.0 | 5.917706 | 10.309634 |
| hdbscan silhouette | | 0.444215 | 0.661961 |

Tabuľka 6.2: Štatistika nájdených skupín

Metódy priradili do skupín jednotlivo: 'agglomerativne' 7103 IP, 'dbscan' 34100 IP, číslo je vyššie pretože sa vyskytlo niekoľko riedkych skupín s veľkosťou $> 500$, 'hdbscan' 7119 IP. Metóda match v dátach odhalila 5464 IP s presnými zhodami. Všetky metódy dobre detekujú presné zhody. Lepší prehľad o kombinácií veľkosti skupiny a kvality poskytuje graf na obrázku v prílohe B.1. Metóda dbscan funguje horšie, pretože hustota rozloženia vzdialeností pri použití jaccardovej metriky je silno závislá na aktivite. Vektory s vyššou aktivitou budú mať v svojej blízkosti väčší počet susedov s malým počtom rozdielov a ten má v percentuálnom vyjadrení veľmi malú hodnotu, preto zadanému $T$ vyhovie viac vektorov. Hdbscan sa s týmto efektom vysporiadava lepšie, pretože variabilnú hustotu podporuje. Dôvod použitia filtrácie podľa susedností je, že sa takto vynúti horná hranica hustoty skupín. Algoritmus síce môže a aj detekuje niekoľko skupín s veľmi nízkou hustotou, vizte vzorku so súhrnnými aktivitami klastrov na obrázku 6.2.

V prípade, ak by začal byť problém variáciou hustoty kvôli jaccardovej metrike neúnosný, napríklad pri použití kratšieho agregačného intervalu je možné použiť inú metriku. Pri skúmaní nastavenia SECT metódy na novej sade alebo s použitím dlhších časových intervalov s inou agregáciou je dobré najprv vykonať detekciu na presné zhody a preskúmať rozloženia. Pokiaľ je žiadúce vynútiť veľmi dobré prekrytie aktivít v skupinách, aglomeratívne klastrovanie vyhľadá globulárne klastre (complete linkage). Pre zisk menej striktných skupín sa viac hodia density based metódy. V prípade, že metrika normalizuje hodnoty je lepšie použiť hdbscan.



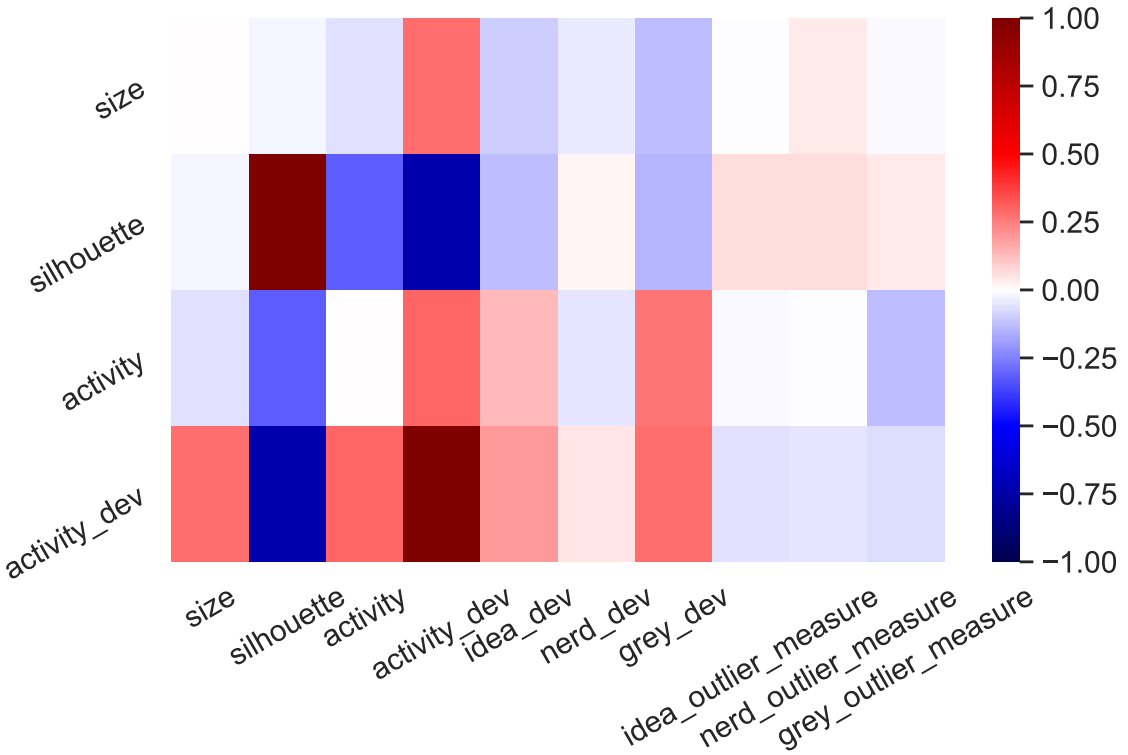

Obr. 6.2: Vzorka aktivít skupín zoradená podľa silhouette metriky

## 6.3 Overenie skupín identifikovaných metódou hdbscan

Overenie správnosti skupín bolo vykonané štatisticky, porovnaním príznakov pripojených k skupinám. Extra príznaky sú z dvoch typov, jedny popisujú mieru konzistencie informácií vnútri skupín, druhé popisujú mieru odlišnosť od pozadia. Príznaky formátu _dev udávajú sumu odchýlky histogramu od centroidu skupiny v danom type informácie (ak je token 'idea_dev' potom histogram je vektor typov z tabuľky udalostí). Príznaky formátu _outlier_measure udávajú priemernú vzdialenosť histogramu od základného histogramu. Základ udáva sadu všetkých, IP ktoré sa vyskytli v tabuľke vektorov. Vzdialenosť sa počíta kosínovou podobnosťou, pretože jej aplikácia leží v identifikácií tematickej podobností textov.

Filtrácia vektorov podľa aktivity zabezpečí, že sa v dátach nebudú vyskytovať náhodné korelácie. Tento experiment predpokladá, že výsledky získané pomocou metódy match udávajú základné hodnoty štatistík. Výsledok sa použije ako 'ground truth' pre porovnanie ostatných metód, respektíve hdbscan-u v tomto experimente.

Na obrázku 6.3, kde je rozdielová kovariancia výsledkov metód je vidieť zaujímavé miesta. Boli vybrané zaujímavé polia, vybrali sa kovariancie príznakov získaných z vektorov aktivít verzus typové príznaky. Mapu treba čítať opatrne, jedná sa o relatívne kovariancie, teda udáva, či je silnejšia alebo slabšia ako pri match metóde. Ideálne by mali byť _dev stĺpce neutrálne, opak znamená zvyšujúcu sa nekonzistentnosť. Stĺpce _outlier_measure




Obr. 6.3: Relatívna kovariancia príznakového vektora. (hdbscan-match)

by mali ostať neutrálne, prípadne sa zvýšiť. Vyššia odchýlka od základných typov znamená vzácnejšie kombinácie, čo sa dá čakať pri použití benevolentnejšieho klastrovania.

Mapa naznačuje, že boli identifikované skupiny s vyššou aktivitou, ktorých typové zloženie sa podobá na pozadie (viac skupín s bežnými typmi). Naproti tomu, skupiny s vysokou shilouette metrikou mávajú menej obvyklé typy a sú aj typovo konzistentnejšie. Kovarianca Activity proti nerd_dev naznačuje, že existujú aj väčšie skupiny, ktoré sú typovo konzistentné.

## 6.4 Vývoj typového zloženia skupín v týždni

Analýza skúma vývoj odhalených koordinovaných skupín v týždni po dňoch. Pretože v praxi dáta pribúdajú postupne blokoch, bola vykonaná postupná analýza dát po jednodenných intervaloch. Detektor bol nastavený na metódu hdbscan v jaccardovou metrikou, agregačný interval 15 min, prah aktivity $[10, 86]$ a vzdialenosť $T = 0.1$ sa použila pre zahodenie nezaujímavých vektorov. Veľkosť skupiny nebola obmedzená. Výsledná séria výsledkov umožnila sledovať ako sa skupiny vyvíjajú v týždni. *Podobnostné skupiny* sa identifikovali klastrovaciou analýzou na základe členov (IP) skupín v jednotlivých dňoch. Výpočet vzdialeností skupín sa získal jaccardovou metrikou. Kritériom pre priradenie do rovnakej podobnej skupiny je zhoda v aspoň 50% IP. V popise experimentu sa na klastre skupín naprieč analýzami odkazuje ako na *podobnostné skupiny*. Zdrojom označení (tagov) boli reputačné databázy NERD a GreyNoise. Agregácia tagov k skupinám prebehla tak aby sa nevýznamne tagy v skupine zahodili. To sa vykonalo prahovaním normalizovaného rozloženia s úrovňou 0.25, čo približne znamená. Agregácia tagov do podobnostných skupín sa vykonala zjednotením zoznamov.

Nasledujúci experiment v prvej časti analyzoval vývoj 15 najväčších podobnostných skupín. Sledoval sa vzor opakovania a typové zloženie. V druhej časti sa analyzujú všetky odhalené skupiny, sleduje sa štatistická odlišnosť skupín tvoriacich podobnostné skupiny oproti ostatným s cieľom zistiť, či v príznakoch skupín existuje prediktor opakovania skupiny.

### Vývoj podobnostných skupín

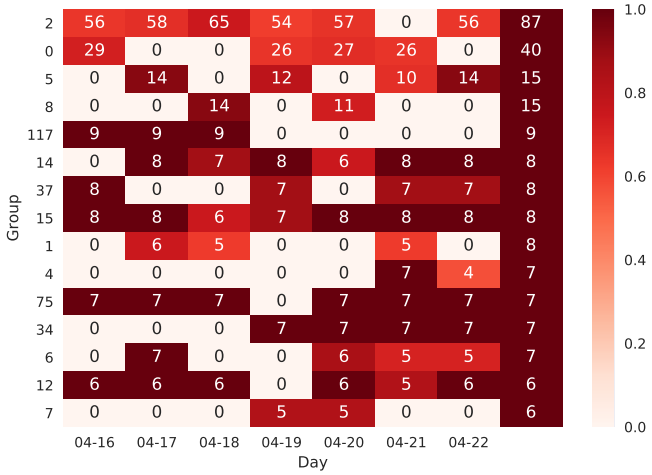
Obrázok 6.4 zobrazuje vývoj veľkostí skupín. Riadky označujú podobnostné skupiny, intenzita farby bunky zobrazuje relatívnu veľkosť skupiny voči maximu (počet pozorovaných IP v poslednom stĺpci) v priradenej podobnostnej skupine. Odpovedajúce typové zloženia zobrazuje tabuľka 6.3, zoradenie je rovnaké, NaN reprezentuje absenciu výskytu skupiny. Je vidieť, že väčšie podobnostné skupiny v priebehu času menia svojich členov avšak zloženie tagov sa u väčšiny nemení (podobnostné skupiny 6 a 7 sú výnimkou).

V 15 najväčších podobnostných skupinách sa vyskytlo spolu 238 IP. GreyNoise označila 25 IP ako škodlivé, 4 IP patrili výskumnej službe ShadowServer (podobnostná skupina 1), zbytok IP nebol klasifikovaný[1]. Podobnostné skupiny identifikované ako škodlivé sú:

- *pod. skupina* Primárne škodlivé tagy (výskyt z počtu)[2]
- *8* - Mirai (11 z 15)

[1] GreyNoise delí IP na rozpoznaný šum a neznáme, šum sa ďalej klasifikuje ako škodlivý, neznámy a benígny.

[2] počet označených ako škodlivé z celkového počtu




Obr. 6.4: Vývoj podobnostných skupín (15 najväčších)

- *6* - Eternablue (7 zo 7) skeny na porte 445
- *7* - SSH Bruteforcer (6 zo 6)

Ostatné podobnostné skupiny sú prevažne skeny rôznych typov. Všetky podobnostné skupiny mali takmer perfektné prekrytie vektoru aktivity. Rovnaké hodnoty tagov naprieč členmi skupín sú považované za príznak potvrdzujúci koordináciu, to isté platí pre počet dní výskytu podobnostnej skupiny.

### Rozdielnosť opakujúcich sa podobnostných skupín

Pri ručnej inšpekcii príznakov (features) podobnostných skupín boli zistené nasledujúce vlastnosti členských skupín:

- *Kompozícia IP* - Malé opakujúce sa skupiny (do 10 IP) bývajú často z jednej siete s prefixom /24
- *Špecifické vzory komunikácie* - Komunikácia je rozprestretá v čase, súvislé bloky aktivity sú výnimkou
- *Kvalita klastra* - Aktivita v rámci skupín má takmer perfektné prekrytie.

Bolo zistené, že zo všetkých 39342 IP priradených do 6913 podobnostných skupín v jednotlivých dňoch sa zopakovalo 324 IP v 36 podobnostných skupinách. To znamená, že v týždňovej sade boli detekované iba 0.82% IP s dostatočne stabilným vzorom naprieč aspoň dvoma dňami.

Vzorka 983 jedinečných IP (200 pod. skupín) z celej sady sa analyzovala dotazom do GreyNoise databázy, identifikovalo sa 794 IP 80%. Z týchto boli klasifikované 567 IP (71%) škodlivých, 211 IP (27%) neznáme a 16 (2%) benígne (Shadowserver, Project Sonar, Net System Research). Prevažne sa jednalo o stroje s OS Windows 7/8 (80%), zbytok boli rôzne verzie linuxu. IP boli označené rôzne ako SMB skeny, Eternalblue, SSH skeny, SSH bruteforce.

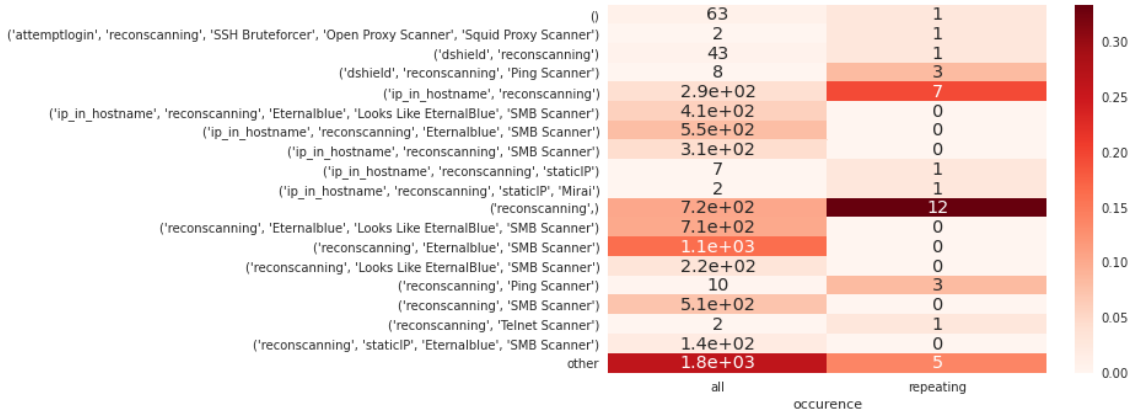
| group | 2020-04-16 | 2020-04-17 | 2020-04-18 | 2020-04-19 | 2020-04-20 | 2020-04-21 | 2020-04-22 |
|---|---|---|---|---|---|---|---|
| 2 | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] | NaN | [reconscanning] |
| 0 | [reconscanning] | NaN | NaN | [reconscanning] | [reconscanning] | [reconscanning] | NaN |
| 5 | NaN | [ip_in_hostname, reconscanning] | NaN | [ip_in_hostname, reconscanning] | NaN | [ip_in_hostname, reconscanning] | [ip_in_hostname, reconscanning] |
| 8 | NaN | NaN | [ip_in_hostname, reconscanning, staticIP, Mirai] | NaN | [ip_in_hostname, reconscanning, staticIP, Mirai] | NaN | NaN |
| 117 | [reconscanning] | [reconscanning] | [reconscanning] | NaN | NaN | NaN | NaN |
| 14 | NaN | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] |
| 37 | [reconscanning, Ping Scanner] | NaN | NaN | [reconscanning, Ping Scanner] | NaN | [reconscanning, Ping Scanner] | [reconscanning, Ping Scanner] |
| 15 | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] | [reconscanning] |
| 1 | NaN | [reconscanning, SSH Scanner] | [reconscanning, SSH Scanner] | NaN | NaN | [reconscanning, SSH Scanner] | NaN |
| 4 | NaN | NaN | NaN | NaN | NaN | [ip_in_hostname, reconscanning] | [ip_in_hostname, reconscanning] |
| 75 | [dshield, reconscanning, Ping Scanner] | [dshield, reconscanning, Ping Scanner] | [dshield, reconscanning, Ping Scanner] | NaN | [dshield, reconscanning, Ping Scanner] | [dshield, reconscanning, Ping Scanner] | [dshield, reconscanning, Ping Scanner] |
| 34 | NaN | NaN | NaN | [ip_in_hostname, reconscanning, staticIP] | [ip_in_hostname, reconscanning, staticIP] | [ip_in_hostname, reconscanning, staticIP] | [ip_in_hostname, reconscanning, staticIP] |
| 6 | NaN | [reconscanning, Eternalblue, SMB Scanner] | NaN | NaN | [reconscanning] | [reconscanning] | [reconscanning, Eternalblue, MSSQL Scanner, SMB Scanner] |
| 12 | [reconscanning] | [reconscanning] | [reconscanning] | NaN | [reconscanning] | [reconscanning] | [reconscanning] |
| 7 | NaN | NaN | NaN | [attemptlogin, reconscanning, SSH Bruteforcer, Squid Proxy Scanner] | [attemptlogin, reconscanning, Open Proxy Scanner, SSH Bruteforcer, Squid Proxy Scanner] | NaN | NaN |

Tabuľka 6.3: Vývoj tagov podobnostných skupín

Analýza výskytu kombinácií tagov ukázala, že cca polovica podobnostných skupín mala priradené pomerne vzácne kombinácie tagov. To je ukázané na obrázku 6.5. Výsledky naznačujú, že by malo byť možné identifikovať opakujúce pod. skupiny už na základe výsledkov z jedného dňa analýzy. Preto vznikla domnienka, že je možné vytvoriť prediktor opakovania výskytu.

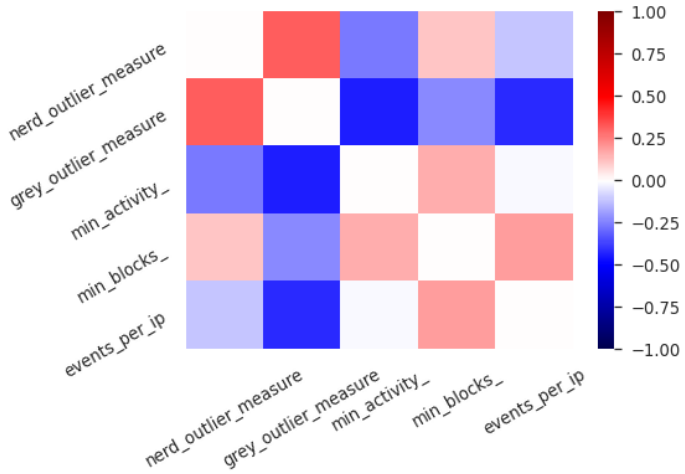
Na obrázku 6.5 sú zobrazené najčastejšie kombinácie tagov podobnostných skupín. Bunky obsahujú hodnoty výskytu danej kombinácie. V ľavom stĺpci sú hodnoty zo všetkých podobnostné skupín, v pravom iba opakujúce sa podobnostné skupiny. Kombinácií typov bolo veľmi veľa, preto boli vybrané najčastejšie[3]. Kombinácie, ktoré sa do grafu nezmestili spadajú do posledného riadku 'other'.

[3]Vybrané na základe normalizovaných histogramov.




Obr. 6.5: Rozdiel výskytu typového zloženia skupín (NERD a GreyNoise)

V snahe zistiť ďalšie rozdielnosti sa štatisticky analyzovali príznaky opakujúcich sa skupín[4] a zisťovali sa rozdiely oproti celej sade. Vypočítala sa rozdielová kovariančná matica, je na obrázku 6.6. Ukazuje, že boli zistené odchýlky od normálu na niektorých dvojiciach príznakov. Príznaky `nerd outlier measure` a `grey outlier measure` medzi sebou korelujú viac a spolu vykazujú vyššiu negatívnu koreláciu voči zbytku príznakov. Príznak `events per IP` viac negatívne koreluje s `grey outlier measure`.




Obr. 6.6: Rozdiel kovariančnej matice vybraných príznakov.

Obrázok 6.6 zobrazuje rozdielovú kovariančnu maticu vybraných príznakov. Nech $X$ kovariancia bola spočítaná z celej sady a $Y$ kovariančná matica bola spočítaná len z opakujúcich sa skupín, potom na obrázku je matica $Y - X$. To umožňuje relatívne porovnanie štatistických hodnôt.

Kovariancia však sama nestačí pre zistenie, či sú skupiny separovateľné, preto boli analyzované rozloženia jednotlivých príznakov. V prílohe na obrázku B.2 je regresný graf. Zobrazuje rozloženia bodov pre páry príznakov. Z vizuálnej analýzy vidieť odlišnosť rozložení. Body sú čiastočne lineárne separované, čo uľahčí trénovanie klasifikátora, pre identifikáciu opakujúcich sa skupiny od zbytku sady.

[4]Opakujúce sa skupiny sú členmi podobnostných skupín, ktoré majú výskyt v aspoň dvoch dňoch.

Opakujúce sa skupiny sú pomerne vzácne a majú špecifické vlastnosti. V porovnaní z ostatnými skupinami nemajú taký vysoký pomer škodlivých IP. V detekovaných skupinách boli nájdení aj výskumní skeneri.

# Kapitola 7

# Záver

Bola vyvinutá metóda pre koreláciu skupín IP na základe vzoru príchodov bezpečnostných udalostí v čase. Prototyp metódy funguje a odhaľuje skupiny vyskytujúce sa v dátach. Bol nájdený spôsob ako minimalizovať vplyv náhodných korelácií a nájsť príznaky, či sú identifikované skupina pravé.

Experimenty s metódou ukázali, že v dátach existuj minimálny podiel opakujúcich sa IP adries, ktoré generujú udalosti kontinuálne, alebo aspoň po dobu niekoľkých dní. V dátach boli nájdené skupiny, ktoré niesli širokú škálu označení, od obyčajných skenov cez bruteforce útoky až po skupiny označené ako botnet Mirai. V dátach sa našli aj skeny známych výskumných organizácií ako napríklad Shadowserver. Studnica znalostí z ďaleka nie je vyčerpaná.

V budúcnosti by mal byť preskúmaný vplyv agregačného intervalu na počet objavených skupín. Mal by byť vylepšený spôsob vyhodnocovania reputačných informácií z tretích strán, tak aby dokázal dodať jasnejšiu informáciu o tom, aký ma tagové zloženie skupin vzťah voči celej dátovej sade. Metóda by mala byť nasadená tak, aby priebežne deň za dňom spracovávala prúd bezpečnostných udalostí a výsledky zdieľala späť do Warden, či iného SIEM systému. Skupiny koordinovaných IP môžu byť využité pre sumarizáciu znalostí o jednotlivých IP do abstraktnejších skupín.

# Literatúra


[1] 2.3. Clustering — Scikit-Learn 0.23.1 Documentation.
URL https://scikit-learn.org/stable/modules/clustering.html

[2] Binomial Distribution. In *The Concise Encyclopedia of Statistics*, Springer New York, s. 44–45.
URL https://doi.org/10.1007/978-0-387-32833-1_34

[3] CESNET Network Topology.
URL https://www.cesnet.cz/services/ip-connectivity-ip/cesnet2-network-topology/?lang=en

[4] Comparing Python Clustering Algorithms — Hdbscan 0.8.1 Documentation.
URL https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html#agglomerative-clustering

[5] Fail2ban.
URL https://www.fail2ban.org/wiki/index.php/Main_Page

[6] Fakultní nemocnice Brno čelí kybernetickému útoku. Denně prověřuje na 20 podezření na koronavirus.
URL https://www.irozhlas.cz/zpravy-domov/fakultni-nemocnice-brno-kyberneticky-utok_2003130756_zit

[7] GreyNoise Intelligence.
URL https://greynoise.io/

[8] How to Accidentally Stop a Global Cyber Attacks.
URL https://www.malwaretech.com/2017/05/how-to-accidentally-stop-a-global-cyber-attacks.html

[9] LaBrea-Intro History.
URL http://labrea.sourceforge.net/Intro-History.html

[10] The Machine Learning Algorithm Cheat Sheet.
URL https://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/

[11] Mapping Mirai: A Botnet Case Study.
URL https://www.malwaretech.com/2016/10/mapping-mirai-a-botnet-case-study.html

[12] Mentat.
URL https://mentat.cesnet.cz/cs/index

[13] Network Entity Reputation Database (NERD).
URL https://nerd.cesnet.cz/

[14] Warden.
URL https://warden.cesnet.cz/cs/index

[15] Abaid, Z.; Kaafar, M. A.; Jha, S.: Early Detection of In-the-Wild Botnet Attacks by Exploiting Network Communication Uniformity: An Empirical Study. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, s. 1–9, doi:10.23919/IFIPNetworking.2017.8264866.

[16] Ahmed, M.; Naser Mahmood, A.; Hu, J.: A Survey of Network Anomaly Detection Techniques. ročník 60: s. 19–31, ISSN 1084-8045, doi:10.1016/j.jnca.2015.11.016.
URL http://www.sciencedirect.com/science/article/pii/S1084804515002891

[17] Bartoš, V.: NERD: Network Entity Reputation Database. In *Proceedings of the 14th International Conference on Availability, Reliability and Security - ARES '19*, ACM Press, ISBN 978-1-4503-7164-3, s. 1–7, doi:10.1145/3339252.3340512.

[18] Berasategi, A.: Earth Mover's Distance.
URL https://towardsdatascience.com/earth-movers-distance-68fff0363ef2

[19] Berba, P.: Understanding HDBSCAN and Density-Based Clustering.
URL https://towardsdatascience.com/understanding-hdbscan-and-density-based-clustering-121dbee1320e

[20] Cejka, T.; Bartos, V.; Svepes, M.; aj.: NEMEA: A Framework for Network Traffic Analysis. In *2016 12th International Conference on Network and Service Management (CNSM)*, IEEE, s. 195–201, doi:10.1109/CNSM.2016.7818417.

[21] CESNET: Intrusion Detection Extensible Alert [IDEA].
URL https://idea.cesnet.cz/en/index

[22] Choi, S.-S.; Cha, S.-H.; Tappert, C.: A Survey of Binary Similarity and Distance Measures. ročník 8.

[23] Dobeš, E.: Analýza síťových bezpečnostních hlášení.

[24] Hofstede, R.; Čeleda, P.; Trammell, B.; aj.: Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX. ročník 16, č. 4: s. 2037–2064.

[25] Husák, M.; Komárková, J.; Bou-Harb, E.; aj.: Survey of Attack Projection, Prediction, and Forecasting in Cyber Security. ročník 21, č. 1: s. 640–660.

[26] Kosub, S.: A Note on the Triangle Inequality for the Jaccard Distance. 1612.02696.
URL http://arxiv.org/abs/1612.02696

[27] McInnes, L.; Healy, J.; Astels, S.: Hdbscan: Hierarchical Density Based Clustering. ročník 2, č. 11: str. 205, ISSN 2475-9066, doi:10.21105/joss.00205.
URL http://joss.theoj.org/papers/10.21105/joss.00205

[28] Mehrotra, K. G.; Mohan, C. K.; Huang, H.: Anomaly Detection. In *Anomaly Detection Principles and Algorithms*, editácia K. G. Mehrotra; C. K. Mohan; H. Huang, Terrorism, Security, and Computation, Springer International Publishing, ISBN 978-3-319-67526-8, s. 21–32, doi:10.1007/978-3-319-67526-8_2.

[29] Morris, A.: The Background Noise of the Internet.
URL https://www.slideshare.net/andrewwantsyou/the-background-noise-of-the-internet

[30] Morris, A.: Identifying and Correlating Internet-Wide Scan Traffic to Newsworthy Events.
URL https://www.slideshare.net/andrewwantsyou/identifying-and-correlating-internetwide-scan-traffic-to-newsworthy-security-events

[31] Ogu, E. C.; Ojesanmi, O. A.; Awodele, O.; aj.: A Botnets Circumspection. ročník 10, č. 11: str. 337, doi:10.3390/info10110337.

[32] Papalexakis, E. E.; Beutel, A.; Steenkiste, P.: Network Anomaly Detection Using Co-Clustering. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, s. 403–410, doi:10.1109/ASONAM.2012.72.

[33] Radford, B. J.; Richardson, B. D.; Davis, S. E.: Sequence Aggregation Rules for Anomaly Detection in Computer Network Traffic. 1805.03735.
URL http://arxiv.org/abs/1805.03735

[34] Rousseeuw, P. J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. ročník 20: s. 53–65, ISSN 0377-0427, doi:10.1016/0377-0427(87)90125-7.
URL http://www.sciencedirect.com/science/article/pii/0377042787901257

[35] Sadoddin, R.; Ghorbani, A.: Alert Correlation Survey: Framework and Techniques. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap between PST Technologies and Business Services*, PST '06, Association for Computing Machinery, ISBN 1-59593-604-1, doi:10.1145/1501434.1501479.
URL https://doi.org/10.1145/1501434.1501479

[36] Schiller, C.; Binkley, J.; Evron, G.; aj.: Botnets: The Killer Web App. In *Botnets: The Killer Web App*, Syngress, prvé vydanie, ISBN 978-1-59749-135-8, s. 29–76.

[37] Siffer, A.; Fouque, P.-A.; Termier, A.; aj.: Anomaly Detection in Streams with Extreme Value Theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, ISBN 978-1-4503-4887-4, s. 1067–1075, doi:10.1145/3097983.3098144.
URL https://dl.acm.org/doi/10.1145/3097983.3098144

[38] Siffer, A.; Fouque, P.-A.; Termier, A.; aj.: Anomaly Detection with Extreme Value Theory.

[39] Swift, D.: A Practical Application of SIM/SEM/SIEM Automating Threat Identification: str. 41.

[40] Tan, Z.; Jamdagni, A.; He, X.; aj.: A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis. ročník 25, č. 2: s. 447–456, ISSN 1558-2183, doi:10.1109/TPDS.2013.146.

[41] Wade, S.; Ghahramani, Z.: Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion). ročník 13, č. 2: s. 559–626, ISSN 1936-0975, 1931-6690, doi:10.1214/17-BA1073.
URL https://projecteuclid.org/euclid.ba/1508378464

[42] Xiang, C.; Yong, P. C.; Meng, L. S.: Design of Multiple-Level Hybrid Classifier for Intrusion Detection System Using Bayesian Clustering and Decision Trees. ročník 29, č. 7: s. 918–924, ISSN 01678655, doi:10.1016/j.patrec.2008.01.008.
URL https://linkinghub.elsevier.com/retrieve/pii/S0167865508000251

# Príloha A

# Ukážka IDEA správ

## Vzorová správa


```
{
   "Format": "IDEA0",
   "ID": "4390fc3f-c753-4a3e-bc83-1b44f24baf75",
   "CreateTime": "2012-11-03T10:00:02Z",
   "DetectTime": "2012-11-03T10:00:07Z",
   "WinStartTime": "2012-11-03T05:00:00Z",
   "WinEndTime": "2012-11-03T10:00:00Z",
   "EventTime": "2012-11-03T07:36:00Z",
   "CeaseTime": "2012-11-03T09:55:22Z",
   "Category": ["Fraud.Phishing"],
   "Ref": ["cve:CVE-1234-5678"],
   "Confidence": 1,
   "Note": "Synthetic example",
   "ConnCount": 20,
   "Source": [
      {
         "Type": ["Phishing"],
         "IP4": ["192.168.0.2-192.168.0.5", "192.168.0.10/25"],
         "IP6": ["2001:0db8:0000:0000:0000:ff00:0042::/112"],
         "Hostname": ["example.com"],
         "URL": ["http://example.com/cgi-bin/killemall"],
         "Proto": ["tcp", "http"],
         "AttachHand": ["att1"],
         "Netname": ["ripe:IANA-CBLK-RESERVED1"]
      }
   ],
   "Target": [
      {
         "Type": ["Backscatter", "OriginSpam"],
         "Email": ["innocent@example.com"],
         "Spoofed": true
      },
      {
```

```
            "IP4": ["10.2.2.0/24"],
            "Anonymised": true
         }
      ],
      "Attach": [
         {
            "Handle": "att1",
            "FileName": ["killemall"],
            "Type": ["Malware"],
            "ContentType": "application/octet-stream",
            "Hash": ["sha1:0c4a38c3569f0cc632e74f4c"],
            "Size": 46,
            "Ref": ["Trojan-Spy:W32/FinSpy.A"],
            "ContentEncoding": "base64",
            "Content": "TVpqdXN0a2lkZGluZwo="
         }
      ],
      "Node": [
         {
            "Name": "cz.cesnet.kippo-honey",
            "Type": ["Protocol", "Honeypot"],
            "SW": ["Kippo"],
            "AggrWin": "00:05:00"
         }
      ]
}
```

## Vzorová IDEA meta správa

Meta správa je agregáciou viacerých ďalších správ

```
{
"Format": "IDEA0",
"ID": "1234",
"CorrelID": [
"3688762d-2efa-44a8-9ea5-34a57b3ae0c7",
"ae6d9ac6-6389-407f-9d7e-58b9692c6eaa"
],
"DetectTime": "2018-08-16T12:17:21.609+00:00",
"Category": ["Recon.Scanning"],
"Confidence": "0.8765",
"Description": "The source IP address follows a known pattern that is
expected to continue with the event described in this message.",
"Note": "cz.cesnet.tarpit_Recon.Scanning_2323,
cz.cesnet.nemea.hoststats_Recon.Scanning_None ==>
cz.cesnet.tarpit_Recon.Scanning_23",
"Source": [{"IP4": ["80.211.238.140"]}],
"Target": [{"Port":[23]}],
```

```
"Node": [{
"Name": "cz.cesnet.iabu",
"SW": "iABU",
"Type": ["Correlation", "Statistical"]
},
{
"Name": "cz.cesnet.tarpit"
}]
}
```

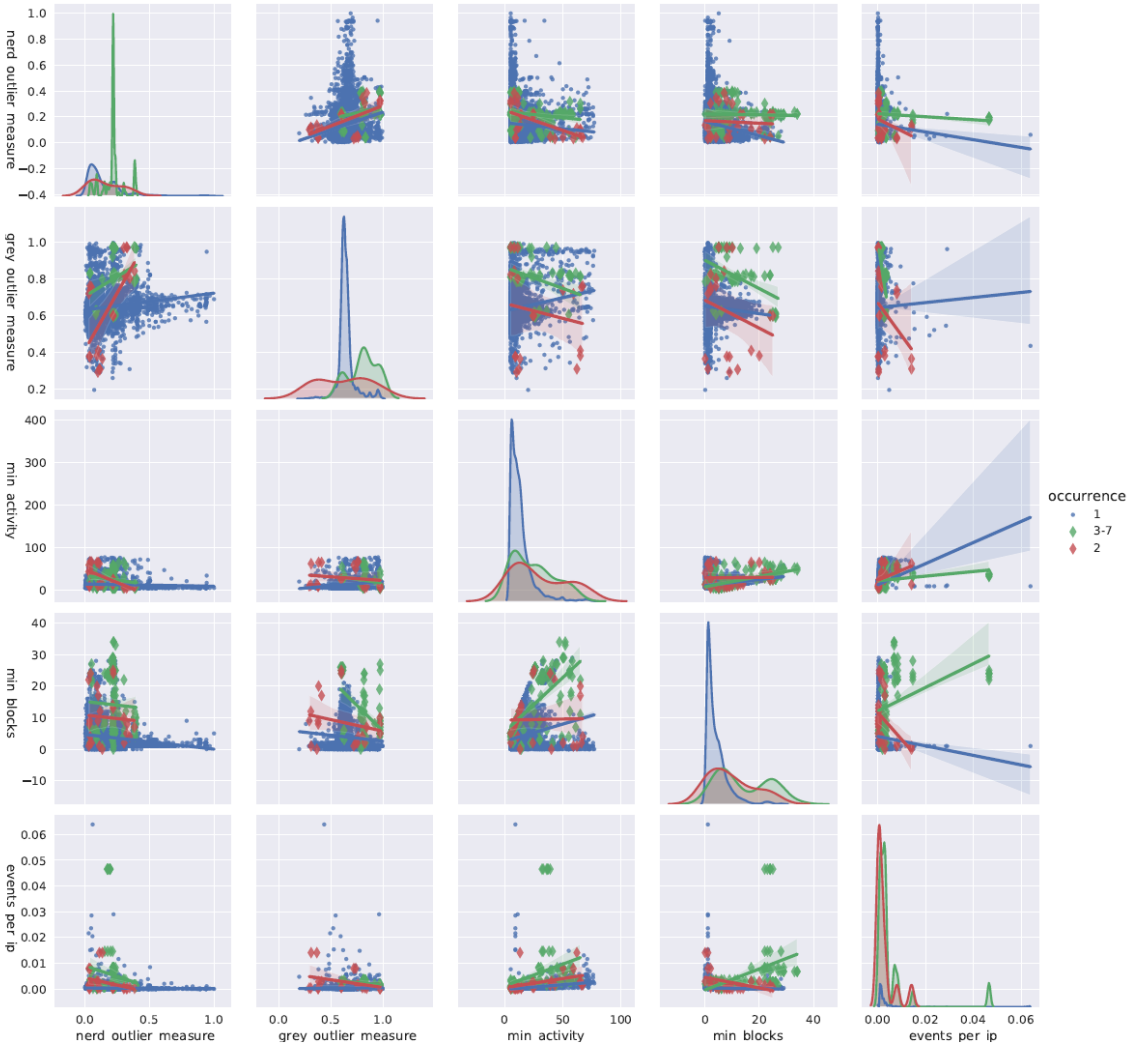
# Príloha B

# Výsledky experimentov - dodatky

## B.1 Porovnanie klastrovacích metód, dodatok


Obr. B.1: Rozloženie kvality klastrov pri rôznych metódach

## B.2 Vývoj skupín v čase, dodatok




Obr. B.2: Regresný párový graf príznakov skupín zobrazuje separovatľnosť opakujúcich sa skupín